Positive and negative association rule mining in Hadoop's MapReduce environment

A Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Technology

in

Computer Science & Engineering

by

Anubhav Kumar Giri (20188004)
Aritra Chatterjee (20184196)
Arvind Kumar Yadav (20184135)
Anshuman Singh Chauhan (20184167)
Ajay Kumar Gond (20184107)

under the guidance of

Dr. Rupesh Kumar Dewang
(Professor)

to the

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
Motilal Nehru National Institute of Technology, Allahabad
Prayagraj, Uttar Pradesh (India)
June 2021

UNDERTAKING

I declare that the work presented in this report titled "*Positive and negative association rule mining in Hadoop's MapReduce environment*", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, Prayagraj, Uttar Pradesh (India) for the award of the ***Bachelor of Technology*** degree in ***Computer Science & Engineering***, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

June 2021
Prayagraj
U.P. (India)

(Anubhav Kumar Giri (20188004))

(Aritra Chatterjee (20184196))

(Arvind Kumar Yadav (20184135))

(Anshuman Singh Chauhan (20184167))

(Ajay Kumar Gond (20184107))

‘

CERTIFICATE

Certified that the work contained in the report titled "*Positive and negative association rule mining in Hadoop's MapReduce environment*", by *Anubhav Kumar Giri (20188004), Aritra Chatterjee (20184196), Arvind Kumar Yadav (20184135), Anshuman Singh Chauhan (20184167), Ajay Kumar Gond (20184107)*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr Rupesh Kumar Dewang
(Professor)
Computer Science and Engineering Dept.
M.N.N.I.T, Allahabad
Prayagraj, Uttar Pradesh (INDIA)

June 2021

Preface

In this paper, we present a Hadoop implementation of the Apriori algorithm. Using Hadoop's distributed and parallel MapReduce environment, we present an architecture to mine positive as well as negative association rules in big data using frequent itemset mining and the Apriori algorithm.

We also analyze and present the results of a few optimization parameters in Hadoop's MapReduce environment as it relates to this algorithm.

The results are presented based on the number of rules generated as well as the run-time efficiency.

Acknowledgements

The completion of this project required a lot of effort, guidance and support from many people. We feel privileged and honoured to have got this all along the development of the project. We would like to express our gratitude to our mentor, Dr. Rupesh Kumar Dewang (Professor) for his perennial support, guidance and monitoring throughout the making of this project. We would also like to acknowledge and thank our professors, colleagues and seniors for supporting us and enabling us to successfully complete our project.

Contents

Chapter 1

Introduction

Association rule mining is a well-known data mining technique used to find associations between items or itemsets. In today's big data environment, association rule mining has to be extended to big data. The Apriori algorithm is one of the most commonly used algorithms for association rule mining. Using the Apriori algorithm, we find frequent patterns, that is, patterns that occur frequently in data. The Apriori algorithm employs an iterative approach where k-itemsets are used to explore (k+1) itemsets. To find the frequent itemsets, first the set of frequent 1-itemsets are found by scanning the database and accumulating their counts. Itemsets that satisfy the minimum support threshold are kept. These are then used to find the frequent 2-itemsets. This process goes on until the newly generated itemset is an empty set, that is, until there are no more itemsets that meet the minimum support threshold. Then the itemsets are checked against a minimum confidence level to determine the association rules. The process of generating the frequent itemsets calls for repeated full scans of the database, and in this era of big data, this is a major challenge of this algorithm. Figure 1 presents a flow chart of how the Apriori algorithm works.

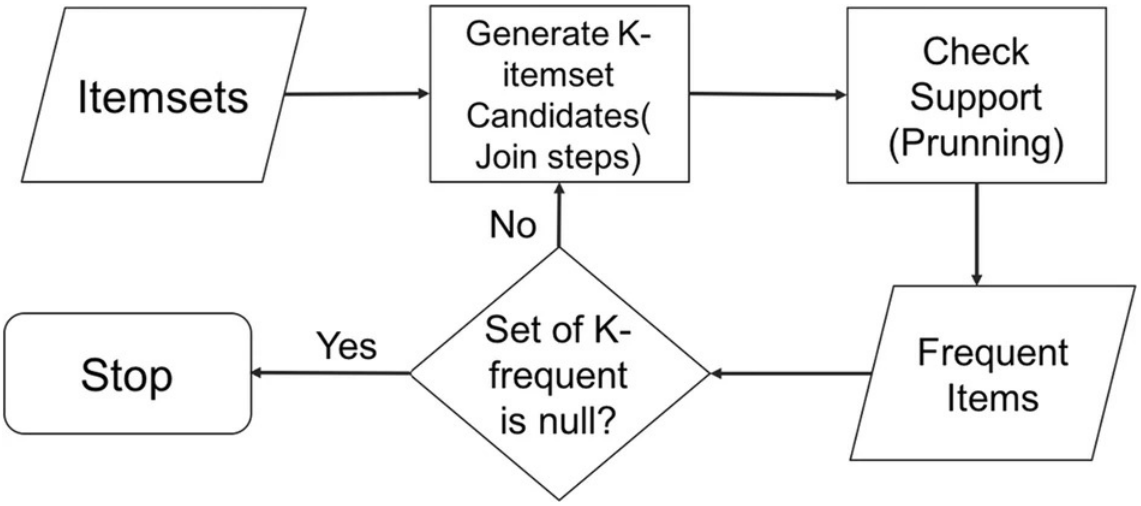

Figure 1: Figure 1

Traditional association rule mining algorithms, like Apriori, mostly mine positive association rules. Positive association rule mining finds items that are positively related to one another, that is, if one item goes up, the related item also goes up. Though the classic application of positive association rule mining is market basket analysis, applications of positive rule mining have been extended to a wide range of areas like biological datasets, web-log data, fraud detection, census data, etc. Negative association rules can be defined as items that are negatively correlated, that is, if one item goes up, the other item goes down. Negative association rule mining also has many applications, including the building of efficient decision support systems, in crime data analysis, in the health care sector, etc.

In this paper we present an architecture for positive as well as negative association rule mining in the big data environment using Hadoop's MapReduce environment using frequent itemset mining. Given the fact that repeated scans of the dataset are needed in the Apriori algorithm, the parallel and distributed structure of Hadoop should be availed of in an optimized way for mining positive as well as negative association rules in big data using the Apriori algorithm.

Chapter 2

Related Work

Positive association rule mining has been implemented in the MapReduce environment by many. Few works have also been done on negative association rule mining, but none have been done in the big data framework. Since so much work has already been done on positive association rule mining over the years, in this related works section we are limiting our discussion to works that have used the less frequently addressed negative association rule mining.

Brin et al. proposed a Chi-square test to find negative association rules. They used a correlation matrix to determine the relationships, positive as well as negative.

Aggrawal and Yu's approach was based on mining strong collective itemsets. They used the collective strength of an itemset I as:

$$\mathrm{C\,(I)} = \frac{1-v(I)}{1-E[v(I)]} \times \frac{E[v(I)]}{v(I)}$$

In the above equation, v(I) is the violation rate of an itemset I. It is the fraction of violations over the entire set of transactions and E[v(I)] is the expected value of v(I). The value of collective strengths range from 0 to , where 0 means that the items are perfectly negatively correlated and means the items are perfectly positively correlated. Aggrawal and Yu claim that this model has good computational efficiency.

Savasere et al.'s approach to finding negative association rules was by combining positive frequent itemsets with domain knowledge in the form of a taxonomy. After getting all the possible positive itemsets, some candidate negative itemsets were

selected based on the taxonomy used. The association rules were generated from the selected negative itemsets. This approach is difficult to generalize because it is domain specific and requires a predefined taxonomy. A similar method is described in [34].

Wu et al.'s algorithm finds both the positive and negative association rules. This algorithm finds rules in the forms: XY, XY and XY. The authors added "mininterest" with the support-confidence framework. Mininterest was used to check the dependency between two itemsets.

Teng et al.'s work, referred to as substitution rule mining (SRM), discovers a subset of negative association rules. This algorithm discovers the negative association rules in the form: XY. This algorithm first discovers the "concrete" items. Concrete items are items that have a high Chi-square value and exceeds the expected support. The correlation coefficient is found for each pair of items [29].

Antonie and Zaiane introduced an algorithm which mines strong positive and negative association rules based on Pearson's correlation coefficient. The correlation coefficient for the association rule XY is:

$$\emptyset = \frac{s(XY)s(\neg X \neg Y) - s(X \neg Y)s(\neg XY)}{\sqrt{(s(X)s(\neg Y)s(Y)s(\neg Y))}}$$

In this algorithm, positive and negative association rules are generated while calculating the correlation between each candidate itemset.

Thiruvady and Webb's algorithm, Generalized Rule Discovery (GRD), discovers top-k positive and negative association rules. They used leverage and the number of rules to be discovered.

The authors in proposed a new Apriori-based algorithm (PNAR) that utilizes the upward closure property to find negative association rules. If the support of X exceeds the minimum support threshold, then every YI and X Y= and (XY) also meets the support threshold.

Mahmood et al. used infrequent itemsets to determine positive as well as negative association rules. Positive association rule mining extracts frequent items or itemsets, but there may be many important items or itemsets with low support which get discarded in positive association rule mining. These infrequent items or itemsets, despite their low support, can produce important negative association rules. Hence

though the mining of negative association rules is important, the search space for negative association rule mining is actually more than for positive association rule mining since items with low support have to be retained. This would be a major challenge for the traditional sequential implementations of the Apriori algorithm, and even more challenging to implement on big data sequentially.

In summary, though there have been a few implementations of negative association rule mining, a parallel implementation of negative association rule mining on the MapReduce environment using big data has not been addressed. Oweis et al. presented an implementation of Lift in the Big Data environment, but used the standard approach that is used in Apriori's sequential implementation.

Chapter 3

Definitions

3.1 Definition 1

(Association rule) An association rule is stated in the form: $X \Rightarrow Y, where X \subseteq I, Y \subseteq I and X \cap Y = \phi$.

3.2 Definition 2

(Support) The support of a rule, s, in transaction set D, is the probability of X occurring in transaction set D.

3.3 Definition 3

(Confidence) The confidence of a rule is the conditional probability that the subsequent Y is true given the predecessor X. The formula for confidence is: $\frac{Support(X \cap Y)}{Support(X)}$

3.4 Definition 4

(Positive item) A positive item, i_k, is an item that is present in a transaction T.

3.5 Definition 5

(Negative item) A negative item, $\neg i_k$, is an item that is not present in a transaction T.

3.6 Definition 6

(Positive association rule) A positive association rule is in the form, $X \Rightarrow Y$, where the rule satisfies a minimum support and confidence threshold.

3.7 Definition 7

(Negative association rule) A negative association rule can also be expressed in the form $X \Rightarrow Y$, where $X \subseteq I, Y \subseteq IandX \cap Y = \phi$, and where X and/or Y contain at least one negative item. A negative association rule would be in one of the following forms: $\neg X \Rightarrow Y, X \Rightarrow \neg Y$ and $\neg X \Rightarrow \neg Y$, where $\neg standsfor \backslash not", andwhere$X $\cap Y = \phi.X \Rightarrow \neg Y$ refers to X occurring in the absence of Y; $\neg X \Rightarrow Y$ refers to Y occurring in the absence of X; $\neg X \Rightarrow \neg Y$ means not X and not Y.

Chapter 4

Experimental Design

In our implementation, we use the support, confidence, and lift framework to determine positive as well as negative association rules using frequent itemset mining. For positive association rules we will use Definition 6 and for negative association rules we will use Definition 7.

Negative association rules will be determined using lift. Given two itemsets, X and Y, lift is defined, by Eq. (1), as the probability of X and Y occurring together divided by the probability of X multiplied by the probability of Y. Lift can be defined by the formula: Lift(X,Y)=P(X U Y)/P(X)P(Y). If the resulting value is less than 1, there is a negative dependency between itemsets X and Y. If the resulting value is greater than 1, there is a positive dependency between itemsets X and Y. If the resulting value is one, the two itemsets have no dependency.

Support of the negative association rules will be of the form: $Supp(X \Rightarrow \neg Y) > min\_supp; Supp(\neg X \Rightarrow \neg Y) > min\_supp; Supp(\neg X \Rightarrow \neg Y) > min\_supp$.

Confidence of negative association rules will be in the form: $Conf(X \Rightarrow \neg Y) > min\_conf; Conf(\neg X \Rightarrow Y) > min\_conf; Conf(\neg X \Rightarrow \neg Y) > min\_conf$.

Chapter 5

Proposed Work

5.1 Handling Big Data

5.1.1 Hadoop and MapReduce

Hadoop is a large-scale distributed framework for parallel processing of big data. Based on the Google File System and Google's MapReduce, Hadoop is an open source project of Apache. Given that Hadoop is deployable on a large cluster of commodity computers, the size of Hadoop's Distributed File System (HDFS) depends on the size of the cluster and the hardware used. HDFS ensures fast and scalable access to the data. Files in HDFS are replicated, hence data stores in HDFS are fault tolerant.

Hadoop's distributed computing environment uses the MapReduce programming paradigm to support parallel processing of high volumes of data. Hadoop has a master/slave architecture. There is one master node and multiple slave nodes on each cluster. The slave nodes or worker machines are usually referred to as DataNodes and the master machine is referred to as the NameNode. The NameNode allocates the block ids and the DataNodes store the actual files. HDFS's file system divides a file into fixed block sizes. The default block size is usually 64 MB, but it this can be varied.

5.1.2 MapReduce

MapReduce has two main components: the Mapper and the Reducer. The Mapper takes the input key-value pair (k1, v1) from HDFS and calculates the output in the intermediate key value pair (k2, v2). Intermediate key value pairs are shuffled and exchanged between the Mapper and Reducer. The Reduce function takes the intermediate key value pairs and produces the final output (k3, v3). There is an optional Combiner function which can be used in between the Mapper and Reducer functions. Combiners are mainly used to reduce the communication cost of transferring the intermediate output from the mappers to the reducers.

5.2 Hadoop Implementation of Apriori Algorithm

In our Hadoop implementation of the Apriori algorithm, first the algorithm is used to discover frequent itemsets. From the frequent itemsets, we find the positive and negative association rules. MapReduce jobs are used to find the frequent itemsets and finally a map only job is used to find the positive as well as negative association rules. Figure 2 presents our Hadoop implementation of the Apriori algorithm diagramatically. The number of MapReduce iterations will depend on the number of itemsets being generated. In this diagram, we presented two MapReduce iterations and one map only function.

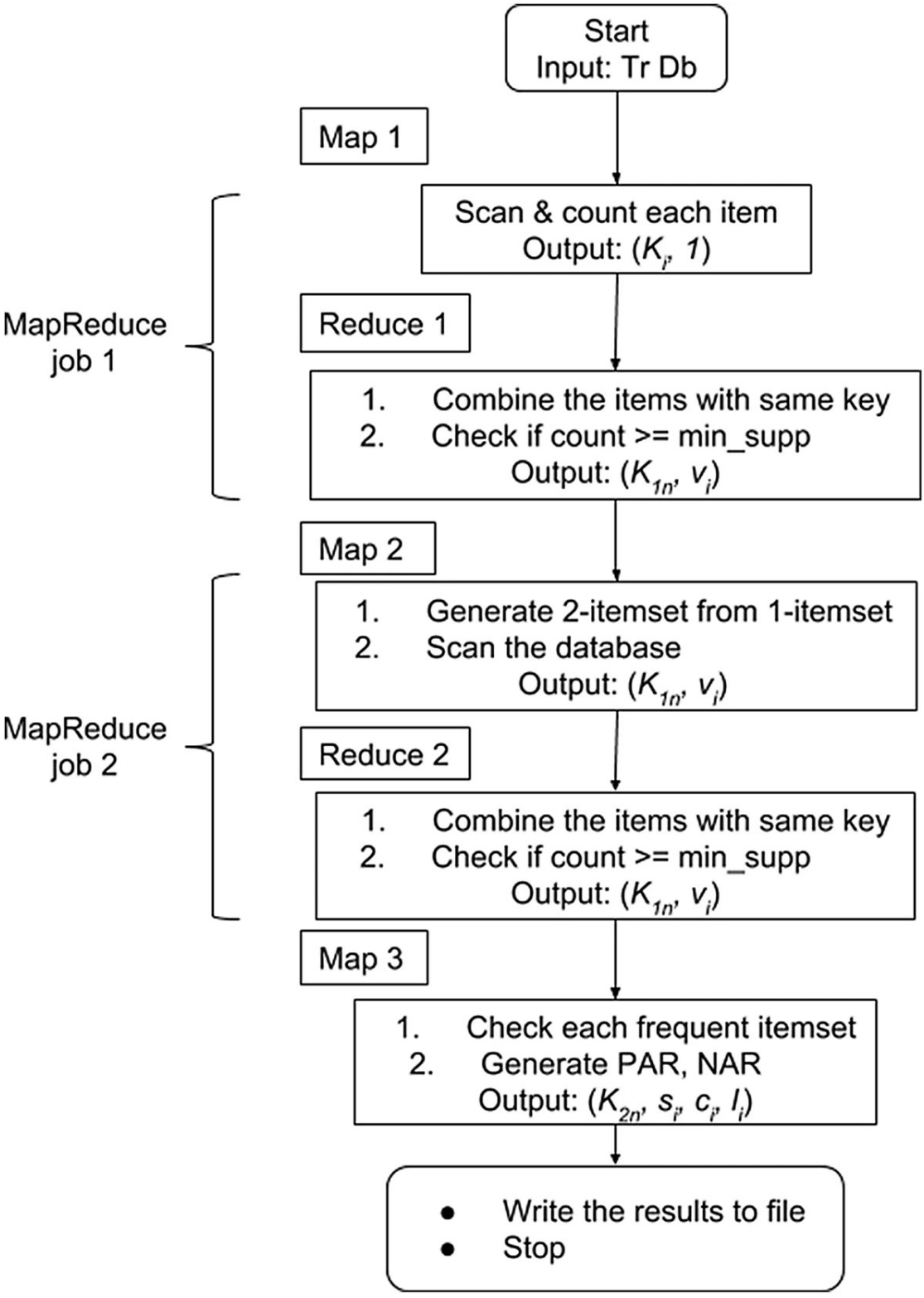

Figure 2: Figure 2

5.2.1 The first MapReduce job

In the first MapReduce job, we determine the frequent 1-itemsets. The input of the first MapReduce job is the transactional dataset. As the data is read into HDFS, data is divided into blocks and distributed over multiple mappers. The mapper reads one transaction at a time and outputs a (key, value) pair where key is the item and value is 1, in the form (item, 1). The (key, value) pairs are then passed to the reduce phase. The reducer will take these pairs and sum up the values of the respective keys. Reducers will output (item, total_count). Total_count is compared with min_supp and those equal to or above the min_supp threshold are kept as the frequent 1-itemset. The frequent 1-itemset and their respective support values are then written to distributed cache. Figure 3 presents the first MapReduce job diagramatically.

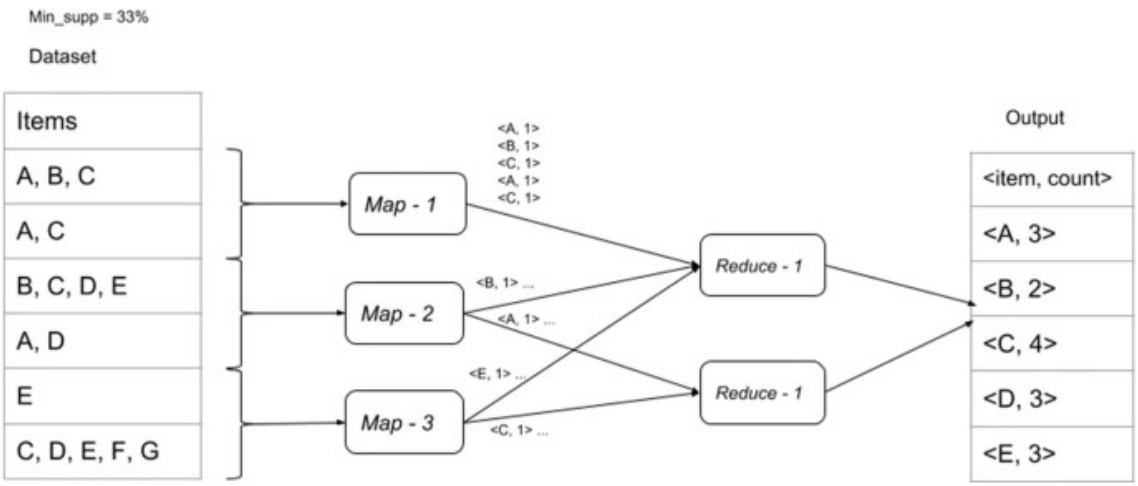

Figure 3: Figure 3

5.2.2 The second MapReduce job

In the second MapReduce job, we generate the frequent 2-itemsets. The input for the second MapReduce job is the frequent 1-itemset from the first MapReduce job as well as the transactions database. The frequent 1-itemset is read in from the

distributed cache. Paralleling the first MapReduce job, the 2-item (key, value) pairs are generated in the map phase. Then the 2-item (key, value) pairs are passed to the reduce phase of the second MapReduce job. Again, paralleling the first MapReduce job, the reducer then takes these pairs and sums up the values of the respective keys. Reducers will output (item, total_count). Total_count is then compared with min_supp and those equal to or above the min_supp threshold will be kept as the frequent 2-itemset. The frequent 2-itemset and their respective support values are then written to distributed cache. Figure 4 presents the second MapReduce job diagramatically.

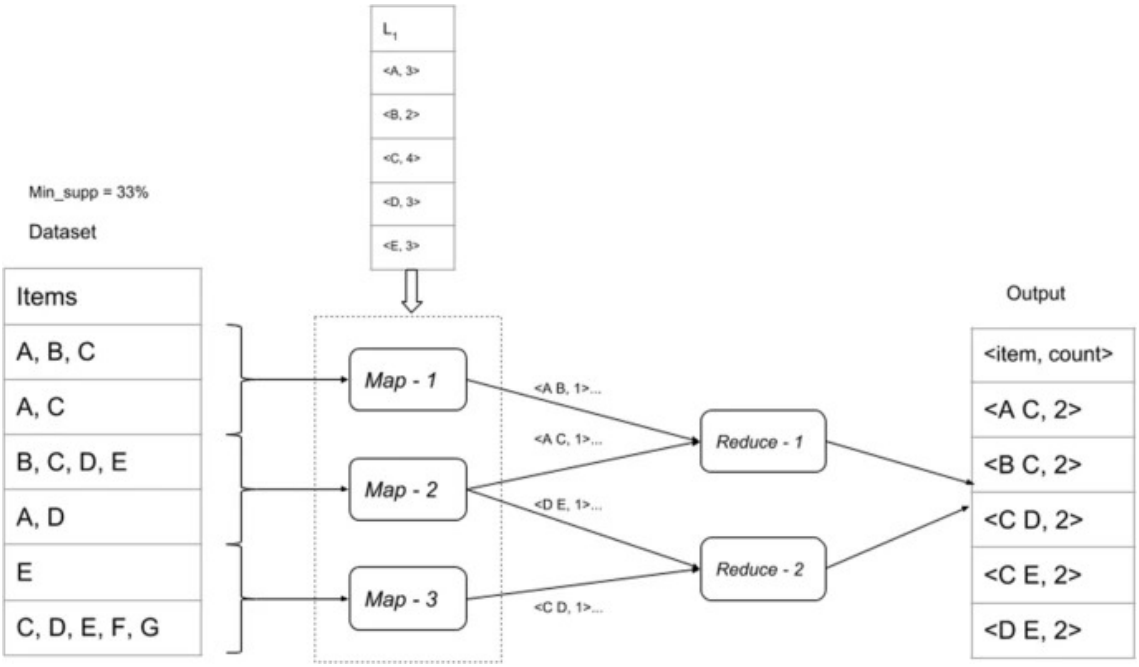

Figure 4: Figure 4

There would be as many similar MapReduce jobs as the number of itemsets required.

5.2.3 The third MapReduce (Map-only) job

The third MapReduce job is a map only operation. The output from MapReduce job 2 (frequent 2-itemsets) is read into MapReduce job 3 from the distributed cache. In MapReduce job 3, the confidence and lift are calculated for the frequent 2-itemsets to determine the positive as well as negative association rules. Figure 5 presents the third MapReduce job diagramatically.

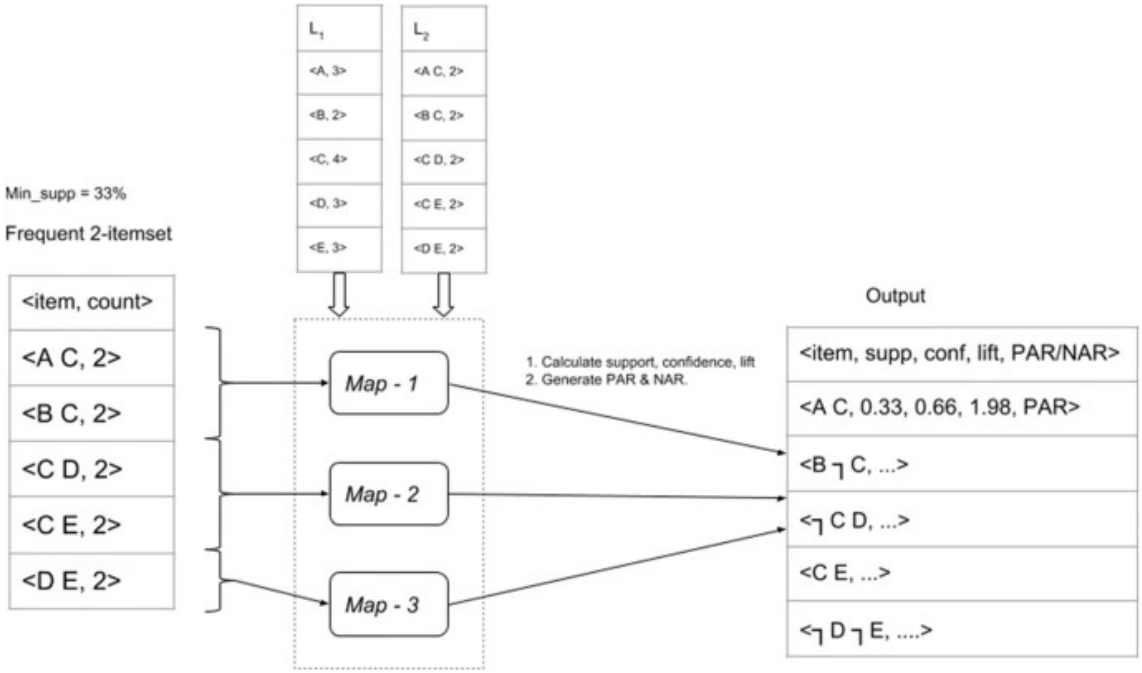

Figure 5: Figure 5

5.2.4 Determining positive and negative association rules

If the confidence of (A$\Rightarrow B$)$is greater than the minimum confidence threshold and the lift of (A \Rightarrow B) is greater than 1, than this is a positive association rule. If the confidence of either (A and not B) or (not A$

```
for each itemset A ∪ B = I, A ∩ B = φ do begin
/* generate rules of the form A ⇒ B. */
      If conf(A ⇒ B) ≥ min_conf && lift(A ⇒ B) ≥ 1
            then output the rule (A⇒B); PAR U (A⇒B)
      else
      /* generate rules of the form (A ⇒ ¬B) and (¬A ⇒ B). */

      if conf(A ⇒ ¬B) ≥ min_conf && lift(A ⇒ ¬B) ≥ 1
            output the rule (A ⇒ ¬B); NAR U (A ⇒ ¬B)
      else if conf(¬A ⇒ B) ≥ min_conf && lift(¬A ⇒ B) ≥ 1
            output the rule (¬A ⇒ B); NAR U (¬A ⇒ B)
      else if conf(¬A ⇒ ¬B) ≥ min_conf && lift(¬A ⇒ ¬B) ≥ 1
            output the rule (¬A ⇒ ¬B); NAR U (¬A ⇒ ¬B)
```

Figure 6: Algorithm in last Map task

5.3 Experimental Setup

For experimental analysis, we have used the following hardware and software specifications:

The software specifications of the setup are as follows:

- Java Development Kit=1.8
- Hadoop=2.7.3

The hardware specifications of the setup are as follows:

- Processor : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
- GPU : NVIDIA GeForce 1060 (6 GB VRAM)
- RAM : 16.00 GB DDR4 Memory
- Storage : SSD

5.4 Dataset

Due to cost and natural limitations, we could not set up a fully distributed Hadoop cluster at the moment, so we are falling back to pseudo-distributed cluster which used different JVM processes to emulate cluster nodes. This resulted in reduced performance and operating on huge datasets became unfeasible. Nevertheless, our algorithm works all the same in both the cases, just it would benefit from the highly parallel nature of the fully distributed cluster.

Due to these limitations, we are using a small dataset (mushroom.dat), with 8124 transactions.

5.5 Results

5.5.1 Graphical Results

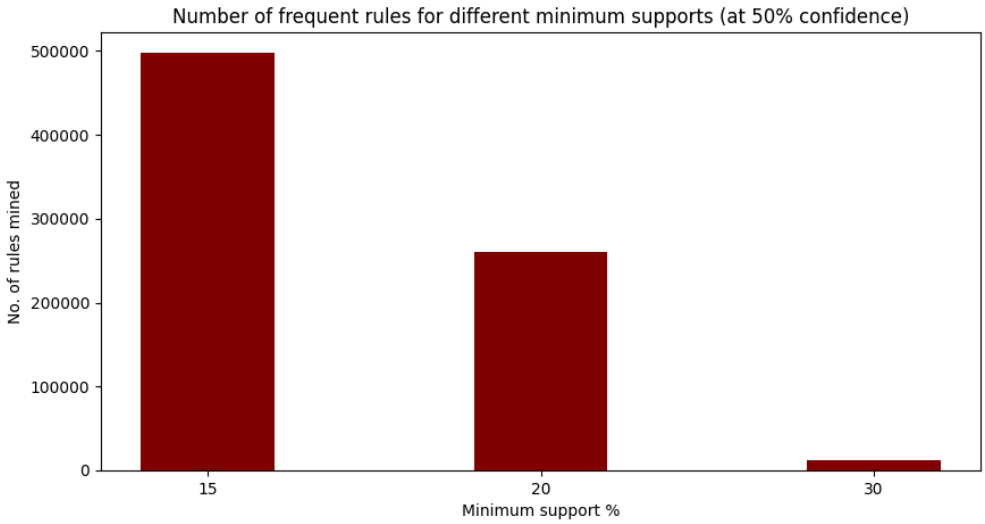

Figure 7: Plot 1

As proved in Figure 7, increasing the minimum support %, decreases the number of frequent item sets generated, and as a result, decreases the number of association rules generated.

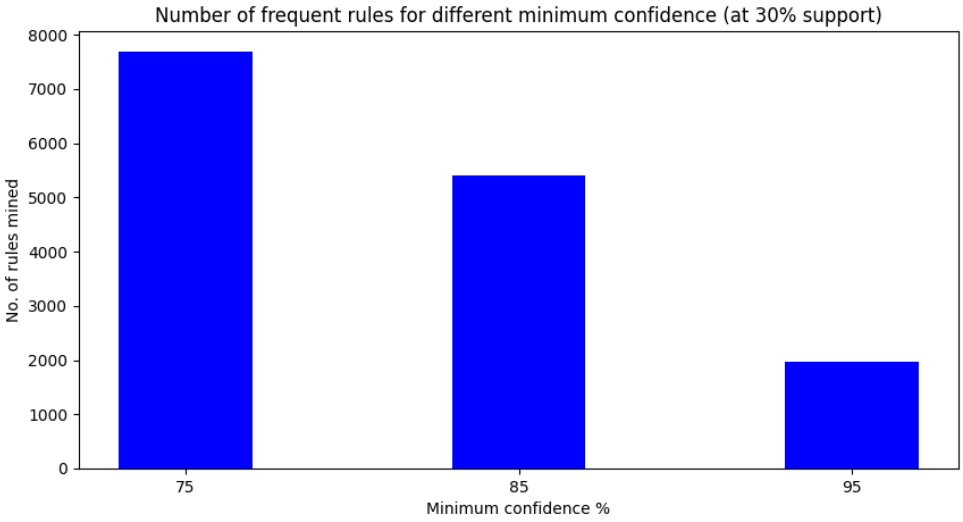

Figure 8: Plot 2

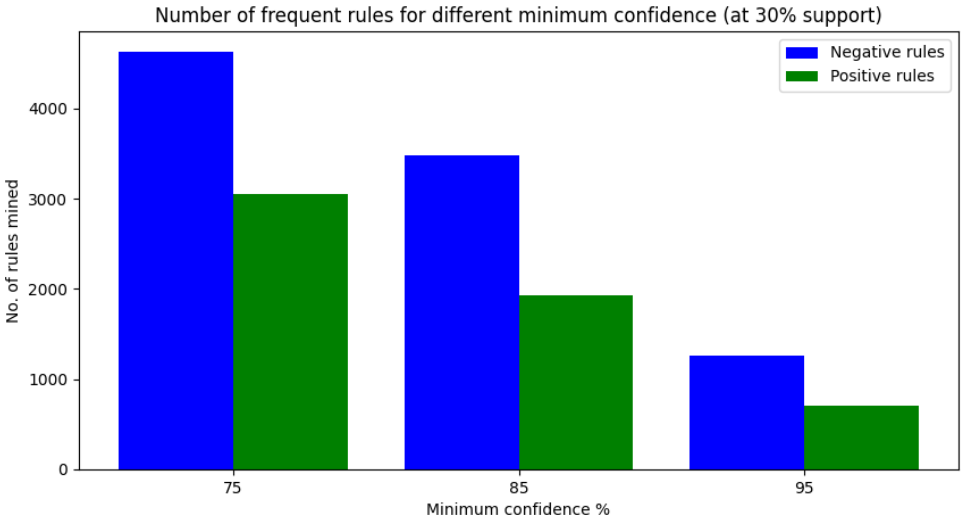

Figure 9: Plot 3

Through Figures 8 and 9, we observe that with increase in minimum confidence threshold, the number of rules generated decreases. We also observe that generally number of negative rules are greater than positive rules. Thus, ignoring negative association rules would lead to great loss of data.

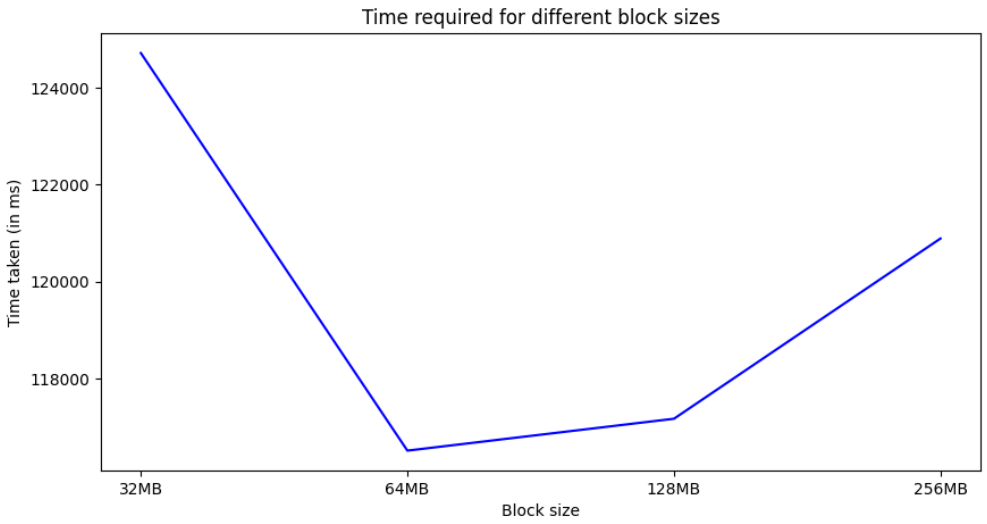

Figure 10: Plot 4

We also tried to find the effect of increasing block size of Hadoop HDFS on our algorithm's run-time. Through the above figure, we observe that by increasing block size, the run time of our algorithm first decreases, then slowly increases. This happens due to trade-off of overhead vs parallelism. Decreasing block size increases parallelism but also increases overhead of managing multiple small chunks of files. Increasing block size will similarly decrease parallelism but also decrease overheads. This, to some extent, justifies the shape of our result plot.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this paper, we presented a Hadoop implementation of the Apriori algorithm to mine positive as well as negative association rules. We performed experiments that showed that, for this dataset, there were more negative association rules than positive association rules, so if we mine just for positive association rules, we could be losing some information.

6.2 Future Work

In this project, we are using pseudo-distributed mode of Hadoop, which is basically just an emulation of a multi-node cluster on a single computer. Thus, the range of experimentation was pretty limited. As a result, we were also unable to test the effect of the number of slave nodes on the running time of our algorithm.

In the future, we hope to use a fully distributed cluster, by either creating it manually, or use a cloud cluster on Cloudera or AWS. This would help us test for even large datasets, and give more accurate results.

References

[1] Aggarwal CC, Yu PS. A new framework for item-set generation. In: Proceedings of the seventeenth ACM SIGACT- SIGMOD-SIGART symposium on principles of database systems, PODS'98. 1998. p. 18–24.

[2] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: ACM SIGMOD conference. New York City: ACM Press; 1993. p. 207–16.

[3] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: VLDB 1994 proceedings of the 20th international conference on very large data bases. 1994. p. 487–99.

[4] Antonie M-L, Zaïane OR. Mining positive and negative association rules: an approach for confined rules. In: Proc. of PAKDD. 2004. p. 27–38.

[5] Antonie L, Li J, Zaiane OR. Negative association rules. In: Frequent pattern mining. Berlin: Springer; 2014. p. 135–45.

[6] Bagui S, Just J, Bagui S. Deriving strong association mining rules using a dependency criterion, the lift measure. Int J Data Anal Tech Strat. 2009;1(3):297–313.

[7] Bala PK. A technique for mining negative association rules. In: Computer2009, Bangalore, India, 2009. p. 1–4.

[8] Brin S, Motwani R, Silverstein C. Beyond market basket: generalizing association rules to correlations. In: Proc. SIGMOD. 1997. p. 265–76.

[9] Cornelis C, Yan P, Zhang X, Chen G. Mining positive and negative association rules from large databases. In: Proc. of CIS. 2006. p. 1–6.

[10] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. ACM Commun. 2008;51:107–13.

[11] Gates A, Natkovich O, Chopra S, Kamath P, Narayanam S, Olston C, Reed B, Srinivasan S, Srivastava U. Building a high-level dataflow system on top of MapReduce: the Pig experience. Proc Very Large Data Bases. 2009;2(2):1414–25.

[12] Ghemawat S, Gobioff H, Leung S. The Google file system. In: Proceedings of ACM symposium on operating systems principles, Lake George, NY. 2003. p. 29–43.

[13] Han J, Kamber M. Data mining: concepts and techniques. Burlington: Morgan Kaufmann Publishers; 2012.

[14] Jiang H, Luan X, Dong X. Mining weighted negative association rules from infrequent itemsets based on multiple support. In: 2012 international conference on industrial control and electronics engineering. 2012. p. 89–92.

[15] Kishor P, Porika S. An efficient approach for mining positive and negative association rules from large transactional databases. In: 2016 international conference on inventive computation technologies (ICICT). 2016.

[16] Lucchese C, Orlando S, Perego R, Silvestri F. WebDocs: a real-life huge transactional dataset. In: FIMI '04, Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations, Brighton, UK. 2004.

[17] Li N, Zeng L, He Q. Parallel implementation of Apriori algorithm based on MapReduce. In: 2012 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing. 2012.

[18] Lin M-Y, Lee P-Y, Hsueh S-C. Apriori-based frequent itemset mining algorithms on MapReduce. In: ICUIMC '12 Proceedings of the 6th international conference on ubiquitous information management and communication. 2012.

[19] Lin X. MR-Apriori: association rules algorithm based on MapReduce, In: 2014 IEEE 5th international conference on software engineering and service science. 2014.

[20] Mahmood S, Shahbaz M, Guergachi A. Negative and positive association rules mining from text using frequent and infrequent itemsets. Sci World J. 2014;2014:1–11.

[21] Oweis NE, Fouad MM, Oweis SR, Owais SS, Snasel V. A novel Mapreduce lift association rule mining algorithm (MRLAR) for big data. Int J Adv Comput Sci Appl. 2016;7(3):151–7.

[22] Ramasubbareddy B, Govardhan A, Ramamohanreddy A. Mining positive and negative association rules. In: 5th international conference on computer science and education, Hefei, China, 2018. p. 1403–6.

[23] Riley E. Indirect association rule mining for crime data analysis. EWU Master's Thesis. 2015.

[24] Sahu AK, Kumar R, Rahim N. Mining negative association rules in distributed environment. In: 2015 International conference on computational intelligence and communication networks. 2015. p. 934–7.

[25] Savasere A, Omiecinski E, Navathe S. Mining for strong negative associations in a large database of customer transactions. In: Proc. of ICDE. 1998. p. 494–502.

[26] Singh S, Garg R, Mishra PK. Review of Apriori based algorithms on MapReduce framework. In: International conference on communication and computing, ICC 2014, Bangalore, India. 2014.

[27] Swesi IMAO, Bakar AA, Kadir ASA. Mining positive and negative association rules from interesting frequent and infrequent itemsets. In: 2012 9th international conference on fuzzy systems and knowledge discovery. 2012. p. 650–5.

[28] Teng W-G, Hsieh M-J, Chen M-S. On the mining of substitution rules for statistically dependent items. In: Proc. of ICDM. 2002. p. 442–9.

[29] Teng W-G, Hsieh M-J, Chen M-S. A statistical framework for mining substitution rules. Knowl Inf Syst. 2005;7(2):158–78.

[30] Thiruvady DR, Webb GI. Mining negative rules using GRD. In: Proc. of PAKDD. 2004. p. 161–5.

[31] White T. Hadoop: a definitive guide. Sebastopol: O'Reilly Publishers; 2012.

[32] Wu X, Zhang C, Zhang S. Efficient mining of both positive and negative association rules. ACM Trans Inf Syst. 2004;22(3):381–405.

[33] Yuan X, Buckles BP, Yuan Z, Zhang J. Mining negative association rules. In: Proc. of ISCC. 2002. p. 623–8.