IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-6 E-ISSN: 2347-2693

Uncertain Big Data Strategical Miner

H. V. Sapte [1*], S. S. Pallati [2], P. P. Pandit [3], A. S. Joshi [4], V. Jumb [5]

[1*] Xavier Institute of Engineering, Mumbai University, Mumbai, India
[2] Xavier Institute of Engineering, Mumbai University, Mumbai, India
[3] Xavier Institute of Engineering, Mumbai University, Mumbai, India
[4] Xavier Institute of Engineering, Mumbai University, Mumbai, India
[5] Xavier Institute of Engineering, Mumbai University, Mumbai, India

**Corresponding Author: harish.sapte10@gmail.com, Tel.: +91-865552252*

Available online at: www.ijcseonline.org

Received: 25/May/2017, Revised: 02/Jun/2017, Accepted: 20/Jun/2017, Published: 30/Jun/2017

***Abstract*—** There are many data mining algorithms which exist today for searching patterns from transactional databases. Most of them work only on precise data. But there are also situations in which these conventional algorithms fail, situations in which Data is uncertain in nature. Uncertain data can be explained as the one where items have probabilistic values associated with them. These probabilities express the likelihood of these items to be present in the transactions. In mining, the search tree produced is also one of the major factor of concern. The search space produced when dealing with uncertain data is much larger due to the presence of existential probabilities. This problem worsens when dealing with Big data. Considering all the above factors and concerns, an algorithm is specified and explained ahead. It allows users to express the interest in terms of constraints and uses the Map Reduce programming model to mine uncertain Big data for frequent patterns that satisfy the user-specified constraints. By using these user-specified constraints as inputs, the algorithm greatly reduces the search space for Big data mining of uncertain data, and returns only those patterns the users are interested in.

***Keywords*—** Big data models and algorithms, Big data analytics, Uncertain data mining, Frequent pattern mining

I. INTRODUCTION

In the current era of Big Data, companies and organizations possess overwhelming amounts of data with them. Data Mining has become crucial for extracting the most relevant strategic knowledge from this available raw data. Data mining is the process of extracting and analyzing data from different sources. Purpose of Data mining is to search for potentially useful information. It allows user to view data from different dimensions and thus helps in perceiving accordingly. Using it, interesting patterns can be easily obtained from raw ordinary data. Some real-life applications include (i) grouping people based on their interests (ii) classifying new insurers based on records of similar old claimants (iii) anomaly detection. Market Basket analysis is another prominent example where customer's behavior is analyzed and attempts are made to increase the sales of different products by understanding the customer's mentality.

A huge amount of valuable data is produced everyday by different real-life applications or industries like banking, finance, medical, telecommunication, and social web applications. This enormous data that needs to be processed has lead us into the new era of Big data [4]. This refers to interesting high-velocity, high-value, and/or high-variety data with volumes beyond the ability of commonly-used software to capture, manage, and process within a tolerable elapsed time. To enable enhanced decision making, insight, process optimization, data mining and knowledge discovery, new forms of processing data are now needed. This drive and motivates research and practices in Big data analytics and Big data mining [5,6,7,8]. Apache Hadoop is an open-source software framework used for distributed storage and processing of big data sets using the MapReduce programming model [17]. This MapReduce has the potential to handle parallel and distributed computing on large clusters or grids of nodes. As the name suggests, MapReduce involves two key functions: mapper and reducer. While implementation user need not worry about anything other than these two functions. Thus, processes like data partitioning, parallel scheduling and execution of programs, failure handling, inter machine communication are internally handled by MapReduce model, and need not be concerned about.

Apache Hadoop is an open-source software framework used for distributed storage and processing of big data sets using the MapReduce programming model [17]. This MapReduce

has the potential to handle parallel and distributed computing on large clusters or grids of nodes. As the name suggests, MapReduce involves two key functions: mapper and reducer. While implementation user need not worry about anything other than these two functions. Thus, processes like data-partitioning, parallel scheduling and execution of programs, failure handling, inter machine communication are internally handled by MapReduce model, and need not be concerned about.

Since frequent pattern mining was introduced, numerous studies have been conducted to mine frequent patterns from precise data. With these traditional databases, users know whether an item is present in (or is absent from) a transaction. However real-life applications involve uncertain data, partially due to inherent measurement inaccuracies, network latencies, sampling and duration errors, and intentional blurring of data to preserve anonymity [3,10,11,12]. This Uncertainty is indicated by the probability of individual items to be present (or not) in a transaction. Algorithms which work well on precise or certain data, are not suitable for uncertain data. This lead to an exploration of appropriate algorithms to accomplish the objective. So, basically the task is to mine user-interesting frequent patterns from Uncertain Big Data which is uncertain in nature.

Several pattern mining algorithms (e.g., the UF-growth, CUF-growth and PUF-growth algorithms) have been proposed for handling uncertain data [13,14,15]. Users may have some phenomena or attributes in mind on which they focus the mining. However, the algorithms mine patterns without user focus. Also, users often need to wait for a long period of time for numerous patterns, out of which only a tiny fraction may be interesting to the users. Hence, constrained pattern mining, which aims to find those frequent patterns that satisfy the user-specified constraints, is needed [16].

This paper is organized as follows. The next section gives information about related works. In Section III, principle concepts are explained. Section IV contains information about Implementation. We propose our algorithm for mining constrained frequent patterns from uncertain data using MapReduce. Evaluation results and conclusions are presented in Sections IV and V, respectively.

II. RELATED WORK

In the paper ‘Reducing the Search Space for Big Data Mining for Interesting Patterns from Uncertain Data' an algorithm and a methodology is proposed which uses MapReduce programming model for implementing constrained uncertain big data mining. It lets users to express their constraints which are succinct and anti-monotone in nature. Thus, the proposed algorithm can be used for mining of frequent patterns which satisfy the user-specified constraints [1].

In the paper Apriori-based Frequent Item set Mining Algorithms on MapReduce, three algorithms, namely SPC, FPC, and DPC, were proposed to investigate the performance of the Apriori-like algorithms in a MapReduce framework in this paper [18]. To enhance the performance of the Apriori-like frequent Item set mining algorithms, many parallelization techniques have come up. SPC is a simple conversion of the serial Apriori algorithm into the distributed MapReduce version. SPC finds the frequent k-itemsets in kth database scan (map-reduce phase), using mappers to generate candidate’s supports and reducers to collect global supports [18].

DPC dynamically collects candidates of variable lengths for counting by mappers according to the number of candidates and the execution time of previous map-reduce phases and thus outperforms both FPC and SPC and has good scalability [18].

FPC improves SPC by using a mapper to count the candidate k, (k+1), and (k+2) itemsets altogether in a map-reduce phase. Consequently, FPC effectively reduces the number of map-reduce phases DPC is proposed to strike a balance between reducing the number of MapReduce phases (by combining variable-length candidates) and increasing the number of pruned candidates [18].

In the paper ‘A Modified Approach to Mine Frequent Patterns from Uncertain Data’ presented by Jigisha V. Patel and Krunal J. Panchal, an algorithm is proposed which reduces the time complexity of PUF-tree algorithm. PUF tree requires less computation time and is compact as compared to other tree based algorithms. The paper proposed above replaces the tree structure with linear list data structure [2]. This further improvises the working performance of the PF tree algorithm. The Objective is to mainly minimize the time complexity.

In the paper ‘Mining Frequent Itemsets from Uncertain Data’ presented by Chun-Kit Chui, Ben Kao, and Edward Hung the problem addressed is to mine frequent Itemsets from uncertain data under a probabilistic framework. As it uncertain data, transactions have existential values associated with each item and a formal definition of frequent patterns is given under such an uncertain data model conventional mining algorithms are proven to be computationally inefficient under such a model [9]. A data trimming framework is proposed to improve mining efficiency. Through some research and experiments, it is proven that the data trimming technique can achieve significant savings in both CPU cost and I/O cost.

III. PRINCIPAL CONCEPTS

These are the key concepts which are important for this algorithm.

Existential Probability - The numerical value within the range (0,1], that represents the probability of the data item to exist in each transaction. Existential Probability of an item x in transaction Tj can be denoted as P (x, Tj)

Minimum Support - A Pattern X is frequent in an uncertain database if expSup(X) ≥ a user-specified minimum support threshold minsup. The presence of minsup helps to discover the frequent patterns from uncertain data.

User Defined constraints - This allows the user to use a set of constraints to specify his interest for guiding the process so that only those frequently occurring sets of items satisfying the user-specified constraints are found. Unnecessary computations, unrequired outputs, wastage of time are avoided by using this constraint. The constraint considered is anti-monotone in nature [17]. A constraint C is anti-monotone if and only if all subsets of a pattern satisfying C also satisfy C. The objective is to find patterns which satisfy user-defined constraints and have expected ≥ minsup, only then that pattern is considered as a valid pattern.

Expected Support - The expected support denotes the support values for itemsets when existential probabilities are involved. The method from C. K. Leung, R. K. MacKinnon, F. Jiang [1] is used to find out expSup(X) of Item set X in the dataset over all n transactions in the database which is given by equation (1):

$$expSup(X)=\sum_{i=1}^{n} P(X,t_i)=\sum_{i=1}^{n}\prod_{x\in X}P(x,t_i) \quad (1)$$

Map Reduce model - As the data-size is huge to handle this kind of data, the algorithm proposed uses high-level programming model called MapReduce [19]. MapReduce model process high volumes of data by using parallel and distributed computing on large clusters or grids of nodes (i.e., commodity machines), which consist of a master node and multiple worker nodes.

There are two important functions involved in this model as the name suggest “mapper” and “reducer”. The map function takes input in the form of <key, value> pair and returns a list of <key, value> pairs as an intermediate result:

map:<k1, v1>→ list of <k2, v2>,

where k1 & k2 are keys in the same or different domains, and v1 & v2 are the corresponding values in some domains.

Later in this process these intermediate results are shuffled and sorted. As the mapper function was carried out on each processor, similarly the reduce function is executed.

The reduce function combines the intermediate results and summarizes it to give the list of values associated with a given key (for all k keys) and returns (i) a list of k pairs of keys and values, (ii) a list of k values, or simply (iii) a single (aggregated or summarized) value:

reduce: <k2, list of value2>→list of <k3, value3>,

reduce: <k2, list of value2>→list of value3, or

reduce: <k2, list of value2>→value3.

IV. IMPLEMENTATION

The problem of mining constrained frequent patterns (i.e., valid frequent patterns) from Big Data that is Uncertain in nature can be done using the proposed system when user defined Minimum Support and user specified Constraints are provided along with the Big-Data Dataset.

In this section, we propose an algorithm that works on the map-reduce programming model to generate valid frequent patterns. The algorithm proposed here works in two sets of Map-Reduce Functions: (A) First one that mines Valid Frequent Singletons and (B) a second one that mines valid frequent non-singleton patterns. The following diagram gives us information about the way in which data flows through the various map and reduce functions.

Our software takes as input the dataset (on which mining is to be done), minimum support values (which the output patterns must possess) and the item values (which are the user defined constraints).

Initially the dataset is given as input to the first map function along with user required constraints. The map function sends items, along with their individual probability values as the key-value pairs to the first reducer. The reducer does the job of finding valid frequent singletons and displays those in an output file.

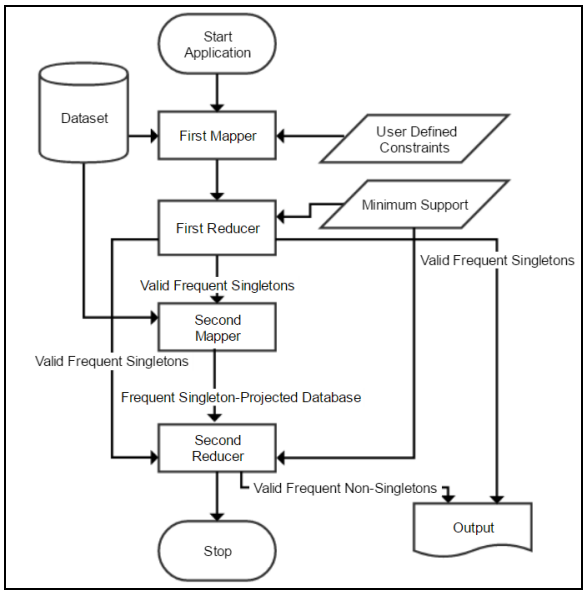


Figure 1. Flowchart showing data flow through various processes

The second map function uses dataset and valid frequent singletons to generate a singleton-projected database. This data is then used by second reducer along with frequent singleton values and minimum support value to generate all the valid frequent non-singletons.

A. Valid Frequent Singleton Mining

Pattern Mining is done by the algorithm using the following sequence of steps: i) Read large volumes of uncertain (big) data. ii) As each item of the data possesses existential probability value, these values are used for computing the Expected Support. iii) The Expected Support Calculation process is done within the Map-Reduce sets of functions. For computing singletons, the equation for Expected Support can be simplified as follows:

$$expSup(\{x\}) = \sum P(x, T_j) \quad (2)$$

where $P(x, T_j)$ denotes the existential probability value of item x in particular Transaction T_j. The Map-Reduce algorithm divides the data into several chunks or blocks of data and then distributes it among different processors. Every Processor that receives a data block, runs the Map function on that block. For every occurrence of Item x, belonging to particular transaction T_j, the first Map function of our algorithm will emit out $<x, P(x, T_j)>$ to the reducer function. Thus, our Map Function can be specified as follows:

For each $T_j \in$ partition of the uncertain Big data do
  for each item $x \in T_j$ and $\{x\}$ satisfies C_{AM}
    emit $<x, P(x, T_j)>$

Thus, we obtain a list of $<x, P(x, T_j)>$ values. Here x and $P(x, T_j)$ act as keys and values. These are grouped and sorted together to form $< x,$ list of $P(x, T_j)>$.

Now on these pairs of $<x,$ list of $P(x, T_j)>$, each processor runs the reduce function to further obtain the final expected support values of x (Singletons). Thus, our reducer function can be specified as follows:

For each $x \in$ valid x, list of $P(x, T_j)$ do
  set $expSup(\{x\}) = 0$;
  for each $P(x, T_j) \in$ list of $P(x, T_j)$ do
    $expSup(\{x\}) = expSup(\{x\}) + P(x, T_j)$;
  if $expSup(\{x\}) \geq$ minsup then
    emit $<\{x\}, expSup(\{x\})>$.

A higher-level abstraction viewpoint can be used to represent our map and reduce functions as follows:

map: <ID of transaction T_j, content of T_j> → list of <valid x, $P(x, T_j)$>

reduce: <valid x, list of $P(x, T_j)$> → list of <valid frequent $\{x\}$, $expSup(\{x\})$>

This output that has been obtained from the reduce function gives us the required Valid Frequent Singletons.

B. Valid Frequent Non-Singleton Mining

From the first set of Map Reduce functions, Valid Frequent Singletons along with their respective associated existential support values were obtained. For every transaction, we emit all valid frequent singletons with expSup values, present in that transaction. The key value is set to 1 for each key-value pair.

Thus, our map function can be specified as follows:

For each $T_j \in$ partition of the uncertain Big data do
  emit <1, Set of $\{\{x\}, expSup(x)\}$ present in T_j>

Now we use an algorithm that produces linear list data structure to mine frequent non-singleton patterns from uncertain data [2]. In this algorithm, the transaction would contain items along with expected support. All transactions of the projected database are scanned and the items are inserted in table in sorted manner, a pointer is maintained with each item and expected support is calculated for each entry in linear list. Considering all the items and all the possible combinations from the item, the ones with expected support more than the minimum support are considered as frequent patterns others are discarded. Thus, this algorithm finds frequent non-singleton patterns from uncertain data with minimum time complexity by using a linear list data structure [2].

The outputs of the mapper are sorted and grouped, thus providing with a key-value pair where key is 1 and value is a set of valid sets.

Thus, the reducer function derived from Patel et al. [2] is as follows:

For each Set of {{x}, expSup(x)} ∈ Set of valid Sets
Build linear list structure to find X
Generate X and expSup(X)

A higher-level abstraction viewpoint can be used to represent the second set of the map and reduce functions as follows:

map: <ID of transaction T_j, content of T_j> → <1, Set of {{x}, expSup(x)})>.

reduce: <1, Set of valid Sets>→ list of <valid frequent {X}, expSup({X})>.

Example: Consider the following example, where the dataset comprises of transactions along with item sets and their probabilities.

Table 4.1. Tiny Sample set of Uncertain Big Data

T1	1:0.98	3:0.2	9:0.36	13:0.14	23:0.51	25:0.32
T2	2:0.87	3:0.4	9:0.67	14:0.75	23:0.44	26:0.76
T3	1:0.63	3:0.76	10:0.23	15:0.8	23:0.5	25:0.3
T4	2:0.06	4:0.7	9:0.21	15:0.89	23:0.59	27:0.64
T5	2:0.99	3:0.66	10:0.87	14:0.38	23:0.68	26:0.51
T6	1:0.36	3:0.43	10:0.03	13:0.49	23:0.78	25:0.28

The user-specified constraints are 1, 2, 4, 10, 23 and the given Min-Support is 0.8. The algorithm used here reads the dataset. After reading the first transaction, the first mapper imparts the output as <1:0.98> and <23:0.52>. For second transaction, the output is <2:0.87> and <23:0.44>, it only takes those items that satisfy the user–defined constraints. Therefore, the first mapper produces the following result by reading one transaction at a time:

X→ invalid items

1: {0.98, 0.63, 0.36}, 2: {0.87, 0.06, 0.99}, 4: {0.7},10: {0.23, 0.87, 0.03}, 23: {0.51, 0.44, 0.5, 0.59, 0.68, 0.78}

The rest items along with probabilities which do not satisfy the user defined constraints are discarded like these ones:

3: {0.2, 0.4, 0.76, 0.66, 0.43}, 9: {0.36, 0.67, 0.21}, 13: {0.14, 0.49}, 14: {0.75, 0.38}, 15: {0.8, 0.89}, 25: {0.32, 0.3, 0.28}, 26: {0.76, 0.51}, 27: {0.64}.

The valid patterns that satisfied user-defined constraints are then shuffled and sorted. The first reducer re-reads this <1:[0.98, 0.63, 0.36]>, <2:[0.87, 0.06, 0.99]>, <4:[0.7]>, <10:[0.23, 0.87, 0.03]>, <23:[0.51, 0.44, 0.5, 0.59, 0.68, 0.78]> and produces the output as <1:1.97>, <2:1.92>, <10:1.13>, <23:3.5> and the key pair value of <4:0.7> is discarded as it does not satisfy min-sup constraint(0.8).

Therefore, the Valid Frequent Singleton patterns so generated are {1:1.97, 2:1.92, 10:1.13, 23:3.5}.

For further processing, the algorithm uses the uncertain big database comprising of transactions consisting of items along with their probabilities and user defined constraints, i.e. 1, 2, 10, 23, 4 and min-support which is 0.8.

Second Mapper remembers the valid singleton sets generated by the first Mapper Reducer function, in this example valid frequent singletons are 1, 2, 10, and 23. It sort singletons in decreasing order based on expected support value i.e. 23, 1, 2, 10. It re-reads transactions from the uncertain big database in the sorted order of singletons and eliminates infrequent singletons, and outputs a list comprising of these singleton items with key value equal to 1. After reading the first transaction, the second mapper gives output as <1: {1:0.98, 23:0.51}>, it does not contain 3 or any other infrequent item.

Similarly, after reading second transaction mapper function outputs {1: {2:0.87, 23:0.44}}. For third transaction, the mapper imparts the output as {1: {{1:0.63, 10:0.23, 23:0.5}} and so on. These pairs are then shuffled and sorted. Afterwards the reducer function reads <1: {frequent items in transactions}>. In this example reducer function reads <1: {{1:0.98, 23:0.51}, {2:0.87, 23:0.44}, {1:0.63, 10:0.23, 23:0.5}, {2:0.06, 23:0.59}, {2:0.99, 10:0.87, 23:0.68}, {1:0.36, 10:0.03, 23:0.78}}>.

Reducer function then reads each sub-transaction and arranges it in order of singletons list which we sorted earlier. A linear list table is created which consist of all valid single-tons and a pointer is maintained. Read first sub-transaction as {23:0.51, 1:0.98}, the first item in the transaction becomes key in the linear list table.

Table 4.2. After scanning first sub-transaction

23	→	1: 0.4998
1		
2		
10		

Table 4.3. After scanning second sub-transaction

23	→	1: 0.4998	→	2:0.3828
1				
2				
10				

Table 4.4. After scanning third sub-transaction

23	→	1:0.4998 +0.315=0.8148	→	2:0.382	→	10:0.115
1	→	10:0.1449				
2						
10						

Table 4.5. After scanning all sub-transactions

23	→	1: 1.0956	→	2:1.0919	→	10:0.73
1	→	10:0.1557				
2	→	10:0.8613				
10						

From this table now generate all possible patterns and check their expected support value if it is greater than or equal to minimum support output that pattern as frequent non-singleton pattern. In our example patterns generated are: (23,1):1.0956, (23,2):1.0919, (23,10):0.73, (23,1,2):1.0919, (23,1,10):0.73, (23,2,10):0.73, (1,10):0.1557, (2,10):0.8613. From these patterns only (23,1), (23,2), (23,1,2), (2,10) satisfies minimum support condition. Hence, the algorithm finds total a total eight frequent patterns satisfying user specified constraints.

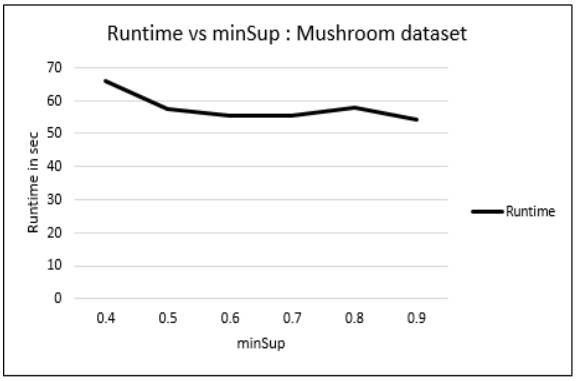
Total frequent patterns: <1:1.97, 2:1.92, 10:1.13, 23:3.5, (23,1):1.0956, (2,10):0.8613, (23,2):1.0919, (23,1,2):1.0919>

This is how the frequent patterns are generated.

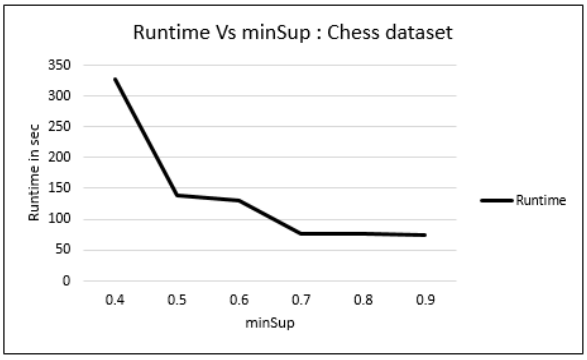
V. RESULTS

In this section, we evaluate our proposed algorithm in mining user-specified constraints from uncertain Big data. We used different benchmark datasets, which include real-life datasets (e.g., accidents, connect4, and mushroom) from the FIMI Repository (http://fimi.ua.ac.be/). For our experiments, the generated data range is about 1M transactions with an average transaction length of 10 items from a do-main of 1K items. As the above real-life and synthetic datasets originally contained only precise data, we assigned to each item contained in every transaction an existential probability from the range (0,1]. All experiments were run using either (i) a single machine with an Intel Core i5 4-core processor (1.73 GHz) and 8 GB of main memory running a 64-bit Windows 10 operating system, (ii) cluster of machines with the similar hardware configuration as mentioned in (i)All versions of the algorithm were implemented in the Java programming language. (ii)The version of Apache Hadoop 2.6.0 was used.

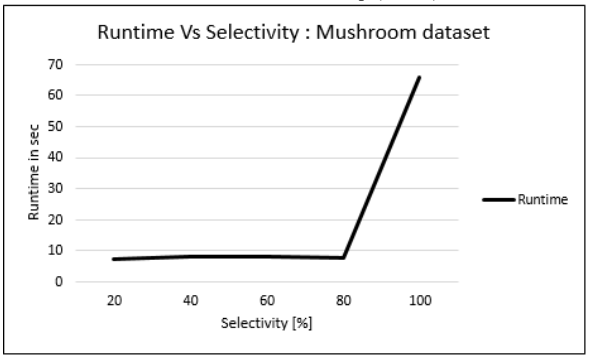
For comparison purpose, a software module called as 'SPMF Open-Source Data Mining Library' (http://www.philippe-fournier-viger.com) was used. This tool has an inbuilt functionality of allowing users to select the algorithms which they desire and then their software returns the expected result after running that user specified algorithm. Using this platform, a dataset having precise da-ta was loaded into the FP-growth algorithm and those resultant patterns were compared with the patterns generated by the proposed algorithm. Similarly, a dataset, which was uncertain, was run using U-Apriori Algorithm and patterns generated were compared with the patterns generated by the proposed algorithm. Resultant patterns from both the outcomes were found to be matching. Experiments were done with 100% selectivity. The benefits become more obvious by Figure. 3.1 to 3.4. They show that, when selectivity decreased (i.e., fewer frequent patterns satisfy the constraints), runtimes also decreased, because (i) fewer pairs were re-turned by the map function, (ii) fewer pairs were shuffled and sorted by the reduce function, and/or (iii) fewer constraint checks were performed.



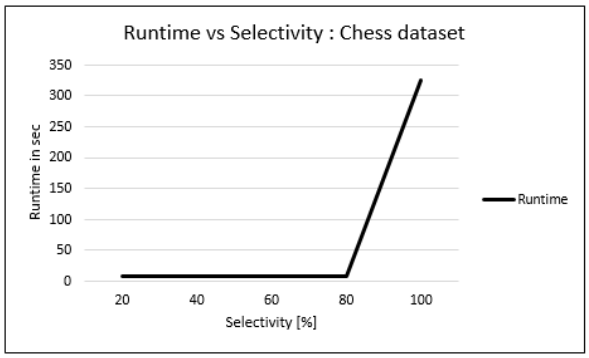
2.1. Runtime Vs minSup (Mushroom)



2.2. Runtime Vs minSup (Chess)



2.3. Runtime Vs Selectivity (Mushroom)



2.4. Runtime Vs Selectivity (Chess)

Figure. 2. Experimental Results

VI. CONCLUSION

Algorithms existing today mostly focus on association analysis enabled by mining interesting patterns from precise databases. However, there are situations in which data are uncertain, for which very few algorithms have been created. The Items in each transaction of these probabilistic data-bases of uncertain data are usually associated with existential probabilities expressing the likelihood of these items to be present in the transaction, making the search space for mining from uncertain data much larger than mining from precise data. This problem worsens as we start working with Big data. Furthermore, in many real-life applications, users may be interested in only a tiny portion of this large search space. To avoid wasting lots of time and space in computing all frequent patterns first and pruning uninteresting ones as a post-processing step, we have implemented a tree-based algorithm that (i) allows users to express their interest in terms of only anti-monotone (AM) constraints and (ii) uses MapReduce to mine uncertain Big data for frequent patterns that satisfy the user-specified constraints. Thus, this algorithm returns all and only those patterns that are interesting to the users.

REFERENCES

[1] C.K.-S. Leung , R. K. MacKinnon, F. Jiang, "*Reducing the Search Space for Big Data Mining for Interesting Pattern from Uncertain Data*" , 2014 IEEE International Congress on Big Data, pp.315-322, 2014.

[2] J.V. Patel, K. J. Panchal, "*A Modified Approach to Mine Frequent Patterns from Uncertain Data*", 2015 1st International Conference (NGCT), pp.612-615, 2015.

[3] C.K.-S. Leung, "*Mining uncertain data*", WIREs Data Mining and Knowledge Discovery, Vol.1 ,Issue.4, pp.316–329, July-Aug. 2011.

[4] S. Madden, "*From databases to big data*," IEEE Internet Computing, Vol.16, Issue.3, pp. 4–6, May–June 2012.

[5] Azzini, P. Ceravolo, "*Consistent process mining over Big data triple stores*", IEEE Big Data Congress 2013, pp. 54–61, 2013.

[6] Ölmezoğullari, I. Ari, "*Online association rule mining over fast data*", IEEE International Congress on Big Data 2013, pp.110–117, 2013.

[7] P. Agarwal, G. Shroff, P. Malhotra, "*Approximate incremental big-data harmonization*", IEEE Big Data Congress 2013, pp.118–125, 2013.

[8] Yang , S. Fong, "*Countering the concept-drift problem in big data using iOVFDT*", IEEE Big Data Congress 2013, pp.126–132, 2013.

[9] C.K.Chui, B.Kao, E.Hung, "*Mining Frequent Itemsets from Uncertain Data*", LNCS 2007, pp.47-58, 2007.

[10] C.K.-S. Leung , F. Jiang, "*Frequent itemset mining of uncertain data streams using the damped window model*", ACM SAC 2011, pp.950–955, 2011.

[11] C.K.-S. Leung , F. Jiang, "*Frequent pattern mining from time-fading streams of uncertain data*", DaWaK 2011 (LNCS 6862), pp. 252-264, 2011.

[12] Y. Tong, L. Chen, Y. Cheng, P.S. Yu, "*Mining frequent itemsets over uncertain databases*", PVLDB, Vol.5, Issue.11, pp.1650–1661, July 2012.

[13] C.K.-S. Leung, M.A.F. Mateo, D.A. Brajczuk, "*A tree-based approach for frequent pattern mining from uncertain data*", PAKDD 2008 (LNAI 5012), pp. 653–661, 2008.

[14] C.K.-S. Leung , S.K. Tanbeer, "*Fast tree-based mining of frequent itemsets from uncertain data*", DASFAA 2012 (LNCS 7238), pp. 272–287, 2012.

[15] C.K.-S. Leung, S.K. Tanbeer, "*PUF-tree: A compact tree structure for frequent pattern mining of uncertain data*", PAKDD 2013 (LNCS 7818), pp.13–25, 2013.

[16] D.N. Goswami, Anshu Chaturvedi., C.S. Raghuvanshi,"*An Algorithm for Frequent Pattern Mining Based On Apriori*", International Journal on Computer Science and Engineering(IJCSE), Vol.2, Issue.4, pp.942-947, 2010.

[17] J.Dean, S.Ghemawat, "*MapReduce: simplified data processing on large clusters*", CACM, Vol.51, Issue.1, pp.107-113, Jan. 2008.

[18] M.Y.Lin, P.Y.Lee, S. C. Hsueh, "*Apriori based frequent itemset mining algorithms on MapReduce*", ICUIMC 2012, art.76, 2012.

Authors Profile

Mr. V Jumb pursued Bachelor of Engineering from University of Mumbai, India in 2010 and Master of Engineering (Computer Engineering) from University of Mumbai, India in 2014. He is currently working as an Assistant Proffessor in the Department of Computer Engineering in Xavier Institute of Engineering, India, and has 6 years of Teaching Experience.

Mr. H V Sapte is currently pursuing a Bachelors degree in Engineering (Computer Engineering) from Xavier Institute of Engineering, India.

Ms. S S Pallati is currently pursuing a Bachelors degree in Engineering (Computer Engineering) from Xavier Institute of Engineering, India.

Mr P P Pandit is currently pursuing a Bachelors degree in Engineering (Computer Engineering) from Xavier Institute of Engineering, India.

Mr. A S Joshi is currently pursuing a Bachelors degree in Engineering (Computer Engineering) from Xavier Institute of Engineering, India.

