CHAPTER-3

TEXT DOCUMENT CLUSTERING

Text clustering is the process of grouping similar documents into clusters. Text clustering is accomplished by representing the documents as a set of terms of indexes associated with numerical weights. The goal is always to cluster the given text documents, such that they get clustered based on the similarity measures with a reasonable accuracy. During text clustering, the documents need to be preprocessed before analyzing the data. The dimensions of the vector that represent the documents need to be reduced.

The following aspects can be considered as the most important issues that need to be looked into and decided upon for the purpose of text clustering.

- Document representation
- Suffix tree representation
- Analysis of similarity measures/distance criteria (Clustering)
- Clustering algorithms

In this chapter, the basic concepts regarding the above aspects which are used in this work are presented.

3.1 Document Representation

It is necessary to pre-process the text documents in order to mine large document collections. The preprocessed information needs to be stored in a data structure, which is more appropriate for further

processing than a plain text file. The documents can be represented using vector space model, suffix trees etc.

3.1.1 Vector Space Model (VSM)

Most text mining approaches are based on the idea that a text document can be represented by a set of words, that is, a bag-of-words representation. However, in order to be able to define at least the importance of a word within a given document, usually a vector representation is used, where for each word a numerical "importance" value is stored. The vector space model [73], the probabilistic model [48] and the logical model [49] are the currently predominant approaches based on this idea.

The representation of a set of documents as vectors in a common vector space is known as the vector space model (VSM). Documents are represented as vectors of features representing the terms that occur within the collection in the vector space model of IR. Words are assumed to appear independently and the order is immaterial in this representation. The value of each feature is called the term weight and is usually a function of term's frequency (or TF-IDF) in the document, along with other factors.

If each document is considered as a multi-dimensional vector and then try to cluster documents based on their word contents, the problem differs from classic clustering scenarios in several ways.

Document clustering data is high in dimensions, characterized by a highly sparse word-document matrix with positive ordinal attribute values and a significant amount of outliers. Here the frequency of each term is used as its weight, which means terms that appear more frequently are more important and descriptive for the document.

VSM representation of a document involves three steps. First step is the document indexing where content bearing terms are extracted from the documents. The second step is to compute the weights of indexed terms to enhance retrieval of documents relevant to the user. The final step is identifying the similarities between the documents.

3.1.1.1 Extracting the Index Terms

The first initiation is to find all the index terms in a text document. The following steps are carried out to do this :

- Removal of all kinds of formatting
- Removal of stop words
- Perform stemming

There is a need to strip all formatting from the documents, if any, including capitalization, punctuation etc. Then the next step is to remove words that are semantically insignificant called stop words in the information retrieval terminology. A list of commonly used English words that doesn't carry semantic meaning like the words, "*to*", "*the*",

"*a*", "*go*", "*an*", "*has*" etc. is the stop list. The amount of noise in the collection is reduced greatly by using a stop list. It also eliminates a large number of words that would make the computation more difficult. Creating a stop list is something of an art and it depends very much on the nature of the data collection. The elimination of stop words will leave an abbreviated version of the document set containing words which are semantically significant.

The next most important step is stemming. In bag of words representation of documents the words that appear in documents often have many morphological variants and in most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of clustering applications. For this reason, a number of stemming algorithms, or stemmers, have been developed, which attempt to reduce a word to its stem or root form. Stemming is the process of removing common endings from words, leaving behind an invariant root form. For example two similar words, "*fishing*" and "*fish*", are not treated as two different words. The algorithm usually applied for stemming is the Porter's stemming algorithm [12]. Consider the example given in table 3.1 showing a corpus of six documents (each line represents a single document):

Table 3.1 Sample document set

Doc #	Text of the document
Doc 1	cricket bat ball cricket common
Doc 2	cricket bat ball cricket common
Doc 3	medicine disease doctor common
Doc 4	medicine disease doctor common
Doc 5	Financial analysis economy common
Doc 6	Financial analysis economy common

After stripping, pruning and stemming the set of documents, the following flat list of words (index words) is obtained in table 3.2.

Table 3.2 Stemmed index words

Word id	Stemmed word
1	Bat
2	Ball
3	Common
4	Cricket
5	Diseas
6	Doctor
7	Medicin
8	Analysi
9	Economi
10	Finance

Preprocessing of the text documents involves applying stemming, removal of stop words and tokenizing the text.

3.1.1.2 Attaching Weights to the Index Terms

Documents in vector space can be represented using Boolean, Term Frequency(TF) and Term Frequency – Inverse Document Frequency(TF-IDF).

3.1.1.2.1 Boolean

In Boolean representation, if a term exists in a document, then the corresponding term value is set to one, otherwise it is set to zero. Boolean representation is used when every term has equal importance and is applied when the documents are of small size.

3.1.1.2.2 Term Frequency

In Term Frequency and Term Frequency Inverse Document Frequency the term weights have to be set. The term weights are set as the simple frequency counts of the terms in the documents. This reflects the intuition that terms occurring frequently within a document may reflect its meaning more strongly than terms occurring less frequently and should thus have higher weights.

Each document d is considered as a vector in the term-space and represented by the term frequency (TF) vector:

$$d_{tf} = [\, tf_1 , tf_2 , \ldots\ldots tf_D \,] \quad (3.1)$$

where tf_i is the frequency of term i in the document and D is the total number of unique terms in the text database.

3.1.1.2.3 Term Frequency-Inverse Document Frequency

The next aspect is to give a higher weight to words that occur only in a few documents. Terms that are limited to few documents are useful for discriminating those documents from the rest of the collection, while terms that occur frequently across the entire collection are not helpful. The inverse document frequency term weight is one way of assigning higher weights to these more discriminative words. IDF is defined via the fraction N/n_i, where, N is the total number of documents in the collection and n_i is the number of documents in which term i occurs.

Due to the large number of documents in many collections, this measure is usually squashed with a log function. The resulting definition IDF is thus:

$$idf_i = \log\left(\frac{N}{n_i}\right) \quad (3.2)$$

Combining term frequency with IDF results in a scheme known as tf-idf weighting.

$$w_{i,j} = tf_{i,j} \times idf_i \quad (3.3)$$

Thus, the TF-IDF representation of the document d is:

$$d_{tf-idf} = [\, tf_1 \log(n / df_1),\ tf_2 \log(n / df_2),\ldots\ldots,\ tf_D \log(n / df_D)\,] \quad (3.4)$$

To account for the documents of different lengths, each document vector is normalized to a unit vector (i.e., $||d_{tf\text{-}idf}|| = 1$).

Given a set C_j of documents and their corresponding vector representations, the centroid vector c_j is defined as:

$$c_j = \frac{1}{|C_j|} \sum_{d_i \in c_j} d_i \quad (3.5)$$

where each d_i is the document vector in the set C_j, and j is the number of documents in Cluster C_j. It should be noted that even though each document vector d_i is of unit length, the centroid vector c_j is not necessarily of unit length. Our research is experimented with three vector model representations: Boolean, Term Frequency and Term Frequency-Inverse Document Frequency.

3.1.2 Suffix Tree Representation

The classical document model is viewed as vectors of words. Another approach is the suffix tree document (STD) model as well as the related similarity measures that are graph-based. The vector-based

document models do not have the information about the order by which the words occur in a document. A document model that is more sophisticated and that preserves the complete word order information is the STD model. Here the similarity between two documents is defined in terms of string overlaps in their common suffix tree.

A suffix tree of a set of strings, called a generalized suffix tree (GST), is a compact trie of all the suffixes of all the strings in the set. Figure 3.1 is an example of the generalized suffix tree of a set of three strings "*cat ate cheese*", "*mouse ate cheese too*" and "*cat ate mouse too*". The internal nodes of the suffix tree are drawn as circles, and are labeled a through f for further reference. Leaves are drawn as rectangles. The first number in each rectangle indicates the string from which that suffix originated and the second number represents the position in that string where the suffix starts.

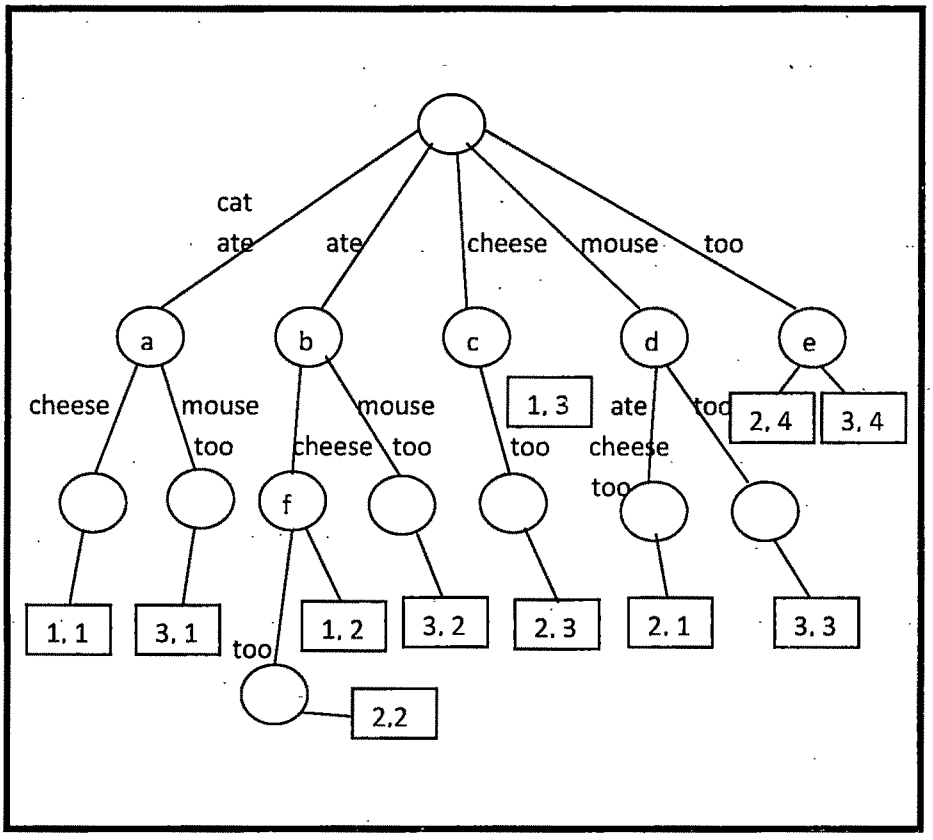

Figure 3.1 An example Suffix Tree

Suffix Tree Clustering (STC) [62], creates clusters based on phrases shared between the documents. STC was found to be faster than standard clustering methods in this domain. Carrot2 [63] is an open extensible research system for examination and development of search results clustering algorithms. Grouper [64], which acts as a document clustering interface to the HuskySearch meta-search service retrieves results from several popular Web search engines.

3.2 Similarity Measures

Document clustering groups similar documents to form a coherent cluster. However, the definition of a pair of documents being similar or different is not always clear and normally varies with the actual problem setting. In document clustering, similarity is typically computed using associations and commonalities among features, where features are typically words and phrases [56]. For example, while clustering research papers, two documents are regarded as similar if they share similar thematic topics. When clustering is employed on web sites, we are usually more interested in clustering the component pages according to the type of information that is presented in the page. For instance, when dealing with university web sites, we may want to separate professor's home pages from student's home pages and pages for courses from pages for research projects. This kind of clustering benefits further analysis and utilize the dataset such as information retrieval and information extraction, by grouping similar types of information sources together.

Accurate clustering requires a precise definition of the closeness between a pair of objects, in terms of either the pair wise similarity or distance [109]. A variety of similarity or distance measures have been proposed and widely applied, such as cosine similarity, Jaccard coefficient, Euclidean distance and Pearson Correlation Coefficient.

3.2.1 Cosine Similarity

For document clustering, there are different similarity measures available. When documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors [23]. It is the most commonly used measure in Document Clustering. For two documents d_i and d_j, the similarity between them can be calculated

$$\cos(\underline{d_i}, \underline{d_j}) = \frac{\underline{d_i} \cdot \underline{d_j}}{\| \underline{d_i} \| \, \| \underline{d_j} \|} \qquad (3.6)$$

where d_i, and d_j are m-dimensional vectors over the term set $T= \{t_1, t_2,...t_m\}$. Each dimension represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between [0, 1]. The cosine similarity is independent of document length. When the document vectors are of unit length, the above equation is simplified to:

$$\cos (d_i , d_j) = d_i . d_j \qquad (3.7)$$

When the cosine value is 1 the two documents are identical, and 0 if there is nothing in common between them (i.e., their document vectors are orthogonal to each other).

3.2.2 Jaccard Coefficient

The Jaccard coefficient [35] [36], which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects. For a text document, the Jaccard coefficient compares the sum weight of shared terms to the sum weight of terms that are present in either of the two documents but are not the common terms. The Jaccard coefficient and Jaccard Index are defined as:

$$\text{Jaccard Coff}(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||^2 + ||d_j||^2 - d_i * d_j} \tag{3.8}$$

$$\text{Jaccard Index}(d_i, d_j) = \frac{d_i \cap d_j}{d_i \cup d_j} \tag{3.9}$$

The Jaccard Coefficient ranges between [0, 1]. The Jaccard value is 1 if two documents are identical and 0 if the two documents are disjoint. The Cosine Similarity may be extended to yield Jaccard Coefficient in case of Binary attributes.

3.2.3 Euclidean Similarity

This is the most usual and intuitive way of computing distance between two documents. It is most widely used measure in clustering tasks. This

is a default distance measure used with k-means algorithm. It takes the difference between two documents measured with a ruler in two or more dimensional space into account. The Euclidean distance is 0, when two documents are identical. This distance type is usually used for data sets that are suitably normalized or without any special distribution problem. The Euclidean distance and Euclidean Similarity is defined as:

$$\text{Euclidean Distance } (d_i, d_j) = \sqrt{\sum_k (d_{ik} - d_{jk})^2} \qquad (3.10)$$

$$\text{Euclidean Similarity } (d_i, d_j) = 1 - \sqrt{\sum_k (d_{ik} - d_{jk})^2} \qquad (3.11)$$

where d_{ik} is the k^{th} term of the document d_i

3.2.4 Pearson Correlation Coefficient

Correlation is a technique for investigating the relationship between two quantitative, continuous variables. For example, age and blood pressure. Pearson's correlation coefficient (r) is a measure of the strength of the association between the two variables. There are different forms of Pearson Correlation Coefficient formula. It is given by

$$\text{Pearson Similarity } (d_i, d_j) = \frac{m\sum_k d_{ik} \text{ X } d_{jk} - TF_i \text{ X } TF_i}{\sqrt{[m\sum_k d_{ik}^2 - TF_i^2][m\sum_k d_{jk}^2 - TF_j^2]}} \qquad (3.12)$$

where $TF_i = \sum_k d_{ik}$ and $TF_i = \sum_k d_{ik}$

The measure ranges from +1 to -1. Positive correlation indicates that both variables increase or decrease together, whereas negative correlation indicates that as one variable increases, so the other decreases, and vice versa. Two documents are identical when Pearson similarity is ±1.

The Euclidean distance is a distance measure, while the cosine similarity, Jaccard coefficient and Pearson coefficient are similarity measures. We apply a simple transformation to convert the similarity measure to distance values. Because both cosine similarity and Jaccard coefficient are bounded in [0, 1] and monotonic, we take D = 1 − SIM as the corresponding distance value. For Pearson coefficient, which ranges from −1 to +1, we take D = 1 − SIM when SIM ≥0 an d D = |SIM| when SIM < 0.

3.2.5 Dice Coefficient

Dice's coefficient (also known as the Dice coefficient) [110] is a similarity measure related to the Jaccard index. It simplifies the denominator from the Jaccard measure and introduces a factor 2 in the numerator. The normalization in the Dice formula is also invariant to the number of terms in common between the two vectors, similar to the cosine formula.

$$Sim(Doc_i, Doc_j) = \frac{2 * \sum_{k=1}^{n} (Doc_{i,k} * Doc_{j,k})}{\sum_{k=1}^{n} Doc_{i,k} + \sum_{k=1}^{n} Doc_{j,k}} \quad (3.13)$$

3.2.6 Manhattan Distance

The Manhattan distance function computes the distance that would be traveled to get from one data point to the other if a grid-like path is followed. The Manhattan distance between two items is the sum of the differences of their corresponding components. In our case of two documents being represented as vectors in an n-dimensional space,

$$\sum_{k=1}^{n} | Doci,k - Docj,k | \quad (3.14)$$

3.2.7 Minkowski Distance

The Manhattan distance (1-norm distance) and the euclidean distance (2-norm distance) are specific forms of the more general minkowski distance (p-norm distance). Between two vectors in n dimensional space, it is defined as

$$dist(Doc_i, Doc_j) = (\sum | Doci,k - Docj,k |^p)^{1/p} \quad (3.15)$$

Here, p can take any value, even less than 1. It can be easily seen that, when p equal to 1, minkowski distance becomes the manhattan distance, and when p equal to 2, it becomes the Euclidean distance

3.3 Partitioning Clustering

The database is partitioned into a predefined number of clusters using partitioning clustering techniques. They attempt to determine k partitions that optimize a certain criterion function. The partition clustering algorithms are of three types:

- K- Means algorithm
- K- Medoid algorithm

3.3.1 K-Means Algorithm

K-Means[13] [54] is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The standard K-means algorithm works as follows. Given a set of data objects D and a pre-specified number of clusters k, k data objects are randomly selected to initialize k clusters, each one being the centroid of a cluster. The remaining objects are then assigned to the cluster represented by the nearest or most similar centroid. Next, new centroids are recomputed for each cluster and in turn all documents are re-assigned based on the new centroids. This step iterates until a converged and fixed solution is reached, where all data objects remain in the same cluster after an update of centroids. The generated clustering solutions are locally optimal for the given data set and the initial seeds. Different choices of initial seed sets can result in very different final partitions. Methods for finding good starting points have been proposed [18].

The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move.

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments. It minimizes intra-cluster variance, but does not ensure that the result has a global minimum of variance.

The K-Means clustering algorithm has been implemented in Java for the current work, for the purpose of clustering the text documents.

3.3.2. K- Medoid Clustering Algorithm

The K-Medoids algorithm [13] is a clustering algorithm related to the K-Means algorithm. Both the K-Means and K-Medoids algorithms are partitional (breaking up the dataset into flat groups) and both attempt to minimize the squared error. The error is computed as the distance between points labeled to be in a cluster and a point designated as the

center of that cluster. In contrast to the K-Means algorithm, K-Medoids chooses data points as centers (medoids or exemplars).

K-Medoid is a classical partitioning technique of clustering that clusters the data set of n objects into K clusters known a priori. It is more robust to noise and outliers as compared to K-Means. A medoid can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal that is it is a most centrally located point in the given data set.

The most common example of K-Medoid clustering is the Partitioning Around Medoids (PAM)[13] algorithm. The algorithm is as follows:

1. The algorithm begins with an arbitrary selection of the k objects as medoid points out of n data points (n>k)
2. After selection of the k medoid points, associate each data object in the given data set to most similar medoid. The similarity here is defined using distance measure that can be Euclidean distance, Manhattan distance or Minkowski distance
3. Randomly select non-medoid object O'
4. Compute total cost S of swapping initial medoid object to O'
5. If S<0, then swap initial medoid with the new one (if S<0 then there will be new set of medoids)
6. Repeat steps 2 to 5 until there is no change in the medoids

3.4 Evaluation Measures

The objective measures are established to assess the performance of different text mining algorithms. The 3 types of quality measures used are: external, internal and relative measures. The external measures involve a priori knowledge about the clusters. Internal measures assume no knowledge about the clusters. Relative measures evaluate the differences between different cluster solutions. The internal and relative measures are applied only to clustering, while external measures are applied to both categorization and clustering.

3.4.1 External Quality Measures

Only when a valid clustering solution is available the external measures can be applied.

3.4.1.1 Precision, Recall and F-Measure

Precision, Recall and F-Measure are the external measures well known for performance assessment from Information Retrieval [100].

- Precision: this measure retrieves the number of correct assignments out of the number of total assignments made by the system.
- Recall: this measure retrieves the number of correct assignments made by the system, out of the number of all possible assignments.

- F-measure: this measure is a combination of the precision and recall measures used in machine learning.

The precision and recall of a cluster Ci and a known class Sj are:

$$\Pr ecision(C_i, S_j) = \frac{|C_i \cap S_j|}{|C_i|} \quad (3.17)$$

$$\text{Re}call(C_i, S_j) = \frac{|C_i \cap S_j|}{|S_j|} \quad (3.18)$$

With these Precision and Recall values F- Measure can be computed as follows:

$$F(C_i, S_j) = \frac{2 * \Pr ecision(C_i, S_j) * \text{Re}call(C_i, S_j)}{\Pr ecision(C_i, S_j) + \text{Re}call(C_i, S_j)} \quad (3.19)$$

Only the cluster with the highest F-measure for each class is selected. Finally, the overall F-measure of a clustering solution is weighted by the size of each cluster:

$$F(S) = \sum_i \frac{n_i}{n} \max\left(F(C_j, S_i)\right) \quad (3.20)$$

3.4.1.2 Entropy

The homogeneity of a cluster is indicated by Entropy: low entropy indicates a high homogeneity, and vice versa. A cluster with one element will have an entropy value of zero. The entropy of a cluster is 1 if it contains an equal number of elements of each class from the original solution. The formula for entropy for a cluster Ci is:

$$E(C_i) = -\sum_{i=0}^{k-1} pr_{ij} \cdot \log(pr_{ij}) \quad (3.21)$$

where pr_{ij} represents the proportion of elements from class j that are contained in cluster i. And, the overall entropy for a clustering solution over a dataset with k classes is obtained by calculated the weighted-average of the entropy for each cluster:

$$Entropy(S) = \sum_i \frac{n_i}{n} E(C_i) \quad (3.22)$$

Here in this research entropy measure is used for evaluating cluster quality.

3.4.1.3 Purity

The average precision of a cluster relative to their best matching classes is represented by Purity. The higher the number of elements in the cluster that belong to the same class, the higher its purity will be. The purity formula for cluster Ci is:

$$P(C_i) = \frac{1}{|C_j|} \max(count_{ij}), j = 0,1...,k-1 \quad (3.23)$$

where $Count_{ij}$ represents the number of elements from class j that belong to cluster i. The overall purity of a clustering solution for a dataset with k classes is:

$$Purity(S) = \frac{1}{n}\sum_{i} \max\left(Count_{ij}\right), j = 0,1,...,k-1, j \neq i \quad (3.24)$$

3.4.2 Internal Quality Measures

When there is no external knowledge about the clustering solution the internal quality measures are used.

- Intracluster Similarity: This is the most common quality measure. The average of the similarity between any two points of the cluster is represented by this measure.
- Partition Index: The ratio of the sum of compactness and separation of clusters is reflected by this Partition Index.
- Dunn Index: It tries to identify compact and well separated clusters. The Dunn Index could be infeasible for large n as it has a $O(n^2)$ complexity. The diameter of the largest cluster can also be influenced by outliers.
- Separation Index: It measures the inter-cluster dissimilarity and the intra-cluster similarity by using cluster centroids. It is computationally more efficient than Dunn Index.

Other statistical measures [97] like the Mean square error the silhouette coefficient [98] also exist.

3.4.3 Relative Quality Measures

Instead of providing an absolute evaluation there is an increasing need to evaluate one clustering solution against another. A relative quality measure is helpful in this aspect. Cluster Distance Index measures the difference between two clusters. The distance is just $||c_i - c_j||$ for centroid-based clusters. If the clusters are not represented by centroids, then the linkage paradigm from hierarchical clustering can be applied: the cluster distance could either refer to the minimum, maximum or average distance between any two points of the two clusters.