A New Algorithm for Extracting Textual Maximal Frequent Itemsets from Arabic Documents

Zeyad hamid1,2, Hussein K Khafaji3
1 College of Information Technology, University OF Babylon, Iraq
2 Ministry of Iraqi defence, Iraq.
3 Computer Communication Engineering Dept, Al-Rafidain University College, Iraq
Zeyadhamid2@gmail.com, hr_hkml811@yahoo.com

Abstract. In this paper, a new technique has been suggested for extracting textual maximal frequent itemsets named Maximal Itemset Miner Algorithm (MIMA). This algorithm begins search process through generating the best initial border in search space depending on minimum support of items in the first level that achieves the general minimum support determined by the user. Our approach for counting itemsets support combines the idea of vertical representation of the data with a queue data structure to store the itemsets. To reduce search space, the algorithm adopted several pruning conditions for each itemsets in the initial border. Experiments performed on standard textual CNN Arabic dataset and proposed method registers less execution time comparing with the Apriori algorithm when applying it on three different size datasets.

1. Introduction

Association rules mining was proposed by Agrawal et al. in 1993 who produced Apriori algorithm for exploring and finding relations among items in database (1). Association rule mining is one of the important data mining tasks used for uncovering hidden relationships among itemsets of the database. This implies existing some items consider a proof of present other items with a specific probability. Confidence and support are two measures used to determine the power of discovered rules. Support of rules is the proportion of transactions that contain items, whereas confidence is the parameter measures the probability of existing specific items in case of present other items in the same transactions (2).

The formal definition of association rules: Let I = \{i1, i2, i3, i4, . . . , im\} be a set of items. Let D be a set of transactions, where each transaction t is a set of items such that t ⊆ I. Each transaction has a unique identifier, called TID. A transaction t contains X, a set of some items in I, if X ⊆ t. An association rule is an implication of the form X ⇒ Y, where X ⊂ I, Y ⊂ I, and X ∩ Y = Ø. The rule X ⇒ Y holds in D with confidence c if the segment of transactions that also contain Y in those which contain X in D is c. The rule X ⇒ Y has supported s in D if the segment of transactions in D that contains X ∪ Y is s. Given a set of transactions D, the aim of mining association rules is to generate all association rules, which have support and confidence equal and larger than the minimum support (minsup) and minimum confidence (minconf), which are defined by the user (3).

Association rules mining involve two steps discovering frequent itemsets that more or equal specific minimum support and then use these discovered frequent itemsets to construct association rules. The process of finding all frequent itemsets is a very challenging task because the search space is exponential in the number of features that existing in the database. Therefore, many algorithms developed for mining frequent itemsets since introduced by Agrawal et al in (1993) like Apriori, aprioriTID, partitioning, sampling and FP growth. Although previously mentioned algorithms, as well as many others algorithms for discovering frequent itemsets, appeared excellent performance, unfortunately, they produce a huge
amount of frequent itemsets especially when set a low value for minimum support and there is a dense database (4). Therefore to overcome this challenge, researchers have been proposed various approaches to find closed frequent itemsets and maximal frequent itemsets. This paper introduced a new algorithm for extracting textual maximal frequent itemsets.

2. Related work

In order to reduced time and memory space requirements for generating all frequent itemsets, many algorithms suggested for extracting these itemsets. Roberto J. (1998) presented Max_Miner algorithm which worked over-ordered and specific items and parent/child relationships. Max_Miner reduced the number of passes over the database by adopting a purely breadth-first search. Pruning approaches in this algorithm depend on subsets in frequency as well as superset frequency. Max_miner divided the candidate node in the search space in two parts are head and tail. The head contains itemsets counting by the node while all items in the tail part do not exist in the head part. For enumerating any candidate node the algorithm computes the support of head union each item in the tail. When generated frequent node superset pruning process will be performed over all subset of that node whereas subset-infrequence pruning strategy will apply when discovering the infrequent node. Experiments results appeared the proposed method was efficient by order of Apriori-like algorithm when frequent itemsets are long (5). MAFIA algorithm introduced by Doug et al (2005) and adopted depth-first traversal strategy and efficient pruning techniques to develop performance. The data represented using vertical schema. MAFIA supposes all database resides in main memory. This algorithm assumes itemsets have lexicographically ordered that mean the first item in the itemset call head and the other items of that itemsets call tail. MAFIA algorithm used PEP, FHUT, and HUTMFI pruning methods as well as dynamic reordering of the itemsets for reducing searching space. PEP pruning strategy depends on the relationship between the head of the node and its tail. So, because the algorithm searches about maximal frequent itemsets, any itemsets containing head items but not tail there is no necessity to count. If head union tail is frequent, FHUT will be pruning whole subsets. HUTMFI counts the support of itemsets and then checks if supersets of HUT in the MFI(maximal frequent itemsets) if MFI already possesses superset of HUT, the algorithm will pruning all subsets of that node. MAFIA algorithm also adopted a dynamic strategy for decreasing search space through increasing the support value rather than of lexicographically. MAFIA appeared better results on the dense datasets while in poor dataset MAFIA is competitive. In dense dataset MAFIA achieved performance that overcome others algorithm by a factor of three to 30(6). Gouda K. et al(2005) developed a GeMax algorithm for enumerating maximal frequent itemsets through exploiting backtracking search and number of enhancement pruning methods. Performance of the algorithm depended on two factors, the pruning strategies and the representation that applied for itemsets support counting. Progressive focusing is a mechanism used in GenMax for extracting maximal frequent itemsets whereas different propagation used for counting frequent items(7). Takeaki Uno et al. produced maximal frequent itemsets from closed itemsets by putting and rearranging indices on the items and used the maximality check method for pruning operation(8). Chen, F. et al (2007) suggested top-down and bottom-up hybrid search approach. The information generated through bottom-up search used to prune search space at top-down search. Bottom-up generates frequent and infrequent itemsets using Apriori-gen algorithm while top-down discovered maximal frequent itemsets by search space decomposition(9). Mir Md Et al. (2015) used genetic algorithm principles for producing maximal frequent itemsets through bitmap database representation. The algorithm transforms itemsets to chromosome code and removed each subset of frequent chromosome for reducing search space (10). Unil Yun et al. (2016) established Incremental Mining of Weighted Maximal Frequent Itemsets (IM_WMFI) for generating maximal frequent itemsets in the incremental database depending on itemsets weights. Itemset weight calculated through multiplying itemset support value by its average weight which is an average value among all the items forms that itemset. Then the algorithm will consider the itemsets if its weight factor value more than minimum support threshold (11).
processed in a parallel way in various nodes to generate best transactions that may contain maximal frequent itemsets. The algorithm employed the relation between best transactions and sub-transactions to find maximal frequent itemsets (12).

3. Maximal frequent itemsets
As mentioned previously mining association rules consists of two separated steps extracting all frequent itemsets and generating interesting association rules. Extracting frequent itemsets process takes a long time comparing with the second step (3). Therefore, many researchers have been focused to produce algorithms to reduce computational time and storage space that needed to discover all frequent itemsets. One of the main notations that have been pointed when applying traditional algorithms for extracting frequent itemsets they are generating a tremendous number of frequent itemsets. Unfortunately, extracting long itemsets will generate an exponential number of subitemsets causing deterioration of algorithms’ performance. For that reason, researchers suggested a maximal frequent itemsets algorithms approach for reducing computational cost and effort for finding frequent itemsets. Maximal frequent itemsets is proposed for reducing redundancy in the generated frequent itemsets by adoption the relation of itemsets. Maximal frequent itemset has minimum support that is equal or more than determined minimum support and all its supersets are infrequent. In other words, all subsets of maximal frequent itemsets are frequent while all its supersets itemsets are infrequent (13). The main reason for developing maximal frequent itemsets algorithms in reducing the redundancy of generated frequent itemsets which result in reducing required time and memory to extracting association rules. Maximal frequent itemsets can be used for driving all other frequent itemset. The goal of developing maximal frequent itemsets algorithms is to reduce search space in order to get frequent itemsets quickly and with minimal memory storage. Maximal frequent itemsets have compressed and lossless information of all frequent itemsets. It compressed since it contains reduced and precise itemsets than all frequent itemsets allowing to produce more concerned itemsets through decreasing minimum support threshold (14).

4. Natural language processing
Natural language processing is searching field related to studying and analyzing written human language by machine and building sophisticated applications for extracting useful information from textual data. Natural language processing involves several tasks includes tokenization, parsing and grammar, part of speech assignment and stemming that can be applied to the documents in order to get precise data for handling what problem at hand. Tokenization breaks document into separated words while determining word’s type whether it is noun, verb or adjective is the part of the speech assignment process. Steaming is the process of removing appended prefixes, infixing and suffixes from the word and return it to its root for reducing the document size because many of words belong to the same root. Parsing and grammar process targets to identify the relationship between a noun and verb phrases which help in the document processing [15]. Natural language processing collaborating with machine learning is used for classification, clustering and summarization of textual data. The main step for building textual classifier based on association rules is Extracting all or maximal frequent itemsets [16].

5. Primary concepts of the Arabic language
The Arabic language is one of Semitic languages family consists of 28 charterers and involves noun, verb and particle parts of speech [17]. The verbs have three forms which are presents, past and order tenses. Suffixes affixes are used for determining the verbs in the past tense, while prefixes affixes are used for specifying verbs in the present or feature tenses for example, “ketebet” means “she wrote”, whereas “tektub” word means “she writes”. The noun involves a singular, dual and plural formula that can be used for both female and male genders. Adjective and adverb in Arabic language can be generated from three primary parts of speech (noun, verb, practical). In the Arabic language, most of the words can be generated from tri-letter root whereas others can be extracted from the quad-letter root, Penta-letter root or Hexa-letter root [18]. Arabic language composes of classical, modern and colloquial forms. The classical form used by ancient’s Arab people for writing poetries and prose as well as, religious’ scripts written in this kind of Arabic language, therefore, acquired especial importance. On the other
hand, the modern form is used for official dealing between the governmental organizations whereas colloquial form used for informal contracting between local residents in a certain region.

6. Challenges of Arabic documents processing

In comparison with other languages, Arabic text is very difficult since it has special characteristics that differ from other languages. Arabic scripts were written from right to left and letter’s form changes depending on its place in the word [18]. Arabic language has complex orthographic and morphology features. Most of Arabic words can be obtained from a root word that consists of three characters (verbs). As well as, the meaning of some Arabic words depend on the signs (Harakat) that are ignored by most of the writers of the Arabic articles increasing the ambiguity of this language [19]. In the Arabic language, we can produce multiple words from one word. Unlike the English language which contains prefixes and suffixes, Arabic language has infixes affixes adding additional complexity for the stemming process. Synonyms are one of the problems that should be taken into consideration when dealing with Arabic articles since there are a lot of them in the Arabic language, for example, there are 52 synonyms for darkness word and Arabic speakers can use 16 alternatives words for expressing the moon word [6]. One of the useful feature used in the English text classification is capitalization which is not existing in the Arabic language. Sometimes the same word gives different meaning depending on its presence in the context of the sentence causing additional confusion for the applications that process this language. Some characters in the Arabic language written in the same way therefore dots with different numbers and places are used in order to distinguish between them. For the aforementioned challenges and others, documents that written in the Arabic language need special processing for getting accurate features can be used for machine learning.

7. Proposed approach for generating maximal frequent itemsets

In this section, we introduce a technique and new algorithm called maximal itemset miner algorithm for extracting textual Arabic maximal frequent itemsets through serial of steps as shown in Figure 1.

![Figure 1. Proposed technique](image)

7.1. Pre-processing steps

In order to reduce dataset volume, we need to conduct pre-processing steps which including tokenization, stop word removing and stemming. Tokenization partitions the text into discrete word. Then the ineffective words in the document are removed through performing the stop word removing step. Stemming is the last step in the pre-processing stage which means removing prefixes, suffixes and infixes from the words and abstract them to their roots. In this research, we used ISRI stemmer to generate the root of each word that obtained from stop word removing step. Items generated from stemming step are weighted using TF/IDF technique to produce items can be converted to two-dimensional matrix consists of a number of columns represent word sets of all document in the dataset.
while rows represent document IDs as shown in figure 2. This matrix used for calculating frequent of each item, for example, item1 in the below matrix repeated 3 times in all documents while item3 repeated 4 times.

| DocID | Item1 | Item2 | Item3 | Item4 |
|-------|-------|-------|-------|-------|
| DocID | 0     | 1     | 1     | 0     |
| DocID | 1     | 0     | 1     | 1     |
| DocID | 1     | 0     | 1     | 1     |
| DocID | 0     | 1     | 0     | 0     |
| DocID | 1     | 1     | 1     | 0     |

**Figure 2.** Represent the document as a matrix.

### 7.2. Generation initial border

Specifying Initial border plays an essential role in the algorithm's performance, so it is important selecting a suitable level to be the starting point for the searching process. The proposed algorithm used an equation permits generating initial border that existing in a middle position between the first level and the maximal level that may contain maximal frequent itemsets. This equation is equal to support the value of first element plus support value of the last element in the first level dividing on general minimum support that specified by the user.

\[
\text{InitialBorder} = \frac{\text{support value of first element in FL} + \text{support value of last element in FL}}{\text{general minimum support}}
\]  

### 7.3. Initial Border Generation Algorithm

After determining the initial border value we should extract the itemsets of that border using a special algorithm developed by authors for this purpose. The algorithm depends on a moving window over the members of first frequent itemsets in the first level. It is written in a self-documentation manner and it guarantees that the generated itemsets are sorted if the frequent 1-itemsets are presented to the algorithm in sorted form as shown in Figure 3. Each of itemsets of the initial border stored in the queue data structure.

```
Algorithm(): Initial border Generator Algorithm
Input: arrayOfItemset[]; // array of frequent 1-itemsets
       Initial_level // initial level in a lattice to start
       the mining
Output: M; // set of (initial level) itemsets
{M=\{ //construct first window
    Window=" ";
    For (i=1: i<Initial_level; i++)
    {Window.itemset=Window.itemset+arrayOfItemset[i].item;
     Window.transactionSet=window.transaction+arrayOfItemset[i].transaction
    }
    Temp=Initial_level;
    For (i=1; i<=len(arrayOfItemset)-Initial_level-1; i++)
    {For (j=Temp; j<= len(arrayOfItemset); j++)
    {
        // construct a new itemset
        Window(t)=window + arrayOfItemset[j];
        M=union(M,window(t));
        Window(t)=" ";
    }
    // construct next window by removing second item and
    //concatenate with the item beyond the window.
    Window =rightTrim(window)+ arrayOfItemset[Temp];
} }
```
Figure 3. Initial border generator algorithm

Example: Suppose that the first frequent itemsets level are \{a,b,c,d,e,f\} and the initial level is 3. This leads to that first window contains a and b. Hence \{a,b,c\}, \{a,b,d\}, \{a,b,e\} and \{a,b,f\} will be generated from the union of the base ab and the rest of frequent 1-item, i.e, c, d, e and f. The first move of the window makes the window contains a and c so, it will generate itemsets \{a, c, d\}, \{a, c, e\}, and \{a, c, f\}, then the window move to bc and this window will be related to rest of the items and so on. Figure 4 shows the process of generating the k-itemsets of the initial level.

Figure 4. Process of generating initial border

Where window1 produces \{ABC, ABD, ABE, ABF\} itemsets of initial border and window2 produces \{ACD, ACE, ACF\} and window3 produces \{ADE, ADF\}, and window4 produces \{AEF\}. Then when the first window access to the last item of the first level, it will be changed and become BC and the process repeated until generating all itemsets of the initial border. The itemsets of the initial border will be kept in a queue data structure in order to send them to maximal frequent itemsets miner algorithm.

7.4. Maximal Itemset Miner Algorithm

The proposed algorithm for generating maximal frequent itemsets is illustrated in figure 5. The algorithm takes the itemset from the queue and classifies it into maximal, frequent and infrequent itemset. If the itemset is maximal will kept it in the queue. If the itemset is frequent the algorithm will generate all superset for that itemset and store them again in the queue, while if the itemset is infrequent the algorithm will generate all subset of that itemset. All these types of itemset will be submitted to some conditions that used to determine whether we should put the supersets or subsets in the queue to reduce the search space. The first condition we can used to avoid adding the itemsets and consequently all subset if \(Z = \{x_1, \ldots, x_{k-1}, y_{k-1}\}\) is an infrequent k-itemset that can be generated from two \((k-1)\)-itemsets; \(X = \{x_1, \ldots, x_{k-1}\}\) and \(Y = \{x_1, \ldots, y_{k-1}\}\) such that \(X\) is frequent and \(Y\) is infrequent or vice versa, then \(Z\) has no sub itemset, \(W\) that is maximal itemset (observation 1). So we can remove \(Z\) itemset from the queue and ignore all its subset and its superset. The second condition we can exploiting to accelerate extracting maximal frequent itemset if \(Z\) is an infrequent k-itemset and all its entire sub \((k-1)\)-itemsets are infrequent then it may have sub itemsets that are maximal itemsets in another level of the lattice. So, all the generated infrequent itemset adding to the queue and ignore the original infrequent itemset. Another condition used in this algorithm if \(Z\) is an infrequent k-itemset, then \(Z\) can be generated from infrequent and a maximal itemset. So, if we face this type of itemset we need just check whether the frequent itemset subsets of that itemsete is maximal or not so we need to add these frequent itemsets and ignore their parents and all its superset. This process continues until all itemsets in queue become maximal.
Experiments and evaluation

In order to prove the performance of the proposed algorithm, a standard CNN textual Arabic datasets consist of 5,070 documents involves sport, business, entertainment, middle-east news, sciences and technology and word news taken from [20] used to apply MIMA. We apply our algorithm on three collections of the dataset: sport, business and world news. Table 1 shows the characters of each group of the dataset.

| Dataset Name    | Number of documents |
|-----------------|---------------------|
| Sport           | 381                 |
| Business        | 836                 |
| World News      | 1010                |

The execution time of proposed algorithm has been compared with the execution time of the Apriori algorithm with 40% to 80% minimum support threshold and time calculation starts when both proposed algorithm and Apriori begin generating frequent itemsets. Table 2 shows experiments results on sports documents with respect to the processing time in second while table 3 and table 4 explain experiments results with business and word news documents respectively. As illustrated in each table, the execution time of our proposed algorithm is highly decreased from the execution time of the Apriori algorithm as
well as execution time decreased when minimum support decreased and verse versa. By performed experiments on three different size datasets, we proved scalability of the proposed algorithm.

| Table 2. Test results for sport documents |
| Minimum support | Apriori | MIMA |
|------------------|---------|------|
| 40%              | 141.70  | 40.56|
| 50%              | 57.81   | 1.96 |
| 60%              | 45.73   | 0.10 |
| 70%              | 42.44   | 0.015|
| 80%              | 45.54   | 0.015|

| Table 3. Test results for business documents |
| Minimum support | Apriori | MIMA |
|------------------|---------|------|
| 40%              | 439.61  | 140.54|
| 50%              | 158.96  | 4.218|
| 60%              | 130.73  | 0.156|
| 70%              | 119.71  | 0.095|
| 80%              | 124.59  | 0.093|

| Table 4. Test results for world news documents |
| Minimum support | Apriori | MIMA |
|------------------|---------|------|
| 40%              | 292.87  | 25.03|
| 50%              | 197.46  | 3.24 |
| 60%              | 172.89  | 1.61 |
| 70%              | 186.0   | 1.43 |
| 80%              | 176.0   | 0.96 |

9. Conclusion
In this paper, we proposed a new mechanism for extracting maximal frequent itemsets from Arabic textual dataset called MIMA. The algorithm starts with the selected initial border in search space and then moving up and down to produce desired maximal frequent itemsets. The performance MIMA actually highly depending on the initial border that should the searching process start from it, so we suggested an equation considerate this fact through take into consideration the number of frequent itemsets in the first level and their minimum support in order to determine searching space initial border. We perform our experiments on three different sizes textual Arabic data sets and compared the results with Apriori results and we found our proposed algorithm was a superior Apriori algorithm. Our future plan is parallelize MIMA algorithm through using parallel /Multi-threaded system.

References
[1] Rakesh Agrawal, Tomasz Imieliński and Arun N. Swami 1993 Data Mining: A Performance Perspective, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, pp. 914–925.

[2] H. Wang, D. Can, A. Kazemzadeh, F. Bar and S. Narayanan 2012 A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle, *Jeju, Repub. Korea*, pp. 115–120.

[3] R. Agrawal and R. Srikant, 1994 Fast Algorithms for Mining Association Rules in Large Databases *J. Comput. Sci. Technol.*, vol. 15, no. 6, pp. 487–499.

[4] J. Han, H. Cheng, D. Xin, and X. Yan 2007 Frequent pattern mining: current status and future directions, *Data Min. Knowl. Discov.*, vol. 15, no. 1, pp. 55–86.

[5] Bayardo, R. J. 1998 Efficiently mining long patterns from databases. ACM SIGMOD Record 27 (2), 85–93.

[6] Burdick, D. Calimlim, M. Flannick, J. Gehrke, and Yiu T. 2005. MAFIA: a maximal frequent itemset algorithm. *IEEE Transactions on Knowledge and Data Engineering* 17(11)

[7] Gouda, K. and Zaki M. J. 2005 GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets *Data Mining and Knowledge Discovery*, 11(3), 223–242.

[8] Takeaki Uno1, Masashi Kiyomi1, Hiroki Arimura2 1 National Institute of Informatics Hitotsubashi, Chiyoda-ku, Tokyo, Japan Information Science and Technology Hokkaido University.

[9] Chen, F. and Li M. 2007 A Two-Way Hybrid Algorithm for Maximal Frequent Itemsets Mining. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*.

[10] Mir Md. Jahangir Kabir, Shuxiang Xu, Byeong Ho Kang and Zongyuan Zhao GeneticMax: An Efficient Approach to Mining Maximal Frequent Itemsets Based on Genetic Algorithms *School of Engineering and ICT University of Tasmania Launceston, Australia*

[11] Yun, U. and Lee G. 2016 Incremental mining of weighted maximal frequent itemsets from dynamic databases. *Expert Systems with Applications*, 54, 304–327.

[12] Liu Z., Hu L. Wu C., Ding Y., Wen Q. and Zhao J. 2018. A novel process- based association rule approach through maximal frequent itemsets for big data processing *Future Generation Computer Systems* 81, 414–424.

[13] Taouil, N. Pasquier, Y. Bastide, and L. Lakhal 2000 Mining bases for association rules using closed sets. *In IEEE Intl. Conf. on Data Engineering*

[14] Ye-In Chang, Chia-En Li, Wei-Hau Peng and Syuan-Yun Wang 2013 Efficient superset-lattice algorithms for mining closed frequent itemsets and maximal frequent itemsets in data streams *international journal of electrical and engineering*, vol.20, no.2 pp.51-63.

[15] Grant S, Ingersoll Thomas S, Morton Andrew L and Farris 2013, taming text *Manning Publications Co.*

[16] Souad Larabi Marie-Sainte , Nada Alalyani, Sihaam Alotaibi , Sanaa Ghouzali , Ibrahim Abunadi 2017 Arabic Natural Language Processing and Machine Learning-based Systems *IEEE Access*

[17] Abdullah Ayedh , Guanzheng TAN Khaled Alwesabi and Hamdi Rajeh, 2016 The Effect of Preprocessing on Arabic Document Categorization.

[18] Alabbas W., Al-Khateeb H., and Mansour A. 2016 Arabic text classification methods: Systematic literature review of primary studies 2016 4th IEEE International Colloquium on Information Science and Technology.

[19] Qasem A. Al-Radaideh and Mohammed A. Al- Abrat 2018 An Arabic text categorization approach using term weighting and multiple reducts *Springer-Verlag GmbH Germany, part of Springer Nature 2018*

[20] [https://sourceforge.net/projects/ar-text mining/files/Arabic- Corpora/](https://sourceforge.net/projects/ar-text mining/files/Arabic- Corpora/)