Clustering Web Documents based on Efficient Multi-Tire Hashing Algorithm for Mining Frequent Termsets

Noha Negm
Math. and Computer Science Dept.
Faculty of Science, Menoufia University
Shebin El-Kom, EGYPT

Passent Elkafrawy
Math. and Computer Science Dept.
Faculty of Science, Menoufia University
Shebin El-Kom, EGYPT

Mohamed Amin
Math. and Computer Science Dept.
Faculty of Science, Menoufia University
Shebin El-Kom, EGYPT

Abdel Badeeh M. Salem
Computer Science Dept. Faculty of Computers and Information, Ain Shams University
Cairo, EGYPT

***Abstract*—Document Clustering is one of the main themes in text mining. It refers to the process of grouping documents with similar contents or topics into clusters to improve both availability and reliability of text mining applications. Some of the recent algorithms address the problem of high dimensionality of the text by using frequent termsets for clustering. Although the drawbacks of the Apriori algorithm, it still the basic algorithm for mining frequent termsets. This paper presents an approach for Clustering Web Documents based on Hashing algorithm for mining Frequent Termsets (CWDHFT). It introduces an efficient Multi-Tire Hashing algorithm for mining Frequent Termsets (MTHFT) instead of Apriori algorithm. The algorithm uses new methodology for generating frequent termsets by building the multi-tire hash table during the scanning process of documents only one time. To avoid hash collision, Multi Tire technique is utilized in this proposed hashing algorithm. Based on the generated frequent termset the documents are partitioned and the clustering occurs by grouping the partitions through the descriptive keywords. By using MTHFT algorithm, the scanning cost and computational cost is improved moreover the performance is considerably increased and increase up the clustering process. The CWDHFT approach improved accuracy, scalability and efficiency when compared with existing clustering algorithms like Bisecting K-means and FIHC.**

Keywords—Document Clustering; Knowledge Discovery; Hashing; Frequent termsets; Apriori algorithm; Text Documents; Text Mining; Data Mining

I. INTRODUCTION

This With the recent explosive growth of the amount of content on the Internet, it has become increasingly difficult for users to find and utilize information and for content providers to classify and index documents. Hundreds or thousands of results for a search are often returned by traditional web search engines, which is time consuming for users to browse. On-line libraries, search engines, and other large document repositories are growing so rapidly that it is difficult and costly to categorize every document manually. In order to deal with these problems, researchers look toward automated methods of working with web documents so that they can be more easily browsed, organized, and indexed with minimal human intervention. To deal with the problem of information overload on the Internet, Clustering and Classification considered the useful and active areas of machine learning research that promise to overcome this problem [1].

Document clustering is known as an unsupervised and automatic organizing text documents into meaningful clusters or group, In other words, the documents in one cluster share the same topic, and the documents in different clusters represent different topics. It is unlike document classification since there is no training stage by using labeled documents. Document clustering has been studied intensively because of its wide applicability in areas such as *Web Mining*, *Search Engines*, *Information Retrieval*, and *Topological Analysis*.

The high dimensionality of the feature space considered a major characteristic of document clustering algorithms, which imposes a big challenge to the performance of clustering algorithms. Next challenge is that not all features are important for document clustering, some of the features may be redundant or irrelevant and some may even misguide the clustering result [2].

Hierarchical and Partitioning methods are categorized as the essentially two algorithms into the clustering technique [3-8]. K-means and its variants are the most well-known partitioning methods that create a flat, non-hierarchical clustering consisting of k clusters. The Bisecting k-means algorithm first selects a cluster to split, and then employs basic k-means to create two sub-clusters, repeating these two steps until the desired number k of clusters is reached [7]. Steinbach in [4] showed that the Bisecting k-means algorithm outperforms basic k-means as well as agglomerative hierarchical clustering in terms of accuracy and efficiency.

A hierarchical clustering method works by grouping data objects into a tree of clusters. These methods can further be classified into agglomerative and divisive hierarchical

clustering depending on whether the hierarchical decomposition is formed in a bottom-up or top down fashion [8]. Steinbach in [9] showed that Unweighted Pair Group Method with Arithmetic Mean (UPGMA) is the most accurate one in agglomerative category.

Both hierarchical and partitioning methods do not really address the problem of high dimensionality in document clustering. Frequent itemset-based clustering method is shown to be a promising approach for high dimensionality clustering in recent literature [10-24]. It reduces the dimension of a vector space by using only frequent itemsets for clustering. The frequent term-based text clustering is based on the following ideas: (1) Frequent terms carry more information about the "cluster" they might belong to; (2) Highly co-related frequent terms probably belong to the same cluster.

Finding frequent itemsets is an important data mining topic, and it was originated from the association rule mining of transaction dataset. The main drawback of frequent itemsets is they are very large in number to compute or store in computer. The very first well known algorithm for frequent itemset generation is Apriori algorithm [10]. It works on the principle of Apriori property, which states that the subset of any frequent itemset is also frequent. Apriori algorithm adopts layer by layer search iteration method to mine association rules. The Apriori algorithm suffers from the following two problems: 1) candidate generation and 2) repeated number of scans.

In this paper, a CWDHFT approach for clustering web documents based on a hashing mining algorithm is proposed. it introduces an efficient Multi-Tire Hashing algorithm (MTHFT) to discover frequent termsets from web text documents. It overcomes the drawbacks of the Apriori algorithm by using new methodology for generating frequent termsets. The multi-tire hash table is building during the scanning process of documents only one time. To avoid hash collision, multi-tire technique is utilized in MTHFT algorithm. The generated set of frequent termsets with varying length is used in the clustering process. Based on the generated frequent termset the documents are partitioned and the clustering occurs by grouping the partitions through the descriptive keywords.

The organization of the paper is as follows. Section 2 discusses the related work to our approach on. In Section 3, we describe the proposed CWDHFT approach and MTHFT algorithm. Section 4 discusses about the results obtained from the comparison of the CWDHFT approach with two other clustering algorithms in this field. Section 5 concludes the work proposed.

II. Related Work

There are many works in the literature that discuss about text clustering algorithms. They address the special characteristics of text documents and use the concept of frequent termsets for the text clustering.

Reference [9], they proposed a new criterion for clustering transactions using frequent itemsets, instead of using a distance function. In principle, this method can also be applied to document clustering by treating a document as a transaction; however, the method does not create a hierarchy for browsing. The novelty of this approach is that it exploits frequent itemsets (by applying Apriori algorithm) for defining a cluster, organizing the cluster hierarchy, and reducing the dimensionality of document sets.

The FTC and HFTC are proposed in [14]. The basic motivation of FTC is to produce document clusters with overlaps as few as possible. FTC works in a bottom-up fashion. As HFTC greedily picks up the next frequent itemset to minimize the overlapping of the documents that contain both the itemset and some remaining itemsets. The clustering result depends on the order of picking up itemsets, which in turn depends on the greedy heuristic used. The weakness of the HFTC algorithm is that it is not scalable for large document collections.

To measure the cohesiveness of a cluster directly using frequent itemsets, the FIHC algorithm is proposed in [15]. Two kinds of frequent item are defined in FIHC: global frequent item and cluster frequent item. However, FIHC has three disadvantages in practical application: first, it cannot solve cluster conflict when assigning documents to clusters. Second, after a document has been assigned to a cluster, the cluster frequent items were changed and FIHC does not consider this change in afterward overlapping measure. Third, in FIHC, frequent itemsets is used merely in constructing initial clusters.

Frequent Term Set-based Clustering (FTSC) algorithm is introduced in [16]. FTSC algorithm used the frequent feature terms as candidate set and does not cluster document vectors with high dimensions directly. The results of the clustering texts by FTSC algorithm cannot reflect the overlap of text classes. But FTSC and the improvement FTSHC algorithms are comparatively more efficient than K-Means algorithm in the clustering performance.

Clustering based on Frequent Word Sequence (CFWS) is proposed in [17]. CFWS uses frequent word sequence and K-mismatch for document clustering. By using the CFWS there are overlaps in the final clusters. With K-mismatch, frequent sequences of candidate clusters are used to produce final clusters. Document Clustering Based on Maximal Frequent Sequences (CMS) is proposed in [18]. The basic idea of CMS is to use Maximal Frequent Sequences (MFS) of words as features in Vector Space Model (VSM) for document representation and then K-means is employed to group documents into clusters. CMS is rather a method concerning feature selection in document clustering than a specific clustering method. Its performance completely depends on the effectiveness of using MFS for document representation in clustering, and the effectiveness of K-means.

A frequent term based parallel clustering algorithm which could be employed to cluster short documents in very large text database is presented in [22]. A semantic classification method is also employed to enhance the accuracy of clustering. The experimental analysis proved that the algorithm was more precise and efficient than other clustering algorithms when clustering large scale short documents. In addition, the algorithm has good scalability and also could be employed to process huge data.

The document clustering algorithm on the basis of frequent term sets is proposed in [23]. Initially, documents were denoted as per the Vector Space Model and every term is sorted in accordance with their relative frequency. Then frequent term sets can be mined using frequent-pattern growth (FP growth). Lastly, documents were clustered on the basis of these frequent term sets. The approach was efficient for very large databases, and gave a clear explanation of the determined clusters by their frequent term sets. The efficiency and suitability of the proposed algorithm has been demonstrated with the aid of experimental results.

Reference [25] a hierarchical clustering algorithm using closed frequent itemsets that use Wikipedia as an external knowledge to enhance the document representation is presented. Firstly, construct the initial clusters from the generalized closed frequent itemsets. Then used the two methods TF-IDF and Wikipedia as external knowledge, to remove the document duplication and construct the final clusters. The drawback in this approach is that it might not be of great use for datasets which do not have sufficient coverage in Wikipedia.

A Frequent Concept based Document Clustering (FCDC) algorithm is proposed in [26]. It utilizes the semantic relationship between words to create concepts. It exploits the WordNet ontology in turn to create low dimensional feature vector which allows us to develop an efficient clustering algorithm. It used a hierarchical approach to cluster text documents having common concepts. FCDC found more accurate, scalable and effective when compared with existing clustering algorithms like Bisecting K-means, UPGMA and FIHC.

To the best of our knowledge, all the previous researchers depend on the Apriori algorithm and their improvements for generating the frequent termsets from text documents. Moreover they don't address the improvements in the execution time as the major factor in the mining process.

III. Clustering Web Documents Based on Hshing Frequent Termsets

The proposed web document clustering approach based on frequent termsets CWDHFT is shown in Fig.1. The main characteristic of the approach is that it introduces a new mining algorithm for generating frequent termsets to overcome the drawbacks of the Apriori algorithm. Moreover it speeds up the mining and clustering process. CWDHFT consists of the four main stages:

- Document Preprocessing
- Mining of Frequent Termsets
- Document Clustering
- Post Processing

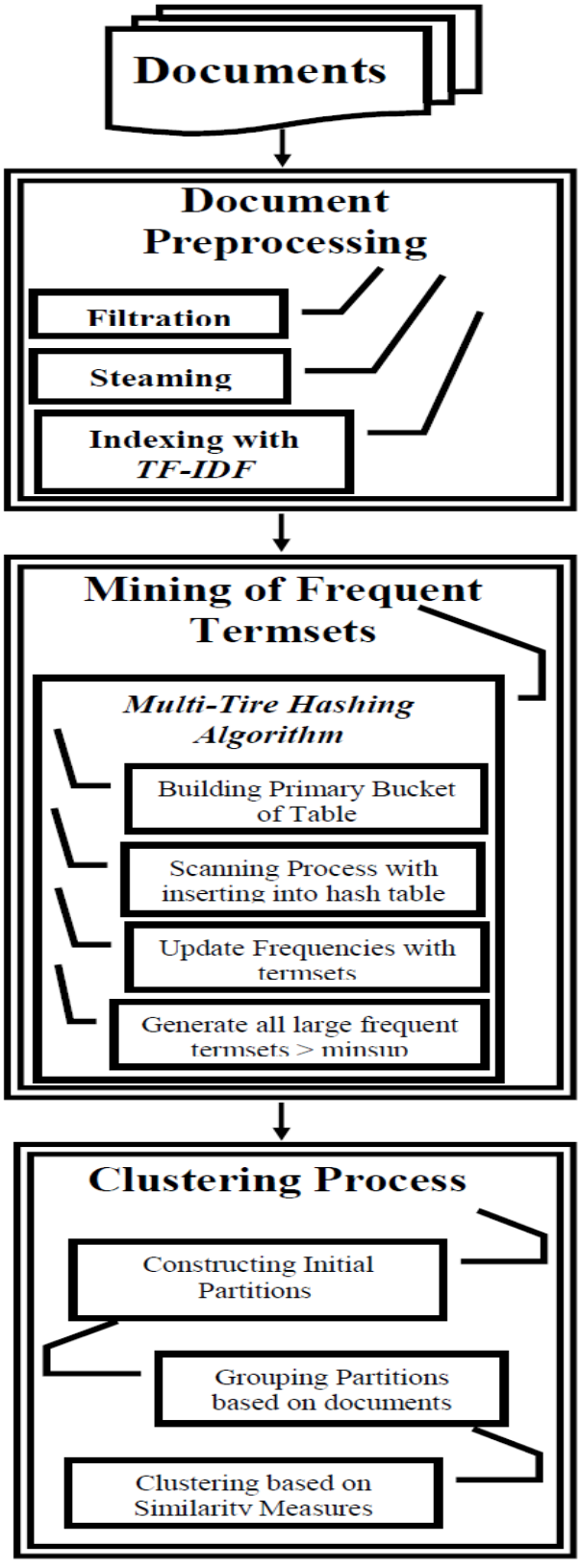

Fig. 1. CWDHFT approach.

A. Document Preprocessing Stage

Our approach employs several preprocessing steps including stop words removal, stemming on the document set and indexing by applying TF*ID:

- Stop words removal: Stop-words are words that from non-linguistic view do not carry information such as (a, an, the, this, that, I, you, she, he, again, almost, before, after). Stop-words removing is to remove this non-information bearing words from the documents and reduce noise. One major property of stop-words is that they are extremely common words. The explanation of the sentences still held after these stop-words are removed. To organize large corpus, removing the stop words affords the similar advantages. Firstly it could save huge amount of space. Secondly it helps to deduce the noises and keep the core words, and it will make later processing more effective and efficient.

- *Stemming:* Removes the affixes in the words and produces the root word known as the stem. Typically, the stemming process will be performed so that the words are transformed into their root form. *For example* connected, connecting and connection will be transformed into connect. A good stemmer should be able to convert different syntactic forms of a word into its normalized form, reduce the number of index terms, save memory and storage and may increase the performance of clustering algorithms to some extent; meanwhile it should try stemming. Porter Stemmer [27] is a widely applied method to stem documents. It is compact, simple and relatively accurate. It does not require creating a suffix list before applied. In this paper, we apply Porter Stemmer in our preprocessing.

- *TF*IDF:* In many weighting schemes the weights as in (1) are the product of two factors, the *term frequency* (*tf*) and the *inverse document frequency* (*idf*) [28]:

$$w_{i,j} = tf_{i,j} * idf_i \qquad (1)$$

- The term frequency is a function of the number of occurrences of the particular word in the document divided by the number of words in the entire document. A word appearing frequently in the text is thus considered more important to describe the content than a word appearing less often. The inverse document frequency models the distinguishing power of the word in the text set; the fewer documents that contain the word the more information about the text in the text set it gives. There are many variants of the *idf*-measure. A simple example is as in (2):

$$idf_i = \log\left({}^{N}/_{Nt_j}\right) \qquad (2)$$

where Nt_j denotes the number of documents in collection N in which t_j occurs at least once. Once a weighting scheme has been selected, automated indexing can be performed by simply selecting the top K of words that satisfy the given weight constraints for each document. The major advantage of an automated indexing procedure is that it reduces the cost of the indexing step.

B. Mining of Frequent Termset Stage

The goal of frequent termset mining is to discover sets of terms that frequently co-occur in the document. The problem is non-trivial in text documents because the documents can be very large, consisting of many distinct terms, and contain interesting termsets of high cardinality. Although the drawbacks of the Apriori algorithm, it still use for generating the frequent termsets that used in the document clustering.

In order to speed up the mining process as well as to address the scalability with different documents regardless of their sizes, we introduce a new algorithm called Multi-Tire Hashing Frequent Termsets algorithm (MTHFT). It is basically different from all the previous algorithms since it overcomes the drawbacks of Apriori algorithm by employing the power of data structure called Multi-Tire Hash Table. Moreover it uses new methodology for generating frequent termsets by building the hash table during the scanning of documents only one time consequently, the number of scanning on documents decreased.

1) Hash Table: The hash table is a data structure that speeds up searching for information by a particular aspect of that information, called a key. The idea behind the hash table is to process the key with a function that returns a hash value; that hash value then determines where in the data structure the terms will (or probably will) be stored. The hash tables can provide constant time $O(1)$ lookup on average, regardless of the number of terms in the table. To avoid hash collision, Multi Tire technique is utilized in this proposed hashing algorithm. It consists of two major components: bucket array and hash function.

a) Bucket Array: A bucket array for a hash table is an array U of size R, where each cell of U is thought of as a " Bucket " and the integer R defines the capacity of the array. If the keys are integers well distributed in the range [0, R-1], this bucket array is all that is needed. An element e with key v is simply inserted into the bucket U[v].

b) Hash Function: The second part of a hash table structure is a function, h, that maps each key v in our dictionary to an integer in the range [0, R-1], where R is the capacity of the bucket array for this table. The main idea of this approach is to use the hash function value, h(v), as an index into our bucket array, U, instead of the key v. That is, we store the item (v, e) in the bucket U [h(v)]. The benefit of the hash function is to reduce the range of array indices that need to be handled. The Division Method (The mod function h(v) = v mod R) used for creating hash function h(v) in hash table.

2) Multi-Tire Hashing Frequent Termsets Algorithm: The MTHFT algorithm as shown in Fig. 2 employs the following two main steps:

- Based on the number of the English alphabet letters R =26, a dictionary table constructed as shown in Table 1

and gives each character a unique numeric number from 0 to 25.

MTHFT Algorithm:

T_m: Set of all termsets for each document *d*
C_m: Candidate termsets for each document *d*
I_k : Frequent termsets of size *k*.

***Input*:** All Text documents.
Process logic: Building Multi-Tire Hash Table and Finding the frequent termsets.
Output: Generating the frequent termsets.

for each document $d_m \in D$ *do begin*
$T_m = \{ t_i : t_i \in d_m , 1 \le i \le n \}$
for each term $t_i \in T_m$ *do*
$h(t_i) = t_i \bmod N;$
$t_i.count++;$
// insert each term in hash table
end
C_k = *all combinations of* $t_i \in d_m$
$C_m = subset(C_k , d_m)$;
for each candidate $c_j \in C_m$ *do*
$h(c_j) = c_j \bmod N;$
$c_j.count++;$
// insert each candidate in hash table
end
end
for given s= minsup in hash table do
$I_1 = \{t \mid t.count \ge minsup \}$
$I_k = \{c \mid c.count \ge minsup, k \ge 2\}$
end

Fig. 2. The MTHFT algorithm.

- There are also two main processes for a dynamic multi-tire hash table: the building process and the scanning process.

a) The Building Process: In the dynamic hash table, a primary bucket is only built at the first. It depends on the number of the English alphabet letter R, not on the number of all terms as shown in Table 2. Their locations in the hash table are determined using the division method of hash function.

TABLE I. THE DICTIONARY TABLE FOR THE ENGLISH ALPHABET LETTERS

Dictionary Table	
Letters	**Location**
A	0
B	1
C	2
D	3
E	4
F	5
.....	.....
V	21
W	22
X	23
Y	24
Z	25

TABLE II. THE PRIMARY BUCKET OF MULTI-TIRE HASH TABLE

***A*(0)**	***B*(1)**	***C*(2)**	**.....**	***Y*(24)**	***Z*(25)**

For example, the alphabet letter E takes the unique numeric number 4 in the dictionary table, and their location is determined by applying the hash function so that its location is also 4 and so on.

b) The Scanning Process: After building a primary bucket, each document is scanned only once as follows:

- For first document, select all terms and make all possible combinations of terms after that determine their locations in the dynamic hash table using the hash function h(v). in hash table, insert them in their locations with their frequencies.
- For each document, all terms and termsets are inserted in a hash table and their frequencies are updated, the process continues until there is no document in the collection.
- Save the multi-tire hash table into secondary storage media for further processing.
- Insert different minimum support values and scan the multi-tire hash table to determine the large frequent termsets that satisfy each threshold support value without redoing the mining process again.
- Insert the generated large frequent termsets in the Clustering process.

3) The advantages of MTHFT Algorithm: The MTHFT algorithm has many advantages summarized as follows:

- Provides facilities to avoid unnecessary scans to the documents, which minimize the I/O. Where the scanning process occurs on the hash table instead of whole documents compared to Apriori algorithm
- The easy manipulations on hash data structure and directly computing frequent termsets are the added advantages of this algorithm, moreover the fast access and search of data with efficiency.
- MTHFT shows better performance in terms of time taken to generate frequent termsets when compared to Apriori algorithm. Furthermore, it permits the end user to change the threshold support and confidence factor without re-scanning the original documents since the algorithm saves the hash table into secondary storage media
- The main advantage of this algorithm is that, it is scalable with all types of documents regardless of their sizes.
- Depending on the multi-tire technique in building the primary bucket, each bucket can store only a single element then we cannot associate more than one term with a single bucket, which is a problem in the case of collisions.

C. Documents Clustring Stage

Document clustering algorithm based on frequent termsets considered a keyword-based algorithm which picks up the core

words with specific criteria and groups the documents based on these keywords. this approach includes three main steps:

- Picking out all frequent termsets
- Constructing partitions
- Clustering documents

1) Picking out all Frequent Termsets: The Multi-Tire Hashing algorithm is used in the previous step to find out the large frequent termsets furthermore to speeding up the mining process. it have ability to determine large frequent termsets at different minimum support threshold values without redoing the mining process again. Therefore, we can pick out different sets of frequent termsets in the clustering process easily. We start with a set of 2-large frequent termsets.

2) Constructing Partitions: Constructing partitions include two sub steps: constructing initial partitions and merging non-overlapping partitions.

a) Constructing initial partitions: initially, we sort the set of 2-large frequent termsets in descending order in accordance with their support level as in (3):

$$Sup(lf_1) > Sup(lf_2) > \ldots\ldots\ldots\ldots\ldots\ldots Sup(lf_k) \quad (3)$$

Then, the first 2-large frequent termsets from the sorted list is selected. Afterward, an initial partition P1 which contains all the documents including the both termsets is constructed. Next, we take the second 2-large frequent termsets whose support is less than the previous one to form a new partition P2. This partition is formed by the same way of the partition P1 and takes away the documents that are in the initial partition this avoid the overlapping between partitions since each document keeps only within the best initial partition. This procedure is repeated until every 2-large frequent termsets moved into partition P(i).

b) Merging non-overlapping partitions: in this step, all partitions that contain the similar documents are merged into one partition. The benefit of this step is reducing the number of resulted partitions.

3) Clustering Documents: In this step, we don't require to pre-specified number of clusters we have a set of non-overlapping partitions P(i) and each partition has a number of documents D. We first identify the words that used for constructing each partition P(i) which called labeling Words Ld [W(i)]. The labeling words are obtained from all 2-large frequent termsets that contained in each partition. For each document, Derived keywords Vd [W(i)] are obtained from taking into account the difference words between the top weighted frequent words for each document with the labeling words. Subsequently the total support of each derived word is computed within the partition.

The set of words satisfying the partition threshold (the percentage of the documents in partition P(i) that contains the termset) are formed as Descriptive Words Pw [c(i)] of the partition P(i). Afterward, we compute the similarity of each document in the partitions with respect to the descriptive words. The definition of the similarity measure plays an importance role in obtaining effective and meaningful clusters. The similarity between two documents Sm is computed as in [8]. Based on the similarity measure, a new cluster is formed from the partitions i.e. each cluster will contain all partitions that have the similar similarity measures.

D. Post processing

Includes the major applications in which the document clustering is used, for example, the recommendation application which uses the results of clustering for recommending news articles to the users.

IV. Expermental Results and Performance Evaluation

Our experiments have been conducted on a personal computer with a 2.50 GHz CPU and 6.00 GB RAM and we have implemented the proposed clustering approach CWDHFT using C#.net language. To evaluate the effectiveness of proposed MTHFT algorithm in the mining process, this section presents the result comparisons between our MTHFT algorithm and Apriori algorithm. Moreover, several popular hierarchical document clustering algorithms Bisecting K-means and FIHC are compared with our CWDHFT approach for clustering web documents. The rest of this section first describes the characteristics of the datasets, then explains the evaluation measures, and finally presents and analyzes the experiment results.

A. Datasets

We have used the largest datasets Reuters-21578 to exam the efficiency and scalability of our algorithm [29]. The Reuters-21578 collection is distributed in 22 files. Each of the first 21 files (reut2-000.sgm through reut2-020.sgm) contain 1000 documents, while the last (reut2- 021.sgm) contains 578 documents. Documents were marked up with SGML tags. There are 5 categories Exchanges, Organizations, People, Places and Topics in the Reuters dataset and each category has again sub categories in total 672 sub categories. We have collected the TOPIC category sets to form the dataset. The TOPICS category set contains 135 categories. From these documents we collect the valid text data of each category by extracting the text which is in between <BODY> ,</BODY> and placed in a text document and named it according to the topic. From Reuters, we have considered 5000 documents the our datasets.

B. Evaluation Methods

The F-measure, as the commonly used external measurement, is used to evaluate the accuracy of our clustering algorithms. F-measure is an aggregation of Precision and Recall concept of information retrieval. Recall is the ratio of the number of relevant documents retrieved for a query to the total number of relevant documents in the entire collection as in (4):

$$Recall\left(K_i, C_j\right) = \frac{n_{ij}}{|K_i|} \quad (4)$$

Precision is the ratio of the number of relevant documents to the total number of documents retrieved for a query as in (5):

$$Precision\left(K_i, C_j\right) = \frac{n_{ij}}{|C_j|} \quad (5)$$

While F-measure for cluster $C_j$and class K_i is calculated as in (6):

$$F\left(K_i, C_j\right) = \frac{2 * Recall\left(K_i, C_j\right) * Precision\left(K_i, C_j\right)}{Recall\left(K_i, C_j\right) + Precision\left(K_i, C_j\right)} \quad (6)$$

where n_{ij} is the number of members of class K_i in cluster C_j. $|C_j|$ is the number of members of cluster $C_j$and $|K_i|$ is the number of members of class K_i .

The weighted sum of all maximum F-measures for all natural classes is used to measure the quality of a clustering result *C*. This measure is called the overall F-measure of C, denoted F(C) is calculated as in (7):

$$F(C) = \sum_{K_i \in K} \frac{|K_i|}{|D|} max_{C_j \in C}\{F(K_i, C_j)\} \quad (7)$$

where *K* denotes all natural classes; *C* denotes all clusters at all levels; $|K_i|$ denotes the number of documents in natural class K_i; and $|D|$ denotes the total number of documents in the dataset. The range of F(C) is [0,1]. A large F(C) value indicates a higher accuracy of clustering.

C. Experimental Results

In this section, we evaluate the performance of our MTHFT algorithm in terms of the efficiency and scalability of finding frequent termsets, moreover the accuracy and efficiency of CWDHFT approach.

1) Evaluation of MTHFT algorithm for finding frequent termsets: We evaluated our MTHFT algorithm of finding frequent termsets in terms of its efficiency and scalability. In our experiment, we compared the MTHFT algorithm with Apriori algorithm, which is the most representative frequent itemset mining algorithm although of its drawbacks. As we know, the efficiency of Apriori is sensitive to the minimum support level and the size of documents. When the minimum support is decreased, the runtime of Apriori increases as there are more frequent itemsets. Moreover, when the size of documents become very large, most time is consuming in the multiple scanning on the documents and generating frequent termsets at different minimum support. In MTHFT algorithm, the time is consumed in building a hash table only one time. After saving the hash table there is no time consuming in generating new different frequent termsets at different minimum support threshold.

Fig. 3 shows a comparison of results of Apriori and MTHFT algorithm for various values of minimum support thresholds at the Reuters datasets. Support is taken as X-axis and the execution time taken to find the frequent termsets is taken as Y-axis. We first chose small value of minimum support equals to 30% then compute the execution time for both algorithms.

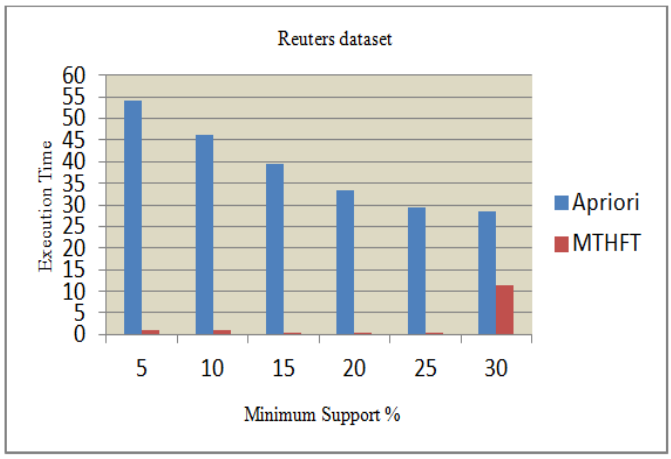

Fig. 3. Time comparison between Apriori and MTHFT algorithms on Reuters dataset.

From the chart, it can be seen that the execution time taken for MTHFT algorithm decreases as the minimum support increased in comparable to Apriori algorithm. At MTHFT algorithm, the whole execution time is consumed in building the hash table the first time. When entering new minimum support, there is no time consumed, however, the time is taken for searching the hash table. We noticed that the execution time decreases as the minimum support increased in comparable to Apriori algorithm. In Apriori algorithm, each time entering a new minimum support it required to redo the mining process from the beginning. We conclude that MTHFT is significantly more efficient than Apriori algorithm in all cases specially for large documents since the complexity of finding the frequent termsets is lower than Apriori.

To examine the scalability of MTHFT algorithm, we create a larger dataset from Reuters. W duplicated the files in Reuters until we get 10000 documents. Fig. 4 illustrates the results of applying MTHFT algorithm and Apriori on different sizes of documents of Reuters at small value of minimum support threshold 15% to ensure that the generated frequent termset in both algorithms is approximately the same. We noticed that MTHFT algorithm is about two to three times faster than Apriori and performs better with large number of documents in contrast Apriori algorithm.

Evaluation of the text clustering algorithm: For a comparison with CWDHFT approach, we also executed Bisecting k-means and FIHC on the same documents. We chose Bisecting k-means because it has been reported to produce a better clustering result consistently compared to k-means and agglomerative hierarchical clustering algorithms. FIHC is also chosen because it uses frequent word sets. For a fair comparison, we did not implement Bisecting k-means and FIHC algorithms by ourselves. We downloaded the CLUTO toolkit [30] to perform Bisecting k-means, and obtained FIHC [31] from their author.

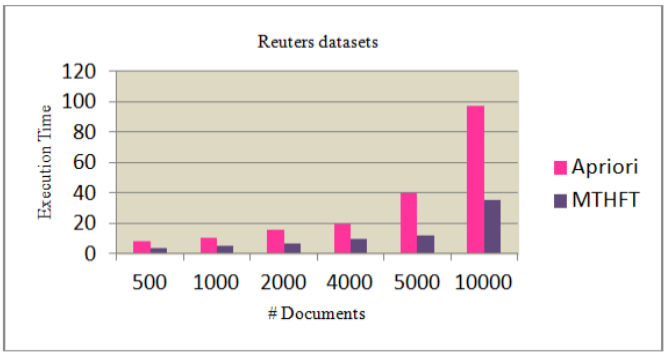

Fig. 4. Time comparison between Apriori and MTHFT algorithms on different sizes of Reuters at MinSup=15%.

Fig. 5 shows the comparison between all the three clustering approaches based on the overall F-measure values with different numbers of clusters. Our CWDHFT approach outperforms all other approaches in terms of accuracy, it has better F-measure because it uses a better model for text documents.

Many experiments were conducted to exam the efficiency of CWDHFT approach. Fig. 6 compares the execution time of CWDHFT approach with FIHC and Bisecting K-means on different sizes of documents of Reuters. The minimum support is set to 15% to ensure that the accuracy of all produced clustering are approximately the same. The number of documents is taken as X-axis and the time taken to find the clusters is taken as Y-axis. CWDHFT approach runs approximately twice faster than the two approaches FIHC and Bisecting K-means. We conclude that CWDHFT is more efficient than other approaches.

Fig. 5. Overall F-measure results comparison with Reuters dataset.

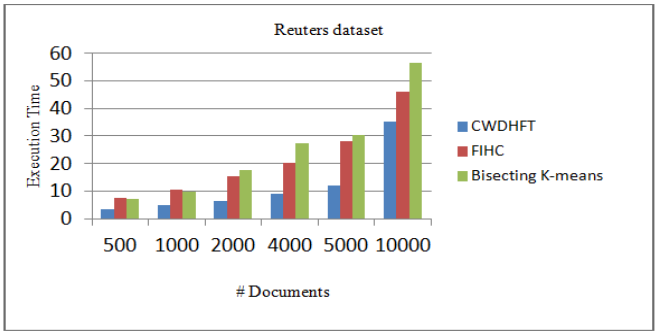

Fig. 6. Efficiency comparison of CWDHFT with FIHC and Bisecting K-means on different sizes of reuters at minsup=15%.

V. Conclusion

In this paper, we presented a novel CWDHFT approach for web document clustering based on hashing algorithm for mining frequent termsets that provides significant dimensionality reduction. The originality of CWDHFT approach is by introducing an efficient MTHFT algorithm for mining frequent termsets. MTHFT algorithm introduced a novel method for mining frequent termsets by building the multi-tire hash table during the scanning process of documents only one time. Furthermore it provided a possibility for mining new frequent termsets at different minimum support threshold without needing to rescan the documents. This is the major factor for speeding up the clustering process.

Experiments are conducted to evaluate MTHFT algorithm in comparison with Apriori algorithm and to evaluate the CWDHFT approach in comparison with Bisecting K-means and FIHC. The largest dataset, Reuters, is chosen to exam the efficiency and scalability of our algorithm. The experimental results show that in mining process, the scanning and computational cost is improved when processing large size of documents. The proposed document clustering, CWDHFT , approach improved accuracy, scalability and efficiency when compared with other clustering algorithms. Moreover, it automatically generates a natural description for the generated clusters by a set of frequent termsets. From all experiments, we conclude that CWDHFT approach has favorable quality in clustering documents using frequent termsets.

VI. Future Work

The area of document clustering has many issues which need to be solved. In this work, few issues e.g. high dimensionality and accuracy are focused. In future work, we intend to propose a novel technique for clustering web documents based on Association Rules instead of using frequent termsets .

References

[1] S. Sharma, and V. Gupta, Recent developments in text clustering techniques, International Journal of Computer Applications, vol. 37, pp. 14-19, 2012

[2] S. Fabrizio, “Machine learning in automated text categorization,” International Conference on ACM Computing Surveys, vol. 34, pp. 1-47, 2002.

[3] K. Jain, N. Murty, and J. Flynn, “Data clustering: a review,” International Conference on ACM Computing Surveys, vol. 31, pp. 264-323, 1999.

[4] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” KDD Workshop on Text Mining, 2000. Avaiable online : http://glaros.dtc.umn.edu/gkhome/node/157

[5] P. Berkhin, “Survey of clustering data mining techniques,” 2004, [Online]. Available: http://www.accrue.com/products/rp_cluster_review pdf

[6] R. Xu, “Survey of clustering algorithms,” International Conference of IEEE Transactions on Neural Networks, vol.15, pp. 634-678, 2005.

[7] F. Benjamin, W. Ke, and E. Martin, “Hierarchical document clustering,” Simon Fraser University, Canada, pp. 555-559, 2005.

[8] J. Ashish, J. Nitin, “Hierarchical document clustering: A review,” 2nd National Conference on Information and Communication Technology, 2011, Proceedings published in International Journal of Computer Applications.

[9] B. Fung, K. Wang, and M. Ester, “Hierarchical document clustering using frequent itemsets,” International Conference on Data Mining, vol.

[10] 30, pp. 59-70, 2003.

[11] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," International Conference on Management of Data, vol. 22, pp. 207-216, Washington 1993.

[12] O. Zaiane and M. Antonie, "Classifying text documents by association terms with text categories," International Conference of Australasian Database, vol. 24, pp. 215-222, 2002.

[13] B. Liu, W. Hsu and Y. Ma, "Integrating classification and association rule mining," International Conference of ACM SIGKDD on Knowledge Discovery and Data Mining, pp. 80-86, 1998.

[14] O. Zamir and O. Etzioni, "Web document clustering: A feasability demonstration," International Conference of ACM SIGIR, pp. 46-54, 1998.

[15] M. Beil, and X. Xu, "Frequent term-based text clustering," International Conference on Knowledge Discovery and Data Mining, pp. 436- 442, 2002.

[16] M. Hassan and K. John, "High quality, efficient hierarchical document clustering using closed interesting itemsets," International IEEE Conference on Data Mining, pp. 991-996, 2006.

[17] L. Xiangwei, H. Pilian, "A study on text clustering algorithms based on frequent term sets," Springer-Verlag Berlin Heidelberg, 2005.

[18] Y.J. Li, S.M. Chung, J.D. Holt, "Text document clustering based on frequent word meaning sequences," Data & Knowledge Engineering, vol. 64, pp. 381–404, 2008.

[19] H. Edith, A.G. Rene, J.A. Carrasco-Ochoa, and J.F. Martinez-Trinidad, "Document clustering based on maximal frequent sequence," FinTal vol. 4139, pp. 257-267, LNAI 2006.

[20] Z. Chong, L. Yansheng, Z. Lei and H. Rong, FICW: Frequent itemset based text clustering with window constraint, Wuhan University journal of natural sciences, vol. 11, pp. 1345-1351, 2006.

[21] L. Wang, L. Tian, Y. Jia and W. Han, "A Hybrid algorithm for web document clustering based on frequent term sets and k-means," Lecture Notes in Computer Science, Springer Berlin, vol. 4537, pp. 198-203, 2010.

[22] Z. Su, W. Song, M. Lin, and J. Li, "Web text clustering for personalized e-Learning based on maximal frequent itemsets," International Conference on Computer Science and Software Engineering, vol. 06, pp. 452-455, 2008.

[23] Y. Wang, Y. Jia and S. Yang, "Short documents clustering in very large text databases," Lecture Notes in Computer Science, Springer Berlin, vol. 4256, pp. 38-93, 2006.

[24] W. Liu and X. Zheng, "Documents clustering based on frequent term sets," Intelligent Systems and Control, 2005.

[25] H. Anaya, A. Pons and R. Berlanga, "A Document clustering algorithm for discovering and describing topics," Pattern Recognition Letters, vol. 31, pp. 502-510, April 2010.

[26] R. Kiran, S. Ravi, and p. Vikram, "Frequent itemset based hierarchical document clustering ung Wikipedia as external knowledge," Springer-Verlag Berlin Heidelberg, 2010.

[27] R. Baghel and Dr. R. Dhir, A Frequent concept based document clustering algorithm, International Journal of Computer Applications, vol. 4, pp. 0975 – 8887, 2010.

[28] http://tartarus.org/martin/PorterStemmer/

[29] M. Berry, "Survey of text mining: clustering, classification, and retrieval," Springer-Verlag New York, Inc., 2004.

[30] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[31] http://glaros.dtc.umn.edu/gkhome/views/cluto

[32] http://ddm.cs.sfu.ca/dmsoft/Clustering/fihc_index.html