A New Suffix Tree Similarity Measure for Document Clustering

Hung Chim
Department of Computer Science
City University of Hong Kong
HKSAR, China
chim@cs.cityu.edu.hk

Xiaotie Deng
Department of Computer Science
City University of Hong Kong
HKSAR, China
csdeng@cityu.edu.hk

ABSTRACT

In this paper, we propose a new similarity measure to compute the pairwise similarity of text-based documents based on suffix tree document model. By applying the new suffix tree similarity measure in *Group-average* Agglomerative Hierarchical Clustering (GAHC) algorithm, we developed a new suffix tree document clustering algorithm (NSTC). Experimental results on two standard document clustering benchmark corpus *OHSUMED* and *RCV1* indicate that the new clustering algorithm is a very effective document clustering algorithm. Comparing with the results of traditional word term weight *tf-idf* similarity measure in the same GAHC algorithm, NSTC achieved an improvement of 51% on the average of *F-measure* score. Furthermore, we apply the new clustering algorithm in analyzing the Web documents in online forum communities. A topic oriented clustering algorithm is developed to help people in assessing, classifying and searching the the Web documents in a large forum community.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering; E.1 [**Data Structure**]: Trees; H.3.1 [**Content Analysis and Indexing**]: Linguistic Processing

General Terms

Algorithms, Experimentation

Keywords

Suffix tree, document model, similarity measure

1. INTRODUCTION

Knowledge collaboration includes contributing to, authoring within, discussing, sharing, exploring, and deploying a collective knowledge base [12]. The World Wide Web opens up new possibilities for people to share knowledge, exchange information, and conduct knowledge collaboration. Numerous kinds of knowledge collaborative online communities sprang up the world. BBS, Weblog and Wiki have become well-known words in our daily life. On the other hand, computers have no understanding of the content and meaning of the submitted information data. Assessing and classifying the information data have mainly relied on the manual work of a few experienced people (the editors or moderators) in these knowledge collaboration systems. With growth of a community, the workload of the manual work will become heavier and heavier. Especially in online BBS forum communities, the more people join the discussion, the heavier workload the forum moderators have to bear.

Document clustering and classification have long been studied as a post-retrieval document visualization technique [16, 3]. Document clustering algorithms attempt to group documents together based on their similarities; the documents that are relevant to a certain topic will hopefully be allocated in a single cluster [27]. 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. For experienced members and editors, an automatic Web document clustering will help them to identify and assess high qualify documents more easily and efficiently. For newbies and guests, an efficient online searching service as well as a categorical index of the whole forum will help them to look for their interested topics.

Any clustering technique replies on four concepts: data representation model, similarity measure, clustering model and clustering algorithm that generates the clusters using the data model and the similarity measure [7]. The Vector Space Document (VSD) model [19] is a very widely used data representation model for document classification and clustering today. The common framework of this data model starts with a representation of any document as a feature vector of the words that appear in documents of the data set. The term-weights (usually term-frequencies) of the words are also contained in each feature vector [22]. The similarity between two documents is computed with one of several similarity measures based on two corresponding feature vectors, e.g. *cosine* measure, *Jaccard* measure, and *Euclidean distance* measure.

Suffix tree document model and Suffix Tree Clustering (STC) algorithm were proposed by Zamir and Etzioni and used in their meta-search engine [26]. STC is a linear time clustering algorithm (linear in the size of the document set), which is based on identifying phrases that are common to groups of documents. A phrase is an ordered sequence of one or more words [27]. There are two distinct characteristics attracting us to study suffix tree document model and

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

STC algorithm. Firstly suffix tree document model proposed a new flexible n-grams approach to identify all overlap nodes (phrases) among the documents as Longest Common Prefixes (LCPs) [11]. Secondly, one or several phrases are naturally selected to generate a topic summary to label the corresponding cluster during building the clusters. After implementing STC algorithm by following the description of Zamir's papers, we found that STC in point of fact can still obtain quite good results in clustering standard documents as well as document snippets. However, STC algorithm sometimes generates some large-sized clusters with poor quality in our experiment results of clustering standard documents. They desperately lower the overall effectiveness of STC algorithm.

Through analyzing the original design of STC algorithm, we identified the reason for dissipating the effort of STC algorithm as that: there is no effective quality measure to evaluate the quality of clusters in STC, neither the base clusters designated by the overlap nodes (phrases) in a suffix tree nor the clusters generated by the cluster merging. Furthermore, the weight of each overlap phrase is individually calculated from its length (the number of words in it) and document frequency *df*, the number of documents containing it (see Section 3.1 or paper [26] for details). In summary, STC algorithm lacks an efficient similarity measure to assess the importance of each phrase in a global view of entire document set. On the other hand, VSD model uses a feature vector to represent a document. The statistical features of all words are taken into account of the word term weights (usually *tf-idf*) and similarity measures. However, in contrast to suffix tree document model, the sequence order of words is seldom considered in the clustering algorithms based on this model. In fact, two document models are isolated in current information retrieval techniques [20].

Therefore we focused our work on how to combine the advantages of two document models in document clustering. By mapping each node of a suffix tree (excludes the root node) into a unique dimension of a M dimensional space (M is the total number of nodes in the suffix tree except the root node), each document is represented by a feature vector of M nodes. Then we rationally apply traditional term weighting schemes to the nodes (phrases) in suffix tree document model. Consequently we find out a simple solution to successfully connect two document data models: firstly the weight (term frequency *tf* and document frequency *df*) of each node are recorded in building the suffix tree from the documents, and then *cosine* similarity measure is used to compute the pairwise similarity of any two documents. With combination of the word's sequence order consideration of suffix tree model and the term weighting scheme of VSD model, the new suffix tree similarity measure works pretty good in GAHC algorithm. Our experimental results show that, new suffix tree document clustering (NSTC) algorithm is very effective in clustering standard documents of the data sets generated from *OHSUMED* [24] and *RCV1* [4] corpus.

Apart from Section 1, this paper is organized as follows: Section 2 discusses related work. Section 3 starts with a brief review of suffix tree model and STC algorithm, and then the detailed design of the new suffix tree similarity measure. A topic oriented Web document clustering approach follows in Section 4. Section 5 illustrates some experimental results for testing the effectiveness and efficiency of the new similarity measure in GAHC algorithm with comparisons of traditional word term weight similarity measure and original STC algorithm. Finally Section 6 summarizes our work with some considerations on future directions.

2. RELATED WORK

Text document clustering has been traditionally investigated as a means of improving the performance of search engines by pre-clustering the entire corpus [22], and a post-retrieval document browsing technique as well [16, 3, 26]. The methods used for document clustering covers several research areas, such as database, information retrieval, and artificial intelligent including machine learning and natural language processing. Agglomerative Hierarchical Clustering (AHC) algorithm might be most commonly used algorithm among the numerous document clustering algorithms. There are several variants from this algorithm, e.g. *single-link*, *group-average* and *complete-link*. In practice, AHC algorithm can often generate a high quality clustering result with a tradeoff of a higher computing complexity [23].

In traditional document models such as VSD model, words or characters are considered to be atomic elements in the statistical feature analysis and extraction. Clustering methods based on VSD model mostly make use of single word term analysis of document data set. In order to achieve more accurate document clustering, the importance of developing more informative features has received considerable attention in information retrieval literatures recently. Bigrams, trigrams and much longer ngrams have been commonly used in statistical natural language processing [2, 9, 25].

Suffix tree document model was firstly proposed in 1997 [13, 26]. Different from document models which treat a document as a set of words and ignore the sequence order of the words [1], suffix tree document model considers a document to be a set of suffix substrings, the common prefixes of the suffix substrings are selected as phrases to label the edges of a suffix tree. STC algorithm is developed based on this model and works well in clustering Web document snippets returned from several search engines. However, the properties of the suffix tree model and STC have not been analyzed in their papers [26, 27]. Eissen's paper [20] continued the work and pointed out, STC algorithm is a fusion heuristic that efficiently evaluates the graph-based similarity measure for large document collections. Furthermore, they also propose several new graph-based similarity measures to compute the document similarities. Their experimental evaluation results show that the similarity measures, especially the hybrid similarity measure have achieved significant performance improvements in MajorClust algorithm and GAHC algorithm.

3. A NEW SUFFIX TREE SIMILARITY MEASURE

In this section, firstly a brief review of suffix tree document model and STC algorithm is given, then the definition new suffix tree similarity measure is explained in details. After that, two important design issues will be discussed, including the time complexity of computing the document similarity based on the suffix tree document model.

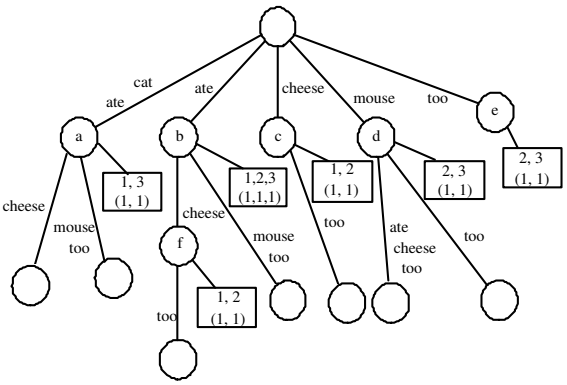

Figure 1: The suffix tree of tree documents "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too"

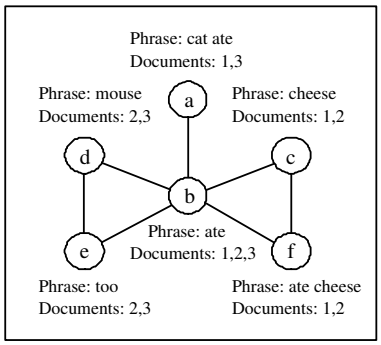

Figure 2: The base cluster graph

3.1 Suffix Tree Document Model and STC Algorithm

In text-based information retrieval, a document model is a concept that describes how a set of meaningful features is extracted from a document. Suffix tree document model considers a document $d = w_1 w_2 ... w_m$ as a string consisting of words w_i, not characters ($i = 1, 2, ..., m$). A suffix tree of document d is a compact trie containing all suffixes of document d. Figure 1 is an example of a suffix tree composed from three documents[1]. The nodes of the suffix tree are drawn in circles. Each internal node has at least two children. Each edge is labelled with a non-empty substring of a document called a phrase, and its suffix node is labelled by the phrase too. Then each leaf node in the suffix tree designates a suffix of a document; each internal node represents an overlap phrase shared by at least two suffixes. The more internal nodes shared by two documents, the more similar the documents tend to be.

In Figure 1, each internal nodes is attached with a box respectively (In the practical implementation, each node including leaf node maintains a list storing the numbers displayed in the box.). The numbers in the box designate the documents which have traversed the corresponding node. Each upper number designates a document identifier, the number below designates the traversed times of the document.

The original STC algorithm is developed based on the suffix tree document model. In detail, STC algorithm has three logical steps.

Step 1. The common suffix tree generating A suffix tree S for all suffixes of each document in $D = \{d_1, d_2, ..., d_N\}$ is constructed. Each internal node containing at least two different documents is selected to be a base cluster, which is composed of the documents designated by the box, and labelled by the phrase of the node.

Step 2. Base cluster selecting Each base cluster B is assigned a score $s(B)$,

$$s(B) = |B| \cdot f(|P|) \tag{1}$$

[1]We use the same example of paper [26] to describe the design of the suffix tree similarity measure. But the numbers in the box of a node designate the node's term frequency and document frequency.

where $|B|$ is the number of documents in B, and $|P|$ is the number of words in P. Then all base clusters are sorted by the scores, and the top k base clusters are selected for cluster merging in *Step 3*.

Step 3. Cluster merging A similarity graph consisting of the k base clusters is generated. An edge is added to connect two base clusters B_i and B_j if the *Jaccard* coefficient of B_i and B_j is larger than 0.5, say, when $\frac{|B_i \bigcap B_j|}{|B_i \bigcup B_j|} > 0.5$. The connected components in this graph form the final clusters. For example, the nodes a, b, c, d, e, f are selected to be the base clusters in the suffix tree of Figure 1. Finally the 6 base clusters form a final cluster as shown in Figure 2 after cluster merging.

3.2 The New Suffix Tree Similarity Measure

By mapping all nodes n of the common suffix tree to a M dimensional space of VSD model ($n = 1, 2, ..., M$), each document d can be represented as a feature vector of the weights of M nodes,

$$d = \{w(1, d), w(2, d), ..., w(M, d)\} \tag{2}$$

Term frequency - inverse document frequency (tf-idf) is a commonly used information retrieval technique for assigning weights to individual word terms appearing in all documents [22] [18]. When we represent each document as a feature vector in the M dimensional space, it's very easy to understand that the document frequency of each node $df(n)$ is the number of the different documents that have traversed node n; the term frequency $tf(n, d)$ of a node n with respect to a document d is the total traversed times of document d through node n. For example in Figure 1, the df of node b is $df(b) = 3$, the tf of the node with respect to document *1* is $tf(b, 1) = 1$ (assuming the document identifiers of the three documents to be *1, 2, 3*).

Therefore we can calculate the weight $w(n, d)$ of node n in document d using the following *tf-idf* formula:

$$tfidf(n, d) = (1 + \log(tf(n, d))) \cdot \log(1 + \frac{N}{df(n)}) \tag{3}$$

After obtaining the term weights of all nodes, it's easy to apply traditional similarity measures like the *cosine* similarity to compute the similarity of two documents. In this paper, we use *cosine* similarity measure to compute the pairwise similarities of all documents. The GAHC algorithm is used to evaluate the effectiveness of the new suffix tree similarity measure in document clustering.

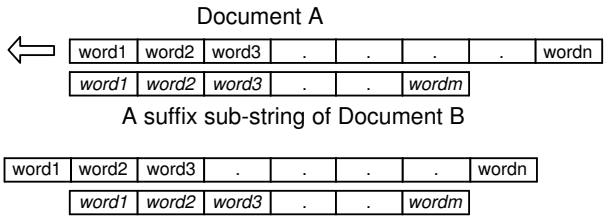

Figure 3: Comparing a suffix sub-string of document B with all suffix sub-strings of document A to find out a maximum length matched path

$$sim_{cos}(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \bullet \vec{d_j}}{|\vec{d_i}| \times |\vec{d_j}|} \quad (4)$$

3.3 A Closer Look to Suffix Tree Document Model

3.3.1 A Simple Efficiency Analysis of the Suffix Tree Similarity Measure

A suffix tree is a data structure that admits efficient string matching and querying. Suffix trees have been studied and used extensively in fundamental string problems such as large volumes of biological sequence data searching [14], approximate string matches [5] and text features extraction in spam email classification [17]. In suffix tree document model, a document is considered as a string consisting of words, not characters. During constructing the suffix tree, each suffix of document B is compared to all suffixes which exist in the tree already to find out a position for inserting it (as depicted in Figure 3). The naive, straightforward method to build a suffix tree for a document of m words takes $O(m^2)$ time. Ukkonen's paper [21] provided a algorithm to build a suffix tree in time linear with the size of a document. The time complexity of building a suffix tree is $O(m)$. Ukkonen's algorithm is argued for lack of space efficiency in building a suffix tree [6]. However, with the trade off of space cost, Ukkonen's algorithm makes it possible to build a large incremental suffix tree online, which allows us to insert a document into the suffix tree and remove it dynamically.

Indeed, suffix link data structure allows the searching algorithms to move quickly from one part of the tree to a distant part of the tree in a large suffix tree. In particular, the doubly-linked list data structure ($node \leftrightarrows edge \leftrightarrows suffix\_node$) also leaves a large room for us to develop different kind of search strategies in building an online clustering algorithm. In our current searching algorithm design, a bottom-up search is chosen to extract all internal nodes that traversed by a document.

Assuming that there are N distinct documents in a data set D, the average length of the documents is m (by words). Then there are a maximum of $N \cdot m$ leaf nodes in the suffix tree generated from N documents (each leaf node represents one or more suffixes of the documents). Thus finding out m leaf nodes representing all suffixes of a document requires a full traverse of $m \cdot N$ leaf nodes (the tree data structure directly maintains a list of all leaf nodes), the average time cost is $m \cdot N$. Because each node in a suffix tree has only one uplink node, the cost for calling back all parent internal nodes of a leave node is trivial in the bottom-up search[2]. Consequently the time cost of extracting the all nodes traversed by two documents and computing the *cosine* similarity is time linear to the size of document set ($2 \cdot m \cdot N$), regardless of the total number of nodes M in the suffix tree. Finally the total time cost for computing all pairwise similarities for all documents is $N \cdot (N-1) \cdot m \cdot N = m \cdot N^3$. In practice, the time cost of manipulating the suffix tree to compute the document similarities is very close to the cost of same operation on an inverted index.

3.3.2 Stopword or Stopnode

Stopwords are frequently occurring, insignificant words that appear in the documents. They are useless to index or use in search engines or other search indexes. Stopwords Lists and stemming algorithms are two commonly used information retrieval techniques for preparing text document. We also use a standard Stopwords List and *Porter* stemming algorithm [15] to preprocess the documents to get "clean" documents. However, we find there still exist some commonly occurring words slightly affecting the precision of the suffix tree similarity measures.

Although *tf-idf* weighting scheme has provided a solution to reduce the negative effect of these words, almost all popular document clustering algorithms including STC algorithm still prefer to consider these words as new stopwords, and ignore them in their document similarity measure. For example, STC algorithm maintains a *stoplist* that is supplemented with Internet specific words in computing the score of each base cluster, e.g. "previous", "java", "frames" and "mail". Words appearing in the *stoplist*, or that appear in too few or too many documents receives a score of zero in computing the score $s(B)$ of a base cluster.

We clearly understand the positive effectiveness of the method in VSD model. The question is, does this idea work in suffix tree document model? Recalling the suffix tree sample in Figure 1 and *Step 2, 3* of STC algorithm, the base clusters generated by the suffix tree are illustrated by Figure 2. The node b is labelled with a phrase "ate", and the phrase "ate" has a maximum document frequency $df = 3$ in the graph. If we consider word "ate" as a stopword, the node b should not be selected to be a base cluster because it gets a zero score. As a result, other 5 base clusters in the graph will not form a single cluster after the cluster merging.

This problem has not been discussed in original STC algorithm. Conventional document models, like VSD model ignores the occurring position of words. Simply ignoring these words in the similarity measure is reasonable. On the contrary, suffix tree document model is trying to keep the sequential order of each word in a document, the same phrase or word might occur in different nodes of the suffix tree. Simply ignoring the words (or phrases) becomes impractical in our approach.

In our document similarity measure, the term of a word is replaced by the term of a node in the suffix tree. We proposed a new definition "stopnode", which applies the same idea of stopwords in the suffix tree similarity measure computation. A node with a high document frequency *df* can

[2]The time cost for finding all nodes traversed by a suffix is decided by the length of its longest common prefix (LCP) in the suffix tree. The maximum length of LCPs in the suffix tree of standard documents is only 5 in our experimental data sets. Thus we say, the time cost is trivial.

be ignored in the similarity measure, but the corresponding phrase and its words might be kept by other nodes in the suffix tree. In our practical document clustering algorithm, a threshold idf_{thd} of inverse document frequency (idf) is given to identify whether a node is a stopnode. The experiments in Section 5 also provide a sufficient proof to this design issue.

4. A PRACTICAL APPROACH: WEB DOCUMENT CLUSTERING IN ONLINE FORUM COMMUNITIES

As mentioned in Section 1, the objective of our work is to develop a clustering algorithm for analyzing the Web documents in an online forum community.

Almost all Web forum systems use the same client-server system design: a Web server works with its inside programs as a preprocessor to handle HTTP requests and compose Web pages, a database server (usually a SQL server) works as a data storage. All forum data are processed into formal text content and stored in some tables with well-defined relationships.

We developed the clustering algorithm with C and PHP languages based on the same platform of a Web based BBS forum system developed by us before [8]. Generally, the Web document clustering algorithm has three logical steps: (1) document preparing, (2) document clustering, and (3) cluster topic summary generating.

4.1 Document Preparing

Different from the Web pages, which are composed by the forum system for people to read in a Web browser (usually *Internet Explorer* or *Firefox*), the content of a topic thread in a forum consists of a topic post and the reply posts. Each post is saved as a tuple in the corresponding table. Besides the text of the post, a tuple also has several fields storing some relevant information of the post, such as the *subject title, submitted time, author, view clicks* (the number of clicks to the post), and *recommend clicks* (the number of recommending clicks to the post). As a Web document, the text of a post might contain HTML tags, BBcode, emotion icon tags or other non-word tokens. There often exist some posts containing only one or several words, that are used to express the author's responsive emotion without any significant meaning, e.g. "thanks", "good post", "well done" and etc.

To prepare a text document with respect to a topic thread, we access the tuples from database table directly and combine all posts of the same thread into a single document. Before adding a post into the document, a document "cleaning" procedure is executed for the post: first all non-word tokens are stripped off; second the text is parsed into words; third all stopwords are identified and removed; forth *Porter* stemming algorithm is applied to all words; finally all stemmed words are incorporated to be a new plain text post. After the document "cleaning", the posts containing at least 3 distinct words are selected for document merging.

During the document merging, the *subject title* of the topic post (the first post which creates the thread in the forum) is selected as the title of the document, the text of all selected posts are added into the document in the order of their *submitted time*. Some fields of each post such as *view clicks and recommend clicks* are also aggregated to the sums respectively. Of course, the constructed document is saved into a table along with the sums for further processing of clustering algorithm.

4.2 Document Clustering

With the description of Section 3, it's quite simple to explain how the suffix tree clustering algorithm works.

Each thread document is fetched from the corresponding table, and inserted into a suffix tree. The tf and df of each node have been calculated during constructing the suffix tree, and the corresponding weight (*tf-idf*) is obtained as well. Thus the pairwise similarity of two documents can be computed with *cosine* similarity measure. Finally these pairwise similarities are used in GAHC algorithm to build a final clustering result.

4.3 Cluster Topic Summary Generating

The topic summary generating characteristic is a very important design issue arousing our study of suffix tree document model and STC algorithm. Unfortunately GAHC algorithm cannot provide this function to the new clustering algorithm. How to generate a new topic summary becomes another interesting design issue in our approach. In fact, topic summary generating concerns two important information retrieval work: (1) ranking the documents in a cluster by a quality score, (2) extracting common phrases as the topic summary of the corresponding cluster.

Evaluating quality of cluster and its documents is still a challenging research topic in modern information retrieval. However, the Web documents of a forum system can provide some additional human assessments for the document quality evaluation. For instance, there are three statistical scores provided in our forum system, *view clicks*, *reply posts* and *recommend clicks*. The quality score of a document d can be calculated with the following formula.

$$q(d) = |d| \cdot v \cdot r \cdot c \tag{5}$$

where $|d|$ is the number of words in the document, v is *view clicks*, r is *reply posts*, and c is *recommend clicks* of the document respectively.

Thus all documents in the same cluster are sorted by their quality scores. We choose the top 10% documents as the representatives of the cluster (at least 5 documents for each cluster). Then the nodes traversed by the representative documents are selected and sorted by their *idf* in ascend order. Finally the top 5 nodes are selected (excluding stopnodes). The original words in their phrases (without stemming) form a topic summary to label the cluster.

5. EVALUATION

In this section, we empirically evaluate the effectiveness of the new suffix tree similarity measure and traditional word term weight (*tf-idf*) similarity measure in the same GAHC algorithm. The original STC algorithm is compared as well. To achieve a fair comparison, at first some standard document collections without any bias must be provided, then some standard clustering quality measures shall be examined.

We choose *F-Measure* for evaluating and comparing three clustering algorithms. The *F-Measure* is commonly used in evaluating the effectiveness of clustering and classification algorithms [22, 10]. It combines the *precision* and *recall* ideas from information retrieval: Let $C = \{C_1, C_2, ..., C_k\}$

be a clustering of document set D, $C^* = \{C_1^*, C_2^*, ..., C_l^*\}$ designate the "correct" class set of D. Then the *recall* of cluster j with respect to class i, $rec(i,j)$ is defined as $|C_j \bigcap C_i^*|/|C_i^*|$. The *precision* of cluster j with respect to class i, $prec(i,j)$ is defined as $|C_j \bigcap C_i^*|/|C_i|$. The *F-Measure* combines both values according to the following formula.

$$F_{i,j} = \frac{2 \cdot prec(i,j) \cdot rec(i,j)}{prec(i,j) + rec(i,j)} \quad (6)$$

Based on this formula, the *F-Measure* for overall quality of cluster set C is defined by the following formula.

$$F := \sum_{i=1}^{l} \frac{|C_i^*|}{|D|} \cdot \max_{j=1,...,k} \{F(i,j)\} \quad (7)$$

Since there is no original binary or source code of STC algorithm and its evaluating document collections provided as the reference for our experiments. We wrote our own code for original STC algorithm following the description in Zamir's paper and PHD thesis, the corresponding document collection of *OHSUMED* [24] is also generated by ourselves as well.

5.1 Document Collections

5.1.1 OHSUMED Document Collection

The *OHSUMED* medical abstracts corpus was created to assist information retrieval research [24]. It is a clinically oriented MEDLINE subset consisting of 348,566 references (out of a total of over 7 million), and covers all references from 270 medical journals from 1987 to 1991. Each *OHSUMED* document has at least one primary and one secondary Medical Subject Heading (MeSH) indexing terms, discriminating between the focused topics and the briefly mentioned topics.

We also use a subset of *OHSUMED* corpus, which is very similar to the one used in paper [26]. However, all documents of the corpus are used to create the document collection. Because the earliest version of Mesh index that we can obtain is 2004, it contains a full version of Mesh index (only the documents in 90-91 are used in the experiments of paper [26]). We only select the documents having at least one MeSH index term in the "C14-Cardiovascular Diseases (C14)" sub-branch of MeSH hierarchy. The corpus provides us a total of 293,856 documents.

We created a set of disjoint groups of *OHSUMED* documents, each relating to a specific topic. These groups of documents are created as follows.

There are 494 index terms under the "C14" term in the MeSH hierarchy. For each term we collected its document group: each selected *OHSUMED* document contains this term as a primary index term, but does not contain any index term that has been selected before. We discarded document groups with less than 100 documents, and also discarded document groups whose term was an ancestor (in the MeSH hierarchy) of another selected term. In the end, we created 8 groups of document sets, each group with 100 documents. The MeSH index terms are: *MSH1058, MSH1262, MSH1473*, *MSH1486*, *MSH1713*, *MSH2025*, *MSH2030* and *MSH2235* as identified by a TREC-9 MeSH topics file, named "query.mesh.1-4904". The document collection that we created has a total of 800 documents, containing 6,281 distinct words after document preprocessing. The average length of the documents is about 110 (by words).

5.1.2 RCV1 Document Collection

We also generated a document collection of *RCV1* corpus [4]. RCV1 is a corpus that was published by Reuters Corporation for research purposes. It contains 806,792 documents, each consisting of hundreds up to thousands words. The documents have been manually enriched by meta information like category (also called topic), geographic region, or industry sector. RCV1 has 103 different categories, arranged within a hierarchy of four top level categories.

Our *OHSUMED* document collection has 8 disjoint groups of documents already. It is not necessary to build a new document collection under such a strict condition again. The purpose of the new *RCV1* document collection is that, we want to test the effectiveness of three clustering algorithms in a more complicated situation near to practice.

We manually identify 10 irrelevant categories according to our knowledge. The category index terms are: *C11, C12, C21, C41, E11, GREL, GSCI, GSPO, GWEA, and M11*. We build a group of documents with regard to each category of *C11, C12, C21, C41, E11, M11*: firstly all documents using the index term as their first class term are selected, then 200 documents are randomly chosen from them to form the document group. For documents of categories *GREL, GSCI, GSPO, GWEA*, because the documents share a same first class term *GCAT*, we randomly select 200 documents from all documents whose second class term are the corresponding term for each category. Finally the document collection has 10 groups of documents, containing 19,229 distinct words. The average length of documents is about 150.

5.2 Results and Discussion

We constructed 3 document sets from *OHSUMED* and *RCV1* document collections respectively. The overview of the 6 document sets is illustrated in Table 1, where *#nodes* designates the total number of nodes in the suffix tree generated by the data set, and *#overlap nodes* designates the number of overlap nodes shared by at least two different documents.

The original STC algorithm selects the 500 highest scoring base clusters for further cluster merging, but only the top 10 clusters are selected from the merged clusters as the final clustering result. Thus we also allowed GAHC algorithm to generate 10 clusters in the our experiments to conduct as fair as possible comparisons. We still recorded the total number of clusters generated by the cluster merging in STC, and also computed the *F-measure* score for each clustering result respectively.

Table 2 lists the *F-measure* scores computed from the clustering results of three clustering algorithms on 6 document sets, where NSTC designates the results of the new suffix tree similarity measure; TDC designates the results of traditional word *tf-idf cosine* similarity measure; STC designates the results of all clusters generated by STC algorithm, and STC-10 designates the results of the top 10 clusters generated by original STC algorithm; *#clusters* designates the amount of clusters generated by STC algorithm for each document set.

Comparing with the results of STC-10, NSTC algorithm has a performance improvement of 80% on the average *F-measure* scores of 6 document sets. Comparing with results (TDC) of traditional word *tf-idf cosine* similarity measure with the same GAHC algorithm, NSTC algorithm also achieved an improvement of 51% on the average *F-measure* scores.

Table 1: Overview of the Document Sets (Corpus Type: O-OHSUMED, R-RCV1)

Document Set	DS1	DS2	DS3	DS4	DS5	DS6
Corpus Type	O	O	O	R	R	R
#categories	3	5	8	4	6	10
#documents	300	500	800	400	600	2,000
#nodes	57,490	100,172	151,289	233,323	313,965	505,405
#overlap nodes	5,342	8,914	13,259	29,644	37,945	57,855

Table 2: F-measure Scores of the clustering results for 6 Document Sets

Document Set	DS1	DS2	DS3	DS4	DS5	DS6	Average
STC	0.70	0.72	0.67	0.73	0.68	0.56	0.68
#clusters	452	473	474	445	451	482	457
STC-10	0.56	0.38	0.28	0.77	0.47	0.28	0.46
TDC	0.73	0.67	0.33	0.62	0.59	0.37	0.55
NSTC	0.91	0.94	0.80	0.83	0.85	0.67	0.83

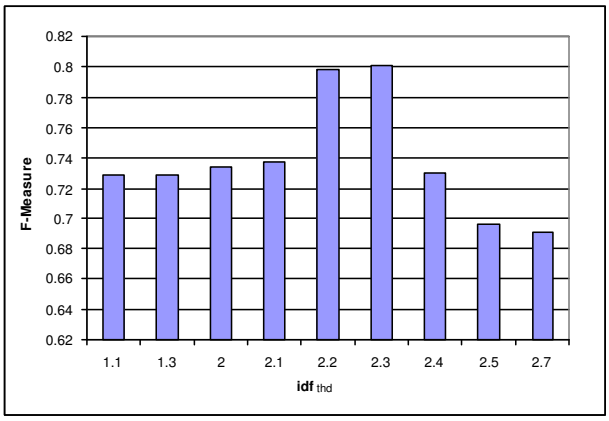

Figure 4: The *F-measure* scores for different idf_{thd} values

The results of STC also discover a potential improvement in STC algorithm, because STC can obtain quite high *F-measure* scores (0.68) in the 6 document sets when all final clusters are taken into account. The experimental results indicate the major reason decreasing the effort of STC algorithm - there is no effective measure to evaluate the quality of the clusters during the cluster merging (*single-link*), eventually the quality of merged clusters cannot be assessed. Thus STC algorithm seldom generated large size clusters with high quality in the experiments. In contrast, NSTC can achieve significant performance improvements with the efficient evaluation measure provided by GAHC algorithm.

Figure 4 shows the effect of threshold idf_{thd} for ignoring the stopnodes in NSTC algorithm, the results are obtained from DS3 document set, which contains all documents of *OHSUMED* document collection. The *F-measure* score reaches the top score of 0.801 while idf_{thd} is set to be 2.3.

DS6 document set contains all documents of 10 groups in *RCV1* document collection. We use *class1-10* to represent the 10 groups respectively. Figure 5, 6, and 7 respectively illustrate the *Precision, Recall, F-measure* scores of each cluster in the clustering result of DS6 (there are only 9 non-empty clusters in the result). It's easy to find that, the 5th cluster is composed of the documents of 3 classes, namely *class1, class2, class4* (*C11, C12, C41*). The cluster shows that some intersections possibly appear among the documents of the 3 classes. In fact, the method that we used to build *RCV1* document collection just ensures the first or second class term of the documents to be disjoint. It is possible that the documents in different classes share a same second or third class term of the corpus category index. The average *F-measure* scores that we obtained in DS4, DS5, DS6 is 0.783, which is very close to the average *F-measure* scores of 0.78 achieved by paper [20]. However, unlike the hybrid measure that they proposed combining single word *tf-idf* similarity measure and graph-based similarity measure, the complexity of our suffix tree similarity measure is simpler and more feasible in practice.

Figure 8 captured a snapshot from a clustering result of the Web document clustering algorithm as presented in Section 4. It demonstrates the *Topic Summary* and the top 5 threads' subject title of two categories. The post data in the experiment are from Apple discussion community (discussions.apple.com). It's a commercial technical support forum for the products of Apple Company. We choose 500 threads in the forum " iPOD with color display - Connecting to Windows" for this experiment. (We wrote a Web crawler to automatically download the Web pages of the online forum community in *Nov.* 2005, all these Web pages were parsed into posts and stored in several tables for our research [8].)

6. CONCLUSIONS AND FUTURE WORK

Both traditional vector space model and suffix tree model play important roles in text-based information retrieval. However, the two models are used in two isolated ways: almost all clustering algorithms based on VSD model ignore the occurring position of words in the document, the different semantic meanings of a word in different sentences are unavoidably discarded. Suffix tree document model keeps all sequential characteristics of the sentences for each document, phrases consisting of one or more words are used to designate the similarity of two documents. But the original STC algorithm cannot provide an effective evaluation method to assess the quality of clusters. This paper proposes a new suffix tree similarity measure to successfully

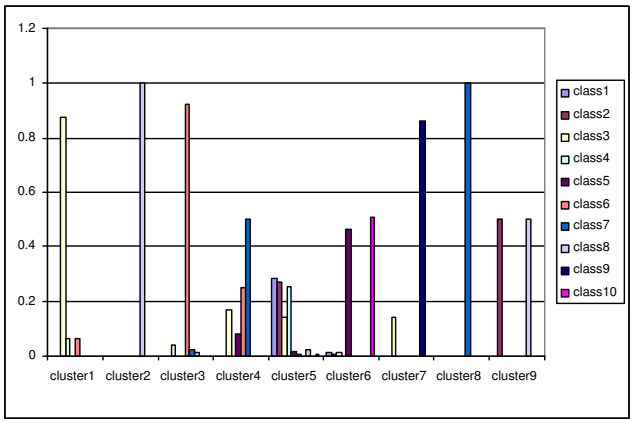

Figure 5: The *Precision* scores for each cluster in the result of DS6 document set

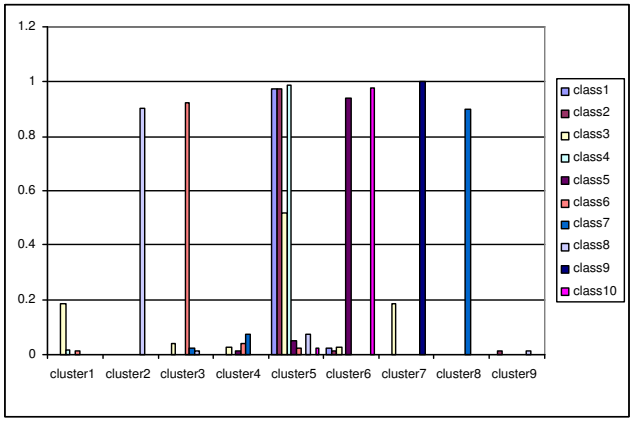

Figure 6: The *Recall* scores for each cluster in the result of DS6 document set

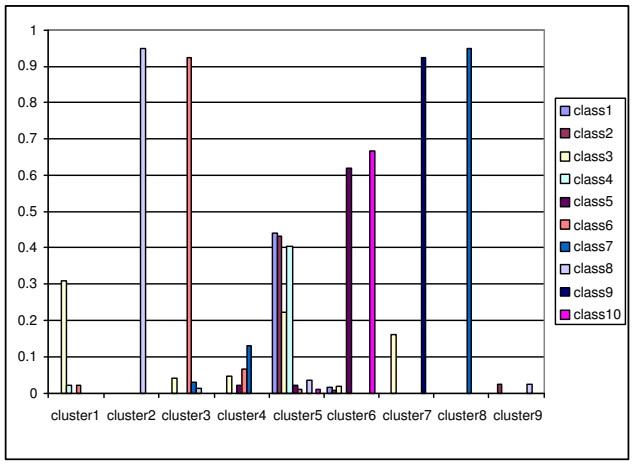

Figure 7: The *F-measure* scores for each cluster in the result of DS6 document set

```
+ Category 1. iPOD, file, update, iTunes, fold
|- 2 ipods on one computer
|- Can I hook up my ipod to a friends computer to get his songs
|- Songs on Hard drive vs.songs on source list.
|- help! all existing songs will be replaced if i update?
|- 60GB Photo iPod Replacement - STILL problems
.  .  .
+ Category 2. iPOD, USB, connect, update, device
|- Cant Format Ipod.
|- iPod 60GB won't power off.
|- Computer eror happened during reformating
|- Screen Frozen w Apple Image
|- Belkin firewire card
```

Figure 8: A demo: the topic summaries of two categories in the Web document clustering result

connect both two document models. By completely mapping all nodes in the common suffix tree into a M dimensional space of VSD model, the advantages of two document models are smoothly inherited in the new document similarity measure. The significant improvement of the clustering performance in our experiments clearly indicates that word order preservation is critical to document clustering and categorization. We believe that the new similarity measure is suitable to not only hierarchical clustering algorithm but also most traditional clustering algorithms based on VSD model, e.g. K-means, single-pass. More performance evaluation comparisons for these clustering algorithms with the new suffix tree similarity measure have been in the consideration of our further work.

The concept of the suffix tree similarity measure is very simple, but the implementation is quite difficult. Our work presented in this paper is mainly focused on improving the effectiveness of document clustering algorithms. Efficiency optimization of the algorithm has been a target of our current work, both the time efficiency and the space efficiency. We can also foresee the potential power of the suffix tree similarity measure in processing documents of other ethnological languages. Applying the new similarity measure in Chinese document clustering is also a part of our future work.

7. ACKNOWLEDGMENTS

The research work described in this paper is partially supported by a City University TDF grant [Project No. 6980080] and a SRG grant of City University of Hong Kong [Project No. 7001975].

8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] E. Charniak. *Statistical Language Learning*. MIT Press, 1993.

[3] W. B. Croft. *Organizing and searching large files of documents*. PhD thesis, University of Cambridge, 1978.

[4] T. G. R. David D. Lewis, Yiming Yang and F. Li. RCV1: A new benchmark collection for text

categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[5] A. Ehrenfeucht and D. Haussler. A new distance metric on strings computable in linear time. *Discrete Applied Math*, 40, 1988.

[6] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

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

[8] X. D. Hung Chim, Min Jiang. A semantics based information distribution framework for large web-based course forum system. *Lecture Notes in Computer Science: Advances in Web Based Learning ICWL 2006*, 4181/2006:93–104, 2006.

[9] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[10] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the KDD-99 Workshop*, San Diego, CA, USA.

[11] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[12] D. K. O'Neill and L. M. Gomez. The collaboratory notebook: A distributed knowledge-building environment for project-enhanced learning. In *Proceedings of Ed-Meida'94*, Vancouver, BC, 1994.

[13] O. M. Oren Zamir, Oren Etzioni and R. M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997.

[14] J. R. Paul Bieganski and J. V. Carlis. Generalized suffix trees for biological sequence data: Application and implentation. In *Proceedings of 27th Annual Hawaii International Conference on System Sciences*, pages 35–44, 1994.

[15] M. Porter. New models in probabilistic information retrieval. *British Library Research and Development Report, no. 5587*, 1980.

[16] P. O. R. Allen and M. Littman. An interface for navigating clustered document sets returned by queries. In *Proceedings of the ACM Conference on Organizational Computing Systems*, pages 166–171, 1993.

[17] B. M. Rajesh Pampapathi and M. Levene. A suffix tree approach to anti-spam email filtering. *Machine Learning*, 65, 2006.

[18] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information retrieval. In *Proceedings of 11th Annual International Conference on Research and Development in Information Retrieval, ACM*, pages 147–160, 1988.

[19] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communication of ACM*, 18(11):613–620, 1975.

[20] D. S. Sven Meyer zu Eissen and M. Potthast. The suffix tree document model revisited. In *Proceedings of the 5th International Conference on Knowledge Management*, pages 596–603, 2005.

[21] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[22] C. J. van Rijsbergen. *Information Retrieval*. Second Edition, Butterworths, London, 1979.

[23] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[24] T. J. L. William Hersh, Chris Buckley and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201, Dublin, Ireland.

[25] M. Yamamoto and K. W. Church. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30, 2001.

[26] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of SIGIR'98*, University of Washington, Seattle, USA, 1998.

[27] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374, 1999.