IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 7, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Frequent Pattern Mining Using Record Filter Approach

D. N. Goswami[1], Anshu Chaturvedi[2] and C.S. Raghuvanshi[3]

[1] SOS in Comp. Appl., Jiwaji University
Gwalior, M.P. 474001, India

[2] Dept. of Comp. Appl., MITS College of Enggineering
Gwalior, M.P. 474001, India

[3] Dept. of Comp. Appl., GICTS College of Professional Education
Gwalior, M.P. 474001, India

Abstract

In today's emerging world, the role of data mining is increasing day by day with the new aspect of business. Data mining has been proved as a very basic tool in knowledge discovery and decision making process. Data mining technologies are very frequently used in a variety of applications. 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. Frequent patterns are the itemsets that are frequently visited in database transactions at least for the user defined number of times which is known as support threshold. Presently a number of algorithms have been proposed in literature to enhance the performance of Apriori Algorithm, for the purpose of determining the frequent pattern. The main issue for any algorithm is to reduce the processing time. Present paper proposes a new record filtering based approach which takes very less time for performing computations during mining process. Experiments have been performed on synthetic datasets and the results have been presented. The results show that proposed approach performs well in terms of execution time and ultimately enhances efficiency as compared to traditional Apriori approach.

Keywords: *Association Rule, Apriori, Frequent Patterns, Record Filtering*

1. Introduction

Data mining is the process of finding interesting trends or patterns in large datasets to steer decision about future activities. It is the analysis of dataset to find unsuspected relationship and to summarize the data in new ways which are both understandable and useful. Evolutionary progress in digital data acquisition and storage technology has resulted in huge and voluminous databases. Data is often noisy and incomplete, and therefore it is likely that many interesting patterns will be missed and reliability of detected patterns will be low. This is where, Knowledge

Discovery in Databases (KDD) and Data Mining (DM) helps to extract useful information from raw data. Frequent patterns are those that occur at least a user-given number of times (referred as minimum support threshold) in the dataset. 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. Frequent pattern mining is one of the most important and well researched techniques of data mining. The mining of association rules is one of the most popular research domain. 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. Such rules can be useful for decisions concerning product pricing, promotions, store layout and many others.

2. Problem

The problem of mining association rules is to generate all rules that have support and confidence greater than or equal to some user specified minimum support and minimum confidence threshold respectively. A formal statement of the association rule problem is given in [1], [9], [10], [11].

Let $I = \{ i_1, i_2, i_3, i_4 \ldots\ldots\ldots i_m \}$ be a set of m distinct literals called items, D is a set of transactions (variable length) over I. Each transaction contains a set of items $i_1, i_2, i_3, i_4 \ldots\ldots\ldots i_k \subseteq I$. Each transaction is associated with

an identifier, called TID. An association rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = 0$. Here X is called the antecedent and Y is called the consequent of the rule. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence α if among those transactions that contain X α% of them also contain Y. The rule $X \Rightarrow Y$ has support S in the transaction set D if S% of transactions in D contains $X \cup Y$. The selection of association rules is based on these two values (some additional constraints may also apply). These are two important measures of rule interestingness. They respectively reflect usefulness and certainty of a discovered rule. They can be described by the following equations:

Support $(X \Rightarrow Y)$ = Frequency $(X \cup Y)$ / /D/
Confidence $(X \Rightarrow Y)$ = Frequency $(X \cup Y)$ / Frequency (X)
where /D/ represents the total number of transactions (tuples) in D.

A frequent itemset is an itemset whose number of occurrences is above a minimum support threshold. An itemset of length k is called k-itemset and a frequent itemset of length k as k-frequent itemset. An association rule is considered strong if it satisfies a minimum support threshold and minimum confidence threshold.

3. Classical Frequent Pattern Mining Algorithms

There are different algorithms and approaches for frequent pattern discovery. Techniques to discover the association among data, such as AIS [1], SETM [2], and Apriori [1][3] have been widely studied.
Apriori is a great achievement in history of association rule mining, Apriori algorithm was first proposed by Agrawal et al. The AIS is just a straightforward approach that requires many passes over the database, generating many candidate itemsets and storing counters of each candidate while most of them turn out to be not frequent. Apriori is more efficient during the candidate generation process for two reasons, Apriori employs a different candidates generation method and a new pruning technique.
There are two processes to find out all the large itemsets from the database in Apriori algorithm. First the candidate itemsets are generated, then the database is scanned to check the actual support count of the corresponding itemsets. During the first scanning of the database the support count of each item is calculated and the large 1-itemsets are generated by pruning those itemsets, whose supports are below the predefined threshold. In each pass only those candidate itemsets that include the same specified number of items are generated and checked. The candidate k-itemsets are generated after the $k\text{-}1^{th}$ passes over the database by joining the frequent K-1 itemsets. All the candidate, k-itemsets are pruned by checking their sub (k-1) –itemsets, this k-itemsets candidate is pruned out because it has no hope to be frequent according to the apriori property. The Apriori property says that every sub (k-1) –itemsets of the frequent k-itemsets must be frequent.
An analysis of Apriori algorithm has let the authors to identify the following limitations-

I. The first issue in Apriori is that it generates a large number of candidate itemsets.
II. The second lacuna is that it takes a large number of database scans in order to discover frequent patterns.

4. The Apriori Algorithm

The Apriori algorithm [4],[5],[6],[7],[8] is also called the level-wise algorithm and was proposed by Agrawal and Srikanth in 1994. It is the most popular algorithm to find all the frequent sets which use the downward closure property. The advantage of the algorithm is that before reading the database at every level, it prunes many of the sets which are unlikely to be frequent sets by using the Apriori property, which states that all nonempty subsets of frequent sets must also be frequent. This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well.
Using the downward closure property and the Apriori property, this algorithm works as follows. The first pass of the algorithm counts the number of single item occurrences to determine the L_1 or single member frequent itemsets. Each subsequent pass, K, consists of two phases. First, the frequent itemsets L_{k-1} found in the (k-1)th pass are used to generate the candidate itemsets C_k, using the Apriori candidate generation algorithm. Next, the database is scanned and the support of the candidates in C_k is determined to ensure that C_k itemsets are frequent itemsets.

4.1 Steps of Algorithm

Initialize: k := 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin

C_k := gen_candidate_itemsets with the given L_{k-1}
prune(C_k)
for ***all transactions t*** $\in$ T do
increment the count of all candidates in C_K that are contained in t;
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;

4.2 Working Example of Apriori

To understand the functioning of classical Apriori algorithm, we consider a database of 15 transactions containing an item set I = {I1,I2,I3,I4,I5} of five items.

Table 1: Database (D)

TID	Items
T1	I1, I3, I5
T2	I1, I4
T3	I4, I5
T4	I2, I3, I4
T5	I1, I2, I3
T6	I2, I4, I5
T7	I2, I5
T8	I2, I3, I4, I5
T9	I4
T10	I2, I3, I4, I5
T11	I3, I4
T12	I1
T13	I2, I4, I5
T14	I4, I5
T15	I1, I2, I3, I4, I5

Before starting the Apriori we assume absolute support count of 3.

In the first step of classical Apriori we take the candidate set of one item and scan the database to count the support of each member of candidate set

Table2

Scan D to count the support of each candidate

Itemset	Sup. count
I1	5
I2	8
I3	7
I4	11
I5	9

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I1	5
I2	8
I3	7
I4	11
I5	9

Candidate set of 1 item **Frequent set of 1 item**

After determining the frequent set of 1 item, we generate the candidate set of 2 items by merging the frequent set of 1 item. After that we again scan the database D to count the support of each element of candidate set and generate the frequent set of 2 items by comparing support count with minimum support count.

Table3

Scan D to count the support of each Candidate

Itemset	Sup. count
I1, I2	2
I1, I3	3
I1, I4	2
I1, I5	2
I2, I3	5
I2, I4	6
I2, I5	6
I3, I4	5
I3, I5	4
I4, I5	7

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I1, I3	3
I2, I3	5
I2, I4	6
I2, I5	5
I3, I4	5
I3, I5	5
I4, I5	7

Candidate set of 2 items **Frequent set of 2 items**

Further we generate a candidate set of 3 items by using frequent 2 item sets and pruning technique. After that we again scan all the transactions in database D to count the support of each element of candidate set in order to get the frequent set by comparing them with the minimum support count.

Table4

Scan D to count the support of each candidate

Itemset	Sup. count
I2, I3, I4	4
I2, I3, I5	3
I2, I4, I5	5
I3, I4, I5	3

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I2, I3, I4	4
I2, I3, I5	3
I2, I4, I5	5
I3, I4, I5	3

Candidate set of 3 items **Frequent set of 3 items**

In the next step we generate candidate set of 4 items by using frequent 3 item sets and pruning technique and determine the support of candidate set by scanning all the transactions available in the database in order to get frequent set of 4 items.

Table5

Scan D to count the support of each candidate

Itemset	Sup. Count
I2, I3, I4, I5	3

Candidate set of 4 items

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I2, I3, I4, I5	3

Frequent set of 4 items

In this way classical Apriori discover all frequent item set by scanning all the transactions in each repetitive scan and thus takes a lot of time.

5. Proposed Record Filter Approach

The author has critically analyzed the apriori algorithm and observed that we have to count the support of itemsets many times during mining process. Since counting the occurrences of itemsets is a time-consuming process hence, the present paper proposes a novel approach for mining frequent patterns that takes less time as compared to Apriori algorithm. In case of Apriori algorithm when we count the support of candidate set of length k, we also check its occurrence in transaction whose length may be greater than, less than or equal to the k. But in the proposed approach support count of candidate sets only in the transaction records whose length is greater than or equal to the length of candidate set is checked, because candidate set of length k, can not exist in the transaction record of length k-1 , it may exist only in the transaction of length greater than or equal to k.

5.1 Steps of Proposed Algorithm

Initialize: k := 1, C_1 = all the 1- item sets;
read the database to count the support of C_1 to determine L_1.
L_1 := {frequent 1- item sets};
k:=2; //k represents the pass number//
while ($L_{k-1} \neq \varnothing$) do
begin
C_k := gen_candidate_itemsets with the given L_{k-1}
prune(C_k)
for ***all transactions t whose length is greater than or equal to k*** $\in$ T do
increment the count of all candidates in C_K that are contained in t;
L_k := All candidates in C_k with minimum support ;
k := k + 1;
end
Answer := $\cup_k L_k$;

5.2 Working Example

To illustrate the working of proposed approach, we use the above mentioned transactional database D Shown in Table1. The transactional database (Table 1) contains 15 transactions with an item set I = {I1, I2, I3, I4, I5} of five items and we consider the same minimum support count of 3.
Initially we consider the candidate set of size one and determine the support count as shown below

Table6

Scan D to count the support of each candidate

Itemset	Sup. count
I1	5
I2	8
I3	7
I4	11
I5	9

Candidate set of 1 item

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I1	5
I2	8
I3	7
I4	11
I5	9

Frequent set of 1 item

Next we generate candidate set of size-two and determine the support count only in the transactions which contain at least two items. Hence the transaction T12, that contains a single item will not be considered during this step.

Table7

Scan D to count the support of each candidate

Itemset	Sup. count
I1, I2	2
I1, I3	3
I1, I4	2
I1, I5	2
I2, I3	5
I2, I4	6
I2, I5	6
I3, I4	5
I3, I5	4
I4, I5	7

Candidate set of 2 items

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I1, I3	3
I2, I3	5
I2, I4	6
I2, I5	5
I3, I4	5
I3, I5	5
I4, I5	7

Frequent set of 2 items

Further we generate a candidate set of size-3 and determine the support count by considering only those transactions which contain at least 3 items. Hence the transaction containing only one or two items will not be

scanned throughout the database (i.e. T2, T3, T7, T9, T11, T12, T14)

Table8

Scan D to count the support of each candidate

Itemset	Sup. count
I2, I3, I4	4
I2, I3, I5	3
I2, I4, I5	5
I3, I4, I5	3

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I2, I3, I4	4
I2, I3, I5	3
I2, I4, I5	5
I3, I4, I5	3

Candidate set of 3 items **Frequent set of 3 items**

In next step, we generate the candidate set of size-4 and determine the support count by considering only those transactions which contains at least 4 items. In this process we ignore those transactions that contain 1, 2 or 3 items.

Table9

Scan D to count the support of each candidate

Itemset	Sup. Count
I2, I3, I4, I5	3

Compare candidate support with minimum support count to get frequent set

Itemset	Sup. count
I2, I3, I4, I5	3

Candidate set of 4 items **Frequent set of 4 items**

In this way proposed approach discovers the frequent itemsets of all size by saving considerable amount of processing time.

6. Performance Evaluation

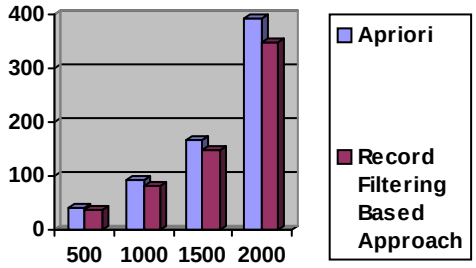
To explore the performance of proposed algorithm, synthetic dataset is used and all the experiments are performed on Pentium IV 2.93 GHz PC machine with 512 MB RAM, running Microsoft Windows 2000. This algorithm is implemented in Java and used hash-set to calculate the candidate itemsets. All the runtime reports include both CPU time and I/O time.

For the comparative study of classical Apriori and proposed approach, we have taken a database of 5000 transactions containing 50 unique items.

During this analytical process we have considered 1000 transactions to generate the frequent pattern with the support count of 10% and the process is repeated by increasing the transaction gradually. Table below (Table 10) shows the execution time corresponding to different transaction sizes.

Table 10: Execution time in seconds for different transaction size

Transaction Size	Execution time (seconds) Apriori	Execution time (seconds) Record Filtering Based Approach
500	42	37
1000	92	82
1500	167	149
2000	392	348



Finally as a result of critical analysis, we can see that proposed approach (Record filtering based approach) takes only 90% time in comparison to classical Apriori. Hence, we save approx 10 % time in the of proposed approach.

7. Conclusion

Present paper proposes a new record filter based algorithm which is a variation of the Apriori algorithm and performs fewer database scans than Apriori and utilizes only transaction of specific sizes for the generation of frequent itemsets. As observed by many researchers counting the occurrences of itemsets is a time consuming activity, this paper introduces a new strategy of considering only those transactions whose length is greater than or equal to the length of candidate set is checked, because candidate set of length k , can not exist in the transaction record of length k-1 , it may exist only in the transaction of length greater than or equal to k. Due to this, proposed approach takes very less time for performing computations during mining process. Experiments have been performed on synthetic datasets and the results have been presented. The results show that proposed approach performs well in terms of execution time and ultimately enhances efficiency as compared to traditional Apriori approach.

References

[1]Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.
[2] Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 487-499.
[3] Agarwal, R. Aggarwal, C. and Prasad V., A tree projection algorithm for generation of frequent itemsets. In J. Parallel and Distributed Computing, 2000.
[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.
[5] A. Amir, R. Feldman, and R. Kashi. A new and versatile method for association generation. Information Systems, 2:333–347, 1997.
[6] R.J. Bayardo, Jr. Efficiently mining long patterns from databases. In L.M. Haas and A. Tiwary, editors, Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, volume 27(2) of SIGMOD Record, pages 85–93. ACM Press, 1998.
[7] S. Parthasarathy, M. J. Zaki, M. Ogihara, S. Dwarkadas; Incremental and interactive sequence mining; Int'l Conf. on Information and Knowledge Management; 1999.
[8] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, Umeshwar Dayal; Multi-Dimensional Sequential Pattern Mining; Int'l Conf. on Information and Knowledge Management; 2001.
[9] Richard Relue, Xindong Wu, Hao Huang; Efficient runtime generation of association rules; Int'l Conf. on Information and Knowledge Management; October 2001.
[10] Assaf Schuster, Ran Wolff, and Dan Trock; Distributed Algorithm for Mining Association Rules; IEEE Int'l Conf. on Data Mining; November 2003.
[11] Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu; Resource-Aware Mining with Variable Granularities in Data Streams; SIAM Int'l Conf. on Data Mining; 2004

Dr. D.N.Goswami

***D.N.* Goswami** is Professor and Head in the School of Studies in Computer Science, Jiwaji University, Gwalior. He has done Master in Computer Applications and Ph.D. in Computer Science from Jiwaji University. His Research interests includes Software Quality and Reliability analysis, Adhoc Networks ,Relational Data base Management Systems and Data Mining. He has guided Ph.D. theses in Computer Science and Applications.

Dr.Anshu Chaturvedi

Anshu Chaturvedi Currently working as Lecturer in Department of Computer Applications at Madhav Institute of Technology and Sciences, Gwalior. She has obtained her Ph. D. in 2009. Her research interests include Adhoc Networks, Data Mining. Operating Systems and Security. She is a life member of Computer Society of India She has seven years of experience in the academic field. She has also won Young Scientist Award in 2009.

Mr. C.S.Raghuvanshi

C.S.Raghuvanshi is doing Ph.D in computer science from jiwaji university Gwalior under the guidance of Dr. D.N.Goswami and also working as lecturer in GICTS college of professional education Gwalior. He has done M.sc (IT) from Jiwaji University Gwalior and M.Tech(IT) From AAI Deemed University Allahabad. His Research interests includes Data Mining, Adhoc Network,Software Engineering.