Revue d'Intelligence Artificielle
Vol. 38, No. 3, June, 2024, pp. 805-813
Journal homepage: http://iieta.org/journals/ria

A New Algorithm for Arabic Document Clustering Utilizing Maximal Wordsets

Khitam A. Salman[1*], Hussein K. Khafaji[2]

[1] Informatics Institute for Postgraduate Studies (IIPS), Iraqi Commission for Computers and Informatics (ICCI), Baghdad 10068, Iraq
[2] Computer Communication Engineering Department, Al-Rafidain University College, Baghdad 10014, Iraq

Corresponding Author Email: khitam.a.salman@gmail.com

https://doi.org/10.18280/ria.380307

Received: 15 September 2023
Revised: 12 January 2024
Accepted: 29 March 2024
Available online: 21 June 2024

Keywords:
MFWs, Maximal Frequent Wordsets, Arabic documents clustering, similarity functions, CNN, OSAC

ABSTRACT

Arabic document clustering (ADC) is a critical task in Arabic Natural Language Processing (ANLP), with applications in text mining, information retrieval, Arabic search engines, sentiment analysis, topic modeling, document summarization, and user review analysis. In spite of the critical needs of ADC, the available ADC algorithms achieved limited success based on the evaluation metrics used for clustering. This paper proposes a novel method for clustering Arabic documents. The method leverages Maximal Frequent Wordsets (MFWs). The MFWs are extracted using the FPMax algorithm, a data mining technique adept at identifying significant recurring word patterns within the documents. These MFWSs serve as features for a new clustering approach that groups documents based on content similarity. Each MFW serves as a data structure housing features, their respective strengths in clustering, and the corresponding documents, simplifying the clustering process to a mere measurement of similarity. The proposed approach offers various clustering results for varying numbers of clusters in one training session. The effectiveness of the proposed method is assessed using two well-known benchmark datasets (CNN and OSAC), achieving accuracy of 80% and 81% respectively. This approach offers a promising contribution to the field of ANLP.

1. INTRODUCTION

Arabic is a Semitic language. It is most often spoken in nations where the majority of the population is Muslim. It is also the language of the "AL-Quran AL-Kareem," the Muslims' holy book. Over 400 million people speak Arabic as their first language, and over 250 million speak it as their second language, and it is recognized as the official tongue of states in North Africa and the Middle East [1]. Traditional, Modern Standard Arabic (MSA), and dialects are the three main categories of Arabic.

The Arabic language presents unique challenges for document clustering tasks compared to languages with simpler structures. Notably, the Arabic script lacks inherent vowel markings, leading to ambiguity. Additionally, Arabic morphology features complex derivational prefixes and suffixes that significantly alter word meaning. These characteristics necessitate specialized preprocessing techniques like disambiguation and stemming to overcome these challenges during document preparation [2, 3]. Document clustering plays a crucial role in data mining and information retrieval, especially for the vast and ever-growing volume of Arabic documents online. By grouping documents based on thematic similarity, clustering facilitates efficient navigation and analysis of these information resources. Effective clustering of Arabic documents is essential for various applications, including information retrieval, text mining, automatic document categorization and user review analysis [4].

Document clustering is an unsupervised machine learning process that groups documents based on similarities by eliminating the intra-similarity among documents in one group and increasing the inter-similarity properties among different groups. This process does not require class labels for the documents [5]. A challenging task in data and text mining is identifying hidden, important, and possible patterns in the document [6]. The most difficult problems in document clustering can be summed up as coping with massive data quantities, high dimensionality, and low retrieval precision.

Dealing with large data volumes, high dimensionality, and low retrieval precision can be summed up as the most challenging issues when dealing with document clustering [7]. Arabic documents are now readily available online in a variety of formats, making it difficult to organize them without the aid of a computer. The clustering of Arabic-language documents has recently attracted the attention of researchers.

To depict the connection between data points and the clusters they belong to, hierarchical clustering creates a structure like a tree. Each data point is initially clustered separately and iteratively merging smaller clusters into bigger clusters when the stopping criteria is satisfied. Hierarchical clustering comes in two forms: agglomerative (bottom-up) and divisive (top-down).

Partitional clustering divides the data into a predetermined

number of distinct, non-overlapping clusters. It aims to keep the distances between the data points and the cluster centroids as small as possible. K-means and K-medoids are two common partitional clustering algorithms. While density-based clustering, creates clusters by locating regions with a high data point density. A cluster is defined as a dense zone surrounded by a sparse region. This type of clustering is helpful in detecting clusters of arbitrary forms. DBSCAN and HDBSCAN are examples of this type of clustering algorithms [8].

Association rule mining (ARM) [9] is a data mining technique that is used to discover potential hidden patterns among data. Apriori and FP-Growth are the most widely used algorithms for extracting all frequent itemsets (FI) and frequent patterns in datasets, respectively [10]. To extract FI from large transactions, the Apriori approach is widely utilized. It works on the assumption that all FI subsets must also be frequent. It continuously counts the support of items and item combinations that can be utilized within a transaction and eliminating any sets that go below a user-defined threshold and by integrating sets from the previous iteration and validating them against the transaction, candidate frequent itemsets are created. Despite being easy to comprehend and utilize, the Apriori technique can be computationally expensive for massive transactions and demands several database searches. On the other hand, the FP-Growth approach is scalable and utilized to mine frequent item sets in large datasets. The algorithm stores the frequent item sets in a tree-like data structure known as the FP-tree. The itemsets are encoded in the tree after a single run over the dataset to generate it, making it fast and quick to mine the common itemsets by utilizing depth-first search to traverse the tree. In order to decrease the size of the tree and improve the algorithm's scalability and performance, the divide-and-conquer strategy and pruning techniques are used. FP-Growth has been shown to be faster than traditional frequent item set mining algorithms, such as Apriori, and is widely used in data mining and machine learning applications.

The conventional frequent itemsets mining algorithms are computationally extensive and generate voluminous sets of items. Therefore, maximal frequent itemsets (MFI) approach is undertaken to overcome the forementioned problems by significantly reducing the search space. Various algorithms were proposed for mining MFI, such as MAFIA [11], FPmax algorithm [12], MaxMining [13], GenMax [14], and MIMA [15, 16] which dedicated to mine textual MFI from Arabic documents.

Document clustering plays a vital role in organizing and analyzing vast collections of Arabic text data. However, the unique characteristics of the Arabic language, such as the lack of inherent vowel markings and complex morphology, pose significant challenges for traditional clustering techniques. This paper proposes a novel algorithm to Arabic document clustering that addresses these challenges and offers promising accuracy.

Our key contribution lies in leveraging Maximal Frequent Word Sequences (MFWSs) for Arabic document clustering. By employing the FPMax algorithm, we extract the most prominent recurring sequences of words within the documents. These MFWSs capture the thematic content of documents more effectively compared to individual words, as they account for the inherent structure and context of the Arabic language.

The remainder of this paper is organized as follows; the second section examines previous studies that have used maximally frequent item sets for document clustering. The proposed approach is thoroughly explained in Section 3 with an example, and the experiments and findings are covered in Section 4. The conclusion is demonstrated in the final section.

2. RELATED WORKS

The interest in Arabic document clustering has increased recently due to the growing amount of Arabic content on the Internet, making manual clustering impractical. Our previous survey [17] on the research of Arabic document clustering approaches and techniques revealed a limited number of studies in this area. To the best of our knowledge, no one has utilized Maximum Frequent Itemset (MFI) for categorizing Arabic documents. So far, only one study has used Frequent Itemsets (FI) to cluster Arabic documents, using a hierarchical clustering approach based on N-grams [18]. The clustering accuracy achieved by the Frequent Itemset-based Hierarchical Clustering (FIHC) was 70%, higher than the 63% accuracy obtained for clustering European languages. The results of the research are not confident due to the lack of information about the dataset used for experiments. Also, the use of only one dataset is insufficient to judge the efficiency and predict the behavior of an algorithm

In literature [19], a study investigated an approach that used K-means and particle swarm optimization (PSO) to group Arabic documents. K-means is sensitive to the selection of the initial clusters producing different results according to the initial points selected. This issue has been overcome by using PSO to analyze the entire dataset and identify the best starting points for K-means and achieve good clustering results. This approach inherited the drawbacks of PSO and K-means such as sensitivity to initial parameters, computational complexity, and convergence to local optima.

Alhawarat and Hegazi [20] utilized Latent Dirichlet Allocation (LDA) and K-means for document clustering, finding that normalization of text data led to substantial improvements in clustering outcomes. When the combined method was applied with normalization, it achieved higher scores (29% and 40% F-score for BBC and CNN respectively) compared to the traditional approach (24% and 29% for the same datasets). However, LDA may face scalability challenges when applied to large-scale datasets, as the model's complexity increases with the number of documents and topics. Also, LDA requires tuning of hyperparameters such as the number of topics (K), Dirichlet priors, and sampling techniques. Improper selection of hyperparameters can impact the quality of clustering results.

Sangaiah [21] proposed unsupervised clustering for Arabic documents. He compared three approaches: supervised, semi-supervised, and unsupervised. These methods utilized K-means, incremental K-means, threshold + K-means, and K-means with dimensionality reduction (DR) for clustering. Unsupervised clustering achieved 70% and 43% for F-measure and entropy, respectively, and it is regarded as effective for Arabic document clustering.

Although k-means is effective, it is sensitive to selecting the initial point, and this may hinder the performance. PSO-K-means solve this issue but poses computation costs. When the semantic is crucial, LDA is sufficient, but it adds complexity. K-means with dimensionality reduction is effective, but information loss may occur during reduction.

Our clustering strategy effectively reduces dimensions in Arabic documents, ensuring accurate text analysis and high-speed reduction of dimensions for effective clustering results, as we will describe in the upcoming sections.

3. THE PROPOSED SYSTEM

The Maximal Frequent Wordset-Based Arabic Document Clustering System (MFW-ADC) is presented in this paper. MFW leverages the FPMax algorithm to extract informative MFWs from Arabic documents. These MFWs capture the thematic content and inherent structure of the language, enabling effective document clustering. The model comprises three modules: preprocessing, dimensionality reduction and MFW mining, and clustering (as shown in Figure 1). These modules are described in detail in the subsequent sections.

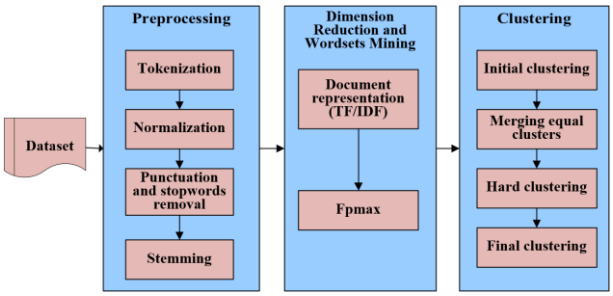

Figure 1. System architecture

3.1 Preprocessing module

Document preprocessing is a crucial step in natural language analysis, ensuring trustworthy and reproducible textual data. It comprises four stages: tokenization, normalization, punctuation and stopword removal, and stemming. Tokenization divides a document into separate words. Normalization converts letters into one form, while special characters and stopword removal reduce data dimensionality and improve analysis accuracy. Stemming generates a morphological variant of the base word, ensuring the same root word is considered the same entity. In Arabic, several stemming algorithms have been developed, such as Tashapheen, Khoja [22], Light Stemmer [23], etc., to achieve optimal results. The algorithm is depicted in Algorithm 1. Table 1. shows an explanation of the variables, symbols, and functions used in the proposed algorithms.

Algorithm 1. Dataset Preprocessing algorithm

Input: D: Dataset (Document text), special_char_list, normalized_letters_list
Output: preprocessed_dataset

```
Begin
  For each document d in D Do
    Read (d)
    td=tokenization(d)
      For each token t in td Do
        //Check token letter
        If letter in normalized_letters_list Do {
          norm_t=normalization(token);
        If norm_t in special_char_list Do
          Remove(norm_t);
          Else{ stemmed_t= stemming(norm_t);
          write stemmed_t to preprocessed_d;}}
        Next token
      preprocessed_dataset ∪= preprocessed_d;
      Next document
  Return preprocessed_dataset;
  End
```

Table 1. Variables and functions explanation

Seq	Variable/ Function	Meaning
1	D	Arabic document dataset
2	special_char_list,	[.... ،"" ، ! ،؟]
3	normalized_letters_list	[أ، إ، آ، ء] to [ا] [ت، ة، ـة] to [ه]
4	preprocessed_dataset	Arabic document dataset after the preprocessing.
5	d	A document in D
6	DID	A file contains the tokens of d.
7	Tokenization function	A function to tokenize a document d.
8	normalization	A function to normlize a lettet
9	Stemming	A stemming function to extract a word's root
10	minsupp	Minimum support
11	T_{doc}	Fp tree of document
12	MFWT	MFW tree of maximum frequent wordsets
13	MFW	Set of all the mined MFW
14	P	Path
15	Best_mfw	Best cluster
16	filtered_clusters	A list containing unique clusters
17	Dinitial_cluster	information about the clusters (mfw (cluster label), ID, DID,mfw_support)
18	Merge_cluster	A function to merge clusters
19	belong_cluster	A percentage at which a document belongs to a cluster
20	doc_feature	Document feature vector

3.2 Dimension reduction and wordsets mining module

This module comprises two steps: dimension reduction and wordset mining. Dimension reduction is accomplished in two stages. The first phase is done by representing the preprocessed documents using TF/IDF approach and the second stage is accomplished by utilizingthe FPmax algorithm to extract the MFWs according to the provided minimum support.

TF/IDF is one of the best metrics used to show how significant a word is to a document in a dataset. Two factors are considered when calculating TF/IDF: word frequency (how many times a word appears in a specific document) and the inverse document frequency (how often this word appears in all document in the dataset). This technique optimizes frequent, rare words in a document to highlight discriminative features, ensuring the content is understood while excluding irrelevant or common words. Eqs. (1) and (2) show how to calculate TF-IDF.

$$IDF(w) = log(N/n\_w) \quad (1)$$

$$TF - IDF(w,d) = TF(w,d) * IDF(w) \quad (2)$$

where, TF(w,d) is the frequency of word w in document d (i.e., the number of times word w appears in document d) and IDF(w) is the inverse document frequency of word w, N is the total number of documents in the corpus, and n_w is the number of documents in the corpus that contain word w.

Our dataset consists of Arabic documents. Each document is identified by a unique document identifierDID and a list of preprocessed words (wordlist). To reduce the data complexity, Fpmax algorithm [12] is used to discover the most frequent word patterns (MFWs) within the documents. This technique relies on two key parameters: minimum support and maximum wordset length. Adjusting these parameters helps in reducing the data dimentionality while still obtaining the crucial inormation from the documents. This will enhance the clustering effeciency and accuracy. The algorithm used is shown in Algorithm 2.

Algorithm 2. Mining maximal frequent wordsets algorithm fpmax (T_{doc})

Input: preprocessed_dataset, *minsupp*, *max_mfw-len*, T_{doc}, $MFWT$, Head: a linked list of words.
Output: *MFWT* that contains all the *MFWs*, *DID* sets
Begin
  Read preprocessed_dataset
  if T_{doc} only contains a single path P
    insert $Head \cup P$ into $MFWT$
  else for each *w* in Header-table of T_{doc}
    Append *w* to *Head*
  Construct the Head-pattern base
  Tail = $\{frequent\ words\ in\ base\}$
  subset_checking($Head \cup Tail$);
  if $Head \cup Tail$ is not in MFW
    construct the FP-tree T_{Head}
  call Fpmax(T_{Head})
remove *w* from $Head$

3.3 Clustering module

This module outlines four key steps: primary clustering, merging equal clusters, hard clustering, and final clustering, which are detailed in the following subsections.

3.3.1 Primary clustering

Initial clusters are created using the Fpmax algorithm's MFWs, with words as labels. The best-fit clusters are determined using similarity functions such as Euclidean distance, cosine similarity, Manhattan distance, overlap, and Jaccard index, as represented by Eqs. (3) to (7). The length of the cluster's label, i.e., MFW's length, will be added to the similarity function. The purpose of adding this factor is to assign the document to the most similar and largest clusters. The process of determining initial clusters is illustrated in algorithm 3.

$$d(d,mfw)\ sqrt(sum((d_i - mfw_i)^2)) + |mfw| \quad (3)$$

$$cos(\theta) = ((d * mfw)/||d|| * ||mfw||) + |mfw| \quad (4)$$

$$d(d,mfw) = \sum|d_i - mfw_i| + |mfw| \quad (5)$$

$$Overlap(D,mfw) = \left[\frac{|D \cap mfw|}{min\ (|D|,|mfw)|}\right] + |mfw| \quad (6)$$

$$J(D,mfw) = \left[\frac{|D \cap mfw|}{|D \cup mfw|}\right] + |mfw| \quad (7)$$

Algorithm 3. Primary clustering algorithm

Input: MFWs, DID sets
Output: initial clusters
Begin
Initial_clusters = []
  For d_i in preprocessed_dataset**:**
    //Determine best initial cluster (best MFW)
    best_mfw = None
    For mfw_i in MFW:
      intersection = find_intersection(mfw_i, d_i)
      if intersection == 0 {score = 0}
      else{score = similarity_metric(mfw_i, d_i)
          +|mfw_i|}
      //End if
      similarity = best_score (depending on similarity metrics)
      best_mfw = mfw_i
      initial_clusters ∪= (best_mfw, d_i)
    End For
    Next d_i
  End for
  Return initial_clusters
End

3.3.2 Merging equal clusters

It's worthy to mention that, after identifying the initial clusters, it is possible to identify numerous clusters with similar DID sets. After that, these clusters can be combined using the approach illustrated in Algorithm 4.

Algorithm 4. Merging equal clusters algorithm

Input: MFWs, DID sets
Output: filtered clusters
Begin
  filtered_clusters = []
  For i=0 to |MFWs| -1 Do
    For j =i+1 to |MFWs| -1 Do
      //determine if two clusters contain same DID (Same documents)
      If (DID_i == DID_j):
        merge_cluster(MFW_i, MFW_j)
    filtered_clusters∪= (MFW_i, filtered_clusters)
    End for
  End for
  Return filtered_clusters
End

3.3.3 Hard clustering

The proposed algorithm uses hard clustering to assign documents to specific clusters based on their similarity. This process calculates the percentage of belonging between documents and MFWs, retaining the highest-percentage-related document and removing the lowest-percentage-related document. The cluster with significant support can incorporate a document when competing with other clusters. The algorithm utilized for hard clustering implementation is depicted in algorithm 5.

Algorithm 5. Hard clustering algorithm

Input: Dinitial_cluster(information about the clusters (mfw(cluster label),DID, mfw_support, ocument_feature
Output: hrad clusters
Begin
  for i=0 to |Dinitial_cluster| -1 Do

```
    for j = i+1 to |Dinitial_cluster| -1 Do
      // check if documents are assigned to many clusters
      value = intersection(initial_cluster[i, DID],
      initial_cluster[j, DID] )
      if value !=0 Do
        //calculate document belong percentage
        belong_cluster_i=intersection( cluster_i_mfw,
        doc_feature)
        belong_cluster_j = intersection(cluster_i_mfw,
        doc_feature)
        if (belong_cluster_ i) > (belong_cluster_j) Do
          remove(DID, cluster_j)
        else if (belong_ cluster_i) < (belong_ cluster_j)
          remove(DID, cluster_i)
          else: //compare clusters size
          if |cluster_i| > |cluster_j| Do
            remove(DID, cluster_j)
          else if |cluster_i| < |cluster_j|
            remove(DID, cluster_i)
          else : if support(cluster_i) > support(cluster_j)
             remove(DID, cluster_j)
            else if support(cluster_i) < support(cluster_j)
             remove(DID, cluster_j)
            else {remove(DID, cluster_j)}
            end if
          end if
        end if
      end for
    end if
  end for
end for
End
```

3.3.4 Final clustering

The final step involves merging clusters (MFWs) from previous stages to create final clusters. The desired number of clusters is specified, and the clusters are merged with the most similar one based on their similarity value. One of the measures described in Section 3.2 is used to calculate the similarities between the MFWs. The length of the MFW is a significant factor that ensures a cluster will be merged with the longest and most similar one. The employed algorithm is depicted in algorithm 6.

Algorithm 6. Final clustering algorithm

```
Input: MFWs, DID sets
Output: final_clusters
Begin
  Final_cluster=[]
    For each mfw_i in MFWs:
        For each mfw_j in MFWs:
          Intersection = intersection(mfw_i, mfw_j)
          If Intersection == 0 Do: {score = 0}
          Else:
              score = similarity_metric(mfw_i, mfw_j) +
                len(mfw_j)
              append(score_list,[mfw_i,mfw_j,score])
              similarity = best_score(score_list) //
                depending on similarity metrics
              mfw_i = merge(mfw_i, mfw_j) // merge the
                clusters' labels
              DID = merge_cluster(DID_i, DID_j)
              final_clusters = append(mfw_i)
        End for
    End for
    Return final_clusters
End
```

3.4 Illustrative example

Suppose we have the following dataset:

Dataset = [[G, I], [F, E, G, I, A], [H, A, D, C, E, G, F], [B, I, F], [H, B, E, F], [H, F], [C, D, H, G]], where A, B, C, D, E, F, G, H, and I represent preprocessed words in the document dataset.

By applying Algorithm 2 that was mentioned in the previous section with a minimum support of 0.2, the mined MFWs and their corresponding DID support values are shown in Table 2.

Table 2. Mined MFWs

MFWs	DID	Support
{A, G, E, F}	[0, 1, 2, 3, 4, 5, 6]	0.285714
{F, B}	[1, 2, 3, 4, 5]	0.285714
{G, D, C, H}	[0, 1, 2, 4, 5, 6]	0.285714
{F, E, H}	[1, 2, 3, 4, 5, 6]	0.285714
{G, I}	[0, 1, 2, 3, 6]	0.285714
{F, I}	[0, 1, 2, 3, 4, 5]	0.285714

When algorithm 3 is applied to the mined MFWs, the initial clusters obtained are shown in Table 3.

Table 3. Initial clusters

MFWs	DID	Support
{A, G, E, F}	[1, 2]	0.285714
{F, B}	[3]	0.285714
{G, D, C, H}	[2, 6]	0.285714
{F, E, H}	[4, 5]	0.285714
{G, I}	[0]	0.285714
{F, I}	[3]	0.285714

From Table 3, we can see that document 2 is assigned to two clusters (cluster 1 and cluster 3), and we can also find two equal clusters, i.e., two clusters contain document 3. By applying algorithms 4 and 5 for merging equal clusters and hard clustering, respectively, we obtain Table 4.

Table 4. Merging and hard clustering

MFWs	DID	Support
{A, G, E, F}	[1, 2]	0.285714
{I, F, B}	[3]	0.285714
{G, D, C, H}	[6]	0.285714
{F, E, H}	[4, 5]	0.285714
{G, I}	[0]	0.285714

By determining the number of clusters to 3, the final clustering process is depicted in Table 5.

Table 5. Final clustering

MFWs	DID	Support
{F, H, A, G, E}	[1, 2, 4,5]	0.285714
{I, F, G, B}	[3, 0]	0.285714
{G, D. C, H}	[6]	0.285714

4. RESULTS AND DISCUSSION

Using Python code running on a core i7 computer with 16GB of RAM, the suggested technique was examined on several datasets. The datasets and assessment metrics utilized for evaluating the effectiveness of the clustering method are described more thoroughly in the following sections.

4.1 Datasets

The proposed algorithm is implemented on two datasets CNN and OSAC [24]. Details of these datasets are briefly described in the following Table 6.

Table 6. Datasets description

Dataset	No. Classes	Total No.Doc.	Classes detail
CNN	6	4689	Business (836), Entertainment (474), Middle East (1462), Scitech (526), Sport (381), World (1010)
OSAC	10	22465	Business (3102), History (3233), family (3608), Religion (3171), Sport (2419), Health (2296), Astronomy (557), Law (944), Stories (762), Foods (2373)

4.2 Experiments and results evaluation

This paper uses precision, recall, and F-score as assessment metrics in clustering to evaluate the performance of a proposed technique. Precision measures the accuracy of document clustering by calculating the percentage of correctly assigned documents, while recall quantifies the completeness of the clustering. F-score balances precision and recall, indicating the accuracy and comprehensiveness of document clustering. A high F-score indicates high precision and recall, indicating the majority of documents are correctly assigned. These metrics are calculated using the following equations:

$$Recall(|CL_i, C_j|) = \frac{n_{ij}}{|CL_j|} \quad (8)$$

$$Precision(|CL_i, C_j|) = \frac{n_{ij}}{|C_j|} \quad (9)$$

$$f - score(CL_i, C_j) = \frac{2 * Recall(|\mathrm{CL_i}, \mathrm{C_j}|) * Precision(|CL_i, C_j|)}{Recall(|\mathrm{CL_i}, \mathrm{C_j}|) + Precision(|CL_i, C_j|)} \quad (10)$$

where, CL is the original class of the dataset. nij is the number of documents of class CLi that are presented in cluster Cj. |CLi| is the number of documents in class i and |Cj| is the number of documents in cluster j.

The experiments were repeated with different predetermined minimum support value (minsupp) for different number of clusters of each of the selected datasets i.e. (CNN_3, CNN_4, CNN_5, CNN_6, OSAC_3, OSAC_4, OSAC_5, and OSAC_10). Table 7 and Table 8 show the experiments details.

The selection of a similarity metric can significantly impact clustering results. Across various cluster sizes (3, 4, 5, and 6), for the CNN dataset, Euclidean distance, overlap similarity, and Jaccard index yielded similar outcomes. However, the best clustering results are obtained with Euclidean distance with different minimum support values for each cluster size (0.42 for CNN-3, varying between 0.38 and 0.40 for CNN-4 and CNN-6) as shown in Table 7.

In contrast, the best performance for the OSAC dataset is obtained using cosine similarity and occasionally Manhattan distance, with minimum support values ranging from 0.40 to 0.42. This led to the conclusion that the optimal similarity metric depends on the specific dataset. The results depicted in Table 8.

Table 7. F-measure for CNN dataset

No. Clusters	Similarity Metric	F-Measure for Minimum Support minSupp =					
		0.43	**0.42**	**0.41**	**0.40**	**0.39**	**0.38**
3	Euclidean	0.75	**0.76**	0.75	0.74	0.75	0.75
	Overlap	0.74	0.74	**0.75**	**0.75**	**0.75**	**0.75**
	Jaccard index	0.66	0.74	**0.75**	**0.75**	**0.75**	**0.75**
	Cosine	0.71	0.70	**0.72**	**0.72**	**0.72**	**0.72**
	Manhattan	0.70	0.70	0.70	0.71	**0.72**	**0.72**
4	Euclidean	0.78	0.78	0.79	**0.80**	0.79	**0.80**
	Overlap	0.72	0.77	0.78	0.78	0.78	**0.79**
	Jaccard index	0.68	0.73	0.77	0.78	**0.80**	0.77
	Cosine	0.70	0.68	0.70	0.70	0.70	**0.71**
	Manhattan	0.68	0.70	0.70	0.70	0.70	**0.71**
5	Euclidean	0.77	0.77	0.79	0.78	**0.80**	0.79
	Overlap	0.77	0.77	0.77	0.78	0.78	**0.80**
	Jaccard index	0.78	0.79	0.78	0.78	**0.80**	0.78
	Cosine	0.69	0.68	0.72	0.72	0.72	0.71
	Manhattan	0.70	0.70	0.70	0.70	0.70	0.70
6	Euclidean	0.77	0.77	0.79	0.78	**0.80**	0.79
	Overlap	0.76	0.77	0.79	**0.80**	0.79	0.79
	Jaccard index	0.71	0.71	0.71	0.77	0.73	0.71
	Cosine	0.71	0.71	0.71	0.72	0.72	0.72
	Manhattan	0.70	0.70	0.70	0.70	0.71	0.70

Table 8. F-measure for OSAC dataset

No. Clusters	Similarity Metric	F-Measure for Minimum Support minSupp =					
		0.45	**0.44**	**0.43**	**0.42**	**0.41**	**0.40**
3	Euclidean	0.70	0.70	0.70	0.70	0.71	**0.72**
	Overlap	0.71	0.70	**0.72**	**0.72**	**0.72**	**0.72**
	Jaccard index	0.70	0.70	0.70	0.71	**0.72**	0.71
	Cosine	0.77	0.72	0.78	0.78	0.77	**0.80**
	Manhattan	0.77	0.77	0.78	0.79	0.77	**0.80**
4	Euclidean	0.71	0.71	0.71	**0.72**	**0.72**	**0.72**
	Overlap	**0.72**	0.70	0.70	0.70	**0.72**	**0.72**
	Jaccard index	0.65	0.68	0.70	0.70	**0.71**	**0.71**
	Cosine	0.73	0.78	0.77	0.78	0.77	**0.80**
	Manhattan	0.62	0.62	0.64	0.66	**0.69**	**0.69**
5	Euclidean	0.66	0.66	0.69	0.70	**0.71**	**0.71**
	Overlap	0.60	0.68	0.66	0.69	0.70	**0.71**
	Jaccard index	0.65	**0.72**	0.67	0.69	0.71	0.70
	Cosine	0.77	0.73	0.77	**0.80**	0.77	**0.80**
	Manhattan	0.72	0.73	0.75	0.75	0.76	**0.78**
10	Euclidean	0.70	0.69	0.70	0.70	0.70	0.71
	Overlap	0.76	0.72	0.70	0.70	0.72	0.75
	Jaccard index	0.62	0.64	0.61	0.65	**0.68**	**0.68**
	Cosine	0.78	0.78	0.80	0.80	**0.81**	0.80
	Manhattan	0.61	0.65	0.69	0.71	**0.73**	**0.73**

Table 9. Comparison of the suggested approach and related work

Ref.	Author	Approach	Dataset	Evaluation Metric	
				Type	Percentage%
[18]	Al-sarrayrih and Al-Shalabi	FIHC	Built in house DS	F-measure	70
[19]	Daoud et al.	PSO-Kmeans	BBC	F-measure	33
			CNN		52
			OSAC		54
[20]	Alhawarat and Hegazi	LDA-Kmeans	BBC	F-measure	29
				Purity	58
			CNN	F-measure	59
				Purity	40
[21]	Sangaiah et al.	K-means	Set of doc.	Entropy	45
				F-measure	70
		Incremental K-means		Entropy	34
				F-measure	43
	The proposed approach	MFWs	CNN	F-measure	80
			OSAC		81

The results presented in Table 7 and Table 8 shows that the minimum support threshold can positively affect the results because it involves the process of choosing the discriminative features, so it clear that the low-level supports provide such features. Also, the Euclidean similarity metric mostly provides best results of F-score due to its appropriateness for the representation of maximal wordsets. The high value of F-score Indicates that the proposed algorithm's performance is good in terms of balancing precision and recall. Also, it suggests that the model is effectively identifying true positives while minimizing false positives and false negatives, implying that the behavior of the proposed algorithm has a good trade-off between precision (accuracy of positive predictions) and recall (sensitivity to true positives), resulting in reliable and balanced clustering outcomes.

The proposed algorithm is compared with the studies mentioned in Section 2, and the results showed that our algorithm achieved better clustering results than these works. When comparing it with the model presented in the study [18] it is found that using frequent itemsets for the clustering process leads to a voluminous number of itemsets, which in turn increases the search space and the computation time. These problems are overcome in our proposed algorithm by using maximal frequent wordsets, which shrink the search space and eliminate the computation time.

Daoud et al. [19] enhanced the selection of the initial clusters by combining K-means with PSO to scan the entire search space. The method supposes that each particle of the swarm represents the centroid of the clusters. The fitness function is minimized at each iteration by measuring the local best position and the global best position. In our proposed algorithm, as we mentioned before, the search space is reduced by using the user-defined threshold to mine MFWs, which will be the search space that will be scanned. Furthermore, the results achieved by us are better than those provided by this study for the same dataset.

K-means and LDA were used in literature [20] for clustering and topic modeling. The documents are represented as a bag of words, and then TF-IDF is applied to the document vector space to eliminate redundant data; afterwards, the data is normalized using Euclidean distance. As a last step, k-means was used for document clustering. On the other hand, topics are modeled using the same normalized dataset as an input to the LDA. The result achieved by this study is less than that

achieved by our model for the CNN dataset. Clustering Arabic documents was accomplished using either unsupervised or semi-supervised approaches suggested by A. K. Sangaiah [21]. K-means or incremental K-means were used for these approaches, and the clustering results were evaluated using the F-measure and entropy. The comparison is figured out Table 9.

It is a noteworthy accomplishment to obtain an F-score value in Arabic document clustering of more than 80%, as this shows that the suggested method performs well in terms of recall and precision. Such a score indicates:

High Accuracy: An F-score of more than 80% is a strong indication of the system's accuracy in correctly clustering Arabic documents because the F-score is a metric that strikes a balance between precision and recall.

Efficient Clustering: A high F-score indicates that the suggested method efficiently clusters related Arabic documents while reducing false positives and misclassifications.

Robustness: The robustness and generalizability of the algorithm are demonstrated by its consistent achievement of an F-score above 80% in a variety of datasets and circumstances.

Comparative Advantage: The proposed algorithm performs better than other algorithms that usually yield F-score values.

5. CONCLUSIONS

In this paper, we introduce a novel approach for clustering Arabic documents. The method utilizes the maximal frequent word sets discovered by the Fpmax algorithm to achieve effective clustering results. This technique addresses the challenge of high-dimensionality datasets by employing MFWs for efficient dimension reduction. The evaluation of the proposed method is done on two benchmark datasets, CNN and OSAC. The experiments revealed different cluster configurations by tuning the specified threshold and the number of clusters. Hence, a diverse set of clustering outcomes result. These promising results contribute to the field of ANLP and encourage further exploration of diverse techniques for enhanced Arabic document clustering. In the future work, we aim to apply other data mining techniques, evaluate their impact for clustering purposes, and evaluate the clustering of Arabic documents.

REFERENCES

[1] Salloum, S.A., AlHamad, A.Q., Al-Emran, M., Shaalan, K. (2018). A survey of Arabic text mining. Intelligent natural language processing: Trends and Applications, 417-431. https://doi.org/10.1007/978-3-319-67056-0_20

[2] Al Sbou, A.M., Hussein, A., Talal, B., Rashid, R.A. (2018). A survey of Arabic text classification models. International Journal of Electrical and Computer Engineering, 8(6): 4352-4355. https://doi.org/10.11591/ijece.v8i6.pp.4352-4355

[3] Zitouni, I. (2014). Natural Language Processing of Semitic Languages. Heidelberg: Springer.

[4] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. arXiv preprint arXiv:1707.02919. https://doi.org/10.48550/arXiv.1707.02919

[5] Witten, I.H., Frank, E., Hall, M.A., Pal, C.J. (2017). Moving on: Applications and beyond. Data Mining, 503-532. https://doi.org/10.1016/B978-0-12-804291-5.00013-1

[6] Chen, Z.L. (2022). Research and application of clustering algorithm for text big data. Computational Intelligence and Neuroscience, 2022: 7042778. https://doi.org/10.1155/2022/7042778

[7] Afzali, M., Kumar, S. (2019). Text document clustering: issues and challenges. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 263-268. https://doi.org/10.1109/COMITCon.2019.8862247

[8] Xu, D., Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2: 165-193. https://doi.org/10.1007/s40745-015-0040-1

[9] Hamdad, L., Benatchba, K. (2021). Association rules mining. SN Computer Science, 2: 449. https://doi.org/10.1007/s42979-021-00819-x

[10] Ai, D., Pan, H., Li, X. Gao, Y.X., He, D. (2018). Association rule mining algorithms on high-dimensional datasets. Artificial Life and Robotics. 23: 420-427. https://doi.org/10.1007/s10015-018-0437-y

[11] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., Yiu, T. (2005). MAFIA: A maximal frequent itemset algorithm. IEEE Transactions on Knowledge and Data Engineering, 17(11): 1490-1504. https://doi.org/10.1109/TKDE.2005.183

[12] Grahne, G., Zhu, J. (2003). High Performance Mining of Maximal Frequent Itemsets Gösta. https://api.semanticscholar.org/CorpusID:13264349.

[13] Wang, H. (2015). MaxMining: A novel algorithm for mining maximal frequent itemset. Applied Mechanics and Materials, 713: 1765-1768. https://doi.org/10.4028/www.scientific.net/amm.713-715.1765

[14] Gouda, K., Zaki, M.J. (2001). Efficiently mining maximal frequent itemsets. In Proceedings 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, pp. 163-170. https://doi.org/10.1109/ICDM.2001.989514

[15] Hamid, Z., Khafaji, H.K. (2021). Classification of Arabic documents depending on maximal frequent itemsets. Journal of Physics: Conference Series, 1804(1): 012009. https://doi.org/10.1088/1742-6596/1804/1/012009

[16] Khafaji, H.K. (2021). A new algorithm for extracting textual maximal frequent itemsets from Arabic Documents. Journal of Physics: Conference Series, 1773(1): 012012. https://doi.org/10.1088/1742-6596/1773/1/012012

[17] Salman, K.A., Khafaji, H.K. (2022). Arabic document clustering: A survey. In 22022 4th International Conference on Current Research in Engineering and Science Applications (ICCRESA), Baghdad, Iraq, pp. 59-64. https://doi.org/10.1109/ICCRESA57091.2022.10352511

[18] Al-sarrayrih, H., Al-Shalabi, R. (2009). Clustering arabic documents using frequent itemset-based hierarchical clustering with an n-grams. Microsoft Word - 520-Paper-Haytham (zuj.edu.jo).

[19] Daoud, A.S., Sallam, A., Wheed, M.E. (2018).

Improving Arabic document clustering using K-means algorithm and Particle Swarm Optimization. In 2017 Intelligent Systems Conference (IntelliSys), London, UK, pp. 879-885.https://doi.org/10.1109/IntelliSys.2017.8324233

[20] Alhawarat, M., Hegazi, M. (2018). Revisiting k-means and topic modeling, a comparison study to cluster Arabic documents. IEEE Access, 6: 42740-42749. https://doi.org/10.1109/ACCESS.2018.2852648

[21] Sangaiah, A.K., Fakhry, A.E., Abdel-Basset, M., El-henawy, I. (2019). Arabic text clustering using improved clustering algorithms with dimensionality reduction. Cluster Computing, 22: 4535-4549. https://doi.org/10.1007/s10586-018-2084-4

[22] Mamoun, R., Ahmed, M. (2016). Arabic text stemming: Comparative analysis. In 2016 Conference of Basic Sciences and Engineering Studies (SGCAC), Khartoum, Sudan, pp. 88-93. https://doi.org/10.1109/SGCAC.2016.7458011

[23] Larkey, L.S., Ballesteros, L., Connell, M.E. (2007). Light Stemming for Arabic Information Retrieval. In: Soudi, A., Bosch, A.v., Neumann, G. (eds) Arabic Computational Morphology, pp. 221-243. https://doi.org/10.1007/978-1-4020-6046-5_12

[24] Saad, M.K., Ashour, W. (2010). OSAC: Open source Arabic corpora. In e 6th International Conference on Electrical and Computer Systems (EECS'10), Lefke, North Cyprus. https://doi.org/10.13140/2.1.4664.9288