**MVEMFI: Visualizing and Extracting Maximal Frequent Itemsets**

Maha Attia Hana*

*Department of Electrical and Communication, Canadian International College ElSheikh Zaid, Deputed from Faculty of computers & Information, Helwan University, Egypt

**ABSTRACT**

Association rule is a data mining technique that has a huge number of applications. One of the crucial steps in association rule is the extraction of frequent itemsets. This research is inspired by simple appealing visualization of itemsets frequencies in the simple well known two dimension matrix representations. This paper proposes a new procedure to extract maximal frequent itemsets called Matrix Visualization and Extraction of Maximal Frequent Itemsets. The procedure consists of two steps. The first step sets the environment to mine data while the second extracts frequent itemsets. MVEMFI procedure has been tested by three synthetic datasets and processing time has been recorded. It has been found that MVEMFI performance is not affected by the number of transactions or the density of items’ occurrences in the dataset.

**Keywords** – Association rules, Data mining, Maximal frequent itemset

**I. INTRODUCTION**

Association rule is a promising tool to expand the scope of data analysis and to reveal hidden relations among data values. Association rule mining has wide spectrum of applications in the area of customer relationship management [1], financial applications [2], tax inspection [3], traffic management [4] and computing in cloud environment [5, 6], education [7, 8].

Association rules mines dataset by discovering frequent patterns, then generates a set of rules that reveals the relationship between dataset items. The first step in association rule mining process is to extract frequent itemsets and the final step is rule generation. The most common used parameters for data mining are support and confidence. There are two ways two identify frequent itemsets either through candidate generation as in Apriori [9, 10] or without candidate generation as in pattern growth methods and their modifications [11, 12, 13, 14, 15, 16]. Also, there are three types of frequent itemsets; typical frequent itemsets [9, 10], closed frequent itemsets [15, 16, 17] and maximal frequent itemsets [11, 15, 18]. A typical frequent itemset is any itemset with frequency above a specific threshold. A closed frequent itemset is a frequent itemset that doesn’t have a superset with the same frequency. A maximal frequent itemset is a frequent itemset which has no proper frequent superset. Pruning the search of frequent itemsets depends on two properties. The first property states that if an itemset is infrequent, then all its supersets must be infrequent. The second property is that if an itemset is frequent, then all its subsets must be frequent. This research proposes a new non-candidate frequent itemsets generation procedure called Matrix Extraction and Visualization of Frequent Itemsets “MVEMFI”. The importance of the current research is to represent and to visualize itemsets’ frequencies in a naive simple two dimension matrix notation. MVEMFI procedure starts by constructing the matrix, then it undergoes a few processing steps and evolves by extracting frequent itemsets. The paper starts by reviewing related work in section two. Section three explains MVEMFI procedure while section four explains the conducted experiments. The results are in section five and the research is concluded in section six along with suggested future work.

**II. RELATED WORK**

There exits many association rule mining algorithms and this section reviews some of these algorithms. Apriori Algorithm [9, 10] employs a bottom-up search that enumerates every single frequent itemset. It starts by examining the count of single k-itemsets, then identifies frequent ones and uses them to produce k+1-itemsets. Apriori repeats the last two steps until no more frequent itemsets can be found. The exponential complexity of algorithm limits its usage to short patterns. Max-Miner [11] extracts only the maximal frequent itemsets by generating all the frequent itemsets using both bottom-up and top-down traversal. After identifying maximal frequent patterns, all frequent patterns are derived by scanning database to determine their frequency. It uses subset pruning for an infrequent itemset and “look ahead” pruning for the subsets of a frequent itemset. It mines long frequent itemsets and needs several scans to database. Pincer-Search [12] identifies maximal frequent itemsets and runs both bottom-up and top-
down searches at the same time. Both previously mentioned properties are used to prune candidates based on information gathered in one direction. If some maximal frequent itemset is found in the top-down direction, then this itemset can be used to eliminate its subsets as they are frequent. Also, if an infrequent itemset is found in the bottom-up direction, then it eliminates all its supersets in the top-down direction. The difference between Max-miner and Pincer-Search is that Max-miner uses a heuristic that looks ahead for the longest itemsets and reorder the items according to their frequency, so that itemset appears in the most candidate group. FP-tree [16, 17] mines closed frequent itemsets. FP-growth uses a divide-and-conquer way. It first scans dataset to get frequent itemset and in the second scan it generates FP-tree. Then, FP-growth starts to mine FP-tree starting by 1-K itemset, constructing conditional pattern base, then conditional FP-tree and mine it. FP-max [18] is a variation of FP-tree by using maximal frequent itemset. FP-max implements maximal frequent itemsets tree to keep track of all maximal frequent itemsets. FP-max reduces the number of subset frequency test operation, thus it reduces the search time. Eclat [14] uses TID(s) and their intersections to determine itemsets’ frequencies. TID are stored in a bit matrix. Eclat uses a prefix tree to search in depth-first order. MAFIA [15] mines maximal or closed frequent itemsets from a transactional database using a depth-first. MAFIA uses the above two properties to prune the search along with a third one. The third pruning method is based on the fact that if one transaction is a subset of another then the frequency of the former conforms to the latter. [19] Proposed HANA algorithm that has two steps. The first step uses TID(s) to count the frequency of the k-itemsets by intersecting transactions and storing the results in a matrix and then identifies the frequent itemsets. The second step generates (k-1)-itemsets for the frequent ones by introducing the concept of multiplex matrices. [20] Proposed HOUIMine algorithm to mine high on-shelf utility itemsets using three tables to speed processing. OS table is used to indicate the items on-shelf information. PTTU table records the transaction utility of all the transactions occurring within a time period. COSUI table records high transaction-weighted-utility of an itemset. The pruning strategy based on the on-shelf utility upper bound. The filtration mechanism for generating itemsets is also designed to prune redundant candidate itemsets early and to systematically check the itemsets. [21] Proposed association rule hiding (ARH) algorithm which deals with sensitive data. It mines the data, extracts rules, identifies sensitive rules, and then modifies the database to hide the transactions that support those sensitive rules. They compared ARH results with the k-anonymity method. [22] Proposed MFIF method that finds the maximal frequent item first by looking for transactions with maximum number of items rather than the minimal frequent itemset that starts by k equals one and increases to get k+1 frequent itemsets. If the frequency of maximal frequent itemset is greater or equal than the support, then it is a maximal frequent itemset. Otherwise, MFIF searches the corresponding subsets for a maximal frequent itemset. [23] Proposed a mining algorithm called Mining Frequent Weighed Itemsets (FWI). FWI assign different weights to all items and uses Weighted Itemset-Tidset tree (WIT-trees). Then, they proposed a Diffset strategy for both efficient computation of the weighted support of itemsets and for mining FWI.

III. MATRIX VISUALIZATION AND EXTRACTION OF MAXIMAL FREQUENT ITEMSETS “MVEMFIT” PROCEDURE

The visualization of frequent itemsets in matrix notation is shown in Fig. 1.
The analysis of Fig. 1 reveals a set of interesting findings. Itemsets are represented in the matrix by their decimal values. All possible itemsets are represented either individually, as one element in the matrix, or by combing two elements. If the matrix is reordered, their will be chunks of k-itemsets in blocks. Itemsets in the last column/row are supersets of all other items in previous column/row. In analogy, all itemsets in the first column/row are subsets of all other items in succeeding column/row. The itemset in the last column and the last row is superset for any itemset in matrix. Therefore, recognizing the relation of two itemsets would be applied rationally to their corresponding column/row.

Step 1: Determine the number of items and divide number of items in almost two equal values; No_row_items and No_column_items.

Step 2: Construct a two dimensional matrix where number of rows and columns equal to the number of itemsets that is generated from No_row_items and No_column_items, respectively. Fig. 1 shows symbolic, binary and decimal itemsets representations in matrix notation.

### Process 1: Mine Processing Preparations

**Input:** number of items  
**Output:** No_row_items, No_column_items, Frequency matrix, Row subsets, Row supersets, Column subsets, Column supersets  
**Steps:**

1. No_row_items = round (No_items/2)  
   No_column_items = No_items – No_row_items  
2. Construct frequency matrix (2^No_row_items, 2^No_column_items)  
3. Sort rows’ indices according to number of bits set to 1  
   Sort columns’ indices according to number of bits set to 1  
4. For each row index  
   Get Row’s index subsets  
   Get Row’s index supersets  
   End  
If (not (No_row_items == No_column_items))  
   For each row index  
   Get Column’s index subsets  
   Get Column’s index supersets  
   End  
Else  
   Column subsets = Row subsets  
   Column supersets = Row supersets  
   End

![Fig. 3 Pseudo code for process 1](image-url)
Column index of subsets for items in a row = \{ \{\}, \{1\}, \{1\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{1,2,3,4,5,6,7\} \}

Row index of subsets for items in a column = \{ \{1\}, \{1\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{1,2,5\}, \{1,3,5\}, \{1,4,5\}, \{1,2,3,4,6,7,8\}, \{1,2,3,5,6,9,10\}, \{1,2,4,5,7,9,11\}, \{1,3,4,5,8,10,11,12\}, \{1,2,3,4,5,15\} \}

Row index of supersets for items in a column = \{ \{6,7,9,10,12,13,14,15,16\}, \{6,8,9,10,12,13,15,16\}, \{7,8,11,12,14,15,16\}, \{12,13,16\}, \{12,14,16\}, \{12,15,16\}, \{13,14,16\}, \{13,15,16\}, \{14,15,16\}, \{16\}, \{16\} \}

Fig. 4 Symbolic itemsets sorted by the number of items

3.2 Maximal Frequent Itemsets Extraction Process

Maximal frequent itemsets extraction process consists of the following five steps and the pseudo code is in Fig. 6.

Step 1: Read the data set and convert each transaction itemset into decimal value. Then, for each transaction calculate the matrix element indices and increments its count.

Step 2: Sort the matrix according to step#3, process#1.

Step 3: Increment all subsets of each itemset in the matrix, using the discovered supersets. This enables the calculation of k-itemset frequency.

Step 4: Remove infrequent itemsets. An infrequent itemset has a frequency less than the support value.

Step 5: Remove all subsets of frequent items, using the discovered subsets and supersets in step#4, process#1. The removal starts with subsets of frequent itemsets in columns, then in rows as the number of columns maybe less than those of rows.

Fig. 6 Pseudo code for process 2

For example, if the number of items is five \{a, b, c, d, e\}, then items are divided into three items \{a, b, c\} and two other items \{d, e\}. The itemsets for \{a, b, c\} are represented in matrix row elements and \{d, e\} are represented in matrix column elements. Next, the Frequency matrix is constructed by initiating No_rows = 8 and No_column = 4. In step 3, both rows and columns values are sorted according to the number of items present in each itemset, Fig. 7. In the last step, the subsets and the supersets for each itemset in a row and a column is identified. Then, step#1, process#2 starts by reading data and converting it into decimal number, Table 1. Next, the matrix element index for each transaction is calculated and the corresponding matrix element is incremented, Fig. 8(a).

Fig. 7 Sorted Symbolic itemsets representation

Table 1 Dataset Sample

<table>
<thead>
<tr>
<th>T No.</th>
<th>Binary Itemset</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1 0 1 0 0</td>
<td>20</td>
</tr>
<tr>
<td>T2</td>
<td>0 1 1 0 1</td>
<td>13</td>
</tr>
<tr>
<td>T3</td>
<td>1 1 0 0 1</td>
<td>25</td>
</tr>
<tr>
<td>T4</td>
<td>1 1 0 1 0</td>
<td>26</td>
</tr>
<tr>
<td>T5</td>
<td>1 0 1 1 0</td>
<td>22</td>
</tr>
<tr>
<td>T6</td>
<td>1 1 0 0 0</td>
<td>24</td>
</tr>
<tr>
<td>T7</td>
<td>1 0 0 0 0</td>
<td>16</td>
</tr>
</tbody>
</table>
For example, in the set of transactions \{AC, CE, ABDE\}, their decimal values are \{5, 20, 27\}. The matrix indices for each one is \{(5,0), (4,2), (3,3)\} and their element frequency in matrix is incremented by one. In step 2, \textit{Frequency\_matrix} is sorted, Fig. 8(b).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{T9} & 0 & 0 & 0 & 1 \\
\textbf{T10} & 0 & 0 & 1 & 0 \\
\textbf{T11} & 1 & 0 & 0 & 0 \\
\textbf{T12} & 0 & 1 & 0 & 0 \\
\hline
\end{tabular}
\caption{Frequency\_matrix before sorting}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{T9} & 0 & 0 & 0 \\
\textbf{T10} & 0 & 0 & 1 \\
\textbf{T11} & 0 & 1 & 0 \\
\textbf{T12} & 0 & 1 & 0 \\
\hline
\end{tabular}
\caption{Frequency\_matrix after sorting}
\end{table}

In step 3, all subsets of each itemset in \textit{Frequency\_matrix} are incremented, using the discovered row’s supersets and column’s supersets. Fig. 9 shows \textit{Frequency\_matrix} after updates.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{T9} & 0 & 0 & 0 \\
\textbf{T10} & 0 & 0 & 1 \\
\textbf{T11} & 0 & 1 & 0 \\
\textbf{T12} & 0 & 1 & 0 \\
\hline
\end{tabular}
\caption{Frequency\_matrix after incrementing all itemsets’ subsets}
\end{table}

Next, infrequent itemsets are removed using a support value equals to one, Fig. 10. Finally, all frequent subsets of a frequent super itemset are removed starting by column then row, Fig. 11. At the last step, the final maximal frequent itemsets are identified along with their frequency. Maximal frequent itemsets are \{abc, ad, cd, bde\}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{T9} & 0 & 0 & 0 & 1 \\
\textbf{T10} & 0 & 0 & 1 & 0 \\
\textbf{T11} & 0 & 1 & 0 & 0 \\
\textbf{T12} & 0 & 1 & 0 & 0 \\
\hline
\end{tabular}
\caption{Frequency\_matrix after removing infrequent itemsets}
\end{table}

IV. EXPERIMENT

Three data sets are used to examine this work which has been used in [19]. Three datasets \{A, B, C\} are generated synthetically. The number of items in each set is 24 and the average number of items presence per transaction per data set is \{2, 7 and 13\} with 2000 transactions per data set.

V. RESULTS

Table 2 and Fig. 12 illustrate the processing time of MVEMFI algorithm for the three datasets for different transactions. Table 3 shows the mean and the standard deviation for each dataset separately and for the three datasets. MVEMFI is characterized by having almost a constant time of processing with mean value equals to 715.57 and standard deviation of 5.03.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Transaction No.} & \textbf{Dataset A} & \textbf{Dataset B} & \textbf{Dataset C} \\
\hline
50 & 696.80 & 702.18 & 706.06 \\
100 & 699.24 & 702.70 & 710.45 \\
150 & 702.19 & 703.36 & 710.94 \\
200 & 705.40 & 707.64 & 711.32 \\
250 & 705.41 & 708.63 & 724.63 \\
500 & 717.31 & 725.26 & 732.60 \\
1000 & 723.29 & 733.80 & 733.09 \\
2000 & 735.97 & 738.33 & 736.96 \\
\hline
\end{tabular}
\caption{MVEMFI procedure processing time}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Dataset A} & \textbf{Dataset B} & \textbf{Dataset C} \\
\hline
\textbf{Mean} & 710.71 & 715.24 & 720.76 & 715.57 \\
\textbf{Standard Deviation} & 13.59 & 14.87 & 12.41 & 5.03 \\
\hline
\end{tabular}
\caption{Table 3 Statistical measures for MVEMFI procedure}
\end{table}
Table 4 and Fig. 13 show the comparison between MVEMFI procedure and HANA algorithm in terms of the processing time. It is noticeable that HANA algorithm outperformed MVEMFI in case of small transaction number and low density datasets. On the other hand, MVEMFI procedure outperformed HANA algorithm in case of high density datasets even in small number of transactions.

Table 4 HANA algorithm processing time

<table>
<thead>
<tr>
<th>Transaction No.</th>
<th>Dataset A</th>
<th>Dataset B</th>
<th>Dataset C</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.14</td>
<td>2.11</td>
<td>74</td>
</tr>
<tr>
<td>100</td>
<td>0.34</td>
<td>9.09</td>
<td>1444</td>
</tr>
<tr>
<td>150</td>
<td>0.61</td>
<td>25</td>
<td>7740</td>
</tr>
<tr>
<td>200</td>
<td>0.97</td>
<td>54</td>
<td>139660</td>
</tr>
<tr>
<td>250</td>
<td>1.34</td>
<td>87</td>
<td>373395</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE WORK

This paper proposes MVEMFI procedure to discover frequent items without candidate generation using matrix notation. It can be inferred that the variation in processing times for all runs is small regardless of number of transaction or the density of the items in the datasets which is better than a solution with high variability in the processing time.

Also, MVEMFI is characterized by using matrix notation which is a simple well known data type. HANA and MVEMFI algorithms interchange the best performance as HANA uses transaction identifiers to extract frequent itemsets which are not the case with MVEMFI. HANA algorithm extracts all frequent itemsets while MVEMFI extracts maximal frequent itemsets. This explains why HANA algorithm outperformed MVEMFI in case of datasets with fewer items per transaction as well as when the number of transactions is few. There are several advantages of the MVEMFI algorithm. The above two characteristics are also two advantages that covers predictability, stability and simplicity. The next advantage is that it divides the number of items into two portions which limits the explosive nature of itemsets generation. Also, three of processing steps when identified for one matrix element are applied to the containing column/row. Those steps are reorder itemsets’ elements, identify subsets and supersets. The last advantage is that the generation of full frequent itemsets is minimized as it is performed after the removal of infrequent ones.

Future work may include adapting MVEMFI procedure to extract typical or closed frequent itemsets. Further work may include hardware MVEMFI procedure and upon successful performance, it may be a solution for stream data mining. Another suggestion is to represent frequent itemsets by higher matrix dimension and access the cost and the benefit of adding more dimensions to the matrix.

References


[16] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, Proceeding of the ACM SIGMOD International Conference on Management of Data (SIGMOD’00), TX, USA, 2000, 1-12.

[17] J. Han, J. Pei, Y. Yin, and R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Mining and Knowledge Discovery, 8, 2004, 53–87.


