Phrase based Clustering Scheme of Suffix Tree Document Clustering Model

Anoop Kumar Jain
Research Scholar (M. Tech),
Samrat Ashok Technological Institute,
Vidisha(SATI), M.P., India

Satyam Maheshwari
Dept. of Computer Application
Samrat Ashok Technological Institute,
Vidisha(SATI, M.P., India

ABSTARCT

Document clustering is one of the difficult and recent research fields in the search engine research. Most of the existing documents clustering techniques use a group of keywords from each document to cluster the documents. Document clustering arises from information retrieval domains, and "It finds grouping for a set of documents belonging to the same cluster are similar and documents belongs to the different cluster are dissimilar". The nformation retrieval plays an important role in data mining for extracting the relevant information for related to user request. Information retrieval finds the file contents and identifies their similarity. It measures the performance of the documents by using the precision and recall. In this paper we proposed a phrase based clustering scheme which based on application of Suffix Tree Document Clustering (STDC) model. The proposed algorithm is designed to use the STDC model for accurate equivalent representation of document and similarity measurement of the similar documents. This method of clustering reduces the grouping time and similarity accuracy as compared to other existing methods.

Keywords

Clustering Techniques, Document Clustering,Phrase Merging,Suffix Tree

1. INTRODUCTION

Document retrieval systems typically present search results in a ranked list, ordered by their estimated relevance to the query. The similarity based relevancy is estimated between the text of a document and the query. Such ranking schemes work well when users can formulate a well-defined query for their searches. The users of search engines often formulate very short queries that often retrieve large numbers of documents. Based on such a condensed representation of the user's search interest, it is impossible for the search engine to identify the specific documents that are of interest to the users. Moreover, many webmasters now actively work to influence rankings. These problems are exacerbated when the users are unfamiliar with the topic they are querying about, when they are novices at performing searches, or when the search engine's database contains a large number of documents. These conditions commonly exist for Web search engine users. Therefore the vast majority of the retrieved documents are often of no interest to the user, such searches are termed as low precision searches. The low precision of the search engines coupled with the ranked list presentation force users to sift through a large number of documents and make it hard for them to find the information they are looking for. As low precision searches are inevitable, tools must be provided to help users "cope" with (and make use of) these large document sets. Such tools should include means to easily browse through large sets of retrieved documents [1] and [2-4].

The objective of paper is to make search engine results easy to make document cluster. Document clustering algorithms attempt to group similar documents together. In this paper the proposed method is a phrase based clustering scheme which based on application of Suffix Tree Document Clustering (STDC) model.

2. BACKGROUND TECHNIQUES

2.1 Clustering and Its Models

Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. The main task of clustering is explorative data mining, and the common technique for statistical data analysis used in many fields, like including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics. The notion of a "cluster" varies between algorithms and is one of the many decisions to take when choosing the appropriate algorithm for a particular problem. The terminology of a cluster seems obvious: a group of data objects. The clusters found by different algorithms vary significantly in their properties, and understanding the cluster models is key to understanding the differences between the various algorithms [3-5].

Typical cluster models include:

Connectivity models: for example hierarchical clustering builds models based on distance connectivity.

Centroid models: for example the k-means algorithm represents each cluster by a single mean vector.

Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.

Density models: for example DBSCAN and OPTICS defines clusters as connected dense regions in the data space.

Subspace models: in Biclustering (also known as Co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.

Group models: some algorithms (unfortunately) do not provide a refined model for their results and just provide the grouping information.

Graph-based models: a clique, i.e., a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques [4-6].

A "clustering" is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example a hierarchy of clusters embedded in each other. Clustering can be roughly distinguished in:

Hard clustering: each object belongs to a cluster or not

Soft clustering (also: fuzzy clustering): each object belongs to each cluster to a certain degree (e.g. a likelihood of belonging to the cluster)

There are also finer distinctions possible, for example:

Strict partitioning clustering: here each object belongs to exactly one cluster

Strict partitioning clustering with outliers: objects can also belong to no cluster, and are considered outliers.

Overlapping clustering (also: alternative clustering, multi-view clustering): while usually a hard clustering, objects may belong to more than one cluster.

Hierarchical clustering: objects that belong to a child cluster also belong to the parent cluster

Subspace clustering: while an overlapping clustering, within a uniquely defined subspace, clusters are not expected to overlap [7-8].

2.2 Document Clustering

Document clustering (or Text clustering) is automatic document organization, topic extraction and fast information retrieval or filtering. It is closely related to data clustering. A web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by Enterprise Search engines. Document clustering involves the use of descriptors and descriptor extraction. Descriptors are sets of words that describe the contents within the cluster. Document clustering is generally considered to be a centralized process. Examples of document clustering include web document clustering for search users. The application of document clustering can be categorized to two types, online and offline. Online applications are usually constrained by efficiency problems when compared offline applications. In general, there are two common algorithms. The first one is the hierarchical based algorithm, which includes single link, complete link, group average and Ward's method. By aggregating or dividing, documents can be clustered into hierarchical structure, which is suitable for browsing. However, such an algorithm usually suffers from efficiency problems. The other algorithm is developed using the K-means algorithm and its variants. Usually, it is of greater efficiency, but less accurate than the hierarchical algorithm [9] and [10].

2.3 Related Works

Lan Yu et. al. In this paper, we use one of the data mining algorithms (clustering) to analyze PHS test data with the help of SQL Server. The purposes of experiments are to discover how to construct a training set and how to set parameters of Microsoft clustering algorithm. Some valuable conclusions are achieved. A good training set should have sufficient records, but a huge number of records could degrade the performance of clustering. That is the reason why the accuracy of case 2 is better than that of case 1. And using all data to compose a training set as case 1 of this experiment is not practical. When the number of the training set is determinate, the more records a ranking has, the higher accuracy of this ranking obtained [2].

M. Asif Naeem et. al. proposed a novel clustering algorithm called Hierarchical Particle Swarm Optimization (HPSO) data clustering. The proposed algorithm exploits the swarm intelligence of cooperating agents in a decentralized environment. The experimental results were compared with benchmark clustering techniques, which include K-means, PSO clustering, Hierarchical Agglomerative clustering (HAC) and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The results are evidence of the effectiveness of Swarm based clustering and the capability to perform clustering in a hierarchical agglomerative manner. The approach to tackling the problem of hierarchical agglomerative clustering through hierarchical particle swarm optimization based clustering is inspired by the collective intelligent behavior of swarms. HPSO-clustering has the properties of both partitioned based data clustering and hierarchical data clustering. Experimental results verify the performance of PSO against PSO, traditional hierarchical agglomerative clustering and K-means clustering [1].

Yun Ling et. al. utilized correspondence analysis algorithm to process matrix decomposition and then make use of Bayesian approach for co-clustering. They find that utilizing the two methods synthetically is very significative to solve actual problems. Experiments on synthetic and real world data demonstrate the efficiency and effectiveness of the algorithm. They proposed a CA method for fast co-clustering on large data. The learned knowledge is useful when synthesizing these two methods. Because of the small size of the approximation matrices, it has runtime complexity equal to orders of magnitude faster than the runtime complexity of the previous co-clustering algorithms. Due to its low complexity and simple implementation the work presented will make a broad application [3].

M. Sudharshan et. al. propose an innovative clustering technique called the Rapid Clustering Method (RCM), which uses Subtractive Clustering combined with Fuzzy CMeans clustering along with a histogram sampling technique to provide quick and effective results for large sized datasets. Rapid Clustering Method can be used to cluster the dataset and analyze the characteristics in a social network. It can also be used to enhance the cross-selling practices using quantitative association rule mining. With the help of RCM they can obtain rapid results and identify structural changes much more quickly. They considered the problem of clustering data over time and proposed a rapid clustering technique so that it can be used to generate quicker results for social network analysis [4].

3. PROPOSED TECHNIQUES

The proposed algorithm is designed to use the STDC model for accurate equivalent representation of document and similarity measurement of the similar documents. This method of clustering reduces the grouping time and similarity accuracy as compared to other existing methods. The main

focus of the paper is implementing the steps of Suffix Tree Document Clustering (STDC) algorithm for information retrieval. We have introduced the Suffix Tree Clustering algorithm, which generates and describes clusters based on common phrases that are found in the document set. STDC uses a suffix tree to efficiently identifying these common phrases. We have shown STDC to be linear in the length of the document set, incremental, and to produce overlapping clusters.

3.1 Proposed STDC Mechanism

Our proposed Suffix Tree Document Clustering (STDC) mechanism have four steps, these are as follows:

I. Document collection as input

II. Document Cleaning

III. Phrase Identification of Clusters

IV. Phrase Merging Clusters

3.1.1 Document Collection

The collection of documents is the very first to perform the searching. The documents are collected in the dataset. The collected documents can be either text documents or the web documents, but the tool performs the clustering on the text documents.

3.1.2 Document Cleaning

In Document cleaning, data is cleaned from the missing values, smoothing noisy data and inconsistencies. Data cleaning is the preprocessing of the data, through which data is cleaned and processed that is input to the next step to the Suffix Tree structure. Preprocessing includes the steps such as:

1. *Tokenization*
2. *Stop-word removal*
3. *Stemming algorithm*

3.1.2.1 Tokenization

Tokenization is the preprocessing step, in which sentences are divided into tokens. Tokenization is the process of identify the word and sentences boundaries in the text. The simplest form of tokenization is the white space character as a word delimiters and selected punctuation mark such as „.“,“?“and „! „. Each word assigns a token id.

3.1.2.2 Stop-word removal

There are many words in the document that contain no information about the topic. Such words don't have any meaning or no use while creating the suffix tree structure. Stop words are also referred to the as function words that have their own identifiable meaning. Such words that occur in the stop list are: and, but, will, have etc. The list of stop word is store in the database.

3.1.2.3 Stemming Algorithm

In the stemming procedure all words in the text document are replaced with their respective stem. A stem is a portion of a word that would be left after removing the affixes (suffixes and prefixes). Different form of words can be reduced into one base form by using the stemmer. Lots of stemmer created for the English language. The process of stemmer development is easy. There is lot of stemmers available for English language. For example: connected, connecting, interconnection is transformed into word connect.

3.1.3 Phrase identification of Cluster

In this step, the STDC algorithm identifies *all* maximal phrase clusters. This is done efficiently using a suffix tree. The identification of phrase clusters can be viewed as the creation of an inverted index of phrases for the document collection. The phrase clusters are scored and the highest scoring ones are selected for further consideration. We create a generalized suffix tree from all the *sentences* (as defined above sequence of words between phrase boundaries) of all the documents in document set. Each leaf is marked with a sentence identifier that also identifies which document it belongs to.

The key feature of the suffix tree is that each internal node *v* represents a group of documents and a phrase that is common to all of them. The label *vp* of internal node *v* is the common phrase. All the leaves in the sub-tree of *v* correspond to sentences that have suffixes that start with the phrase *vp*. Therefore, the group of documents containing *vp* can be determined from these leaves (actually, we can also determine how many times the phrase *vp* appears in each document and where). This leads us to the following observation: every possible *maximal* phrase cluster corresponds to an internal node in suffix tree. By this we mean that the phrase of the phrase cluster equals the label of the node, and the set of documents of the phrase cluster equals the set of documents designated by the leaves in that node's sub-tree. To see why this is true take a maximal phrase cluster *m* with phrase *mp*. Phrase cluster *m* contains at least two documents, say *i* and *j*, that contain *mp*. Therefore documents *i* and *j* contain sentences which have suffixes that contain *mp*. It means that there exists a path in the suffix tree, starting from the root, whose label starts with the phrase *mp*. As there are two sentences (from different documents) that share this phrase, there must be an internal node *u* on this path, whose label *up* either equals *mp* or else is the first label from the root that has *mp* as a prefix. The phrase cluster corresponding to node *u* must include all the documents that contain the phrase *mp*, therefore as *m* is a maximal phrase cluster, *mp* must be equal to *up* otherwise a word could have been added to *mp* (the next word in *up* after the prefix *mp*) without decreasing the number of documents in the phrase cluster.

The reverse is not true. First, an internal node in the suffix tree might not correspond to a phrase cluster as all the leaves in its sub-tree might originate from suffixes of sentences from a single document. An internal node that has leaves in its sub-tree from at least two different documents does correspond to a phrase cluster, but not necessarily to a maximal phrase cluster. This could happen if the phrase *vp* of node *v* is found in all the documents it appears in as a prefix of longer phrase, say *vpx*, with *x* being some word, but *vp* also appears in at least one of these documents in itself (*i.e.*, ending a sentence or followed by a word other than *x*). In this case, in the phrase cluster corresponding to node *v*, the word *x* could be added to the phrase *vp* without changing it document set, thus it is not maximal.

We can however state a weaker point of an internal node for which all the leaves in its sub-tree originate from different documents does correspond to a maximal phrase cluster. This is true because if such an internal node *v* did not correspond to a maximal phrase cluster, then it would necessarily have a child whose set of leaves (in its sub-tree) equals the set of

leaves of node *v*. This would mean that *v* has a single child, which is contrary to the definition of a suffix tree.

After creating the suffix tree, we determine which nodes correspond to the maximal phrase clusters, and assign to each of these a score that is a function of the number of documents it contains, and its phrase. This is done in a single traversal of the tree. The score of a phrase cluster attempts to estimate its usefulness for clustering task. The score $s(m)$ of maximal phrase cluster m with phrase mp is given by:

$$s(m) = |m| \cdot f(|mp|) \cdot \sum \text{tfidf}(wi)$$

where $|m|$ is the number of documents in phrase cluster m, wi are the words in mp, tfidf(wi) is a score we calculate for each word in mp, and $|mp|$ is the number of words in mp that are not stop-words (*i.e.*, the effective length of the phrase). The function f penalizes single word phrases, is linear for phrases that are two to six words long, and becomes constant for longer phrases.

Term frequency – inverse document frequency (tfidf) is a commonly used IR technique for assigning weights to individual words. The rationale is that the more frequent the word is in a document (the term frequency portion) the more important it is in representing the document's *aboutness*. Likewise, the less frequent the word is in the whole document collection (the inverse document frequency portion) the more likely it is to be a good differentiator between documents, and thus more important to the IR task. We calculate the tfidf(wi,d) score of word wi in document d using the following formula:

$$tfidf(wi,d) = (1+ \log(tf(wi,d))) \cdot \log(1 + N/df(wi))$$

where tf(wi,d) is the number of occurrences of word wi in document d, N is the total number of documents in our document set and df(wi) is the number of documents that term wi appears in. However, we have found that the inverse document frequency component appears to be more accurate when we use the statistics collection, rather than of the retrieved document set. Therefore N will equal the number of documents collection and df(wi) is the number of documents collection that term wi appears in.

Finally, we select only the k highest scoring phrase clusters. This selection is designed to keep the cost of the next step constant, but we have found that it also has another important advantage: it prevents the next step from being influenced by low scoring, and thus presumably less informative, phrase clusters.

3.1.4 Phrase Merging Clusters

In the previous step, we have identified groups of documents that share phrases. However, documents may share more than one phrase. Therefore, the document sets of distinct phrase clusters may overlap and may even be identical. To avoid the proliferation of nearly identical clusters, the third step of the STC algorithm merges phrase clusters with a high overlap in their document sets (phrases are not considered in this step). We define a binary similarity measure between phrase clusters based on the overlap of their document sets. Given two phrase clusters mi and mj, with sizes $|mi|$ and $|mj|$ respectively, and $|mi \cap mj|$ representing the number of documents common to both phrase clusters, we define the similarity sim(mi,mj) of mi and mj to be:

sim(mi,mj) = 1 if $|mi \cap mj| / |mi| > \alpha$ and $|mi \cap mj| / |mj| > \alpha$

sim(mi,mj) = 0 otherwise

where α is a constant between 0 and 1 (we typically use α = 0.6). Next, we look at the *phrase cluster graph*, where nodes are phrase clusters, and two nodes are connected if and only if the two phrase clusters have a similarity of 1. A cluster is defined as being a connected component in the phrase cluster graph. We call these merged clusters. Each merged cluster contains the union of the documents of all its phrase clusters.

3.2 Performance Evaluation of proposed STDC Algorithm

The phrase clusters using the equivalent of a single-link clustering algorithm, where a predetermined minimal similarity between phrase clusters serves as the halting criterion. We do not encounter the undesired chaining effect of single-link clustering because in the realm of phrase clusters we typically find only small connected components.

We process the phrase clusters one at a time, in descending order, based on their scores. For each phrase cluster, we check its similarity to all phrase clusters already processed, and update the phrase cluster graph and the resulting clusters accordingly. Thus we can produce meaningful results even if an impatient user halts the algorithm in the middle of this phase.

As mentioned before, in this step we process only the k highest scoring phrase clusters. We also have a minimum phrase cluster score threshold and in addition discard the lower scoring 20% of the phrase clusters. This is important to keep the cost of this step constant, but it is also important in preventing this step from being influenced by low scoring, and thus presumably less informative, phrase clusters.

The final merged clusters are scored and sorted based on the scores of their phrase clusters and their overlap. Currently we use a simple scoring method: the score of a merged cluster is equal to the sum of the scores of its phrase clusters. As the final number of merged clusters can vary, we consider (and report) only the top few clusters. Typically, only the top 10-15 clusters are reported. We show that the non-incremental version of STDC has a linear time complexity. The incremental version of the algorithm is presented and analyzed in the following section.

3.3 Incremental Clustering

The cost of the first step, document parsing, is obviously linear with the document set length (the sum of the lengths of all the documents in the clustered document set). If we assume the length of each document to be bound by some maximal length, this step is linear with the number of documents to be clustered.

To construct the suffix tree algorithm, which is also linear with the document set length. In one depth-first traversal of the suffix tree we can determine which nodes correspond to maximal phrase clusters, calculate their scores, and select the top k phrase clusters. To determine maximal phrase clusters we must calculate the number of documents represented in the sub-tree of each node. For each node we determine the set of documents that is represented in its sub-tree by performing a union of the sets of all of its children. This process has the potential to be quadratic in the number of documents. But in our domain, sentences are very rarely more than twenty words long, and any longer sentences are split into sections of twenty words in length. This ensures us that the longest path from the root to any leaf in the suffix tree includes no more than twenty internal nodes. This, in turn assures that

calculating the number of documents represented the sub-tree of each node is linear.

The third step of the algorithm is quadratic in the number of phrase clusters being compared, but as we bound this by a constant *k*, we can treat this step as being of constant time. Thus, the overall time complexity of STDC is linear with regard to the document set length, or if we assume the length of the documents is bounded by some maximal length, STC is linear with the number of documents to be clustered.

We have presented the STDC algorithm and have shown it to be linear in the document set length, as well as incremental, if needed. These are crucial for any interactive clustering system. We now focus on two additional characteristics of the algorithm: being phrase based and generating overlapping clusters.

As mentioned previously, most clustering algorithms treat a document as a set of words. STDC identifies common phrases in the documents and uses them as the basis for clustering. This can improve the quality of the clusters by utilizing more information present in the text of the documents, and is useful in constructing concise and accurate descriptions of the clusters. We have noticed in own experiments that users often had a hard time interpreting the clusters when the clustering algorithm has a non-intuitive definition of a cluster. STDC used a simple cluster definition: all documents that contain at least one of the cluster's (hopefully related) phrases are members of that cluster. Allowing a document to appear in more than one cluster acknowledges that documents are complex objects that may be grouped into multiple potentially overlapping, but internally coherent, groups. This is actually the reason many IR system use some form of dot-product document similarity measure: it allows a document to be similar to multiple distinct documents or centroids that could in turn be very dissimilar from each other.

In STDC, as documents may share more than one phrase with other documents, each document might appear in a number of phrase clusters. Therefore, a document can appear in more than one cluster. Note that the overlap between clusters cannot be too high; otherwise they would have been merged into a single cluster. We believe that this feature is highly advantageous to the user in terms of understanding the clusters. But there is another advantage to this feature – if a document is erroneously placed in the wrong cluster (*e.g.*, because of a fluke high similarity with unrelated documents) the effects of this are not as bad as in algorithms with non-overlapping clusters. This is because in STDC the document also be placed in its "correct" cluster. The STDC algorithm does not require the user to specify the required number of clusters, as do many of the clustering algorithms. It does require, on the other hand, the specification of the threshold used to determine the similarity between phrase clusters.

3.4 Pseudo-code of Suffix Tree Document Clustering Technique:

1. Let Ni denote the intermediate tree that encodes all the suffixes from 1 to i

2. Tree N1 consists of a single edge between the root of the tree and a leaf labeled 1. The edge is labeled with the string S.

3. Tree Ni+1 is constructed from Ni as follows:

I. Starting at the root of Ni the algorithm finds the longest path from the root whose label matches a prefix of S[i+1..m]. This path is found by successively comparing and matching words in suffix S[i+1..m] to words along a unique path from the root, until no further matches are possible.

II. When no further matches are possible, the algorithm is either at a node, say w, or it is in a middle of an edge.

III. If it is in the middle of an edge, say (u,v), then it breaks (u,v) into two edges by inserting a new node w just after the last word on the edge whose label matches a prefix of the suffix S[i+1..m].

IV. In either case, the algorithm creates a new edge (w,i+1) running from w to a new

leaf labeled i+1, and labels the new edge with the unmatched part of suffix S [i+1..m].

Figure 1: Pseudo-code of Suffix Tree Document Clustering Technique

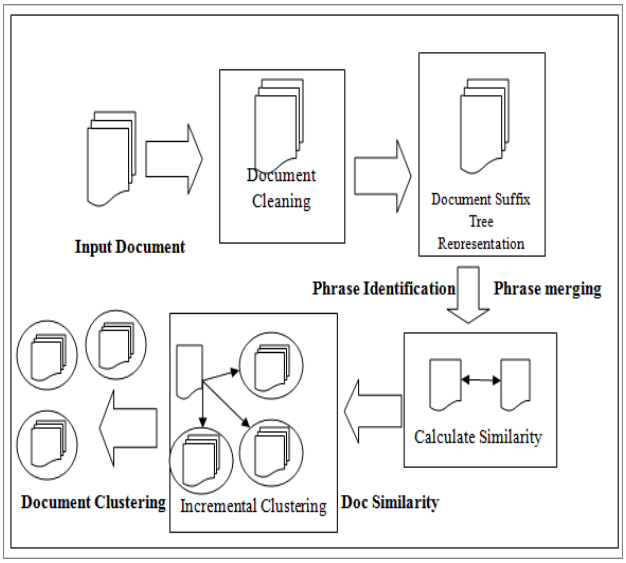

Figure2 Suffix Tree Document Clustering System Design

4. EVALUATION MATRICES AND RESULTS ANALYSIS

4.1 Evaluation Metrics for effectiveness of Cluster

In this paper we implement proposed Suffix Tree Document Clustering technique using MATLAB tool.

In order to evaluate the quality of the clustering, there are many different quality measures to access the cluster effectiveness. There are three quality measures are adopted, which are widely used in the text mining literature for the purpose of document clustering. The first is the F-measure, which combines the Precision and recall ideas in information retrieval. The second measure is the Purity which indicates the

percentage of the dominant in the given cluster. And finally measure the Entropy, which tells how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. To sum up, the approach maximizes the F-measure and minimizes the entropy score of the clusters to achieve a high-quality clustering.

In an information retrieval scenario, the instances are documents and the task is to return a set of relevant documents given a search term or equivalently, to assign each document to one of two categories, "relevant" and "not relevant". In this case, the "relevant" documents are simply those that belong to the "relevant" category. Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents, while precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search.

In information retrieval contexts, precision and recall are defined in terms of a set of retrieved documents (e.g. the list of documents produced by a web search engine for a query) and a set of relevant documents (e.g. the list of all documents on the internet that are relevant for a certain topic), cf. relevance.

4.2 Precision

In the field of information retrieval, precision is the fraction of retrieved documents that are relevant to the search:

$$\text{precision} = \frac{|\{\text{relevent document}\} \cap \{\text{retrieved document}\}|}{|\{\text{retrieved document}\}|}$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. Precision is also used with recall, the percent of all relevant documents that is returned by the search. The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system. Note that the meaning and usage of "precision" in the field of Information Retrieval differs from the definition of accuracy and precision within other branches of science and technology.

4.3 Recall

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{precision} = \frac{|\{\text{relevent document}\} \cap \{\text{retrieved document}\}|}{|\{\text{relevent document}\}|}$$

In binary classification, recall is called sensitivity. So it can be looked at as the probability that a relevant document is retrieved by the query. It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by computing the precision. Precision is the probability that a (randomly selected) retrieved document is relevant. Recall is the probability that a (randomly selected) relevant document is retrieved in a search.

4.4 F-measure

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Based on this formula the F-measure for overall quality of cluster set C is defined by the following formula:

$$F = \sum_{i=1}^{i} \left(\frac{|C|}{N} \cdot \text{MAX}\{F\} \right)$$

4.5 Purity

Cluster purity indicates the percentage of the document class members in the given cluster. For measuring the overall clustering purity we weighted average purity as follows:

$$\text{Purity} = \sum_{i=1}^{i} \left(\frac{|C|}{N} \cdot \text{MAX}\{\text{Precision}\} \right)$$

4.6 Entropy

Entropy is a measure of un-predictability or information content. To get an informal, intuitive understanding of the connection between these three English terms, consider the example of a poll on some political issue. Usually, such polls happen because the outcome of the poll isn't already known.

Shannon denoted the entropy H of a discrete random variable X with possible values $\{x_1, ..., x_n\}$ and probability mass function P(X) as,

$$H(X) = E\big(I(X)\big) = E(-\ln P(X)).$$

Here E is the expected value operator, and I is the information content of X.

I(X) is itself a random variable. The entropy can explicitly be written as

$$H(X) = \sum_{i=1}^{n} P(x_i)\, I(x_i) = \sum_{i=1}^{n} P(x_i) \log_b \frac{1}{P(x_i)} = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i).$$

Where b is the base of the logarithm used.

To sum up we would like to maximize the F-measure and Purity and the entropy score of the cluster to achieve the effectiveness.

4.7 Results Analysis

Our proposed fast and efficient phrase based Suffix Tree Document Clustering (STDC) techniques results analysis using performance evaluation matrices like F-measure, Purity and Entropy, table given below showed the values of all parameters and all the graphs described the performance of our proposed clustering techniques. The parameters are total number of files, word count, total number of nodes, sorted nodes, purity, entropy, F-measure, total number of strings, sorted string, total numbers of clusters, and total time (s). Total number of file in our results analysis are ten, word count are 0, 5, 5, 7, 8, 11, 12, 13, 14, and 14 respectively, total number of nodes are 0, 12, 14, 19, 23, 35, 37, 41, 47, and 52 respectively and all the parameters values are given in table 1.

Table 1 represents the parameters values of total number of files, word count, total number of nodes, sorted nodes, purity, entropy, F-measure, total number of strings, sorted string, total numbers of clusters, and total time (s).

Total Files	Word Count	Total Nodes	Sorted Node	**Purity**	**Entropy**	**F-Measure**	Total String	Sorted String	Total Clusters	**Time in Sec.**
1	0	0	0	0	0	0	1	0	0	0
2	5	12	2	0.968	0.515	0.711	12	2	2	0.046
3	5	14	5	0.903	0.599	0.712	14	5	2	0.075
4	7	19	5	0.915	0.542	0.957	19	5	2	0.118
5	8	23	7	0.808	0.672	0.762	23	7	4	0.152
6	11	35	9	0.831	0.533	0.989	35	9	5	0.311
7	12	37	10	0.848	0.628	0.796	37	10	5	0.346
8	13	41	12	0.84	0.906	0.771	41	12	5	0.462
9	14	47	13	0.808	0.672	0.771	47	13	6	0.658
10	14	52	16	0.831	0.989	0.989	52	16	4	0.788

Figure 3 represent the numbers of files vs. F-measure graph. The F-measure value linearly increases when numbers of files are low. With increase the files the F-measure values are varies in between 0.7 to 1.0.

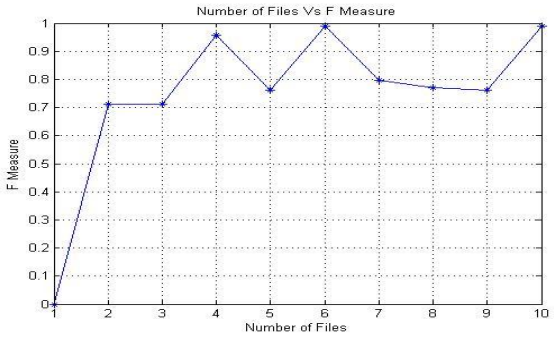

Figure 3 F-measure vs. Number of Files

Figure 4 represent the numbers of files vs. purity graph. The purity values are linearly increases up to 0.98 when numbers of files are low. With increase the files the purity values are varies in between 0.8 to 0.91.

Figure 5 represent the numbers of files vs. Entropy graph. The entropy value linearly increases up to 0.5 when numbers of files are low. With increase the files the entropy values are varies in between 0.5 to 0.9. In the graph the entropy value (0.9) maximum when numbers of files 8.

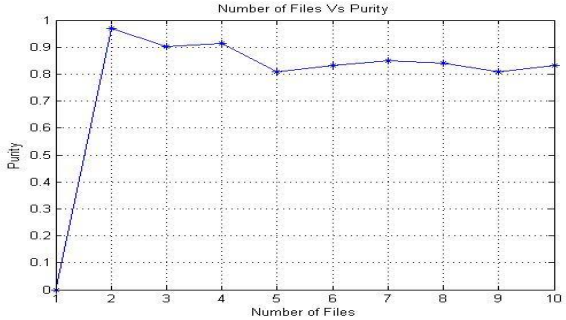

Figure 4 Purity vs. Number of Files

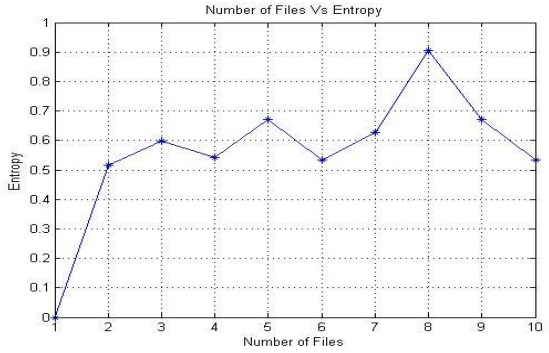

Figure 5 Entropy vs. Number of Files

Figure 6 represent the comparison graph of total string & sorted string with different numbers of files. In the graph blue line represents sorted string and the red line represents total strings in the document file. The sorted string graph represents more efficiency than total string graph.

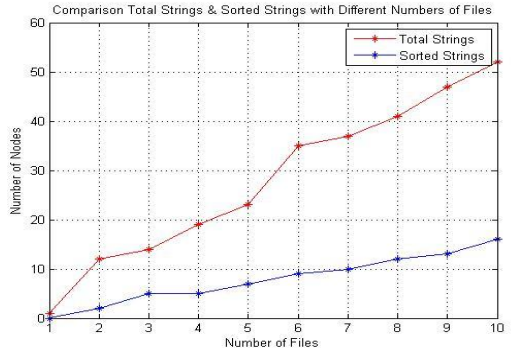

Figure 6 Comparison of total string & sorted string with different numbers of files

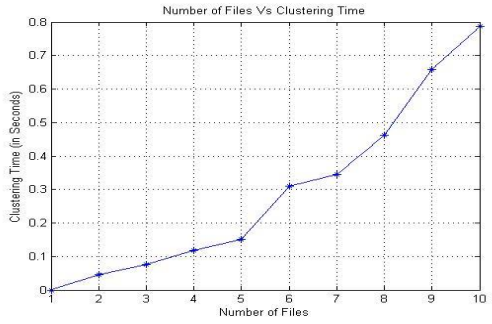

Figure 7 Numbers of Files vs Clustering Time (Seconds)

Figure 7 represent the numbers of files vs. clustering time in seconds' graph. In the graph of clustering time are slowly increases (0 to 0.15 sec) when number of files varies from 1 to 5. After then clustering times linearly (highly means 0.15 to 0.8 seconds) increases with respect to numbers of files increases (5 to 10).

5. CONCLUSION

Document clustering has initially been investigated in Information Retrieval mainly as a means of improving the performance of search engines by pre-clustering the entire corpus. The cluster hypopaper stated that similar documents will tend to be relevant to the same queries, thus the automatic detection of clusters of similar documents can improve recall by effectively broadening a search request. In our paper we are proposed a phrase based clustering scheme which based on application of Suffix Tree Document Clustering (STDC) model. Our proposed Suffix Tree Document Clustering (STDC) mechanism have four steps, these are Document collection as input, Document Cleaning, Phrase Identification of Clusters and Phrase Merging Clusters. We conclude that our proposed STDC technique is better perform based on document clustering than k-means clustering.

6. REFERENCES

[1] Shafiq Alam, Gillian Dobbie, Patricia Riddle, M. Asif Naeem, "Particle Swarm Optimization Based Hierarchical Agglomerative Clustering", 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 64-68.

[2] David Pettinger and Giuseppe Di Fatta, "Scalability of Efficient Parallel K-Means", IEEE e-Science 2009 Workshops, pp. 96-101.

[3] Yun Ling and Hangzhou, "Fast Co-clustering Using Matrix Decomposition", IEEE 2009 Asia-Pacific Conference on Information Processing, pp. 201-204.

[4] J. Prabhu and M. Sudharshan and M. Saravanan and G.Prasad, "Augmenting Rapid Clustering Method for Social Network Analysis", 2010 International Conference on Advances in Social Networks Analysis and Mining, pp. 407-408.

[5] F. Yang, T. Sun, C. Zhang, An efficient hybrid data clustering method based on K-harmonic means, and Particle Swarm Optimization, Expert Systems with Applications 2009, pp. 9847–9852.

[6] Y.-T. Kao, E. Zahara, I.-W. Kao, A hybridized approach to data clustering, Expert Systems with Applications 2008, pp. 1754-1762.

[7] Madjid Khalilian, Farsad Zamani Boroujeni, Norwati Mustapha, Md. Nasir Sulaiman, "K-Means Divide and Conquer Clustering", IEEE 2009, International Conference on Computer and Automation Engineering, pp. 306-309.

[8] Lan Yu, "Applying Clustering to Data Analysis of Physical Healthy Standard", 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010), pp. 2766-2768.

[9] Vignesh T. Ravi and Gagan Agrawal, "Performance Issues in Parallelizing Data-Intensive Applications on a Multi-core Cluster", 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 308-315.

[10] Maryam hajiee, "A New Distributed Clustering Algorithm Based on K-means Algorithm", 2010 3rd International Conference on Advanced Computer Theory and Engineering (1CACTE), pp. 408-411 (V2).