# A NOVEL DOCUMENT CLUSTERING FOR ORGANIZING THE WEB PAGES

**Subhashini R.**[1] **and Jawahar Senthil Kumar V.**[2]

[1] Research Scholar, Sathyabama University, Chennai-119, India
[2] Assistant Professor, Anna University, Chennai, India
E-mail: [1]subhaagopi@gmail.com

**Abstract**

The exponential growth of information on the World Wide Web has prompted for developing efficient and effective methods for organizing and retrieving the information. Clustering techniques play an important role in searching and organization of web pages. In this paper we proposed an approach for web search results clustering based on a new document clustering algorithm. It is an alternative to a single ordered result of search engines. This approach presents a list of clusters to the user. This believes that clusters of search results are easier to browse than a single ordered list. Experimental results show that the new approach has better performance than that of conventional web search result clustering.

**Keywords:** Information Retrieval, Search results organization, document clustering, incremental clustering, new suffix tree.

## I. INTRODUCTION

Document clustering is widely applicable in areas such as search engines, web mining, information retrieval, and topological analysis. Most document clustering methods perform several preprocessing steps including stop words removal and stemming on the document set. Document clustering is an automatic grouping of text documents into clusters so that documents within a cluster have high similarity in comparison to one another, but are dissimilar to documents in other clusters. The IR community has explored document clustering as an alternative method of organizing retrieval results [13]. The majority of search engines [3, 6] give a long list of ranked documents; most of them are irrelevant. Most of the web search engines are also characterized by extremely low precision. Typical queries retrieve hundreds of documents, most of which have no relation with what the user was looking for. Hence it is difficult for the users to find relevant document and it is also time consuming. Clustering algorithms attempt to group documents together based on their similarities; thus documents relating to a certain topic will hopefully be placed in a single cluster. Clustering technique relies on four concepts: data representation model, similarity measure, clustering model and clustering algorithm [16] that generates the clusters using the data model and the similarity measure. This can help users both in locating interesting documents more easily and in getting an overview of the retrieved document set. Several researchers have suggested that the clustering techniques are feasible for web mining. The objective of our work is to develop a document clustering algorithm to categorize the Web documents in an online community. Such a clustering result is absolutely helpful in speeding up the knowledge collaboration in the online community.

## II. LITERATURE SURVEY

Document clustering [7] has been traditionally investigated mainly as a means of improving the performance of search engines by pre-clustering the entire corpus [2] (the cluster hypothesis - van Rijsbergen, 79).Numerous documents clustering algorithms appear in the literature [1]. Agglomerative Hierarchical Clustering (AHC) algorithms are probably the most commonly used. These algorithms are typically slow when applied to large document collections. Single-link and group-average methods typically take O(n2) time, while complete-link methods typically take O(n3) time [15]. As our experiments demonstrate, these algorithms are too slow to meet the speed requirement for one thousand documents. Linear time clustering algorithms are the best candidates to comply with the speed requirement of on-line clustering. These include the K-Means algorithm - O(nkT) time complexity where k is the number of desired clusters and $T$ is the number of iterations [9] and the Single Pass method - O(nK) were $K$ is the number of clusters created [4](Hill, 68). One advantage of the K-Means algorithm is that, unlike AHC algorithms, it can produce overlapping clusters. Its chief disadvantage is that it is known to be most effective when the desired clusters are approximately spherical with respect to the similarity measure used. Document clustering has been investigated as a post-retrieval document browsing technique. Most clustering algorithms base on two document model : the vector space document (VSD) model [11] and the suffix tree document model [5]. Clustering method base on VSD model such as K-Means and agglomerative hierarchical clustering (AHC) cluster the document according to the similarity of vectors which represent documents in the defined vector space. There methods can improve the clustering quality, but they are suitable for “off-line” clustering situation due to time efficiency.

Zamir [5] has shown that STC outperforms other algorithms. The main advantages of STC over other clustering algorithms are that it uses phrases rather than words, and that it allows clusters to overlap. Hau Jun Zeng and etc. [12] introduced an improved suffix tree with n-gram to deal with the problem of the original suffix tree. But STC with n-gram give too many base clusters. Chim and Deng [10] have proposed a new clustering algorithm combine the advantages of two document models in document clustering. The research on document clustering has been stared over the past several years. This work is primarily based on the work of zamir’s. Compared to aforementioned work, the new document clustering algorithm we proposed is to improve the quality in clustering web page snippets, and the clustering speed can meet the demand of “on the fly” mode.

## III. OVERVIEW OF STC

In this section, firstly the key requirements of web document clustering pointed out by Zamir [5] is discussed and then a brief review of suffix tree document model and STC algorithm is given.

### *A. Key Requirements for Web Document Clustering*

1. Relevance: The method ought to produce clusters that group documents relevant to the user’s query.

2. Browsable Summaries: The user needs to determine at a glance whether a cluster’s contents are of interest. We do not want to replace sifting through ranked lists with sifting through clusters. Therefore the method has to provide concise and accurate descriptions of the clusters.

3. Overlap: Since documents have multiple topics, it is important to avoid confining each document to only one cluster.

4. Snippet-tolerance: The method ought to produce high quality clusters even when it only has access to the snippets returned by the search engines, as most users are unwilling to wait while the system downloads the original documents off the Web.

5. Speed: A very patient user might sift through 100 documents in a ranked list presentation. We want clustering to allow the user to browse through at least an order of magnitude more documents. Therefore the clustering method ought to be able to cluster up to one thousand snippets in a few seconds. For the impatient user, each second counts.

6. Incrementality: To save time, the method should start to process each snippet as soon as it is received over the Web.

### *B. STC Algorithm*

This algorithm was developed by Zamir and Etzioni [5] in 1998. Based on this algorithm they have developed the clustering engine named Grouper. Suffix Tree Clustering (STC) is a linear time clustering algorithm that is based on identifying the phrases that are common to groups of documents. A phrase is an ordered sequence of one or more words. The base cluster is a set of documents that share a common phrase.STC has three logical steps: (1) document "cleaning", (2) identifying base clusters using a suffix tree, and (3) combining these base clusters into clusters.

**Step 1 – Document "Cleaning"**

In this step, the string of text representing each document is transformed using a light stemming algorithm (delete word prefixes & suffixes and reducing plural to singular).Sentence boundaries (identified via punctuation and HTML tags) are marked and non word tokens (such as numbers, HTML tags and most punctuation) are stripped.

**Step 2 – Identifying Base Clusters**

The identification of base clusters can be viewed as the creation of an inverted index of phrases for the document collection. This is done efficiently using a data structure called a suffix tree. This structure can be constructed in time linear with the size of the collection, and can be constructed incrementally as the documents are being read. The idea of using a suffix tree for document clustering was first introduced in 1997. Each node of the suffix tree represents a group of documents and a phrase that is common to all of them. Therefore, each node represents a base cluster. Furthermore, all possible base clusters (containing 2 or more documents) appear as nodes in our suffix tree. Each base cluster is assigned a score that is a function of the number of documents it contains and the words that make up its phrase.

**Step 3 – Combining Base Clusters**

Documents may share more than one phrase. As a result, the document sets of distinct base clusters may overlap and may even be identical. To avoid the proliferation of nearly identical clusters, the third step of the algorithm merges base clusters with a high overlap in their document sets (phrases are not considered in this step The STC algorithm is incremental and order independent. As each document arrives from the Web, we "clean" it and add it to the suffix tree. Each node that is updated (or created) as a result of this is tagged. We then update the relevant base clusters and recalculate the similarity of these base clusters to the rest of the base clusters. if there is any changes in the base cluster graph result in any changes to the final clusters. The final clusters are scored and sorted based on the scores of their base clusters and their overlap. As the final number of clusters can vary, the top few clusters need to be reported. Typically, only the top 10 clusters are of interest. For each cluster reported, the number of documents it contains, and the phrases of its base clusters. In STC, as documents may share more than one phrase with other documents, each document might appear in a number of base 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.

## IV. NOVEL DOCUMENT CLUSTERING

This method has three logical steps: (1) document preprocessing (2) Build the suffix tree and selection of Base cluster (3) Merging base cluser.

### *A. Preprocessing*

Pre-processing is selecting the most suitable terms that describing better content. The terms are transformed using stemming algorithm. Non-word tokens, such as Articles, pronouns, prepositions, and etc., are eliminated [14]. Here the snippets are obtained from the search engine's result and is given to stopword removal module. As mentioned in the "document cleaning", stopwords are removed from the given snippet. Html tags are also removed from this snippet. Extracted snippet information is given to STC module for insertion. The suffix tree data structure was introduced as an efficient string processing technique. A suffix tree allows us to insert a string into the suffix tree incrementally.STC is available as a dll here. The given snippet is split into more documents as mentioned in the STC algorithm and formed Suffix tree along with cluster information that also will be updated.

### *B. Selection of base clusters*

Once the STC formation is completed, Then we evaluate the phrase importance by statistical method which is usually applied in VSD model. In this paper, the equation for computing an interesting score, shown in Eq.1, is modified from hung chim and et.al [10]. The suffix tree constructed from documents usually contains lots of internal nodes (phrases). Not all internal nodes (phrases) are useful for document clustering, and some of the nodes (phrases)may give irrelevant results. So selecting a subset of original nodes as document features can reduce the high dimensionality of the feature space and also improve the accuracy of clustering results.

For each internal node $n_i$ , the phrase designated by $n_i$ is $p_i$. When we are constructing the suffix tree, use variable $tf(p_i)$ accumulate the times traverse through the node $n_i$ by the suffix in the documents corpus, then $tf(p_i)$ is the term frequency of phrase $p_i$ ; the times of different documents that traverses through the node $n_i$ is $df(p_i)$ , then $df(p_i)$ is the document frequency. Therefore the weight of phrase $p_i$ in documents corpus i.e., the weight w(n,d) of node $n$ in document $d$ can be calculated using the classic tf / idf scheme in formula (1), where $N$ is the total number of documents in corpus.

$$tf/idf = tf(p_i) \cdot \log (N/df(p_i)) \qquad \text{... (1)}$$

A phrase is an ordered sequence of one or more words. The more number of words a phrase contains, the richer meaning it can express. Therefore, the importance of a phrase should incorporate a factor about the length of phrase $p_i$, which designated by $|p_i|$. We calculate the factor using a heuristic utility function in following formula:

$$f(|p_i|) = \log_2 |p_i| \qquad \text{... (2)}$$

The score $s(n_i)$ of node $n_i$ (phrase $p_i$) is given by formula(3). To speed up the follow clustering, we only choose the *k* highest scoring phrases as key phrases (we take k to be 500 in our experiment).

$$S(n_i) = tf/idf \cdot f(|p\,i|) \cdot |d| \qquad \text{... (3)}$$

Where $|d|$ is the number of snippets in cluster $n_i$. Then the base clusters containing the highest scoring phrases i.e., top n ranked base clusters should be selected for merging. The initial time complexity of cluster merging process is $O(n^2)$. To keep this cost as constant, the similarity is not calculated for all base clusters but only for top n ranked base clusters. Here, the value of $n = 200$ was sufficient to ensure better performance.

*C. Merging base Clusters*

For merging base clusters, Zamir's STC defines a binary similarity measure between base clusters based on the overlap of their document sets. Given two base clusters Bm and Bn , with sizes |Bm | and |Bn | respectively, and representing the number of documents common to both base clusters. The similarity of Bm and Bn to be 1 if:

$$|Bm \cap Bn| / |Bm| > 0.5 \text{ and}$$

$$|Bm \cap Bn| / |Bn| > 0.5$$

Otherwise, their similarity is defined to be 0.

We found that the "and" Boolean operator is not suitable in the following condition if one base cluster is the subset of the other. So we change the "and" operator to "or" operator:

$$|Bm \cap Bn| / |Bm| \ \alpha \text{ or}$$

$$|Bm \cap Bn| / |Bn| > \alpha$$

This is essentially identical to Yang's [8] improvement. Here the $\alpha$ varies from 0.3 to 0.7 and it shows better performance at $\alpha = 0.6$. Since it uses the top n ranked base clusters for merging, only less number of clusters are formed than the existing method.

## V. EVALUATION

In this section, we evaluate the effectiveness and efficiency of our algorithm. The average precision of the results are calculated for different values of similarity constant and it provides better performance at $\alpha = 0.6$. The algorithms to be compared are the original STC and the search engine result list. The STC algorithm does not require the user to specify the required number of clusters. However, we found that the performance of STC is not very sensitive to this threshold, unlike AHC algorithms that showed extreme sensitivity to the number of clusters required. The base clusters are retrieved and corresponding document snippet with multiple links are displayed on the output screen. We use C # to implement algorithm, and use Excel to plot all figures. In order to evaluate the quality of the clustering, we adopted three quality measures widely used in the text mining literature for the purpose of document clustering [2].

*A. Experimental setup*

Due to lacks of standard dataset for testing web search results clustering, we have to build a test dataset. For this purpose, we have defined a set of queries such as computer, Internet, Games and so on, for which search results were collected from the search engine. We manually assigned a relevance judgement (relevant or not) to each document in these collections.

*B. Quality measure*

We use commonly used F-measure for evaluating and comparing different clustering results. F-measure combines the Precision and Recall ideas from the Information Retrieval literature. The precision and recall of a cluster j with respect to a "correct" class i are defined as:

$$P = \text{Precision}(i, j) = N_{ij}/N_j \qquad \text{... (4)}$$

$$R = \text{Recall}(i, j) = N_{ij}/N_i \qquad \text{... (5)}$$

where

$N_{ij}$: is the number of members of class *i* in cluster *j*,

$N_j$: is the number of members of cluster *j*, and

$N_j$: is the number of members of class i.

The standard F-measure F is used as the overall measure of how well the clustering matches the ideal clustering:

$$F = 2PR/P + R \quad \text{... (6)}$$

### *C. Data Preparation*

Web documents dataset are collected with the help of Yahoo

API [17]. This generates a dynamic collection of documents. Yahoo! Search BOSS (Build your Own Search Service) is an initiative in Yahoo! Search to open up Yahoo!'s search infrastructure and enable third parties to build revolutionary search products leveraging their own data, content, technology, social graph, or other assets. This release includes Web, News, and Image Search as well as Spelling Suggestions. Developers have expressed interest in altering the result order, removing results they do not want, and blending in their own data. All of these activities are allowed and encouraged with BOSS but not in the existing search API.

SYNTAX
http://boss.yahooapis.com/ysearch/web/v1/{query}?appid= {your BOSS appid}[&param1=val1&param2=val2&etc]

EX
http://boss.yahooapis.com/ysearch/web/v1/animals?appid=123
45&format=xml&start=1&count=10

Where the parameter start represents the ordinal position of first result where first position is zero. The parameter count represents the total number of results to return; maximum value is 100.

### *D. Experimental Results*

In this experiment, we submitted 10 different queries to the yahoo API and the search results of 100 snippets are collected for each query from the yahoo search engine. This API returns the XML, that dump the search results and it contains the URL, Title, Snippet, Description. From which,web documents are collected by parsing the XML document. After that, we created a suffix tree and merge the base clusters to get the final clusters of the query that is displayed to the user. While merging, the base clusters are merged with different values of á. This is essentially identical to Yang's [8] improvement. Here the á varies from 0.3 to 0.7 and it shows better performance at $\alpha = 0.6$. And identified the documents which are not in any of the merged clusters, those are listed in a new cluster. GUI will show all information, clustered information along with multiple HTML links.

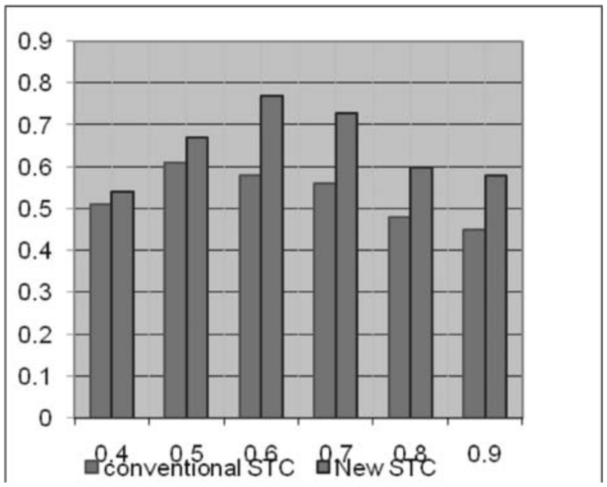

Fig. 1. F-Measure Comparision

The Figure.1. shows that the F-Measure value is better for the similarity constant of 0.6 than the other similarity values. The top n clusters returned by the algorithm were evaluated with Six different values of the similarity constant. New STC outperforms the conventional STC in all the similarity range.

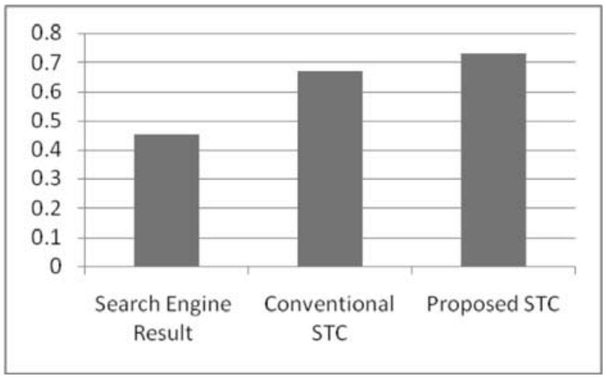

Fig. 2. The average of precision

As shown in Figure.2. the performance of Proposed STC is improved and compared with the two other results. This is mainly due to the selection of the base clusters for merging. We extract high discrimination phrases from documents, and remove the meaningless phrases which misguide the clustering result. Our new document similarity measure has the ability to accurately judging the relation between snippets

## VI. CONCLUSION

In this work we have shown a Novel document clustering algorithm. We modified the STC by the new definition of cluster score. This method is mainly focused on improving the effectiveness of document clustering. According to the preliminary experiment results, the new approach provides smaller clusters and more readable cluster label than that approach using the previous STC algorithms. Further experiments are also necessary to confirm modified STC's apparent advantage over existing clustering algorithms.

## REFERENCES

[1] Willet P. 1988, Recent trends in hierarchical document clustering: a critical review. Information Processing and Management, 24:577-97.

[2] van Rijsbergen C. J., 1979 Information Retrieval, Butterworths, London, 2nd ed.,.

[3] Cutting D. R, Karger D. R , Pedersen J. O and Tukey Scatter J.W / 1992 Gather: a cluster-based approach to browsing large document collections. In Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 318-29.

[4] Hill. D. R. 1968 A vector clustering technique. In Samuelson (ed.), Mechanised Information Storage, Retrieval and Dissemination, North-Holland.

[5] Zamir,1998 Oren and Etzioni, Oren. Web document clustering: a feasibility demonstration. Annual ACM Conference on Research and Development in Information Retrieval Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval Melbourne, Australia P. 46 - 54.

[6] Brin, S 1998 and Page, L. The anatomy of a large scale hypertexual web search engine. Proc. Of 7th WWW conference.

[7] Hammouda, K.M. and M.S. Kamel, 2004 Efficient phrase-based document indexing for Web document clustering. IEEE Transactions on Knowledge and Data Engineering, 16(10): p. 1279-1296.

[8] Yang J. W. 2004 A Chinese Web Page Clustering Algorithm Based on the Suffix Tree. Wuhan University Journal of National Sciences [M]. 9 (5):817-822.

[9] Rocchio, J.J 1966 Document retrieval systems - optimization and evaluation. Ph.D. Thesis, Harvard University,

[10] Hung, C. and D. Xiaotie, 2007 A new suffix tree similarity measure for document clustering, in Proceedings of the 16th international conference on World Wide Web.,ACM: Banff, Alberta, Canada.

[11] Salton, G., A. Wong, and C.S. Yang, 1975 A vector space model for automatic indexing. Commun. ACM, 18(11): p.613-620.

[12] Hua-Jun Zeng and et.at., 2004 "Learning to Cluster Web Search Results ", SIGIR'04 , Peking University.

[13] Branson S. and Greenberg A. Clustering Web Search Results Using Suffix Tree Methods. homepage: http://stanford.edu/lass/archive/cs/cs276a/cs276a

[14] Porter M. F, 1980 "An algorithm for suffix stripping ", Program, 14(3), pp.130-137.

[15] Voorhees E.M. 1986 Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. Information Processing and Management, 22:465-76.

[16] Hammouda, K.M. and M.S. Kamel, 2004 Efficient phrase-based document indexing for Web document clustering. IEEE Transactions on Knowledge and Data Engineering, 16(10): p. 1279-1296.

[17] http://developer.yahoo.com/search/boss/

**R.Subhashini,** received her B.E degree from Madurai Kamaraj University, Madurai, India in 2000, ME degree specialized in the field of Computer Science and Engineering from Sathyabama University, Chennai, India in 2005. She is pursuing her Ph.D degree in the area of Information Retrieval at Sathyabama University, Chennai, India. She is presently working as an Associate Professor in the Department of Information Technology at Sathyabama University, Chennai, India. Her research interests include Information Retrieval, Document Clustering, Data Mining.