Chapter 4

Document Clustering Algorithm using a Fuzzy Approach

Document clustering is an automatic grouping of text documents into clusters so that documents within a cluster have high similarity values among one another, but dissimilar to documents in other clusters. It has wide applications in areas such as search engines, web mining, information retrieval and topological analysis. This chapter presents a new document clustering algorithm using the concept of fuzzy sets, where each cluster is viewed as a fuzzy set over some finite universal set. We implemented the algorithm and the results are reported. We also have discussed the efficiency and time complexity of the algorithm.

4.1 Motivation

Knowledge discovery in databases often requires clustering the data into a number of distinct segments or groups in an effective and efficient manner. While clustering is a classical and well studied area, it turns out that several data mining applications pose some unique challenges. The real world data is dynamic in nature with the database being constantly updated. It is not practical to run the clustering algorithm on the database each time a new data is added to it. The clustering algorithm must be incremental so that the whole database need not be used for clustering every time when the database is being updated.
In this chapter we propose a new document clustering algorithm where the concepts of fuzzy sets have been used. In the proposed algorithm at any given stage of the algorithm there are small clusters and the decision at the current stage is to merge the incoming document with the most similar cluster. The algorithm is agglomerative. The clusters obtained are represented as fuzzy sets over a finite universal set. A similarity measure based on the fuzzy representation of the clusters is defined. The algorithm requires just one pass through the dataset and only the compact representations of the clusters are kept in the memory at any given time. The algorithm starts by considering each input data point as a cluster and goes on merging with the most similar cluster that satisfies the user specified threshold.

In Section 3 we discuss our method of representing a cluster as a fuzzy set, the similarity measure used and the merge function used to merge pairs of clusters. In Section 4 we describe the proposed algorithm and its complexity. In Section 5 we report the experimental result.

During the last few years the concept of fuzzy sets has been used in different areas including clustering or pattern recognition [6, 10, 48, 50]. Conventional clustering techniques assume that an object or data point can belong to one and only one cluster. However there may be overlapping of clusters and thus the separation of clusters is a fuzzy notion and hence the concept of fuzzy sets has come into picture. In fuzzy clustering each data point is associated with each cluster using a membership value. Larger membership values indicate higher confidence in the assignment of the object to the cluster. So in this approach each cluster is a fuzzy set of all data points.
In the paper [18] the authors proposed an approach of fuzzy clustering of web documents. The documents are represented as vectors of variable lengths. Each element of the vector is a pair of key phrase and an importance weight associated with this key phrase in a particular document. Using this representation of documents, fuzzy clustering algorithm was applied.

In [13] the authors proposed a fuzzy set approach for clustering large categorical data. For study of clusters, the underlying dataset was considered as a market-basket dataset where each transaction is a set of items bought by a particular customer. If \( I \) be the set of all items under consideration, each point in a cluster is a subset of \( I \) and each cluster is a collection of such subsets. In this context the clusters were defined as fuzzy sets over the set of all items.

4.2 Clusters as Fuzzy Sets

4.2.1 Fuzzy set representation of clusters

After some preprocessing step is applied to the documents, each document is represented as a finite list where each element in the list is of the form \((w : \text{number of occurrence of } w \text{ in the document})\) for each distinct keyword \( w \) occurring in the document. Let \( W \) be the set of all distinct key-words appearing in the documents under consideration. Let \(|W| = m\). We number the keywords in some order and thus get a sequence of the form \( W = \{w_1, w_2, w_3, ..., w_m\} \). Now keeping this ordering in mind we can represent any document \( d \) as the \( m \)-tuple \((o_1, o_2, o_3, ..., o_m)\) where \( o_i \) indicates the number of occurrence of the word \( w_i \) in the document \( d \). If a word \( w_i \) is not present in a document then \( o_i = 0 \) for that document. In this context we define each
cluster as a fuzzy set over \( W \). The fuzzy set representation of the cluster \( C \) consisting of only one document say \( d \) represented as the \( m \)-tuple \((\alpha_1, \alpha_2, \alpha_3, ..., \alpha_m)\) is computed as follows. Let \( F_C \) be the fuzzy set and \( \mu_{F_C} \) be the associated membership function, then \( \mu_{F_C} : W \rightarrow [0,1] \) is defined as

\[
\mu_{F_C}(w_i) = \frac{\alpha_i}{\sum_{i=1}^{m} \alpha_i}
\]  

(4.1)

Obviously \( 0 \leq \mu_{F_C}(w_i) \leq 1 \) for each \( i \). The fuzzy set \( F_C \) represented by the membership function \( \mu_{F_C} \) together with \( \alpha_{\text{sum}} = \sum_{i=1}^{m} \alpha_i \) is a compact representation of the cluster \( C \) and this is how we represent clusters. In Section 3.3 we show how to obtain this representation for the cluster obtained by merging two clusters whose compact representations are given. So we get the compact representation of a cluster \( C \) as \((F_C, \alpha_{\text{sum}})\).

### 4.2.2 Merging of Clusters

Let \( C_1 \) and \( C_2 \) be the two clusters and \( \alpha_{\text{sum}1} \) and \( \alpha_{\text{sum}2} \) be the total number of terms appearing in \( C_1 \) and \( C_2 \) respectively. Let \((\mu_{F_{C_1}}, \alpha_{\text{sum}1})\) and \((\mu_{F_{C_2}}, \alpha_{\text{sum}2})\) be their compact representations. Let \( C \) be the cluster obtained by merging \( C_1 \) and \( C_2 \), and let \( F_C \) be the fuzzy set representing \( C \). Then the Fuzzy membership function \( \mu_{F_C} \) for \( F_C \) can be computed as follows
for \( k = 1, 2, ..., m \). Thus the compact representation of \( C \) is \((F_c, o_{sum})\) where 
\[
o_{sum} = o_{sum1} + o_{sum2}.
\]

### 4.2.3 Similarity measure between 2 clusters

A similarity function of pairs of clusters is defined which can be calculated from the fuzzy set representation of the clusters. Let \( C_1 \) and \( C_2 \) be two clusters and let \( F_{C_1} \) and \( F_{C_2} \) be the fuzzy sets representing \( C_1 \) and \( C_2 \) respectively. Let \( \text{sim}(C_1, C_2)_{\text{fuzzy}} \) be the value of the similarity function, then
\[
\text{sim}(C_1, C_2)_{\text{fuzzy}} = \frac{|F_{C_1} \cap F_{C_2}|}{|F_{C_1} \cup F_{C_2}|}
\]

where union, intersection and cardinality of fuzzy sets are computed as defined in [42].

We also used the cosine measure as a similarity function. It is computed as
\[
\text{sim}(C_1, C_2) = \frac{F_{C_1} \cdot F_{C_2}}{|F_{C_1}| |F_{C_2}|}
\]

### 4.3. Proposed Clustering Algorithm

#### 4.3.1 Preprocessing step

All document clustering methods require several steps of preprocessing of the input data before performing the actual clustering. We first removed the non-textual
elements from the documents. Stopword removal was done using a standard stopword list. We then created a master word list containing every word in the document set, associated with its overall frequency. We cut down the master word list by removing infrequent words. In each document, words that do not appear in the master word list are removed. Finally, a feature vector was created for each document where each element had two fields. The first being the word present in that document and the second is the frequency of the word in the document. After the preprocessing step each document is represented as a finite list where each element in the list is of the form \((w: \text{number of occurrence of } w \text{ in the document})\) for each distinct keyword ‘\(w\)’ occurring in the document. The ordering of the words in the document is maintained.

4.3.2 Proposed Algorithm

The algorithm accepts as input the following:

- The value of \(n\), which is the size of the input data set.
- The \(n\) input data points (i.e. documents after the preprocessing step)
- The value of \(\theta\) which is the threshold used for merging clusters

Let \(S\) denote the set of clusters obtained at any time during the execution of the algorithm. Each cluster in \(S\) is represented as fuzzy set as described in the previous section. The algorithm is described below.
begin
    set $S = \emptyset$
    input $n$, $\theta$
    for $i = 1$ to $n$ do
        begin
            input a data point $d$
            compute the cluster $C$ consisting of the data point $d$ only
            while there is $C_i \in S$ with $\text{sim}(C, C_i) \geq \theta$
                begin
                    $C_2 \leftarrow \text{merge}(C, C_i)$
                    remove $C_i$ from $S$
                    delete cluster $C$
                    $C \leftarrow C_2$
                end
            add $C$ to $S$
        end
    end
end

The set $S$ gives the final set of clusters.

In the algorithm the function $\text{sim}(C, C_i) \geq \theta$ is as described in Equations (4.5) and (4.6). The function $\text{merge}(C, C_i)$ gives a new cluster after merging the clusters $C$ and $C_i$ as described in equation (4.4). The algorithm requires just one pass through the database and it is not necessary to keep the data points in memory. Only the summary of the clusters using fuzzy sets are kept in memory at any time.

4.3.3 Complexity of the algorithm

Let $n$ be the total size of the input data set. The complexity of computing the summary of a cluster containing 1 element is $O(m)$ where $m$ is the size of the feature vector to which the input data points are converted. In the whole algorithm $n$
such computations are necessary. Thus the overall complexity of this process is $O(mn)$. The complexity of computing the similarity value between a pair of clusters is $O(m)$. In the algorithm at most $n^2$ such computations will be needed. Thus the complexity of this process is $O(mn^2)$. The complexity of the procedure of merging two clusters using cluster summary is $O(m)$. During the execution of the whole algorithm at most $n-1$ times this procedure will be executed. Thus the overall complexity of the algorithm is $O(mn + mn^2)$ i.e. $O(mn^2)$.

4.4 Experimental Result

We tested the proposed clustering algorithm described in section 4.3.2 on the following datasets. Preprocessing of the documents is done as mentioned in section 4.3.1.

4.4.1 Dataset used

We used three datasets for our experimental evaluation. The datasets are BankSearch, 20NewsGroup and Reuters21578.

BankSearch

The BankSearch dataset is a collection of 11,000 documents divided into 11 categories with each category having 1000 documents. The dataset is available at http://lib.stat.cmu.edu/datasets/. We used 3 subsets of the BankSearch Dataset [58] for evaluating our algorithm namely:

(i) ADJ consists of 3 distinct categories namely Banking and finance, Programming language and Sports

(ii) ABC consists of 3 categories on related theme of Banking and Finance and
(iii) ADGHJ consists of 5 distinct categories of Banking and finance, Programming language, Astrology, Biology and Sports.

20NewsGroup

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. Each of the 20 newsgroup topics contains roughly 1000 postings. We used two subsets of the dataset:

(i) A2 consists of alt.atheism and comp.graphics

(ii) B2 consists of talk.politics.guns and talk.politics.mideast

The dataset is available at

http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

Reuters21578

The Reuters-21578 dataset is a collection of 21578 documents that appeared on Reuters news service in the year 1987. We have taken only a subset of the Reuters-21578 dataset consisting of the topics (i) coffee, (ii) gold, (ii) interest, (iv) ship and (v) sugar. The Reuters-21578, Distribution 1.0 is available from Lewis at http://www.research.att.com/~lewis/. We use the primary topic keyword as the category. There are 82 unique primary topics in the data. The categories are highly imbalanced.
Table 4.1 Information about the dataset used

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of Documents</th>
<th>No. of Terms</th>
<th>No. of Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banksearch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADJ</td>
<td>2846</td>
<td>10521</td>
<td>3</td>
</tr>
<tr>
<td>ABC</td>
<td>2897</td>
<td>6201</td>
<td>3</td>
</tr>
<tr>
<td>ADGHJ</td>
<td>4707</td>
<td>22403</td>
<td>5</td>
</tr>
<tr>
<td>20Newsgroup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>1772</td>
<td>3826</td>
<td>2</td>
</tr>
<tr>
<td>B2</td>
<td>1715</td>
<td>5131</td>
<td>2</td>
</tr>
<tr>
<td>Reuters21578</td>
<td>Subset 1</td>
<td>989</td>
<td>3620</td>
</tr>
</tbody>
</table>

4.4.2 Implementation Details

The implementation of the proposed algorithm and k-means is done in Perl. Perl provides three fundamental data types: scalar, array, and hash.

- A scalar is a single value; it may be a number, a string, or a reference.
- An array is an ordered collection of scalars.
- A hash, or associative array, is a map from strings to scalars; the strings are called *keys*, and the scalars are called *values*.

Perl's three built-in types combine with references to produce arbitrarily complex and powerful data structures. In Perl, each defined variable has a name and the address of a chunk of memory associated with it. This idea of storing addresses is fundamental to references because a reference is a value that holds the location of another value. The scalar value that contains the memory address is called a *reference*. References are scalars, so to make an array of arrays, make an array of array references. Similarly, hashes of hashes are implemented as hashes of hash references, arrays of hashes as
arrays of hash references, hashes of arrays as hashes of array references, and so on. Once you have these complex structures, you can use them to implement records. A record is a single logical unit composed of different attributes similar to structs in C and RECORDs in Pascal.

The preprocessing steps consist of (i) removal of non-textual content from the document dataset, (ii) stopword removal, (iii) word filtering and (iv) data preparation. Perl codes were written for the purpose. In the implementation of the proposed algorithm, cluster information (i.e., vector representing the cluster and sum of word occurrences in the vector) was maintained as a record data structure. The record data structure for each cluster was then kept as a list.

4.4.3 Results

The proposed algorithm was implemented and tested on the datasets mentioned above. We compared the clustering results of the proposed algorithm with the results of k-means algorithm. The clustering results are evaluated using rand index [11] which is shown in the table below. We had noticed that if increase the similarity threshold value, then the quality of the clusters obtained improves to a high extent. Majority of the clusters obtained were pure clusters. But the clusters obtained were of small sizes, thereby producing a large number of clusters.

Our next attempt was then to see how the large number of small pure clusters could be merged to form bigger pure clusters. This is reported in the next chapter.
Table 4.2 Performance of the proposed clustering algorithm

<table>
<thead>
<tr>
<th>Dataset</th>
<th>k-means Cosine Measure</th>
<th>Fuzzy Similarity Measure</th>
<th>Proposed Algorithm Cosine Measure</th>
<th>Fuzzy Similarity Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>0.76296</td>
<td>0.75494</td>
<td>0.86221</td>
<td>0.80213</td>
</tr>
<tr>
<td>ABC</td>
<td>0.62575</td>
<td>0.62723</td>
<td>0.68574</td>
<td>0.73207</td>
</tr>
<tr>
<td>ADGHJ</td>
<td>0.70324</td>
<td>0.70227</td>
<td>0.81798</td>
<td>0.80593</td>
</tr>
<tr>
<td>A2</td>
<td>0.61972</td>
<td>0.63288</td>
<td>0.75582</td>
<td>0.76484</td>
</tr>
<tr>
<td>B2</td>
<td>0.61576</td>
<td>0.65297</td>
<td>0.71836</td>
<td>0.77489</td>
</tr>
<tr>
<td>Reuters</td>
<td>0.70262</td>
<td>0.71534</td>
<td>0.723577</td>
<td>0.80338</td>
</tr>
</tbody>
</table>

4.5 Conclusion

In this chapter, we present an algorithm for clustering document collections. With the dynamic nature of real world data, any algorithm must be able to deal with new data that is constantly added to the databases. Since our algorithm is incremental, whole database need not be used for clustering every time when the database is being updated. The results of our experimental study are quite encouraging with the algorithm classifying with good accuracy. In the future we propose to work out a method to merge the relatively large number of small clusters to form bigger ones.