An Efficient Incremental Text Clustering Approach Based on Frequent Itemsets

S. Murali Krishna¹ and S. Durga Bhavani²
¹Research Scholar, JNTUA, Anantapur, Andhra Pradesh, India
Email: smuralikrishnaphd@gmail.com
²Professor, Department of Computer Science and Engineering, School of Information Technology (SIT)
JNTU-Hyderabad, Andhra Pradesh, India

Abstract: Nowadays, the amount of text documents available on digital libraries, internet, news sources, and company-wide intranets and so on are growing incessantly at an incredible rate. Because of this there is an increased interest in developing new methods for enabling the users to effectively navigate, summarize, and organize these text documents. Among the developed methods a vital role is played by the fast and high-quality document clustering algorithms towards the goal of document organization. However, high update rates are still a major issue in document databases. Researchers have found that existing clustering algorithms are not appropriate for preserving the clusters in such dynamic environment, and they have recognized the problem of updating the clusters without habitually performing the complete re-clustering. We have developed a frequent itemsets based incremental clustering approach, which avoids complete re-clustering of the entire database each time when a change is made in the database. The proposed approach consists of two phases, namely (1) primary clustering and (2) incremental clustering. In the primary clustering phase, initial clusters are obtained using the frequent itemsets mined by the Apriori algorithm. Subsequently, using frequent words of the incremental document, suitable cluster for the incremental documents are identified in the incremental clustering phase of the approach. Finally, experimental validation results of the proposed approach on real datasets showed that it was capable of achieving clusters with significantly improved precision.

Keywords: Data mining, Clustering, Incremental clustering, Frequent itemset, Apriori algorithm, Similarity measure, Webkb dataset, 20-newsgroups.

1. INTRODUCTION

Today, the quantity of textual information existing in electronic form is growing at an overwhelming rate and a very good example for this is the World Wide Web (WWW), which is supposed to give access to at least three terabytes of text (that is, three million megabytes). The text compendium sizes which were unthinkable a few years ago are common now in viable and personal hands, and the challenge is to capably mine interesting patterns, style and potential information that are of interest to the user [1]. Text mining, also known as Intelligent Text Analysis, Text Data Mining or Knowledge-Discovery in Text (KDT), refers usually to the method of extracting, attractive and less important information and knowledge from unstructured text [2]. It is therefore vital for an excellent text mining model to recover information needed by the users with appropriate competence [3]. Generally, data mining deals with structured data (for example relational databases), but text is unstructured because it shows special characteristics [4]. The database used for storing unstructured data are entirely different from the databases used for storing structured data, hence conventional data mining techniques cannot be applied [5]. Text mining will be able to work with unstructured or semi-structured data sets such as emails, full-text documents and HTML files and more [6]. Some common approach regarding text mining and knowledge discovery in texts can be found in [7-9].

Text mining generally deals with knowledge discovery from huge text collections and it is a rising study area that combines knowledge discovery and text processing techniques. Text mining is concerned primarily with the discovery of attractive patterns such as clusters, associations, deviations, resemblance, and dissimilarities [10-13]. In the midst of them, text clustering is described as a proficient way for categorizing several documents
to help users navigate, sum up, and organize text documents [17-19]. Cluster analysis is the process of separating data objects (e.g.: document and records) into noteworthy clusters or groups such that objects within a cluster not only have analogous uniqueness but also conflicts with the objects in other clusters [14], [20] The main attraction of cluster analysis is because of the fact that it is capability of positioning clusters directly from the given data without relying on any pre determined information for instance training examples provided by domain experts [15]. Clustering of large collections of text documents is a vital procedure for obtaining advanced level of information related to the basics behind the inherent categorization of the documents [16].

Document clustering is a type of text data mining and organization technique that automatically groups associated documents into clusters [21]. Document clusters are broadly used in a variety of functions such as browsing [22] and screening of document outcome [23] or topic detection [24][25], it also replicates the association of terms and documents. A document is commonly characterized by a feature vector. Typically each feature corresponds to a keyword or phrase that exists in the document set. Every access of the vector stores a numeric weight for the consequent characteristic of the document. After removing the feature vectors of the document, clustering algorithm is applied on this set of vectors as in conventional high dimensional data clustering. The resulting document clusters along-with the representative features (i.e., the key words or phrases with sufficient document held up inside the cluster) are reported to the user [15]. Document clustering is an effective tool to manage information overload.

Vigorously changing databases of huge sizes are presently accessible in a majority of organizations [27]. As an alternative of applying the clustering algorithm to the (very large) updated database [28], it is of huge aspiration that these updates are to be carried out incrementally by taking into account only the old clusters and the data inserted or deleted. Incremental clustering [32, 29, and 30] has evolved as a main concern to carry out data analysis on existing databases because of the fact that the sizes of most existing databases extend to grow quickly. An incremental clustering algorithm is defined to a concept that shares the data instances created by the previous run of the algorithm and hence capable of enduring well with the ever-growing accessible databases [31]. Incremental clustering algorithms make one or hardly any passes over the whole dataset and they choose the cluster of a text document as they see it. In recent times, quite a few researches exist in the literature for incremental text clustering, which are intended to surmount the re-clustering.

In this paper, a proficient incremental text clustering approach is proposed for clustering the newly updated database. The proposed incremental clustering approach executes in two phases namely, primary clustering and incremental clustering. For primary clustering, we have used our prior research [33] that is an effective frequent item set-based document clustering approach. In incremental clustering, at first, the top-p frequent words of the incoming text document from the incremented database are extracted. Then, the top-p frequent words are matched with the familiar words of the initial cluster and, the corresponding partitions are chosen with regard to the exact matching result. Based on the number of retrieved partitions, the procedure for finding the appropriate cluster is changed. (1) If multiple numbers of retrieved partitions are present, then with respect to the representative keywords of the retrieved partitions the similarity value of the top-p frequent words of the documents are determined. According to the similarity measure, the exact partition is identified and the incoming text document is put into its sub-cluster based on a similarity threshold value. (2) If only one retrieved partition is present, then based on a similarity threshold value, the incoming document is put into its sub-cluster. (3) If no partitions are retrieved, then the incoming document is put into a new cluster created for it. This procedure is repeated for every documents present in the incremental database.

The structure of the paper is organized as follows: A brief review of the researches related to the incremental clustering is given in Section 2. The proposed approach for effectual clustering of incremental database is given in Section 3. The experimental results of the proposed approach are presented in Section 4. Finally, the conclusion is given in Section 5.

2. REVIEW OF RELATED RESEARCH

A handful of researches have been presented for clustering the static textual database. Recently, the
incremental text clustering has received a great deal of attention among data mining researchers. A brief review of some recent researches for text clustering and incremental text clustering is presented here.

Zhou Chong et al. [34] have proposed a method named Frequent Itemset-based Clustering with Window (FICW), which makes use of the semantic information for text clustering with a window constraint. FICW has outperformed the technique with which it was compared in both clustering accuracy and efficiency and this was illustrated by the experimental outcome obtained from the tests on three (hypertext) text sets. Yanjun Li et al. [35] have proposed a supervised feature selection method, named CHIR, which is based on the Chi-square statistic and statistical data that measures the positive term-category dependency. They have also proposed a text clustering algorithm called TCFS (Text Clustering with Feature Selection). TCFS is capable of incorporating CHIR to recognize appropriate characteristics (i.e., terms) iteratively, and clustering turns out to be a learning procedure. TCFS and the k-means clustering algorithm in combination with various feature selection techniques have been compared for various real data sets. Their experimental results demonstrated that TCFS with CHIR obtains enhanced clustering precision in terms of F-measure and purity.

Congnan Luo et al. [17] have proposed the usage of neighbors and bond for the family of k-means algorithms in three aspects: a technique to choose initial cluster centroids as per the ranks of candidate documents; a similarity measure which uses a combination of the cosine and link functions; and a heuristic function for selecting a cluster to split as per the neighbors of the cluster centroids. The experimental outcome on real-life data sets have illustrated that the proposed process considerably enhances the performance of document clustering in terms of precision without further increasing the execution time. Wen Zhang et al. [37] have conducted a study on text clustering using frequent itemsets. Mainly three manifolds contribution has been presented by the paper. First, they have presented a review of the present document clustering that use frequent patterns. Second, they have proposed a method for document clustering called Maximum Capturing. Two procedures are involved in Maximum Capturing namely constructing document clusters and assigning cluster topics. Third, the planned technique has been assessed in comparison with CFWS, CMS, FTC and FIHC methods by conducting some experiments. Experimental outcome has illustrated that the clustering performance of Maximum Capturing was better than that of the other techniques presented above. The best performance especially has been achieved by Maximum Capturing with representation by means of individual words and resemblance measure using asymmetrical binary similarity. Moreover, topics produced by Maximum Capturing differentiated clusters from one another and document clusters can use them as their labels.

Kil Hong Joo and SooJung Lee [38] have proposed an incremental document clustering method for an incrementally increasing data set of documents. To deal with the varying number of words in each document, normalized inverse document frequency of a word in the data set has been introduced. In addition, for document clustering the average link technique as an alternative to using one similarity measure has used two similarity measures: a cluster cohesion rate and a cluster participation rate. Furthermore, a category tree for a group of identified clusters has been launched to support the incremental document clustering of recently added documents. The performance of the proposed technique has been examined by a sequence of experiments to recognize their different characteristics.

Sascha Hennig and Michael Wurst [39] have presented an incremental clustering system to organize and manage Newsgroup articles. It enables the administrators and readers of a Newsgroup to archive vital postings and to get an ordered synopsis on existing growth and topics. Such a system must accomplish two conditions to be practically appropriate. First, it must be capable to process hastily altering text streams, adjust the cluster organization vigorously by adding, erasing and reorganizing clusters. Second, the user must be regarded in the incremental process. Intense alterations in the organization structure are obnoxious for the majority of users, even though they are most favorable from the point of view of a theoretical clustering criterion. They have proposed an approach to explicitly model the cost to accommodate modifications in the cluster structure. Users then may control the changes that are satisfactory to them.

Tu Anh Nguyen Hoang and Kiem Hoang [40] have presented how to use graph model for clustering Vietnamese document incrementally.
Graph based model has allowed to totally model the organization of not only single documents but also the entire group of documents. The graph structure can be updated easily whenever a new document is added. Representative sub graph features can be identified when constructing the graph incrementally, which has been later used for calculating the hybrid pair-wise document similarity. These sub-graph characters have made clustering process less responsive to the Vietnamese word segmentation step. Without any assumption on the quantity of clusters and without recovering previous documents the documents have been sorted into clusters on-the-fly based on the hybrid similarity measure.

Tu-Anh Nguyen-Hoang et al. [41] have proposed an approach based on graph model and enhanced Incremental DBSCAN to solve incremental document clustering problem. As a replacement for traditional vector-based model, a graph-based model document representation has been used. With the usage of graph model, the approach has been capable of updating the graph structure as soon as a document is added to the database. Whereas, Incremental DBSCAN is an efficient incremental clustering algorithm appropriate for mining in dynamically evolving databases. Similarity among two documents has been measured by hybrid similarity of their adapting feature vectors and shared-phrase information. The experimental outcome has proved the efficacy of their proposed method.

3. A PROFICIENT INCREMENTAL TEXT CLUSTERING APPROACH BASED ON FREQUENT ITEMSET

In general, data always arrives at a database in a multiple, continuous, rapid and time varying flow. In such environment, clustering applications needs to be performed frequently to overcome complete and costly re-clustering. In this dynamic environment with high volume of updates, maintaining clusters is the highest challenge for applications of clustering. Incremental methods are of great interest to cope with this challenge. In particular, incremental algorithm for clustering differs from traditional clustering methods in that it allows incremental tracking of the ever-increasing large scale information without the necessity of performing complete re-clustering. In the proposed algorithm the process is carried out in two phases, they are

1. Primary Clustering
2. Incremental Clustering

In primary clustering, static database is clustered after applying the preprocessing techniques and then, incremental text clustering approach is carried out. Fig.1 shows the entire process of the proposed incremental text clustering algorithm.

3.1. Primary Clustering

(1) Text Preprocessing: Text, as commonly known is in the unstructured format. All the approaches of text clustering require several steps of preprocessing for changing the unstructured format of text documents into structured format. In the proposed approach, the following steps are used to covert the text documents into structured format.

(a) At first, stop words, i.e. words that appear frequently but have low content discriminating power, are removed. Example of such words are ‘a’, ‘an’, ‘and’, ‘the’, ‘that’, ‘it’, ‘he’, ‘she’, etc. [42].

(b) The document is tokenized into a set of strings separated by some delimiters, e.g. white spaces. These tokens (terms) may represent words, phrases or any key word patterns.

(c) To avoid treating different forms of the same word as different attributes, the tokens are stemmed to their roots (i.e. replacing each token with its base form). For example both, “promotion” and “promoting” are converted to the same base form “promote” [43].

Figure 1: Proposed Incremental Text Clustering Approach
(2) Mining of Frequent Item sets: Let \( D = \{ d_1, d_2, d_3, ..., d_n \}; 1 \leq i \leq n \) be a database of text documents and \( n \) be the number of documents in the database \( D \). Every document in the database is preprocessed using preprocessing techniques and then, frequent item sets are mined from the preprocessed text documents for primary clustering. Frequent term (item) sets are sets of terms co-occurring in more than a threshold percentage of the documents that exist in the database [44]. To find frequent itemset, the frequency of each word or term in every preprocessed document \( d_i \) in the database \( D \) is computed and \( \top - p \) frequent words are taken out.

\[
K_w = \{ d_i | p(d); \forall d_i \subseteq D \}
\]

where,

\[
p(d) = Tw; 1 \leq j \leq p
\]

Binary database \( B_r \) is formed by obtaining unique keywords from the \( \top - p \) frequent words list. Binary database \( B_r \) contains \( n \) number of documents as transactions \( (T) \) and the attributes (unique words) are represented as items, \( U = \{ u_1, u_2, ..., u_q \} \). Binary database \( B_r \) consists of binary data that represents whether each of the particular unique words is present or not in the document \( d_i \).

\[
B_r = \begin{cases} 
0 & \text{if } u_i \notin d_i; 1 \leq j \leq q, 1 \leq i \leq n \\
1 & \text{if } u_i \in d_i; 1 \leq j \leq q, 1 \leq i \leq n
\end{cases}
\]

The database \( B_r \) is given to the Apriori algorithm that was first introduced in [45] for mining the frequent item sets \( F_r \) which, are of different length varying from 1 to \( k \).

\[
F_r = \{ f_1, f_2, f_3, ..., f_k \} ; 1 \leq l \leq k
\]

\[
f_i = \{ f_{i(1)} | 1 \leq i \leq t \}
\]

where, \( \sup(f_{i(1)}) \geq \sup(f_{i(2)}) \geq ... \geq \sup(f_{i(1)}) \) and \( t \) denotes the number of frequent itemsets in the set \( f_r \). The support value is a real number that varies from 0 to 1 (0 \( \leq \sup \leq 1 \)).

(3) Partitioning the Text Documents Based on Frequent Itemsets: Frequent itemsets are sorted in descending order based on their support level. Initially, the first element \( f_{(k/2)}(1) \) from the sorted list \( f(k/2) \), is selected and an initial partition \( C_i \) which contains all the documents that has the item set \( f_{(k/2)}(1) \), is constructed. Subsequently, the second partition is formed by identifying all the documents that has the frequent set \( f_{(k/2)}(2) \) and the documents already present in the initial partition \( C_i \) is not included in this second partition. This procedure is repeated until all text documents in the input database \( D \) are moved into partition \( C_i \). If all the documents in the database are not moved into an appropriate partition, we make use of subsequent sorted list \( f_{(k/2)}-1, f_{(k/2)}-2 \) and so on). This procedure results in a set of partitions \( c \), where each partition \( C_i \) contains a collection documents \( D^{(i)}_{(i)} \).

\[
c = \{ c_{(i)} | c_{(i)} \in f_{(i)} \}; 1 \leq i \leq m, 1 \leq i \leq k
\]

\[
C_{(i)} = Doc[f_{(i)}]; C_{(i)} = \{ D^{(i)}_{(i)}; D^{(i)}_{(i)} \in D, 1 \leq x \leq r \}
\]

Here, \( m \) denotes the number of partitions and \( r \) denotes the number of documents present in a partition.

(4) Clustering of Text Documents within the Partition: In this phase, we first identify the documents \( D^{(i)}_{(i)} \) and the familiar words \( f_{(i)} \) (frequent item set used for constructing the partition) of each partition \( C_i \). Then, the derived keywords \( K_w[D^{(i)}_{(i)}] \) of document \( D^{(i)}_{(i)} \) are obtained by taking the absolute complement of familiar words \( f_{(i)} \) with respect to the top-\( p \) frequent words of the document \( D^{(i)}_{(i)} \). The support of each derived unique keyword within the partitions is computed. The set of keywords satisfying the cluster support \( (cl \_sup) \) are formed as representative words of the partition \( C_i \).

\[
R_w[c(i)] = \{ x : p(x) \}
\]

where,

\[
p(x) = [K_w[D^{(i)}_{(i)}]] \geq cl \_sup
\]

The similarity measure of documents \( D^{(i)}_{(i)} \) with respect to the corresponding representative words is calculated. The similarity measure is calculated as follows.

\[
S(K_w[D^{(i)}_{(i)}], R_w[c(i)]) = \frac{|K_w[D^{(i)}_{(i)}] \cap R_w[c(i)]|}{|R_w[c(i)]|}
\]

\[
S_w(K_w[D^{(i)}_{(i)}], R_w[c(i)]) = \frac{S(K_w[D^{(i)}_{(i)}], R_w[c(i)])}{|R_w[c(i)]|}
\]

According to the threshold value specified by the user, the cluster \( C_i \) is sub clustered into \( C' \) and \( C'' \) (\( C = C' \cup C'' \)). The documents having greater similarity value than the threshold value are sub
clustered into \( C' \) and the documents, which are having lesser similarity value than the threshold, are sub clustered into \( C'' \).

3.2. Incremental Clustering

This section describes the proposed incremental text clustering approach in detail. From the primary clustering phase, we obtain initial cluster \( C_0 \) and representative keywords of the cluster list, \( R_w[c(i)] \). Let \( \Delta D = \{ \Delta d_1, \Delta d_2 \cdots \Delta d_m \} \); \( 1 \leq j \leq \Delta m \) be the incremental database. In the incremental clustering phase, incoming document \( \Delta d_1 \) in the incremental database \( \Delta D \) is preprocessed using the preprocessing steps described in sub-section 3.1.1. Subsequently, top-\( p \) frequent words \( T_{w_j} \) are identified from the document \( \Delta d_1 \). Then, the identified top-\( p \) frequent words \( T_{w_j} \) are compared against the familiar words \( f_{c(i)} \) (frequent item set used for constructing the partition) of the initial cluster \( C_0 \) and the measure \( M_s^{c(i)} \) is calculated for each cluster against the document \( \Delta d_1 \).

\[
M_s^{c(i)} = T_{w_j} \cap f_{c(i)}
\]

From the initial partition list, we retrieve a set of partitions that satisfy the condition \( |M_s^{c(i)}| = |f_{c(i)}| \). The retrieved partitions are represented in a set \( Cl_{0'} \) where, \( 1 \leq i \leq \Delta t \). Then, we use three different cases for finding the suitable cluster based on the number of retrieved partition (\( \Delta t \)).

**Case 1 (if \( \Delta t = 1 \))**: In this case, the representative words of retrieved partition are identified and similarity measure is computed utilizing the top-\( p \) words of the document \( \Delta d_1 \) and representative words of retrieved partition. The similarity measure is computed using the equations given below.

\[
S(T_{w_j}, R_w(Cl_{(i)})) = |T_{w_j} \cap R_w(Cl_{(i)})|
\]

\[
Sim(T_{w_j}, R_w(Cl_{(i)})) = \frac{S(T_{w_j}, R_w(Cl_{(i)}))}{|R_w(Cl_{(i)})|}
\]

The comparison of the similarity value with the similarity threshold value \( T \) determines the appropriate cluster in which the document \( \Delta d_1 \) is to be added. (i.e.) the document \( \Delta d_1 \) may cluster in the cluster \( Cl_{(i')} \) or \( Cl''_{(i')} \).

**Case 2 (if \( \Delta t > 1 \))**: At first, the representative words of each retrieved partitions are identified. For each partition, similarity measure is computed based on the top-\( p \) words of the document \( \Delta d_1 \) and representative words of retrieved partition. The similarity measure is computed using the equation given in case 1. Then, we select one partition, which has maximum similarity value among the retrieved partitions. Based on the similarity threshold value, the document \( \Delta d_1 \) is put into the suitable cluster (within the partition).

**Case 3 (if \( \Delta t = 0 \))**: It means that the incoming document \( \Delta d_1 \) is not suitable to cluster against anyone of the existing cluster list. Hence, the document is formed as a new cluster and ‘ \( m \)’ (number of cluster) is incremented by one.

The proposed procedure is repeatedly carried out for the remaining documents in the incremented database \( \Delta D \). The pseudo code for the proposed incremental clustering approach is given below.

**Input**: Incremental Database \( \Delta D \), Similarity threshold \( T \)

**Output**: The Resultant cluster

**Parameters**:
- \( T_{w_j} \rightarrow \text{Top-}\( p \) frequent words
- \( C_{(i)} \rightarrow \text{Initial cluster list}
- \( \Delta d_i \rightarrow \text{Incoming document}
- R_w(Cl_{(i)}) \rightarrow \text{Representative words of the cluster}
- f_{c(i)} \rightarrow \text{Familiar keywords}

**Pseudocode**

For each \( \Delta d_i \in \Delta D \)

\[
T_{w_j} \leftarrow \text{get_top-}\( p \) (\text{preprocess} (\Delta d_1))
\]

\[
Cl_{(i)} \leftarrow \text{get_cluster list} (T_{w_j}, f_{c(i)})
\]

If count \( (Cl_{(i)}) = 1 \)

\[
\text{Sim} \leftarrow \text{Find Sim} (T_{w_j}, R_w(Cl_{(i)}))
\]

If Sim > \( T \)

\[
Cl_{(i')} \leftarrow Cl_{(i')} \cup \Delta d_1
\]

Else

\[
Cl_{(i'')} \leftarrow Cl_{(i'')} \cup \Delta d_1
\]

Endif
Else if count \( (Cl_i) > 1 \) 

For each \( Cl_i \) 

\[ \text{Sim} \leftarrow \text{Find Sim}(Tw, Rw(Cl_i)) \] 

End for 

\[ P \rightarrow \text{Max} (\text{Sim}(Cl_i)) \] 

If \( \text{Sim}(Tw, Rw(p)) > T \) 

\[ Cl_i' \leftarrow Cl_i \cup \Delta d_i \] 

Else 

\[ Cl_i'' \leftarrow Cl_i'' \cup \Delta d_i \] 

Endif 

Else \( Cl_i = 0 \) then 

\[ C_{i+1} \leftarrow C_{i+1} \cup \Delta d_i \] 

End if 

End for 

4. EXPERIMENTAL RESULTS 

This section details the experimentation and performance evaluation of the proposed frequent itemset-based incremental text clustering approach. We have implemented the frequent itemset-based incremental text clustering approach using Java (JDK 1.6). The performance of the text clustering approach is analyzed with different real life datasets such as, 20 newsgroups and webkb datasets. 

1) 20 newsgroups dataset: This data set (20NG) [46] contains a total of 20000 messages, 1000 messages obtained from each of the 20 newsgroups. The various newsgroups that participated in the dataset are alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc and talk.religion.misc. 

2) Webkb dataset: This data set [47] consists of WWW-pages collected from computer science departments of different universities namely, Cornell, Texas, Misc, Washington, Wisconsin in January 1997 by the World Wide Knowledge Base (Web->Kb) project of the CMU text learning group. The 8,282 pages were manually categorized into the following categories: student (1641), faculty (1124), staff (137), department (182), course (930), project (504) and other (3764). 

For experimentation of the proposed approach, we took the static text databases such as 20-newsgroups dataset and Webkb dataset. To convert these static problems into dynamic problems, the text documents from each database are divided into 2 groups. In 20-newsgroups dataset, we took the 200 documents and divided it into identical number for static and incremental data base documents. In webkb dataset, 200 documents were treated as that of static database and 195 documents were dealt with as incremental database documents. Initially, the static data base is clustered based on the procedure depicted in section 3.1. It provides a set of initial clusters. Then, the incremental database is clustered by using the proposed approach. Here, at first, the top-\( p \) frequent word of first document \( \Delta d_i \) is extracted and then, the presence of these top-\( p \) frequent words within the familiar words \( f_{ij} \) of the initial cluster is identified. Then, the corresponding partition of these frequent words is retrieved and similarity measure is computed utilizing the top-\( p \) words of the document \( \Delta d_i \) and representative words of retrieved partition. Based on the similarity measure obtained, either the incremental document \( \Delta d_i \) is assigned to the appropriate cluster or it is put it into a new cluster.

4.1. Performance Evaluation 

The performance of the proposed incremental text clustering approach was evaluated on 20 newsgroups dataset and Webkb dataset using Precision, Recall and F-measure. We have used the Precision, Recall and F-measure described in [26, 36] for evaluating the performance of the proposed incremental text clustering approach. The definition of the evaluation metrics is given below, 

\[ \text{Precision (i, j)} = \frac{M_{ij}}{M_j} \] 

\[ \text{Recall (i, j)} = \frac{M_{ij}}{M_i} \] 

\[ \text{F - measure (i, j)} = \frac{2 \times \text{Recall (i, j)} \times \text{Precision (i, j)}}{\text{Precision (i, j)} + \text{Recall (i, j)}} \] 

where \( M_{ij} \) is the number of members of topic \( i \) in cluster \( j \), \( M_i \) is the number of members of cluster \( j \) and \( M_j \) is the number of members of topic \( i \).

For both datasets, we calculate the precision, recall and F-measure of the resultant clusters and the obtained result is given in Table 1 and table 2. Finally, the results are plotted as graphs shown in fig. 2 and fig. 3. From these graphs it is evident that the precision obtained by the proposed method is highly accurate for several cluster.
<table>
<thead>
<tr>
<th>Table 1</th>
<th>Precision, Recall and F-measure of 20-newsgroups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partition</strong></td>
<td><strong>Cluster</strong></td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>0.39286</td>
</tr>
<tr>
<td>P2</td>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
<td>0.33333</td>
</tr>
<tr>
<td>C5</td>
<td>0.33333</td>
</tr>
<tr>
<td>C6</td>
<td>1</td>
</tr>
<tr>
<td>C7</td>
<td>0.33333</td>
</tr>
<tr>
<td>C8</td>
<td>1</td>
</tr>
<tr>
<td>C9</td>
<td>0.37931</td>
</tr>
<tr>
<td>C10</td>
<td>1</td>
</tr>
<tr>
<td>C11</td>
<td>0.54545</td>
</tr>
<tr>
<td>P7</td>
<td>C12</td>
</tr>
<tr>
<td>P8</td>
<td>C13</td>
</tr>
<tr>
<td>P9</td>
<td>C14</td>
</tr>
<tr>
<td>P10</td>
<td>C15</td>
</tr>
<tr>
<td>P11</td>
<td>C16</td>
</tr>
<tr>
<td>P12</td>
<td>C17</td>
</tr>
<tr>
<td>P13</td>
<td>C18</td>
</tr>
<tr>
<td>P14</td>
<td>C19</td>
</tr>
<tr>
<td>P15</td>
<td>C20</td>
</tr>
<tr>
<td>P16</td>
<td>C21</td>
</tr>
<tr>
<td>P17</td>
<td>C22</td>
</tr>
<tr>
<td>P18</td>
<td>C23</td>
</tr>
<tr>
<td>P19</td>
<td>C24</td>
</tr>
<tr>
<td>P20</td>
<td>C25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Precision, Recall and F-measure of WebKb Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partition</strong></td>
<td><strong>Cluster</strong></td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>0.807229</td>
</tr>
<tr>
<td>P2</td>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
<td>0.6875</td>
</tr>
<tr>
<td>C5</td>
<td>0.75</td>
</tr>
<tr>
<td>C6</td>
<td>0.764706</td>
</tr>
<tr>
<td>C7</td>
<td>0.5</td>
</tr>
<tr>
<td>C8</td>
<td>0.833333</td>
</tr>
<tr>
<td>C9</td>
<td>0.363636</td>
</tr>
<tr>
<td>C10</td>
<td>0.5</td>
</tr>
<tr>
<td>C11</td>
<td>0.5</td>
</tr>
<tr>
<td>P7</td>
<td>C12</td>
</tr>
<tr>
<td>P8</td>
<td>C13</td>
</tr>
<tr>
<td>P9</td>
<td>C14</td>
</tr>
<tr>
<td>P10</td>
<td>C15</td>
</tr>
<tr>
<td>P11</td>
<td>C16</td>
</tr>
<tr>
<td>C17</td>
<td>1</td>
</tr>
<tr>
<td>C18</td>
<td>0.875</td>
</tr>
<tr>
<td>P12</td>
<td>C19</td>
</tr>
<tr>
<td>P13</td>
<td>C20</td>
</tr>
<tr>
<td>P14</td>
<td>C21</td>
</tr>
<tr>
<td>P15</td>
<td>C22</td>
</tr>
<tr>
<td>P16</td>
<td>C23</td>
</tr>
<tr>
<td>P17</td>
<td>C24</td>
</tr>
<tr>
<td>P18</td>
<td>C25</td>
</tr>
<tr>
<td>P19</td>
<td>C26</td>
</tr>
<tr>
<td>P20</td>
<td>C27</td>
</tr>
</tbody>
</table>

5. CONCLUSION

In this paper, we have developed a proficient approach for clustering the incremental database using frequent itemsets. The proposed approach consists of two modules namely, (1) primary clustering (2) incremental clustering. Initially, frequent itemset were mined from the text document after applying the preprocessing techniques and initial clusters were obtained by using the mined frequent item set. Then, we clustered the incremental database by making use of the initial clusters and the frequent keywords. The frequent keywords of the incoming text document from the incremental database was used for finding the relevant partition and in accordance with the similarity measure, the appropriate cluster was identified for the incoming text document.
An Efficient Incremental Text Clustering Approach based on Frequent Itemsets

Utilizing real datasets of 20-newsgroups and webkb, the proposed approach was experimented. The experimental results showed that the proposed incremental clustering approach is efficient in terms of precision, recall and f-measure.

REFERENCES


