International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Construction of suffix tree using key phrases for document using down-top incremental conceptual hierarchical text clustering approach

Balaji Dhashanamoorthi *

Master of Engineering, Control and Instrumentation, CEG, Anna University, Chennai, India.

International Journal of Science and Research Archive, 2022, 06(01), 294–307

Publication history: Received on 23 May 2022; revised on 26 June 2022; accepted on 29 June 2022

Article DOI: https://doi.org/10.30574/ijsra.2022.6.1.0143

Abstract

With development of technologies in the World Wide Web, usage of document increases day by day. In order to access the document easily, document clustering technique is introduced. In the field of data mining, document clustering plays a vital role. Organizing the unstructured and unlabeled document is one of the major problems and it is ever growing and complex. Handling of such unorganized documents causes more expensive. Hence, challenges raised by the continuing growth of unstructured and unlabeled documents are handled in this proposed work. Document clustering is one of the most powerful methods to solve the problem of organizing unstructured documents. There are numerous clustering methods available. In this we were proposed phrase-based clustering algorithm, which is based on the applications of suffix tree document clustering model. The proposed algorithm is designed to use the suffix tree document clustering (STDC) model for accurate representation of document and similarity measurement of the similar documents. This proposed algorithm gives more than 90% accuracy.

Keywords: Suffix tree document clustering; Retrieval System; Document; Clustering Algorithm; Suffix Tree; Document Clustering

1. Introduction

The purpose of the paper is to improve the quality of document clustering by reducing the noise in the data. We first study the effectiveness of pre-processing of data representation and then applying the classical clustering methods. We then detect the effectiveness of a new clustering algorithm in which the noise is reduced by first clustering the features of the data and then clustering the data on the basis of their feature clusters.

The objective of a paper is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). A distance measure (or, dually, similarity measure) thus lies at the heart of document clustering. The large variety of documents makes it almost impossible to create a general algorithm which can work best in case of all kinds of datasets.

1.1. Analysis of datamining

Data mining (the analysis step of the "Knowledge Discovery in Databases" process, or KDD), an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets ("big data") involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing and model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, virtualization and online updating.

* Corresponding author: D. Balaji

1.2. Document clustering

It is an application of cluster analysis to textual document which is nothing but it divides a set of text into cluster, so that text within each cluster is similar in content

1.2.1. Clustering

The clustering problem can be formalized as an optimization problem, i.e. the minimization or maximization of a function subject to a set of constraints. The goal of clustering can be defined as follows:

Given

a dataset X = {X1, X2, …. , Xn}

the desired number of clusters k

a function that evaluates the quality of clustering

we want to compute a mapping

$$\gamma : \{1,2,.....,n\} \longrightarrow \{1,2,.....,k\}$$

that minimizes the function subject to some constraints.

The function that evaluates the clustering quality are often defined in terms of similarity between objects and it is also called distortion function or divergence. The similarity measure is the key input to a clustering algorithm.

1.2.2. Clustering application

Clustering is the most common form of unsupervised learning and is a major tool in a number of applications in many fields of business and science.

Hereby, we summarize the basic directions in which clustering is used.

Finding Similar Documents: This feature is often used when the user has spotted one “good” document in a search result and wants more-like-this. The interesting property here is that clustering is able to discover documents that are conceptually alike in contrast to search-based approaches that are only able to discover whether the documents share many of the same words.

Organizing Large Document Collections: Document retrieval focuses on finding documents relevant to a particular query, but it fails to solve the problem of making sense of a large number of uncategorized documents. The challenge here is to organize these documents in a taxonomy identical to the one humans would create given enough time and use it as a browsing interface to the original collection of documents.

Duplicate Content Detection: In many applications there is a need to find duplicates or near-duplicates in a large number of documents. Clustering is employed for plagiarism detection, grouping of related news stories and to reorder search results rankings (to assure higher diversity among the topmost documents). Note that in such applications the description of clusters is rarely needed.

Recommendation System: In this application a user is recommended articles based on the articles the user has already read. Clustering of the articles makes it possible in real time and improves the quality a lot.

Search Optimization: Clustering helps a lot in improving the quality and efficiency of search engines as the user query can be first compared to the clusters instead of comparing it directly to the documents and the search results can also be arranged easily.

1.2.3. Challenges in Document Clustering

Document clustering is being studied from many decades but still it is far from a trivial and solved problem. The challenges are:

- Selecting appropriate features of the documents that should be used for clustering.
- Selecting an appropriate similarity measure between documents.
- Selecting an appropriate clustering method utilizing the above similarity measure.
- Implementing the clustering algorithm in an efficient way that makes it feasible interms of required memory and CPU resources.
- Finding ways of assessing the quality of the performed clustering.

Furthermore, with medium to large document collections (10,000+ documents), the number of term-document relations is fairly high (millions+), and the computational complexity of the algorithm applied is thus a central factor in whether it is feasible for real-life applications. If a dense matrix is constructed to represent term-document relations, this matrix could easily become too large to keep in memory - e.g. 100, 000 documents × 100, 000 terms = 1010 entries ∼ 40 GB using 32-bit floating point values. If the vector model is applied, the dimensionality of the resulting vector space will likewise be quite high (10,000+). This means that simple operations, like finding the Euclidean distance between two documents in the vector space, become time consuming tasks.

1.2.4. Motivation

We are developing an application for recommendations of news articles to the readers of a news portal. The following challenges gave us the motivation to use clustering of the news articles:

- The number of available articles is large.
- A large number of articles are added each day.
- Articles corresponding to same news are added from different sources.
- The recommendations had to be generated and updated in real time.

By clustering the articles we could reduce our domain of search for recommendations as most of the users had interest in the news corresponding to a few number of clusters. This improved our time efficiency to a great extent. Also we could identify the articles of same news from different sources.

The main motivation of this work has been to investigate possibilities for the improvement of the effectiveness of document clustering by finding out the main reasons of ineffectiveness of the already built algorithms and get their solutions.

1.3. Existing system

Text clustering is of great practical importance and is widely employed for automatically structuring large document collections today. With the rapid growth of information and the explosion of electronic text from the complex World Wide Web, more and more knowledge you needed is included. But, the massive amount of text also takes so much trouble to people for finding useful information. For example, the standard Web search engines have low precision, since typically some relevant Web pages are returned mixed with a large number of irrelevant pages. So, one appropriate way of organizing this overwhelming amount of document is necessary. Clustering algorithms take a set of unlabeled items and group them into some larger sets. Unlike text classification, which is the process of assigning predefined category labels to new documents based on the classifier learnt from a large, often prohibitive, number of labeled training. text clustering does not require any training data. But, it is often an ill-defined problem. That is, clustering does not have very clear objectives. For another example, k-means which mainly focuses on distance-based cluster analysis, can be applied to find clusters in many statistical analysis systems only when the mean of a set of objects is defined. And, the necessity for users is how to specify k, the number of clusters, which can be seen as a disadvantage. The problem of specifying k is one of the most challenging issues involved with it, since there is really no good solution that can predict the optimal number of clusters to use in every possible situation. The best assignment method of k largely depends on the experiments on the task and data set being considered. So, in other words, most of the time, users have no explicit clues to choose the best k.

1.3.1. Disadvantage

- Clustering does not have very clear objectives.
- The necessity of users is how to specify k, the number of clusters.
- Users have no explicit clues to choose the best k.

1.4. Proposed system

We present a novel down-top incremental conceptual hierarchical text clustering approach. The method is implemented based on a CFU-tree that represents a cluster hierarchy, which makes it effective for incremental and dynamic clustering of incoming objects. The down-top hierarchical clustering approach starts with each input as a separate cluster. That is, an input item is defined a single cluster. The clustering process will proceed by joining two or several existing clusters to form a new one, until certain termination conditions are satisfied. We use the term-based feature extraction to summarize a cluster and use data matrix to store the cluster feature vectors in text clustering. In order to solve the problem of "how to choose k", during clustering, a measure criterion, called Comparison Variation (CV), is also used for judging whether the clusters can be merged or split. The text clustering method is not sensitive to the input data order.

The proposed algorithm is designed to use the STDC model for accurate representation of document and similarity measurement of the similar documents. This method will reduce 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 used a suffix tree to efficiently identifying these common phrases. We have shown STDC algorithm to be linear in the length of the document set, incremental, and to produce overlapping Clusters.

1.5. Modules

- Document Collection
- Pre-Processing
- Phrase Identification
- Phrase Merging

1.6. Block diagram

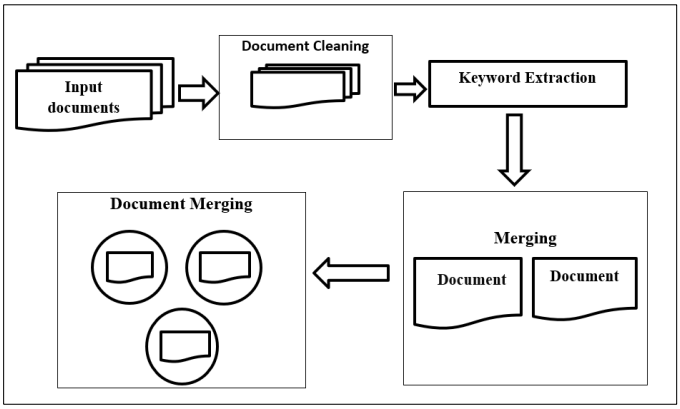

Figure 1 Block diagram of document clustering

1.7. Module description

1.7.1. Document collection

The documents are collected and maintained as datasets. The collected document can be either text documents or web document but the proposed algorithm performs the clustering on the text documents.

1.7.2. Pre-processing

Data scrubbing is the pre-processing of the data, through which data is cleaned and pre-processed that is input to the next step to the Suffix Tree structure. Pre-processing include the steps such as Tokenization, Stop word removal and Stemming.

Tokenization

Tokenization is the pre-processing step, in which sentence are divided into tokens. It is a process of identifying the word and sentence boundary in the text. In simple word, Tokenization means the white space character as a word delimiters and selected punctuation mark such as .",?" And, Token-id is assigned to each word.

Stop-word removal

There are many words in the document that contain no information about the topic, shortly stop words are not helpful to us find the concept of the documents. Such word doesn't carry any semantic meaning. The words occur in the stop list -are, and, but, will, have etc.

Stemming algorithm

In the stemming procedure all words in the text document are replaced with their respective stem. It is a process of reducing some derived words into their root term. Different form of words can be reduced into one base form by using the stemmer. Lots of stemmer algorithm is created for the English language. The process of stemmer development is easy.

1.7.3. Phrase identification

The STDC algorithm is used in this step to identify all maximal phrase clusters by using a suffix tree. The identification of phrase clusters is similar to the creation of an inverted index of phrases for the document collection. The phrase identification clusters are assigned with score value and they are selected based on their highest score. A suffix tree is created from all the sentences (as defined above sequence of words between phrase boundaries) of all documents in document set. Each leaf in the suffix tree is marked with a sentence identifier in order to identify which document it belongs to.

Each internal node v in the suffix tree represents a group of documents and a phrase that is common to all of them. The label vp is the common phrase of an internal node v. All the leaves in the sub-tree of v correspond to the sentences in which the suffixes start with the phrase vp. By using this approach, 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 of the document and where). It is clear from the above that every possible maximal phrase cluster corresponds to an internal node in suffix tree, the phrase of the phrase cluster equals the label of a node, and the set of documents in the phrase cluster equals the set of documents designated by the leaves in that node's sub-tree.

In order to prove the truth mentioned above, a maximal phrase cluster m with phrase mp is taken. Consider a phrase cluster m with at least two documents i and j that have sentences with suffixes that contain mp. It emphasizes that there exists a path in the suffix tree from the root and its label starts with the phrase mp. 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 because there are two sentences (from different documents) that share this phrase.

The phrase cluster must include all the documents that contain the phrase mp corresponding to node u in which m is a maximal phrase cluster. mp must be equal to a word and it will be added to mp (the next word in up after the prefix mp) without decreasing the number of documents in the phrase cluster. In the above case, the reverse is not true.

An internal node in the suffix tree might not correspond to a phrase cluster because all the leaves in its sub-tree might originate from suffixes of sentences from a single document. An internal node with leaves in its sub-tree from at least two different documents corresponds to a phrase cluster but not necessarily to a maximal phrase cluster.

This problem takes place 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). To overcome this difficulty, a phrase cluster corresponding to node v, the word x is added to the phrase vp without changing it document set, thus it is not maximal.

A weaker point of an internal node for which all leaves in its sub-tree originate from different documents does correspond to a maximal phrase cluster. If 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. It is made clear that v has a single child, which is contrary to the definition of a suffix tree. After creating the suffix tree, the nodes corresponding to the maximal phrase clusters are assigned a score in a single traversal of the tree that is a function of the number of documents it contains and its phrase. The score of a phrase cluster is used to estimate its usefulness for clustering task.

The score s(m) of the maximal phrase cluster m with the phrase mp is given by:

$$s(m) = |m| \cdot f(|mp|) \cdot \Sigma\, tfidf(wi)$$

where |m| is the number of documents in phrase cluster m, wi are the words in mp, tfidf(wi) is a score 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 fines single word phrases which is linear for phrases that are two to six words long and becomes constant for longer phrases. The term frequency inverse document frequency (tfidf) is a commonly used IR technique for assigning weights to individual words.

If there are more frequent words in a document (the term frequency portion), then it is more important in representing the document's abruptness. If the words are less frequent in the whole document collection (the inverse document frequency portion), then there is more probability to be a good differentiator between documents, and thus more important to the IR task.

The tfidf(wi,d) score of word wi in document d is calculated 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 document set and df(wi) is the number of documents in which the term wi appears in.

However the inverse document frequency component appears to be more accurate in the case of 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 in which the term wi appears in.

Finally, the k highest scoring phrase clusters is selected. This collection is designed to keep the rate of the next step constant. It also has another important advantage that it prevents the next step from being influenced by low scoring and thus presumably less educational phrase clusters.

1.7.4. Phrase merging

In phrase identification, the groups of documents that share phrases are identified. The document sets of different phrase clusters may intersect and may even be identical because documents may segment more than one phrase. To avoid this 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).

A binary parallel measure between phrases clusters are defined based on the overlap of their document sets. Consider two phrase clusters mi and mj, with sizes |mi| and |mj| respectively where |mi ∩ mj| represent the number of documents common to both phrase clusters. A similarity sim(mi,mj) of mi and mj is defined to be:

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

$$sim(mi, mj) = 0 \text{ otherwise}$$

where α is a constant between 0 and 1 (we typically use α = 0.6). In phrase cluster graph, the nodes are phrase clusters. The two nodes in the phrase cluster graph are connected if and only if the two phrase clusters have a similarity of 1. A cluster is a connected component in the phrase cluster graph. In which, the cluster contains the union of the documents of all its phrase clusters is called merged cluster.

1.8. Pseudo-code of suffix tree document clustering technique

Given a string S of length m, enter a single edge for suffix S[l...m]$ (the entire string) into the tree, then successively enter suffix S[i..m]$ into the growing tree, for i increasing from 2 to m. Let Ni denote the intermediate tree that encodes all the suffixes from 1 to i.

So Ni+1 is constructed from Ni as follows:

Start at the root of Ni

Find the longest path from the root which matches a prefix of S[i+1..m]$

Match ends either at the node (say w) or in the middle of an edge [say (u, v)].

If it is in the middle of an edge (u, v), break the edge (u, v) into two edges by inserting a new node w just after the last character on the edge that matched a character in S[i+l..m] And just before the first character on the edge that mismatched. The new edge (u, w) is labeled with the part of the (u, v) label that matched with S[i+1..m], and the new edge (w, v) is labeled with the remaining part of the (u, v) label.

Create a new edge (w, i+1) from w to a new leaf labeled i+1 and it labels the new edge with the unmatched part of suffix S[i+1..m]

This takes O(m2) to build the suffix tree for the string S of length m.

Suffix extension is all about adding the next character into the suffix tree built so far. In extension j of phase i+1, algorithm finds the end of S[j..i] (which is already in the tree due to previous phase i) and then it extends S[j..i] to be sure the suffix S[j..i+1] is in the tree.

There are 3 extension rules:

- Rule 1: If the path from the root labeled [j..i] Ends at leaf edge (i.e. S[i] is last character on leaf edge) then character S[i+1] is just added to the end of the label on that leaf edge.
- Rule 2: If the path from the root labeled [j..i] ends at non-leaf edge (i.e. there are more characters after S[i] on path) and next character is not S[i+1], then a new leaf edge with label S{i+1] and number j is created starting from character S[i+1].A new internal node will also be created if S[1..i] ends inside (in-between) a non-leaf edge.
- Rule 3: If the path from the root labeled [j..i] Ends at non-leaf edge (i.e. there are more characters after S[i] on path) and next character is S[i+1] (already in tree), do nothing.

One important point to note here is that from a given node (root or internal), there will be one and only one edge starting from one character. There will not be more than one edges going out of any node, starting with same character.

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.

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 analysed in the following section.

2.1. Evaluation metrics for effectiveness of cluster

In this paper, we implement a proposed Suffix Tree Document Clustering Technique using Java tool. In order to evaluate the quality of the clustering. There are many different quality measures to access the cluster effectiveness. Here we are going to use F-measure which is widely used in the text mining literature for the purpose of document clustering. F-measure which combines the Precision and Recall ideas in information retrieval.

2.2. Precision

Precision for clustering is the fraction of objects assigned that belong to an explicit class, which measures how well it is doing at rejecting irrelevant class. Let C = {C1, . . ., Cr} be a set of clusters after clustering, and K = {K1, . . ., Kt} be the set of the true classes of the collection. For each cluster Ci, MaxCi (Kj) is the set of objects associated with class label Kj in Ci, and the number of MaxCi (Kj) is the most in the set of C.

$|MaxCi\ (Kj)| = \max \{||Ci \cup Kj\ ||Ci \in C, Kj \in K\}$

So, the precision of cluster Ci is shown as follows:

Precision (Ci) = |MaxCi (Kj)| |Ci|

2.3. Recall

. Recall is the proportion of objects associated with label Kj and all of the Kj class, which measures how well it is doing at finding all the objects in cluster with the label Kj. Similarly, we define Recall as follows

Recall (Ci) = |MaxCi (Kj)| |Kj |

2.4. F-measure

F-Measure, which is defined as the harmonic mean of Precision and Recall value, is also used to measure the performance of our method. For different specific request, according to the importance of the Precision and Recall, we define F-Measure as follows:

$F=2*\frac{Precision.Recall}{Precision+Recall}$

2.5. Performance analysis

Our proposed fast and efficient phrase based Suffix Tree Document Clustering (STDC) techniques and agglomerative hierarchical clustering result analysis using performance evaluation matric. Table below shows the value of parameters and graph described the performance of our proposed clustering techniques.

2.6. Agglomerative hierarchical clustering

Performance of the proposed agglomerative hierarchical clustering table

Table 1 Precision, Recall and total number of cluster in the agglomerative hierarchical clustering.

Number of Cluster	Precision (%)	Recall (%)
C1	92.3	95.2
C2	94.23	92
C3	90.28	94.11

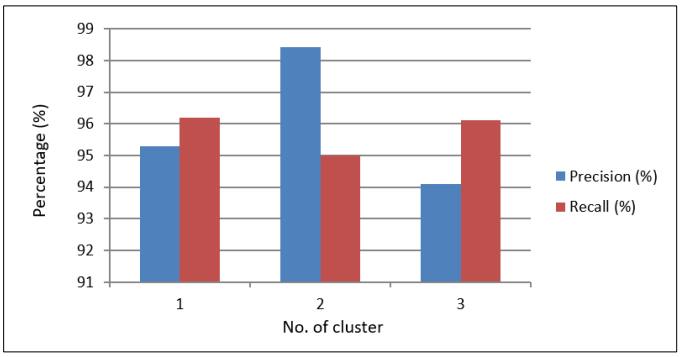

Figure 2 Performance of the proposed agglomerative hierarchical clustering graph

Figure 2 represents the performance graph of proposed agglomerative hierarchical clustering. This graph is constructed with precision and recall values of proposed technique.

2.7. Suffix tree document clustering

Table 2 Performance of the proposed Suffix tree document clustering table

Number of Cluster	Precision (%)	Recall (%)
C1	95.3	96.2
C2	98.42	95
C3	94.1	96.11

Table 3 represents the Precision, Recall and Total number of cluster in the suffix tree document clustering.

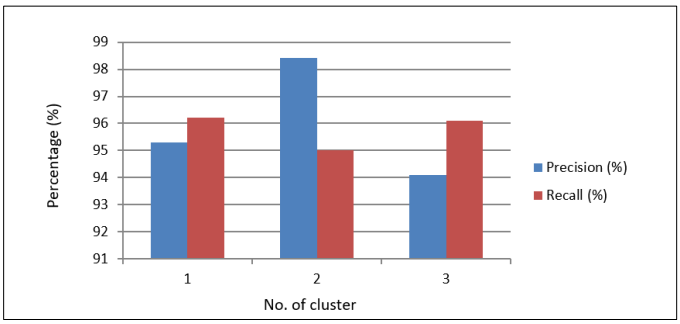

Figure 3 Performance of the proposed Suffix tree document clustering graph

Figure 3 represents the performance graph of proposed suffix tree document clustering. This graph is constructed with precision and recall values of proposed technique.

2.7.1. Comparison of agglomerative heirarchical clustering and suffix tree document clustering

Table 3 Compared performance of the proposed precision table

Number of Cluster	Precision (%) (AHGC)	Recall (%) (STDC)
C1	92.3	95.3
C2	94.23	98.42
C3	90.28	94.1

Table 3 represents the precision values of the proposed Agglomerative hierarchical clustering and Suffix tree document clustering.

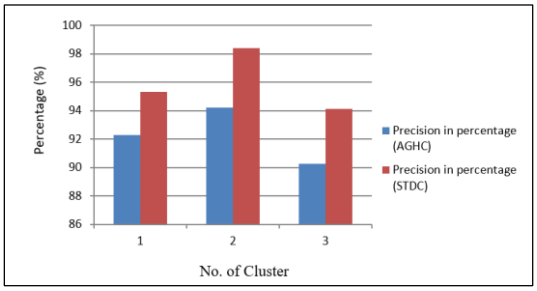

Figure 4 Compared performance of the proposed Precision graph

Figure 4 represents the compared performance graph of the proposed agglomerative hierarchical clustering and suffix tree clustering techniques. This graph is constructed with precision values of proposed techniques. Fig 5.3 shows that the proposed STDC is better than the AGHC technique.

Table 4 Compared performance of the proposed Recall table

Number of Cluster	Precision (%) (AHGC)	Recall (%) (STDC)
C1	95.2	96.2
C2	92	95
C3	94.11	96.11

Tables 4 represents the number of cluster and recall values of the proposed Agglomerative hierarchical clustering and Suffix tree document clustering.

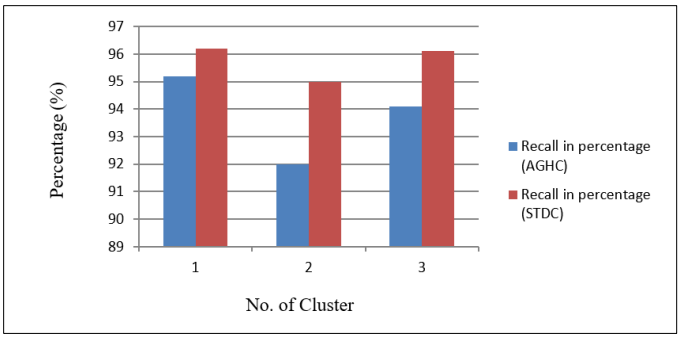

Figure 5 Compared performance of the proposed Recall graph

Figure 5 represents compared performance graph of the proposed agglomerative hierarchical clustering and suffix tree document clustering. This graph is constructed with recall values of proposed techniques. Fig 5.4 shows that the proposed STDC is better than the AGHC technique.

Table 5 Compared performance of the proposed techniques table

Number of Cluster	Precision (%) (AHGC)	Recall (%) (STDC)
C1	95	98
C2	94.11	96.12
C3	93.25	97.26

Table 5 represents the values of total number of cluster and F-measure of the proposed Agglomerative hierarchical clustering and Suffix tree document clustering.

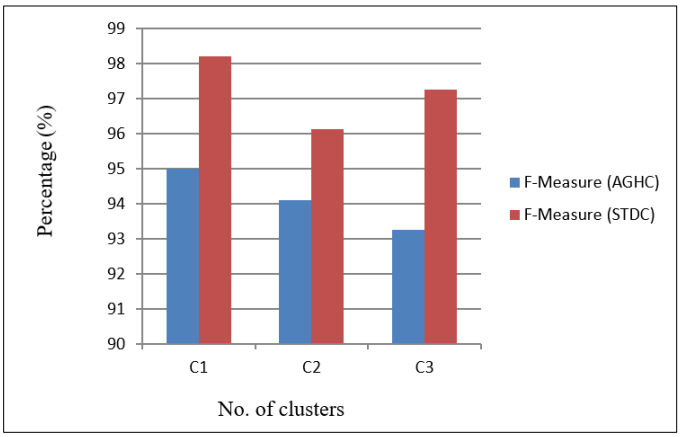

Figure 6 Compared performance of the proposed techniques graph

Figure 6 represent the compared performance graph of the proposed agglomerative hierarchical clustering and suffix tree document clustering. This graph is constructed with F-Measure values of proposed techniques. Fig 5.5 shows that the proposed STDC is better than the AGHC technique.

Table 6 Overall performance of the proposed techniques table

F-Measure in %(AGHC)	F-Measure in % (STDC)
94.12	97.19

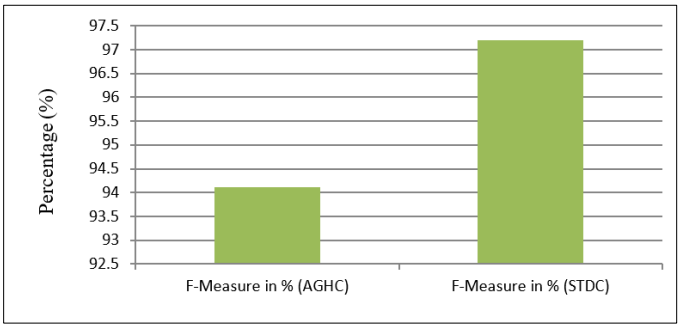

Figure 7 Overall performance of the proposed techniques graph

Table 6 represents the average F-measure values of the proposed Agglomerative hierarchical clustering and Suffix tree document clustering.

Figure 7 represent the overall performance graph of the proposed agglomerative hierarchical clustering and suffix tree document clustering techniques. This graph is constructed with F-Measure values of proposed techniques. Fig 5.6 shows that the proposed STDC is better than the AGHC technique.

In experiments, totally 100 documents of three different categories are used for clustering purpose. The system was clustered according to the category selected by the user. The categories include: (1) Data mining documents, (2) operating system documents, (3) computer documents.

- Cluster (C1) consists of 50 documents in which 28 are relevant and 22 are irrelevant to the category.
- Cluster (C2) consists of 35 documents in which 20 are relevant and 15 are irrelevant to the category.
- Cluster (C3) consists of 15 documents in which 11 are relevant and 4 are irrelevant to the category.

Table 1, 2, 3, 4, 5, 6 represents the calculation of F-Measure using the values of relevant and irrelevant documents of clusters which are mentioned above. The corresponding graph is shown in figure 5.1, 5.2, 5.3, 5.4, 5.5, 5.6

Proposed system omits nearly 500 stop words. The stop words in document corpus are removed before distinct words extraction. Distinct words are extracted and their frequency of occurrence is calculated. Based on the frequency, The probability of the keyword is calculated. The proposed document clustering algorithm performs with 90% efficiency for all categories of documents. On the whole, Suffix tree document clustering is better than the Agglomerative hierarchical clustering which is based on the performance measure.

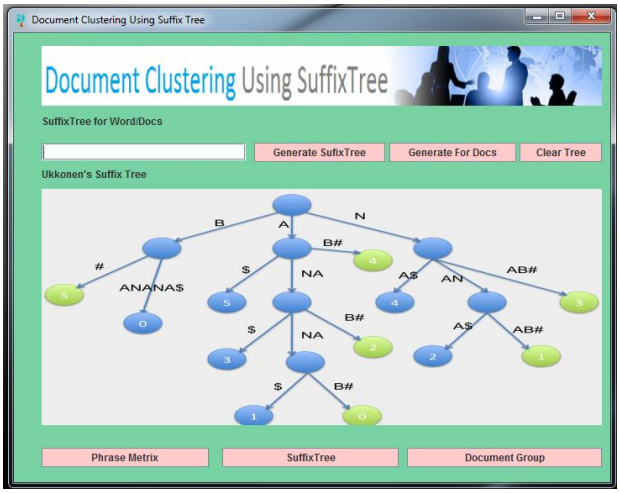

Figure 8 Document clustering

Figure 8 represents front page of the document clustering. This page is constructed using java swing.

Figure 9 Selecting the documents

Figure 9 represents the selection of documents for document clustering. This page is constructed using java swing.

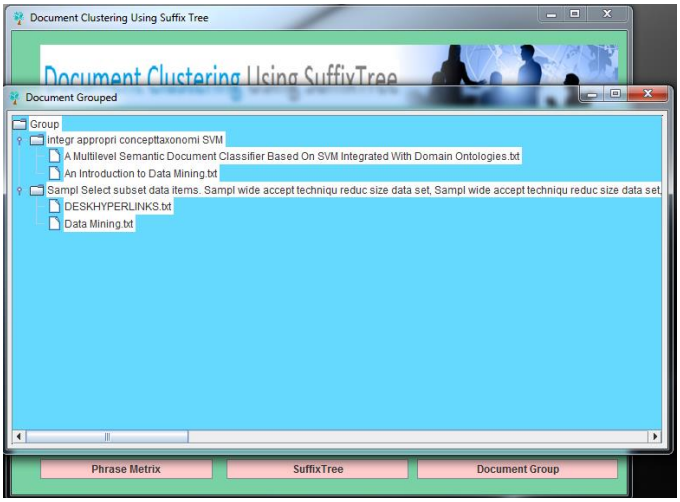

Figure 10 Grouping the document

Figure 10 represent the construction of document clustering. This page is constructed using java swing.

3. Conclusion

In this paper an algorithm was proposed for document clustering. This study explains the possibility of Phrase based clustering scheme which is based on application of suffix tree as a vital approach for document clustering. Our proposed Suffix Tree Document Clustering (STDC) mechanism have four steps, they are document collection as input, Document cleaning, Phrase identification of clusters and phrase Merging clusters. On the whole, the challenges caused by the continuing growth of unstructured unlabeled documents are reduced by this proposed algorithm. The performance graph represented in the paper shows, the proposed STDC technique is better than the AGHC technique.

Future enhancement

In this work suffix tree clustering is used to cluster the documents. It generates and describes clusters based on common phrases that are found in the document set. In future, different methods can be used to extract the keywords and clustering the documents to improve the overall performance of document clustering. To make the document clustering more efficient, merging can be done by some other measures.

References

[1] B. Liu, Y. Dai, X. Li, W.S. Lee, P.S. Yu, "Building text classifiers using positive and unlabeled documents", in: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), Melbourne, FL, USA, Vol. 16, 2003, pp. 179–188.

[2] K.A. Heller, Z. Ghahramani, "Bayesian hierarchical clusters", in: Proc. 22nd Int. Conf. Machine Learning (ICML'05), Bonn, Germany, Vol. 22, 2005, pp. 297–304.

[3] J. Hu, L. Fang, Y. Cao, H.J. Zing, H. Li, Q. Yang, Z. Chen, "Enhancing text clustering by leveraging Wikipedia semantics", in: SIGIR'08 Proceedings of the 31stAnnual International ACM SIGIR Conference on Research and Development in Information Retrieval, Vol. 15, 2008, pp. 179–186.

[4] F. Beil, M. Ester, X. Xu, "Frequent term-based text clustering", in: KDD'02Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Vol. 11, 2002, pp. 436–442.

[5] K. Dave, S. Lawrence, D.M. Pennock, "Mining the peanut gallery: opinion extraction and semantic classification of product reviews", in: Proceedings of the 12thInternational Conference on World Wide Web, Vol. 38, 2003, pp. 519–528.

[6] Z. Molnar, P. Berate, "Enhancing competitiveness of innovative SME through knowledge system using competitive intelligence (case: business cluster Omni pack)", in: 7th International Conference on Strategic

Management and its Support by Information Systems, Calando, Czech Republic, Vol. 27, September 05–06,2007, pp. 59–68.

[7] G. H. John, P. Langley, “Estimating continuous distributions in bayesian classifiers”, in: Eleventh Conference on Uncertainty in Artificial Intelligence, Vol. 11, 1995, pp. 338–345.

[8] K. Kira, L. A. Rendell, “A practical approach to feature selection”, in: Ninth International Workshop on Machine Learning, Morgan Kaufmann, Vol. 19, 1992, pp. 249–256.

[9] F. Pereira, N. Tishby, L. Lee, “Distributional clustering of english words”, in: Proceedings of the 31st Annual Meeting on Association for Computational Linguistics, Vol. 3, 1993, pp. 183–190.

[10] N. Tishby, F. Pereira, W. Bialek, “The information bottleneck method”, in: Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing, Vol. 22, 1999, pp. 368–377.

[11] L. Baker, A. McCallum, “Distributional clustering of words for text classification”, in: Proceedings of the 21st Annual international ACM SIGIR conference on Research and development in information retrieval, ACM, Vol. 6, 1998, pp. 96–103.

[12] Guimei, et al., “ A flexible approach to finding representative pattern sets” , IEEE Trans. on Knowledge and Data Mining, Vol. 26, No. 7, 2014, pp. 1562-1574.

[13] Victorial Nebot et al., “Finding association rules in semantic web data”, Knowledge based systems, Elsevier, Vol.30, No, 1, 2012, pp. 51-62.

[14] T. Kohonen, “Self-organization of very large document collections: State of the art”, in Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks, Vol. 1, 1998, pp. 65–74.

[15] R. Feldman, I. Dagan, and H. Hirsh, “Mining text using keyword distributions”, Journal of Intelligent Information Systems, Vol. 10, 2002, pp. 281–300.

[16] Dhashanamoorthi, Balaji. "EFFICIENCY IMPROVEMENT ON WIND TURBINE THROUGH BUMP UP STEPPER MOTOR" (2022).