Enhancing Algorithm for Dynamic Data Mining Association Rules

Hebah H.O. Nasereddin

Abstract— **The paper; discuss the concept of dynamically estimating and building the model process using association rule model, scanning itemsets with their counts and design a novel, efficient, dynamic mining algorithm. Enhancing (ARBSI) will not require rescanning the original database after collecting the data, even if a number of transactions have been newly inserted, and this will work regardless of the support value used and regardless of the confidence value used. Enhancing (ARBSI) can work in both dynamic and conventional form, this is more efficient and will reduce the time when its performance is compared with the previous techniques used, in such away as: It will know the number of items used from the last process after normalization sub-process which will reduce the time for scanning each transaction, It will know the types of modification insert, update, and/or delete, In case there is an new inserted record Enhancing (ARBSI) can translate this record to numeric using dummy table for attribute without duplicate (especially for nominal values)**

Index Terms— **Data mining, stream data mining, insert, update, delete.**

I. INTRODUCTION

Data mining is the task of discovering interesting and hidden patterns from large amounts of data where the data can be stored in databases, data warehouses, OLAP (on line analytical process) or other repository information **[1].** It is also defined as knowledge discovery in databases (KDD) [2]. [3] Data mining involves an integration of techniques from multiple disciplines such as database technology, statistics, machine learning, neural networks, information retrieval, etc. Data mining process is a step in Knowledge Discovery Process consisting of methods that produce useful patterns or models from the data [3]. In some cases when the problem is known, correct data is available as well, and there is an attempts to find the models or tools which will be used, some problems might occur because of duplicate, missing, incorrect, outliers values and sometimes a need to make some statistical methods might arise as well. [4] Explained the KDD procedures, in a way to help us focus on data mining process. It includes five processes: 1) Defining the data mining problem, 2) Collecting the data mining data, 3) Detecting and correcting the data, 4) Estimating and building the model, 5) Model description, and validaion.while [5] discuss Estimating and Building the Model as: This process includes four parts: 1) select data mining task, 2) select data mining method, 3) select suitable algorithm 4) extract knowledge. Many Data mining techniques have been developed over the last 50 years. Depending on the type of databases processed, these mining approaches may be classified as working on transaction databases, relational databases, and multimedia databases, among others. On the other hand, depending on the classes of knowledge consequent, the mining approaches may be classified as finding association rules, classification rules, and clustering rules [6], among others. From past research, it is clear that association rules in transaction databases are the most common in data mining [7]

In previous research, mining association rules algorithms form transactions were proposed, most of which were executed by scanning single items first, then scanning with two items, and this was repeated, continuously adding one more item each time, until some criteria were met. These algorithms are designed to work with static database. However In real-world applications, new transactions are usually inserted into databases, and designing a mining algorithm that can maintain association rules as a database grows is thus critically important. One application of data mining is to induce association rules from transaction data, such that the presence of certain items in a transaction will imply the presence of certain other items. To achieve this purpose, Agrawal and his co-workers proposed several mining algorithms based on the concept of large itemsets to find association rules in transaction data [8], [9][10]. They divided the mining process into two phases. In the first phase, candidate itemsets were generated and counted by scanning the transaction data. If the count of an itemset appearing in the transactions was larger than a pre-defined threshold value (minimum support), the itemset was considered as a large itemset. Itemsets containing only one item were processed first. Large itemsets containing only single items were then combined to form candidate itemsets containing two items [11]. This process was repeated until all the large itemsets have been found. In the second phase, association rules were induced from the large itemsets found in the first phase. All possible association combinations for each large itemset were formed, and those with calculated confidence values larger than a predefined threshold (minimum confidence) were given out as association rules. This paper is closely related to dynamic data mining, more specifically, to Association Rules. The paper is divided into five sections. Section 2 discusses, Dynamically Estimating and Building the Model Process Using Association Rules. Section 3 presents Dynamic Association Rules Based on Scanning the Itemsets Enhancing (ARBSI).while Section 4 presents Illustrative Examples, at the end conclusion.

II. DYNAMICALLY ESTIMATING AND BUILDING THE MODEL PROCESS USING ASSOCIATION RULES

The original association rules may become invalid, when new transactions are added to databases, or new valid rules

Manuscript received Feb 18, 2017
Hebah H.O. Nasereddin, Middle East University (MEU), Jordan

may appear in the resulting updated databases [12][13][14][15]. In these cases, mining algorithms must re-process the entire updated databases to find final association rules. This will cause two problems: Algorithms do not, however, use previously mined information and require rescanning the database which cost nearly twice the computational time to mine the databases. If new transactions appear often and the original databases are large, these algorithms are thus inefficient in maintaining association rules.[16] .Transactions databases grow over time in real-world applications, which means re-evaluated association rules mined because new association rules may be generated and old association rules may become invalid when the new entire databases are considered. Apriori [8] and DHP [7] solved this problem by re-processing entire new databases when new transactions are inserted into the original databases. These algorithms have two disadvantages: First, increasing the computation time for each insert / update and/or delete transaction. If the original database is large, much computation time is wasted in maintaining association rules whenever update transactions are generated. Second, information previously mined became meaningless [17]. Nassereddin [17] proposed several mining algorithms based on the concept of estimating and building the model process using association rule model, scanning itemsets with their counts and design a novel, efficient, static mining algorithm. (ARBSI) will not require rescanning the original database after collecting the data, even if a number of transactions have been newly inserted, and this will work regardless of the support value used and regardless of the confidence value used. (ARBSI) can work in conventional form, this is more efficient and will reduce the time when its performance is compared with the previous techniques used, in such away as: It will know the number of items used from the last process after normalization sub-process which will reduce the time for scanning each transaction, It will know the types of modification insert, update, and/or delete, In case there is an new inserted record (ARBSI) can translate this record to numeric using dummy table for attribute without duplicate (especially for nominal values) .

The importance of dynamic estimating and building process becomes essential due to the time consumption problem. Many researchers tried to solve these problems. Such as The Fast Update Algorithm (FUP) [12], Pre-large itemsetes [18] and Record Deletion Based on the Pre-Large itemsetes [19] which will be discussed in next Section s. They provided solution for the insert operation but failed to do the same for the other two cases namely update and delete.

A. The Fast Update Algorithm (FUP)

Cheung et al. proposed the FUP algorithm to incrementally maintain association rules when new transactions are inserted [12] [13]. Using FUP, large itemsets with their counts in preceding runs are recorded for later use in maintenance. When new transactions are added, FUP first scans them to generate candidate 1-itemsets (for the new transactions), and then compares these itemsets with the previous ones. FUP partitions candidate 1-itemsets into two parts according to whether they are large for the original database. If a candidate 1-itemset from the newly inserted transactions is also among the large 1-itemsets from the original database, its new total count for the entire updated database can easily be calculated from its current count and previous count since all previous large itemsets with their counts are kept by FUP. Whether an original large itemset is still large after new transactions are inserted is determined from its support ratio as its total count over the total number of transactions.

By contrast, if a candidate 1-itemset from the newly inserted transactions does not exist among the large 1-itemsets in the original database, one of two possibilities arises. If this candidate 1-itemset is not large for the new transactions, then it cannot be large for the entire updated database, which means no action is necessary. If this candidate 1-itemset is large for the new transactions but not among the original large 1-itemsets, the original database must be re-scanned to determine whether the itemset is actually large for the entire updated database.

Using the processing plans mentioned above, FUP is thus able to find all large 1-itemsets for the entire updated database. After that, candidate 2-itemsets from the newly inserted transactions are formed and the same procedure is used to find all large 2-itemsets. This procedure is repeated until all large itemsets have been found. On the other hand, although the FUP algorithm focuses on the newly inserted transactions and thus saves much processing time by incrementally maintaining rules, it must still scan the original database to handle, when a candidate itemsets is large for new transactions but is not recorded in large itemsets already mined from the original database. This situation may occur frequently, especially when the number of new transactions is small. In an extreme situation, if only one new transaction is added each time, then all items in this transaction are large since their support ratios are 100% for the new transaction. In additional to the problem of being not flexible (for example when the support value changed that means the FUP technique will be meaningless). Any way the technique start after static association rules mining (after scanning and finding the large itemsets and dependent on the support value from the beginning.

B. Pre-large Itemsets

In the Pre-large algorithm [18], the large itemsets with their counts in preceding runs are recorded for later use in maintenance. As new transactions are added, the proposed algorithm first scans them to generate candidate 1-itemsets (only for these transactions), and then compares these itemsets with the previously retained large and pre-large 1-itemsets. It partitions candidate 1-itemsets into three parts according to whether they are large or pre-large for the original database. If a candidate 1-itemset from the newly inserted transactions is also among the large or pre-large 1-itemsets from the original database, its new total count for the entire updated database can easily be calculated from its current count and previous count since all previous large and pre-large itemsets with their counts have been retained. Whether an originally large or pre-large itemset is still large or pre-large after new transactions have been inserted is determined from its new support ratio, as derived from its total count over the total number of transactions.

On the contrary, if a candidate 1-itemset from the newly inserted transactions does not exist among the large or pre-large 1-itemsets in the original database, then it is absolutely not large for the entire updated database as long as

the number of newly inserted transactions is within the safety threshold. In this situation, no action is needed. When transactions are incrementally added and the total number of new transactions exceeds the safety threshold, the original database is re-scanned to find new pre-large itemsets in a way similar to that used by the FUP algorithm.

On the other hand, although the Pre-large Itemsets algorithm focuses on the newly inserted transactions and thus saves much processing time by incrementally maintaining rules, it must still scan the original database to handle when transactions are incrementally added and the total number of new transactions exceeds the safety threshold; the original database is re-scanned to find new pre-large itemsets in a way similar to that used by the FUP algorithm.

Another disadvantage is if the number of newly inserted transactions is less than the safety threshold, no action is done in this case, this situation may occur frequently, especially when the number of new transactions is small. In an extreme situation, if only one new transaction is added each time, then this transaction will not be maintained until safety threshold are reach. The algorithm calculates the safety threshold by the equation:

$$t \leq \frac{(S_u - S_l)d}{1 - S_u} \qquad (1)$$

Where
d : the number of transactions in D;
t : the number of transactions in T;
S_l : the lower support threshold for pre-large itemsets;
S_u : the upper support threshold for large itemsets, Su >Sl;

This algorithm depends on the values of S_u and S_l which will constrain the dynamic run, whether they are large or small; (the distance between them large or small). Another point is that the value of S_l depend on what value (This value may change the results of dynamic run).In additional to the problem of being not flexible (for example when the support value changed, is means the Pre-large Itemsets technique will be meaningless). Also the technique start after static association rules mining after scanning and finding the large itemsets and dependent on the support value from the beginning.

C. Record Deletion Based on the Pre-Large

Hong et al [19] proposed an algorithm that maintains a generalized association rules based on the concept of pre-large itemsets [18] for deleted data. The concept of pre-large itemsets is used to reduce the need for rescanning original databases (does not need to rescan the original database until a number of transactions have been deleted) to save maintenance costs and time.

On the other hand, although the Record Deletion Itemsets algorithm focuses on the newly deleted transactions and thus saves much processing time by maintaining rules, it must still scan the original database to handle when transactions are incrementally deleted and the total number of delete transactions exceeds the safety threshold; the original database is re-scanned to find new pre-large itemsets in a way similar to that used by the FUP algorithm.

Another disadvantage is if the number of newly deleted transactions is less than the safety threshold, no action is done in this case, this situation may occur frequently, especially when the number of deleted transactions is small. In an extreme situation, if only one new transaction is deleted each time, then this transaction will not be maintained until safety threshold are reached. The algorithm calculates the safety threshold using equation 1; this technique depends on the values of Su and Sl which will constrain for the dynamic run, whether they are large or small. Another point is that the value of S_l depend on what value. In additional to the problem of being not flexible (for example when the support value changed that means the Pre-large Itemsets technique will be meaningless). Also the technique start after static association rules mining (after scanning and finding the large itemsets and dependent on the support value from the beginning.

III. DYNAMIC ASSOCIATION RULES BASED ON SCANNING THE ITEMSETS ENHANCING (ARBSI)

Although the FUP algorithm [12] and Pre-large Itemsets algorithm [18] focused on the newly inserted transactions and thus save much processing time by incrementally maintaining rules, both of them must still scan the original database to handle cases of newly inserted transactions, both of them solve the insertion case but ignore the update and delete cases. Another disadvantage is if the number of newly inserted transactions [18] is less than the safety threshold, no action is done in this case, this situation may occur frequently, especially when the number of new transactions is small. In additional; to the problem of being not flexible, for example when the support value changes that means both techniques will be meaningless. Any way their techniques start after static association rule mining, after scanning and finding the large itemsets and it is dependent on the support value from the beginning.

Enhancing (ARBSI) presents solutions to the disadvantages of the above techniques. It deals with:

1- The new transactions (insert/ update/delete).
2- The support value is flexible it depends on the user as he/she chooses this value before and/or during running the data mining process.
3- It only scans the original database once to find all itemsets with their appropriate counts.

Also Enhancing (ARBSI) can work either in this dynamic process from scratch, which is more efficient than previous techniques such as: it knows the number of itemsetes from the last process after normalization sub-process which will reduce the time for scanning each transaction, it knows the types of modification insert, update, and/or delete, Enhancing (ARBSI) after generates a mathematical summation value for each transaction [16]. If a new transaction is to take place, a new summation value will be generated based on the new status, which will also be reflected in a dedicated file stored in a predefined local database, which will be used to compare with itemsets selected in the initial scan.

A. Definition of the Proposed algorithm Enhancing (ARBSI)

The proposed algorithm is to induce association rules from transaction data, such that the presence of certain items in a transaction will imply the presence of certain other items by dividing the mining process into two phases. In the first phase, all itemsets will be generated and counted by scanning of the original database without any consideration to the threshold value (minimum support) as in [8] [9] [10]. Number of all itemsets will be equal ($2^{\#items} - 1$). Number of items will be easy to calculate when we run the last normalization sub-process in previous pre-processing process. This process will be repeated until all the itemsets and there counts have been found.

In the second phase, association rules are induced from the large itemsets found in the first phase, after setting the sets that contain the count of each set and the total number of the transactions, we can activate the association rule any time as follows:

- Input the support values (changeable).
- Divide every set by the total number of transactions (Support {set} = count {set}/ count of transactions).
- Find the sets where Support {set} >= support value.
- Calculate the confidence.

All possible association combinations for each large itemset are formed, and those with calculated confidence values larger than a predefined threshold (minimum confidence) are given out as association rules.

Note:

- Itemsets with their counts in preceding runs are recorded for later use in maintenance.
- For the original database is scanned once only at the beginning and the counts are keep for any modifications in later stages.
- No support value will be added until running data mining, it will be inserted manually.

In the case were a new transaction is taken place, a new summation value is calculated for this transaction. This is stored in a predefined location (file). Scan the new transaction; calculate the number of all sets that equal ($2^{\#\text{ of new items}} - 1$).

Once the numbers of itemsets are calculated the following may take place based on the individual new transaction.

1- **Input a new transaction**:

If the transaction contains the same items that exist in the original set, add (+1) to each set and (+1) to the total number of transactions. If the transaction contains a new item that does not exist in the original set, break the transaction into $\{2^{\#\text{ of new items}} - 1\}$ and add this new sets to the original sets, add (+1) to each old set, and (+1) to each new set and (+1) to the total number of transactions.

2- **Delete an exist transaction:**

There is no interpretations, cause the transaction and the sets already exists, so add (-1) to each set and (-1) to the total number of transactions.

3- Update an existing transaction:

In case of update an existing transaction all we have to do is delete an exist transaction (Delete exist transaction step), and then input a new transaction (Input a new transaction step). Note here we can continue as above; we have all the updated sets and there counts and the total number of updated transactions.

[16] Proposed an algorithm to generate a mathematical summation for each transaction. Based on these summation values the exact transaction in the local database that have been modified and needs to be replaced can be identified. In other words, if there are any modification affecting one or a number of transactions, it simply selects the transactions summation for the particular transaction; delete the old transaction then insert the new updated one, and make the changes needed related to the transaction with the modified summation value, this will result in the replacement of the transactions by their changed value from the source DB

B. Presentation of the Enhancing (ARBSI)

The Enhancing (ARBSI) is presented; the notations used in the algorithm are:

D: the original database;
T: the set of new transactions;
d: the number of transactions in D;
t: the number of transactions in T;
S: the support threshold;
C_k *: the set of all candidate k-itemsets from D;*
#items: # of items from normalization sub process;
#new items: the number of updated items;

The Enhancing (ARBSI) steps are explained as follows:

INPUT: A support threshold *S*, is a set of transaction in D consisting of (*d*) transactions, and a set of *t* new transactions, and #items.

OUTPUT: A set of final association rules for the D and T.

STEP 1: Calculate the number of all sets equal $2^{\#items} - 1$.
STEP 2*:* Find all k-itemsets C_k and their counts from the transactions.
STEP 3: Input S.
STEP 4: divide every set by the total number of d.
Support {set} = count {set}/ count of d.
STEP 5: Set the sets where Support {set} >= S. All possible association combinations for each large itemset are formed.
STEP 6: Calculate the confidence, those with calculated confidence values larger than a predefined

threshold (minimum confidence) are given out as association rules.

STEP 7: If T is not empty (there is a new transaction): from the previous technique [16] we can find:

1. Wither it's an insert, delete and/or update case.
2. The item-summation, are recalculated and stored along with modification time.

Sup step 7.1: If Input is a new transaction:

1. Calculate the item-summation value.
2. Calculate the # of all sets equal ($2^{\#\text{new items}} - 1$).
3. Scans the sets to generate sets itemsets.
4. If the transaction contains some of the items that exist in the original sets, add (+1) to each set and (+1) to the total number of transactions.
5. If the transaction contains a new item that doesn't exist in the original set, break the transaction into $\{2^{\#\text{new items}}\}$ and add these sets to the original set. Addition of (+1) to each new set and (+1) to the total number of transactions.

Sup step 7.2: If deleting an exist transaction:

1- Select the transaction from the old item-summation
2- Calculate the number of all sets equal ($2^{\#\text{new items}} - 1$).
3- Scans the sets to generate sets itemsets.
4- Break the transaction into its sets and add (-1) to each set and (-1) to the total number of transactions.

Sup step 7.3: If updating an exist transaction:

1. Select the transaction from the old item-summation.
2. Calculate the item-summation for the new modified transaction.
3. Calculate the number of all sets equal ($2^{\#\text{new items}} - 1$) for the old transaction, scans the sets to generate itemsets, break the transaction into its sets and add (-1) to each set and (-1) to the total number of transactions.
4. Calculate the number of all sets equal ($2^{\#\text{new items}} - 1$) for the modified transaction, scans the sets to generate itemsets, if the transaction contains some of the items that exist in the original sets, add (+1) to each set and (+1) to the total number of transactions, if the transaction contains a new item that doesn't exist in the original set, break the transaction into $\{2^{\#\text{new items}}\}$ and add these sets to the original set. Addition of (+1) to each new set is necessary and (+1) to the total number of transactions.

End

The Enhancing (ARBSI) can thus find all large 1-itemsets for the entire updated database. After that, candidate 2-itemsets from the newly inserted transactions are formed and the same procedure is used to find all large 2-itemsets. This procedure is repeated until all large itemsets have been found.

IV. Illustrative Examples

In this Section, an example is given to illustrate Enhancing (ARBSI) .Assume the initial data set includes 8 transactions, which are as shown in table 1. Note that TID number (200 and 800) and TID number (400,500) are having the same items which mean the same item-summation. From the previous sub process (normalization) we know that the number of items is (5) which mean number of sets will be = ($2^{\#\text{ of items}} - 1$), and equal ($2^5 - 1$) = 31. The minimum support threshold *S* is 50%. All itemsets were generated and counted by scanning the original database (just once) without the consideration of the threshold value (minimum support), the sets of itemsets and there counts. , using a conventional mining algorithm such as the Apriori algorithm, all large itemsets with counts larger than or equal to 4; (8∗50% = 4) are found, Since the user specified minimum confidence is 80%, the final association rules are shown in Table 2. More details in [17].

Table 1: An original database with *TID* and *Items*

TID	Items	Item-summation
100	ACD	I
200	BCE	I I
300	ABCE	V
400	ABE	I I I
500	ABE	I I I
600	ACD	I
700	BCDE	X I
800	BCE	I I

Table 2: The final association rules for this example

Rule	Confidence
IF B,C, Then E	Count(B,C,E)/Count(B,C)= 1
IF C,E, Then B	Count(B,C,E)/Count(C,E)= 1
IF B, Then E	Count(B, E)/Count(B)= 1
IF E, Then B	Count(B,E)/Count(E)= 1

A. Illustrative Example for Insertion Case

Assuming that there are **two new transactions** as shown in Table 3 these are inserted after the initial data set is processed, the new transactions are stored in file table with 3 added columns for; the item-summation, type of modification and time of modification.

Table 3: Two new transactions

New transactions				
TID	Items	Item-summation	Type-mod	Time of modification
900	ABCD	X I I	insert	Day/hour
1000	DEF	V I I	insert	

Enhancing (ARBSI) is proceeds as follows: For transaction number 900, 1000; calculate their new summation value, calculate the number of all sets equal ($2^{\#\text{ of new items}} - 1$), and scans them to generate candidate itemsets (only for these new transactions and according to their new summation number), add (+1) to the candidate itemsets count, add (+1) to the total number of transactions. Note: in this transaction 1000 the item F is new in this case no need to calculate the number of all sets equal ($2^{6} - 1$), = 63 which will consume C.P.U time, all we need to do is add only 4 sets {F, DF, FE, DEF}, instead of 63 set it will be 35 set. The rest of the transactions still have the same summation value calculated at the start of the process. Calculate their new summation value, calculate the number of all sets equal ($2^{\#\text{ of new items}} - 1$), = 15 and scans them to generate candidate itemsets (only for these new transactions), add (+1) to the candidate itemsets count, add (+1) to the total number of transactions. Table 4, 5 show the itemsets and there counts.

Table 4: The new sets and there counts for transaction 900

Items	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD
Add	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
count	6	7	7	4	4	4	3	5	2	4	2	1	3	2	1

Table 5: The new sets and there counts for transaction 1000

Items	D	E	F	DE	DF	EF	DEF
Add	1	1	1	1	1	1	1
count	5	7	1	2	1	1	1

The minimum support threshold *s* is set at 50%; all large itemsets with counts larger than or equal to 5; ((8+2) ∗50% =5) are found, No large 3-itemsets were found in this example. Since the user specified minimum confidence is 80%, the final association rules are shown in Table 6.

Table 6: The final association rules for this example

Rule	Confidence
IF B, Then E	Count(B, E)/Count(B)=6/7
IF E, Then B	Count(B,E)/Count(E)=6/7

As a result of the insert of two transactions: the final association rules for this example (10 transaction, support = 50% and confidence =80% value, two rules) is different from the final association rules (8 transaction, support = 50% and confidence =80% value, four rules). Note: after adding the new transactions in the original database:

1. No need for rescanning the original database scanning.
2. For each new transaction the new calculated item-summation will be saved.
3. The modified table will be empty (it contain only the new added transaction which will cleared after the algorithm deals the transactions).
4. The new normalized transactions will be saved in table2

B. Example for Changing the Minimum Support Threshold Case

Assume after this results, suppose the minimum support threshold is changed and set at 40%, which mean we have the same sets of itemsets and there counts in the original database shown in Table 7

Table 7: All large itemsets from an original database with *s*=40%

Large itemsets					
1 item	Count	2 items	Count	3 items	Count
A	6	AB	4	BCE	4
B	7	AC	4		
C	7	BC	5		
D	5	BE	6		
E	7	CE	4		
		CD	4		

{B, C, E} can be found to be a large 3-itemset. Next, the large itemsets are used to generate association rules. Since the user specified minimum confidence is 80%, the final association rules are shown in Table 8.

Table 8: The final association rules for this example

Rule	Confidence
IF B,C, Then E	Count(B,C,E)/Count(B,C)=4/5
IF C,E, Then B	Count(B,C,E)/Count(C,E)= 4/4
IF B, Then C	Count(B,C)/Count(B)=6/7
IF B, Then E	Count(B, E)/Count(B)=6/7
IF E, Then B	Count(B,E)/Count(E)=6/7
IF D, Then C	Count(C,D)/Count(D)=4/5

As a result for changing the minimum support threshold to 40%: the final association rules for this example (10 transaction, support = 40% and confidence =80% value, six rules) shown in table 8 are different from the final association rules (10 transaction, support = 50% and confidence =80% value, two rules) .Note: after changing the minimum support threshold no need for rescanning the original database in order to find the association rule.

C. Illustrative Example for Deletion Case

Assume the transaction 400 shown in Table 1 is deleted after the initial data set is processed.

Table 9: The delete transaction

New transactions				
TID	Items	Item-summation	Type-mod	Time of modification
400	ABE	I I I	delete	Day/hour

Enhancing (ARBSI) algorithm proceeds as follows: the new delete transaction is stored in file table with 3 added columns

for; the item-summation (the item-summation for delete existing transaction is already calculated, the key for delete the transaction), type of modification and time of modification. Calculate the number of all sets equal ($2^{\#\text{ of delete items}} - 1$), = 7 and scans them to generate candidate itemsets (only for the delete transaction), add (-1) to the candidate itemsets count, add (-1) to the total number of transactions, the new sets of itemsets and their counts are shown in Table 10.

Table 10: The new sets and there counts

Items	A	B	E	AB	AE	BE	ABE
subtract	1	1	1	1	1	1	1
count	5	6	6	3	2	5	2

The minimum support threshold *s* is set at 50%; all large itemsets with counts larger than or equal to 4.5 ((10-1) ∗50%) are found, as shown in Table 21. No large 3-itemsets were found in this example. Next, the large itemsets are used to generate association rules. Since the user specified minimum confidence is 80%, the final association rules are shown in Table 11.

Table 11: The final association rules for this example

Rule	Confidence
IF B, Then C	Count(B,C)/Count(B)= 5/6
IF B, Then E	Count(B, E)/Count(B)=5/6
IF E, Then B	Count(B,E)/Count(E)=5/6

As a result of delete the transactions: the final association rules for this example (9 transaction, support = 50% and confidence =80% value, three rules) shown in table 11 is different from the final association rules (8 transaction, support = 50% and confidence =80% value, four rules)

Note: after deleting the transaction in the original database:

1. No need for rescanning the original database scanning.
2. For each new delete transaction the calculated item-summation will be the key for delete the transaction from table2
3. The modified table will be empty (it contain only the new deleted transaction which will cleared after the algorithm deals the transactions).
4. The deleted normalized transaction will be removed from table2 .

D. Illustrative Example for Update Case

Assume the transactions 500 shown in Table 12 is updated after the initial data set is processed, the last sets and there counts in the original database are shown in table 13, 14

Table 12: Update new transactions

New transactions			
TID	Items	Item-summation	Type-mod
500	ABE	I I I	delete
500	ABF		insert

Enhancing (ARBSI) for update case proceeds as follows: first it is delete case for the old transaction then it will be insert case for the modify transaction.

1. Select the old modified transaction by the item-summation.
2. Calculate the number of all sets equal ($2^3 - 1$), = 7 and scans them to generate candidate itemsets (only for the old modified transaction), the new sets of itemsets and there counts are shown in Table 12.
3. For each old modified transaction the calculated item-summation will be the key for delete the transaction from table2 [16]

Table 13: The new sets and there counts

Items	A	B	E	AB	AE	BE	ABE
subtract	1	1	1	1	1	1	1
count	4	5	5	2	1	4	1

4. Calculate the item-summation for the new modified transaction.
5. Calculate the number of all sets equal ($2^3 - 1$) = 7 for the modify transaction, scans them to generate candidate itemsets (only for these new transactions), add (+1) to each set and (+1) to the total number of transactions, the new sets of itemsets and there counts are shown in Table 14.

Table 14: The new sets and there counts

Items	A	B	F	AB	AF	BF	ABF
add	1	1	1	1	1	1	1
count	5	6	1	3	1	1	1

Note here the sets {F, AF, BF, ABF} are added to the original database. The minimum support threshold *s* is set at 50%; all large itemsets with counts larger than or equal to 4.5 ((9-1+1) ∗50%). No large 3-itemsets were found in this example. Next, the large itemsets are used to generate association rules. According to the condition probability, . Since the user specified minimum confidence is 80%, the final association rules are shown in Table 15

Table 15: The final association rules for this example

Rule	Confidence
IF B, Then C	Count(B,C)/Count(B)= 5/6

As a result of update the transactions: the final association rules for this example (9 transaction, support = 50% and confidence =80% value, one rules) shown in table 5.38 is different from the final association rules (8 transaction, support = 50% and confidence =80% value, four rules) shown in table 5.12. Note: after updating the transaction in the original database:

1. No need for rescanning the original database scanning.
2. For each old modified transaction the calculated item-summation will be the key for delete the old transaction from table2 [16].

3. The modify table will be empty (it contain only the new updated transaction which will cleared after the algorithm deals the transactions).
4. The old normalized transaction will be removed from table2.
5. For each new modified transaction the new calculated item-summation will be saved with transaction in the original database.
6. The new normalized transactions will be saved in table2.

As a conclusion for the above examples:

1. For the first 8 transaction (the original database) with minimum support threshold = 50%, minimum confidence is 80%, 4 rules.
2. For the two inserted transaction (8+2=10 transactions), with minimum support threshold = 50%, minimum confidence is 80%, 2 rules .
3. For the same transactions but changing the minimum support threshold to 40%, minimum confidence is 80%, 10 transaction, 6 rules.
4. When delete one transaction (10-1=9 transactions), with minimum support threshold = 50%, minimum confidence is 80%, 3 rules.
5. With update transaction (9-1+1 =9 transactions), with minimum support threshold = 50%, minimum confidence is 80%, one rule. .

V. CONCLUSIONS

Data mining algorithms have at least two issues that characterize a database perspective of examining data mining concept: Efficiency and Scalability. Ideally any solution to data mining problems must be able to perform well against real-world databases. As far as the efficiency is concerned some parallelization is used to improve or overcome this issue. Dynamic data mining pose significant challenges. It can discover up-to-date patterns invaluable for timely strategic decisions, but this has to be done accurately and quickly with limited computation resources. Mining process can expose long-term trends and more complicated patterns that lead to deeper insights, but more than often meaningful patterns can only be found in subspaces, which incur high complexity in pattern mining.

This paper presents a two part solutions to the problem of Dynamic data mining. The first is concerned with process of detecting an update on the data after it has been collected for the data mining from its original source. The second deals with the process of maintaining the association rules based on the updates that have taken place on the original data in its original location. These two solutions when combined will allow the Enhancing (ARBSI) to solve the problem of dynamic data mining only one scan to the original source of data. This will provide an efficient dynamic data mining technique. Enhancing (ARBSI) works with massive real-world databases regardless of the amount of data and/or the amount of memory available. This algorithm also copies all updates that might take place in the original database to a dummy table specially created. This dummy table will contain a copy of the update records plus their summation value. And based on the summation value all the updated records are identified and all the necessary updates (insert, update, and delete) are carried out on the data used in the data mining process. The second part of the algorithm is used to maintain the association rules produced by the data mining process according to all updates carried out on the original sources of data. This process carries out this process using the data available in the dummy database containing the updated records and their summation value. Once it finished its task it clears the dummy database and waits for any new updates to take place. The paper also presents several examples to support the claims made. The results of the test showed that Enhancing (ARBSI) is capable of carrying out a data mining process on a dynamic database that is being continuously updated, covering all the three updates (insert, update, and delete) transactions. This algorithm was also tested using both static and dynamic databases in both cases the proposed algorithm achieved its task with high efficiency. From the above it is clear that the goal of this paper has been accomplished, in the form of the development of a unique technique to deal with both static and dynamic Data Mining process. The results obtained proved that Enhancing (ARBSI) is able to solve some of the problem related to the Dynamic Data Mining process.

REFERENCES

[1] Maria Halkidi, "Quality assessment and Uncertainty Handling in Data Mining Process" http://www.edbt2000.uni-konstanz.de/phd-workshop/papers/Halkidi.pdf

[2] Fayyad, U. M., G. P. Shapiro, P. Smyth. "From Data Mining to Knowledge Discovery in Databases", 0738-4602-1996, AI Magazine (Fall 1996): 37–53

[3] Jiawei Han, Micheline Kamber. "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers, Champaign: CS497JH ,Fall 2001, www.cs.sfu.ca/~han/DM_Book.html.

[4] Hebah H. O. Nasereddin , "Stream Data Mining", International Journal of Web Applications, Volume 3, Number 2, June 2011.pp 90- 97.

[5] Hebah H. O. Nasereddin, "New Technique to Deal with Dynamic Data Mining in the Database", International Journal of Research and Reviews in Applied Sciences, Volume 13, Issue3, December 2012. Pp 806-814

[6] Mehmed Kantardzic J. B. "Data Mining: Concepts, Models, Methods, and Algorithms", ISBN: 0471228524, IEEE Computer society, Wiley-Interscience, Hoboken, NJ, 2003.

[7] J.S. Park, M.S. Chen, P.S. Yu, "Using a hash-based method with transaction trimming for mining association rules," IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 5, pp. 812-825, 1997

[8] R. Agrawal, T. Imielinksi and A. Swami, "Mining association rules between sets of items in large database," The ACM SIGMOD Conference, pp. 207-216, Washington DC, USA, 1993.

[9] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," The International Conference on Very Large Data Bases, pp. 487-499, 1994

[10] R. Agrawal, R. Srikant and Q. Vu, "Mining association rules with item constraints," The Third International Conference on Knowledge Discovery in Databases and Data Mining, pp. 67-73, Newport Beach, California, 1997.

[11] Hebah H. O. Nasereddin , "Dynamic Data Mining Process" has been published in the ICITST-2008 conference, June in Dublin, Ireland. pp 23-26.

[12] D.W. Cheung, J. Han, V.T. Ng, and C.Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating approach," The Twelfth IEEE International Conference on Data Engineering, pp. 106-114, 1996.
[13] D.W. Cheung, S.D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," In Proceedings of Database Systems for Advanced Applications, pp. 185-194, Melbourne, Australia, 1997.
[14] M.Y. Lin and S.Y. Lee, "Incremental update on sequential patterns in large databases," The Tenth IEEE International Conference on Tools with Artificial Intelligence, pp. 24-31, 1998
[15] S. Zhang, "Aggregation and maintenance for database mining," Intelligent Data Analysis, Vol. 3, No. 6, pp. 475-490, 1999.
[16] Hebah H. O. Nasereddin , "An Enhanced Item-Summation for Dynamic Data Mining Algorithm", International Journal of Web Applications, Volume 4, Number 4, December 2012.pp 173- 184.
[17] Hebah H. O. Nasereddin "(ARBSI): Proposed Algorithm Association Rules Based on Scanning Itemsets", INTERNATIONAL JOURNAL OF CURRENT RESEARCH, Volume 9 issue 2, 2017 .pp 110- 121.
[18] Tzung-Pei Hong, Ching-Yao Wang, Yu-Hui Tao. "A new incremental data mining algorithm using pre-large itemsets" Intelligent Data Analysis, Issue: Volume 5, Number 2 / 2001, pages: 111–129.
[19] TZUNG-PEI HONG, TZU-JUNG HUANG. "Maintenance of Generalized Association Rules for Record Deletion Based on the Pre-Large Concept", Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, February16-19,2007 ,www.wseas.us/e-ibrary/ conferences/2007corfu/papers/ 540-410.pdf

Biographical notes: **Hebah H.O. Nasereddin** holds a PhD and is an Associate Professor in the Department of Computer Information Systems, and a Faculty member of Information Technology in Middle East University (MEU), , Jordan. Her research interests include dynamic data mining, e-business models and management, e-commerce, information systems development, e- learning and knowledge management, e-health and, recently, in e-disability. She has to her credit 33 publications in Scientific Conferences and Journals. She is a member of several professional and scientific societies. Nasereddin is a reviewer for several National and International journals and a keynote speaker for many conferences, general chair for ICNVICT. She is supervising many MSc, and Diploma thesis. Nasereddin published in Computer Philosophy and other Computer topics publications. She is Chief Editor and Editor for several Magazines in addition to her participation in project research evaluations.