Efficient Phrase-Based Document Similarity for Clustering

Hung Chim and Xiaotie Deng, *Senior Member*, *IEEE*

Abstract—Phrase has been considered as a more informative feature term for improving the effectiveness of document clustering. In this paper, we propose a phrase-based document similarity to compute the pairwise similarities of documents based on the Suffix Tree Document (STD) model. By mapping each node in the suffix tree of STD model into a unique feature term in the Vector Space Document (VSD) model, the phrase-based document similarity naturally inherits the term *tf-idf* weighting scheme in computing the document similarity with phrases. We apply the phrase-based document similarity to the *group-average* Hierarchical Agglomerative Clustering (HAC) algorithm and develop a new document clustering approach. Our evaluation experiments indicate that the new clustering approach is very effective on clustering the documents of two standard document benchmark corpora *OHSUMED* and *RCV1*. The quality of the clustering results significantly surpasses the results of traditional single-word *tf-idf* similarity measure in the same HAC algorithm, especially in large document data sets. Furthermore, by studying the property of STD model, we conclude that the feature vector of phrase terms in the STD model can be considered as an expanded feature vector of the traditional single-word terms in the VSD model. This conclusion sufficiently explains why the phrase-based document similarity works much better than the single-word *tf-idf* similarity measure.

Index Terms—Suffix tree, document model, similarity measure, document clustering.

1 Introduction

Document clustering has long been studied as a postretrieval document visualization technique to provide an intuitive navigation and browsing mechanism by organizing documents into groups, where each group represents a different topic [1], [2]. In general, the clustering techniques are based on four concepts: data representation model, similarity measure, clustering model, and clustering algorithm. Most of the current document clustering methods are based on the Vector Space Document (VSD) model [3]. The common framework of this data model starts with a representation of any document as a feature vector of the words that appear in the documents of a data set. A distinct word appearing in the documents is usually considered to be an atomic feature term in the VSD model, because words are the basic units in most natural languages (including English) to represent semantic concepts. In particular, the term weights (usually *tf-idf*, term-frequencies and inverse document-frequencies) of the words are also contained in each feature vector [4]. The similarity between two documents is computed with one of the several similarity measures based on the two corresponding feature vectors, e.g., *cosine* measure, *Jaccard* measure, and *euclidean distance*.

To achieve a more accurate document clustering, a more informative feature term—phrase—has been considered in recent research work and literature. A phrase of a document is an ordered sequence of one or more words [5]. Bigrams and trigrams are commonly used methods to extract and identify meaningful phrases in statistical natural language processing [6]. Yamamoto and Church [7] presented a method to compute all substrings' (phrases) term frequencies and document frequencies in large document corpora by using suffix array [8]. Reference [9] proposed a phrase-based document index model namely Document Index Graph (DIG), which allows for the incremental construction of a phrase-based index for a document set. The quality of clustering achieved based on this model significantly surpassed the traditional VSD model-based approaches in the experiments of clustering Web documents.

Particularly, the Suffix Tree Document (STD) model and Suffix Tree Clustering (STC) algorithm were proposed by Zamir et al. [10] and Zamir and Etzioni [11]. The STC algorithm was used in their metasearching engine to real-time cluster the document snippets returned from other search engines. It is a linear time clustering algorithm (linear in the size of the document set), which is based on identifying the phrases that are common to groups of documents. However, the STD model and STC algorithm have not been analyzed in their work as pointed out by Sven Meyer zu Eissen and Potthast's paper [12]. The STC algorithm got poor results in clustering the documents in their experimental data sets of *RCV1* corpus [13].

By studying the STD model, we find that this model can provide a flexible n-gram method to identify and extract all overlap phrases in the documents. The STD model considers a document as a sequence of words, not characters. A document is represented by a set of suffix substrings, the common prefixes of the substrings are selected as phrases to label the edges (or nodes) of a suffix tree. Despite all previous work on the STD model [10], [11], [5], [12], an effective method has not been found to evaluate the effect of

• *The authors are with the Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong SAR. E-mail: {h.chim, csdeng}@cityu.edu.hk*

Manuscript received 26 June 2007; revised 26 Nov. 2007; accepted 12 Feb. 2008; published online 25 Feb. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-06-0291.
Digital Object Identifier no. 10.1109/TKDE.2008.50.

each phrase in document clustering algorithms. In contrast, the VSD model uses a feature vector to represent a document. The statistical features of all words are taken into account of the term weights (usually *tf-idf*) and similarity measures, whereas the sequence order of words is rarely considered in the clustering approaches based on the VSD model. The two document models are isolated in the current information retrieval techniques.

Thus, we focus our work on how to combine the advantages of two document models in document clustering. As a result of our work, a phrase-based document similarity is presented in this paper. By mapping each node of a suffix tree (excludes the root node) into a unique dimension of an M-dimensional term space (M is the total number of nodes except the root node), each document is represented by a feature vector of M nodes. Consequently, we find a simple way to compute the document similarity: First, the weight (*tf-idf*) of each node is recorded in building the suffix tree, then the *cosine* similarity measure is used to compute the pairwise similarities of documents. By applying the new document similarity to the *group-average* HAC algorithm (GHAC), we developed a new document clustering approach. In our experiments for evaluating the performance of various clustering algorithms, the new clustering approach achieves a significant improvement on clustering the document data sets of two standard document corpora *OHSUMED* [14] and *RCV1.* For *F-measure*, which is used to evaluate the overall quality of clustering result, the average score is 0.892. For *Purity*, which is used to measure the overall precision of the clusters, the average score is 0.928. For *Entropy*, which is used to count how various classes of documents are distributed within each cluster, the average score is 0.079. The quality of the clusters generated by the new clustering approach significantly surpasses the single-word term *tf-idf* similarity measure in the same HAC algorithm (the average scores of above three measures are 0.654, 0.631, and 0.268, respectively), especially on large document data sets.

In addition, we also study the STC algorithm to understand why the algorithm cannot work in clustering standard documents as well as document snippets. By implementing the STC algorithm according to Zamir's papers [10], [11], we find that the STC algorithm can still obtain some fairly good results in clustering standard documents. However, the STC algorithm sometimes generates some large-sized clusters of poor quality in our experiments, they apparently lower the overall effectiveness of STC algorithm. By analyzing the original design of STC algorithm, we find the reason for dissipating the effort of STC algorithm as that: there is no effective quality measure to evaluate the quality of clusters in the algorithm, whether for the base clusters designated by the overlap nodes (phrases) of the suffix tree or for the clusters generated by the cluster merging. Furthermore, the weight of each overlap phrase is individually calculated by its length and document frequency (see Section 3.1 or [11] for details). In summary, the STC algorithm lacks an effective measure to assess the importance of each phrase in a global view of the entire document set.

Furthermore, we also investigate the reason for the phrase-based document similarity to surpass the traditional single-word similarity measure. We find that the feature vector, which is used to represent a document in the STD model, can be considered as an extended version of the feature vector of the traditional single-word terms. By mapping each node of the suffix tree into a unique feature of the VSD model, the sequential order of all words can be preserved by the nodes (phrases) in the expanded feature vector. We believe this is the key point leading to the significant improvement of the phrase-based document similarity in document clustering. On the other hand, the higher dimensional feature vectors often introduce much more computation cost in computing the document similarities for clustering. To reduce the complexity of the phrase-based document similarity, we conduct a series of experiments to examine the role of each kind of nodes in the suffix tree, and investigate the impact on the overall quality of document clustering. Finally, our investigation finds an empirical result: The document similarities are mainly decided by the overlap nodes (phrases), which appear in at least two different documents in the document set. All other nodes or phrases are trivial to the phrase-based document similarities, with a slight effect on the overall quality of document clustering.

The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 starts with a brief review of the STD model and STC algorithm, and then presents the detailed design of the new phrase-based document similarity. Section 4 illustrates the experimental results of the effectiveness and efficiency evaluation for the new document similarity, with the comparisons of traditional word *tf-idf* similarity measure. Some technical analysis and discussions are also provided based on the experimental results. Finally, Section 5 summarizes the contributions of our work.

2 Related Work

Text document clustering has been traditionally investigated as a means of improving the performance of search engines by preclustering the entire corpus [4], and a postretrieval document browsing technique as well [1], [2], [11]. Hierarchical Agglomerative Clustering (HAC) algorithm might be the most commonly used algorithm among numerous document clustering algorithms. Generally, there are three variants from this algorithm: *single-link*, *complete-link*, and *group-average.* In practice, the HAC algorithm can often generate high-quality clusters with a tradeoff of the higher computation complexity [15]. K-Nearest Neighbor (K-NN) algorithm is well known for classification [16]. It has also been used for document clustering [17], [9].

In the traditional document models such as the VSD model, words or characters are considered to be the basic terms in statistical feature analysis and extraction. To achieve a more accurate document clustering, developing more informative features has become more and more important in information retrieval literature recently. Bigrams, trigrams, and much longer n-grams have been commonly used in statistical natural language processing [6], [18].

Suffix tree is a data structure that admits efficient string matching and querying. It has been studied and used extensively in fundamental string problems and applications such as large volumes of biological sequence data searching [19], approximate string matches [20], and text features extraction in spam e-mail classification [21]. The STD model was first proposed in 1997 [10], [11]. Different from document models which treat a document as a set of words and ignore the sequence order of the words [22], the STD model considers a document to be a set of suffix substrings, and the common prefixes of the suffix substrings are selected as the phrases to label the edges of the suffix tree. The STC algorithm is developed based on this model and works well in clustering Web document snippets. However, the properties of STD model and STC algorithm have not been analyzed in their papers [5], [10], [11]. Sven Meyer zu Eissen and Potthast's paper [12] continued the work and pointed out that the STC algorithm was a fusion heuristic that efficiently evaluated the graph-based similarity measure for the document collections. Furthermore, their work also proposed several new graph-based similarity measures to compute the document similarities. Their experimental evaluation results showed that the similarity measures, especially the hybrid similarity measure had achieved significant performance improvements in the MajorClust algorithm and the GHAC. Yamamoto and Church [7] presented a method to compute all substrings' (phrases) term frequencies and document frequencies in large document corpora by using suffix array [8]. Their approach provided an efficient way for extracting phrases and computing their *tf-idf* weights in a static document set. Another relevant work is the DIG proposed by Hammouda and Kamel [9], which allows for the incremental construction of a phrase-based index for a document set. The quality of clustering achieved by using the hybrid similarity measure based on this model significantly surpassed the approaches based on the traditional VSD model.

3 The Phrase-Based Document Similarity

Throughout this paper, we use the symbols N, M, and k to denote the number of documents, the number of terms, and the number of clusters, respectively. We use the symbol D to denote the document set of N documents that we want to cluster, the $C_1, C_2, \ldots, C_k$ to denote each one of the k clusters.

In text-based information retrieval, a document model is a concept that describes how a set of meaningful features is extracted from a document. Most of the current document clustering methods use the VSD model to represent documents. In the model, each document d is considered to be a vector in the M-dimensional term space. In particular, we usually employ the term *tf-idf* weighting scheme [4], [23], in which each document can be represented as

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

where $w(i,d) = (1 + \log tf(i,d)) \cdot \log(1 + N/df(i))$, $tf(i,d)$ is the frequency of the ith term in the document d, and $df(i)$ is the number of documents containing the ith term.

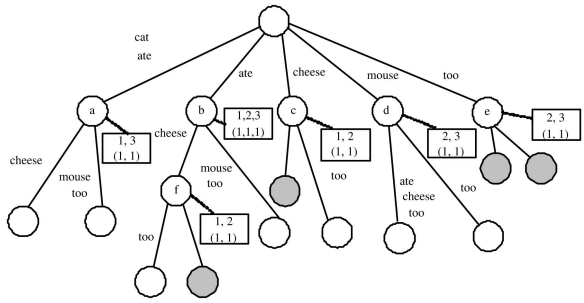

Fig. 1. The suffix tree of tree documents "cat ate cheese," "mouse ate cheese too," and "cat ate mouse too."

In the VSD model, the *cosine* similarity is the most commonly used measure to compute the pairwise similarity of two document d_i and d_j, which is defined as

$$sim_{i,j} = \frac{\vec{d_i} \bullet \vec{d_j}}{|\vec{d_i}| \times |\vec{d_j}|}. \tag{2}$$

3.1 Standard Suffix Tree Document Model and STC Algorithm

The STD model considers a document d as a string consisting of words $w_1 w_2 \ldots w_m$, not characters. The suffix tree of a document d is a compact trie containing all suffix substrings of the document d. Fig. 1 is an example of a suffix tree composed from three documents. The nodes of the suffix tree are drawn in circles. There are three kinds of nodes in the suffix tree: the root node, internal nodes, and leaf nodes. Each internal node has at least two children. Each edge is labeled with a nonempty substring of a document called a phrase. Then, each leaf node in the suffix tree designates a suffix substring of a document; each internal node represents a common phrase shared by at least two suffix substrings. The similarity of two documents is defined as the more internal nodes shared by the two documents, the more similar the documents tend to be.

In Fig. 1, each internal node is attached to an individual box. The numbers in the box designate the documents that have traversed the corresponding node. Each upper number designates a document identifier, the number below designates the traversed times of the document. (In the implementation of our approach, the node data structure has a list storing the numbers directly.)

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

Step 1. The common suffix tree generation. A suffix tree S for all suffixes of each document in $D = \{d_1, d_2, \ldots, d_N\}$ 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 labeled by the phrase of the node.

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

$$s(B) = |B| \cdot f(|P|), \tag{3}$$

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*.

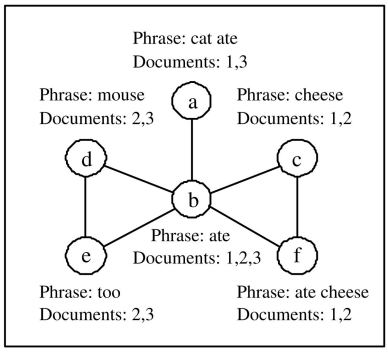

Fig. 2. The base cluster graph.

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, and f are selected to be the base clusters in the suffix tree of Fig. 1. Finally, the six base clusters form a final cluster as shown in Fig. 2 after cluster merging.

In practice, the STC algorithm works well in a meta-searching engine for real-time clustering the document snippets returned from other search engines. It is a linear time clustering algorithm based on identifying the phrases that are common to groups of documents. We think that there are two distinctive characteristics in the STD model and STC algorithm: First, the STD model provides a flexible n-gram method to identify and extract all overlap phrases among the documents as Longest Common Prefixes (LCPs) [8]. Second, one or several phrases are naturally selected to form a topic summary to label the corresponding cluster during the cluster generation. However, the STD model and STC algorithm have not been analyzed in their papers as pointed out by Sven Meyer zu Eissen and Potthast's paper [12]. The STC algorithm also gets poor results in clustering the documents of *RCV1* corpus in their experiments.

3.2 The Phrase-Based Document Similarity Based on the STD Model

As mentioned in the previous section, there are three different kinds of nodes in a suffix tree. In particular, there exist some leaf nodes labeled with an empty phrase (usually a *NULL* in the suffix tree implementations). They are generated by Ukkonen's algorithm [24], which is used to build suffix tree, and denote the end of the corresponding documents. For example, in Fig. 1, the four leaf nodes represented by gray circles are such nodes. We call these leaf nodes "terminal nodes" in our work. With the exception of the terminal nodes and the root node, each node in the suffix tree, either an internal node or a leaf node represents a nonempty phrase that appears in at least one document in the data set. The same phrase might occur in different edges of the suffix tree. For instance, there are three different edges labeled with the same phrase of "cheese" in the suffix tree of Fig. 1.

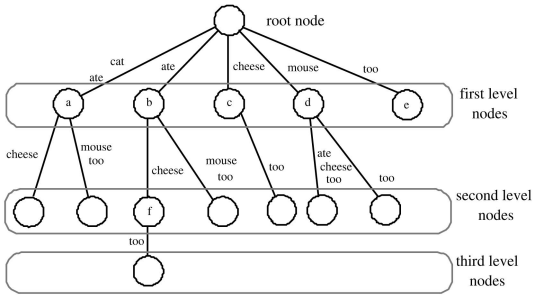

Fig. 3. Three different levels of nodes in the suffix tree of three documents.

The definition of the phrase-based document similarity is simple and understandable: By mapping each node v labeled by a nonempty phrase into a feature of M-dimensional term space, each document d can be represented as a feature vector of the weights of M node terms in the VSD model as illustrated by (1). It is very easy to understand that the document frequency of each node $df(v)$ is the number of the different documents that have traversed node v; the term frequency $tf(v,d)$ of a node v with respect to document d is the total traversed times of the document d through node v. In the example of Fig. 1, the df of node b is $df(b) = 3$, the tf of the node with respect to the document 1 is $tf(b,1) = 1$ (assuming the document identifiers of three documents to be *1, 2, 3*). Therefore, we can calculate the weight of node b with respect to document 1 as $w(b,1) = (1 + \log 1) \cdot \log(1 + 3/3) = 0.693$.

After obtaining the term weights of all nodes, it is easy to apply traditional similarity measures such as the *cosine* similarity to compute the similarity of any two documents. In this paper, the *cosine* similarity measure is used to compute the pairwise similarities of all documents.

Let vectors $\vec{d_x} = \{x_1, x_2, \ldots, x_M\}$, and $\vec{d_y} = \{y_1, y_2, \ldots, y_M\}$ denote two documents d_x and d_y, where x_i and y_i are the weights of corresponding node term v_i, respectively. Then, the similarity of two documents is calculated by the following formula:

$$sim_{x,y} = \frac{\vec{d_x} \bullet \vec{d_y}}{|\vec{d_x}| \times |\vec{d_y}|} = \frac{\sum_{i=1}^{M} x_i y_i}{\sqrt{\sum_{i=1}^{M} {x_i}^2 \sum_{i=1}^{M} {y_i}^2}}. \quad (4)$$

3.3 Properties of the STD Model

Suffix tree is considered to be one of the well-known full text index data structures. It has been studied for decades and is used in many algorithmic solutions and practical applications. Nevertheless, the property of STD model does not appear to be further studied since the model was proposed, including Zamir's papers [5], [10], [11].

The suffix tree of a document set is a compact trie containing all suffix substrings of the documents in the data set. During the suffix tree construction, the root node is the initial node and the parent of all other nodes. All other nodes are created and stored in a hierarchical order to follow their LCP nodes, respectively. In this paper, we define all child nodes of the root node as *first-level nodes* of the suffix tree, the child nodes of the *first-level nodes* as *second-level nodes*, and so on. For instance, Fig. 3 illustrates three levels of nodes in the suffix tree of Fig. 1. The nodes a, b, c, d, and e are *first-level*

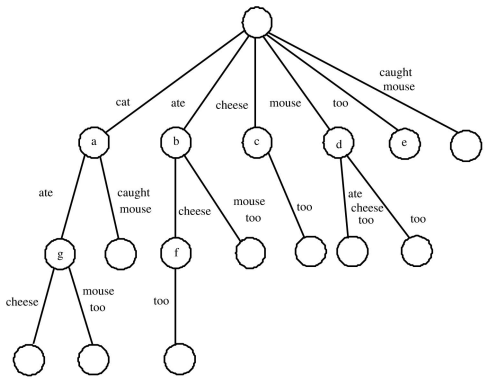

Fig. 4. The suffix tree of four documents after inserting a document "cat caught mouse."

nodes, and the node f is a *second-level node*, while all terminal nodes in the suffix tree are ignored.

In the viewpoint of the VSD model, each node of the suffix tree (except the root node and terminal nodes) is a distinct feature term in the term space. Each node denotes a phrase that appears at least once in the document set. In the example of Fig. 1, the three documents contain five single-word terms {*ate, cat, cheese, mouse, too*}. In the corresponding suffix tree, the set of phrases denoted by the *first-level nodes* is {*ate, cat ate, cheese, mouse, too*}. The word term of *cat* is replaced by the phrase of *cat ate* due to the compact property of suffix tree data structure.

Now, we add a new document "cat caught mouse" into the suffix tree, then we get a new suffix tree as shown in Fig. 4. Consequently, the set of the phrases denoted by the *first-level nodes* is expended to be {*ate, cat, caught mouse, cheese, mouse, too*}. The word *cat* appears in the phrase terms denoted by the *first-level nodes*.

Empirically, we conclude three properties for the STD model. In more precise terms, we let T denote the suffix tree generated by all documents in the data set D. Each internal node v has at least two children and each edge is labeled with a nonempty substring of document set D. No two edges out of a node can have edge labels beginning with the same word. The concatenation of the edge labels on the path from the root node to the internal node v constitute a phrase P_v. In addition, we use L_v to designate the level of internal node v in the suffix tree T, then for each *first-level node* v under the root node, $L_v = 1$; for each *second-level node* v, $L_v = 2$; and so on.

Property 1. *Each internal node of the suffix tree T represents an LCP of the document data set D, and each leaf node represents a suffix substring of a document in the data set D.*

Property 2. *Each first-level node in suffix tree T is labeled by a distinct phrase that appears at least once in the documents of data set D. The number of the first-level nodes is equal to the number of keywords (distinct single-word terms in the VSD model) in the data set D.*

Property 3. *Each phrase P_v denoted by an internal node v at a higher level ($L_v \geq 2$) in suffix tree T contains at least two words. The length of the phrase $|P_v| \geq 2$ (by words).*

We also observe that the larger the size of the data set D, the more single-word phrases appear in the *first-level nodes*. The set of single-word phrases denoted by the *first-level nodes* in the suffix tree T will be very close to the set of keywords in the data set D if the data set is large enough (please see Section 4.6 for details). The multiple-words phrases denoted by the other internal nodes expand the term space of keyword terms based on the traditional VSD model. Thus, we can consider the feature vector representing a document in STD model as an extended version of the traditional keyword feature vector in the VSD model. Hence, the new phrase-based document similarity not only inherits all features extracted by the traditional keyword approaches based on the VSD model but also preserves the sequential order of words in the phrases of STD model. This explains why the new document similarity can exceed the single keyword similarity measure in the same document clustering algorithms. This conclusion might explain why Sven Meyer zu Eissen and Potthast's [12] and Hammouda and Kamel [9] want to use hybrid similarity measures in their work. The phrase-based similarity measures that they proposed must cooperate with the traditional keyword *tf-idf* similarity measure to obtain the best clustering results in their approaches.

Furthermore, the conclusion introduces a study for reviewing the effects of different nodes (phrases) in the phrase-based document similarity. Recall that in the popular clustering approaches using the keyword *tf-idf* similarity measure, the words that occur in no more than two documents are often eliminated. We applied this idea on our phrase-based document similarity and conducted a series of experiments to see the impact on document clustering if we only used the overlap phrases shared by at least two documents to compute the document similarities. The experimental results indicate that the quality of clusters generated by the clustering algorithm is slightly undermined. Finally, our investigation and experiments discover an optimization method: The overlap nodes or phrases, which appear in at least two different documents predominantly decide the document similarities. All other nodes are trivial to the document similarities, with slight effect on the overall quality of document clustering.

3.4 Computation Complexity of the Phrase-Based Document Similarity

To build a suffix tree, the naive and straightforward method compares each suffix substring of the document d to all suffix substrings which already exist in the tree, and finds a position to insert it. The time complexity of building the suffix tree for a document of m words is $O(m^2)$. Ukkonen's paper [24] provides an algorithm to build a suffix tree in linear time with the size of a document. The time complexity of building the suffix tree is $O(m)$. Zamir et al. declared that the time cost for building a suffix tree of N documents is $N \log N$, where the length of a document m is assumed to be bounded by a constant [10].

Indeed, the suffix link data structure allows the searching algorithms to move quickly from one part of the tree to a distant part in a large suffix tree. In particular, the doubly linked list data structure ($node \rightleftharpoons edge \rightleftharpoons suffix\_node$) also provides a large room for developing different kinds of search strategies in building an efficient clustering algorithm. In our current searching algorithm implementation, a

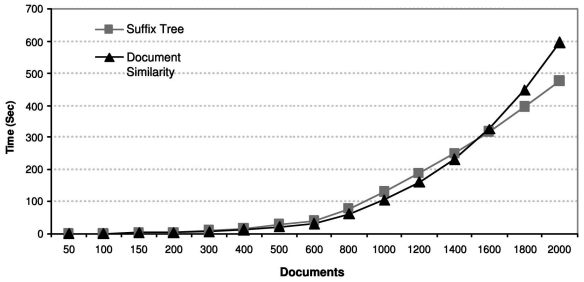

Fig. 5. The time cost for suffix tree construction and computing the similarity matrix.

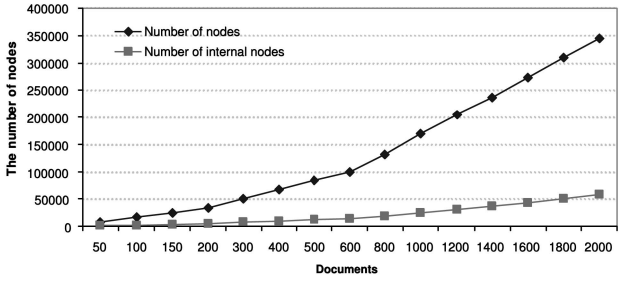

Fig. 6. The size of suffix tree scales linearly to the size of document data set.

bottom-up search is developed to collect all nodes that traversed by a document.

Algorithm: Bottom-up search for collecting all nodes traversed by a document
Input id: document id
Output $Node$: a node list
1: $Leaf \leftarrow$ List of leaf nodes
2: $Node \leftarrow$ Empty List of nodes
3: $lnode \leftarrow$ Empty List of nodes ($lnode$ is a list for all leaf nodes matching id)
4: **for** each leaf node $leaf_i$ in $Leaf$ **do**
5: **if** id exist in $leaf_i$ **then**
6: Add $leaf_i$ to $lnode$
7: **end if**
8: **end for**
9: **for** each node v_i in $lnode$ **do**
10: **if** v_i has an nonempty phrase **then**
11: Add lf_i to $Node$
12: **end if**
13: $v_i \leftarrow$ parent node of v_i
14: **while** v_i is not the root node **do**
15: Add v_i to $Node$
16: $v_i \leftarrow$ parent node of v_i
17: **end while**
18: **end for**
19: return $Node$

Assuming the average length of the documents be m (words), then there are a total of $N \cdot m$ leaf nodes in the suffix tree generated from the N documents. Thus, finding out m leaf nodes representing all suffix substrings of a document requires a full traverse of $m \cdot N$ leaf nodes (see **for** loop at lines 4-8, the tree data structure directly maintains a list of all leaf nodes in Ukkonen's algorithm), the time cost is $m \cdot N$. Because each node in a suffix tree has only one parent node, the cost for calling back all parent internal nodes of a leaf node is trivial in the bottom-up search.[1] Consequently, the time cost of building a node list (a list of pointers to the nodes), which contains all the nodes traversed by a document, is linear time to the size of document set $(O(mN))$, regardless of the total number of nodes M in the suffix tree.

1. The time cost for finding all nodes traversed by a suffix substring is decided by the length of its LCP. The average length of LCPs in the suffix tree of standard documents is only 2.565 in our experimental data sets, so the average time cost for calling back all nodes matching a document is $2.565 \cdot m$. Thus, we say, the time cost is trivial if $N \gg 2.565$.

After obtaining the node lists of all documents, the pairwise similarity of any two documents can be computed based on the node lists. The time complexity of computing all pairwise similarities for N documents is $O(N^2)$. In practice, the time cost of manipulating the suffix tree to compute the document similarities is very close to that on an inverted index data structure. Fig. 5 exhibits the time costs of building the suffix tree from 50 documents to 2,000 documents in a data set, the time costs of computing the document similarities based on the suffix tree are illustrated in Fig. 5 as well. The data set is DS6, which contains 2,000 documents of *RCV1* corpus. The experimental results show that the time cost of building a suffix tree is closely linear to the number of documents (for the small number of documents, the time cost of accessing documents on the disk is nontrivial). Actually, the time cost of computing all pairwise document similarities appears to be $O(N^2)$ of the documents in the data set.

For the space complexity, both suffix tree data structure and the constructing algorithms, including Ukkonen's algorithm, are criticized for the large memory redundancy [25]. However, with the trading off of memory redundancy, 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. There are two basic elements in a suffix tree: nodes V and edges E. Each edge connects two nodes, the source node and the destination node. Thus, we can estimate the memory cost (size) of a suffix tree by counting the total number of nodes in the suffix tree. Fig. 6 illustrates the total numbers of nodes and internal nodes in the suffix tree of various numbers of documents. Both the two numbers increase proportionally to the number of documents, especially the number of internal nodes in the suffix tree.

3.5 Stopwords or Stopnodes

Stopwords are frequently occurring, insignificant words that appear in documents. They are useless to index or use in search engines and other information retrieval systems. Stopwords Lists and stemming algorithms are two commonly used information retrieval techniques for preprocessing text documents. We also use a standard stopwords List and the *Porter's* suffix-stripping algorithm [26] to process the documents to get "clean" documents. However, we note that there still exist some frequently occurring words slightly affecting the accuracy of the phrase-based document similarities.

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

We fully understand the positive effect of the method in clustering approaches based on the VSD model. The question is, does this idea work in the approaches based on STD model too? Recall the suffix tree sample in Figs. 1 and 2 and *Steps 2 and 3* of the STC algorithm, the base cluster of node b is labeled with a phrase of "ate," which has a maximum document frequency $df = 3$ in the graph. If the word "ate" is identified 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 five base clusters in the graph will not form a single cluster in the cluster merging.

There is no mention of this problem in the original STC algorithm. The conventional document models, like the VSD model, ignore the occurring position of words in the documents. Simply ignoring these words in computing the similarity measure is reasonable. To the contrary, the STD model is trying to keep the sequential order of each word in a document, the same phrase or word might occur at several different nodes in the suffix tree. Simply ignoring the words (or phrases) becomes unreasonable in our approach.

In our phrase-based document similarity, the word term is replaced by the node term in the suffix tree. We propose a new definition of "stopnode" to apply the same idea of stopword to the phrase-based document similarity. A node with a high document frequency df can be ignored in computing the document similarities, but the corresponding phrase and its words therein might be kept by other nodes in the suffix tree. In our document clustering approach, a threshold idf_{thd} of the inverse document frequency (idf) is introduced to identify whether a node is a stopnode. All the stopnodes (with $idf < idf_{thd}$) are excluded in computing the document similarities.

The experiments for evaluating the effect of different threshold idf_{thd} values show that the stopnodes will be a problem undermining the quality of the clustering results when the document data set becomes larger. Fig. 7 shows the F-measure scores of clustering results obtained from clustering the documents of two data sets DS3 and DS6 with different idf_{thd} values. The variable range of the F-measure scores is over 20 percent on DS3 and 12 percent on DS6. The document data sets DS3 and DS6, respectively, contain all documents of the *OHSUMED* and *RCV1* document collections that are used to evaluate the document clustering algorithms in our experiments.

4 Experimental Results

In order to test the effectiveness and efficiency of the phrase-based document similarity, we conducted a series of experiments to compare the new document similarity with the traditional keyword *tf-idf* similarity measure in the same GHAC and STC algorithm. For fair comparison, first some standard document data sets without any bias must be provided, then some standard clustering quality metrics shall be examined. In the comparison experiments, we used seven data sets from three document collections. The first two document collections are generated, respectively, from two public benchmark document corpora for Text REtrieval Conferences (TREC), namely, *OHSUMED* and *RCV1*. The third document collection is a subset (mini_20newsgroup) of the well-known 20-newsgroups collection.[2] To evaluate efficiency of the new document similarity in document clustering, we also conducted a series of experiments on a larger document data set generated from *RCV1* corpus.

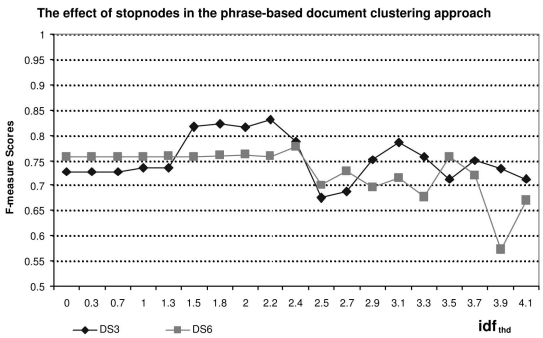

Fig. 7. The *F-measure* scores for different idf_{thd} values on data sets DS3 and DS6.

We write our codes in C language and conducted all experiments on a Pentium D 840 3.2-GHz Linux server with 2-Gbyte memory. For the HAC algorithm, we use an open source library, i.e., the *C Clustering library* [27]. This open source library is a collection of numerical routines that implement the clustering algorithms for bioinformatics data processing and analysis. Since there is no binary or source code of the original STC algorithm available, we write our own code to implement the STC algorithm by following the description in Zamir's papers [11], [10].

4.1 Evaluation Metrics

We evaluate the effectiveness of the document clustering with three quality measures. The first is *F-Measure.* The *F-Measure* is commonly used in evaluating the effectiveness of clustering and classification algorithms [4], [28]. It combines the *precision* and *recall* ideas in information retrieval. Recalling $C = \{C_1, C_2, \ldots, C_k\}$ is a clustering of data set D of N documents, we let $C^* = \{C_1^*, C_2^*, \ldots, C_l^*\}$ 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 $|C_j \bigcap C_i^*|/|C_j|$. *F-Measure* combines both values according to the following formula:

$$F(i,j) = \frac{2 \cdot prec(i,j) \cdot rec(i,j)}{prec(i,j) + rec(i,j)}. \quad (5)$$

2. The document collection can be download from UCI KDD Archive: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

TABLE 1
Overview of the *RCV1* Document Collection

Document Set	$C11$	$C12$	$C21$	$C41$	$E11$	$GREL$	$GSCI$	$GSPO$	$GWEA$	$M11$
#documents	323	431	625	525	482	629	390	3688	454	4635

#documents is the number of documents in the category

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

$$F = \sum_{i=1}^{l} \frac{|C_i^*|}{N} \cdot \max_{j=1,\ldots,k} \{F(i,j)\}. \quad (6)$$

The second measure is Purity. The cluster purity indicates the percentage of the dominant class members in the given cluster. For measuring the overall clustering purity, we use the weighted average purity as shown below:

$$Purity = \sum_{j=1}^{k} \frac{|C_j|}{N} \cdot \max_{i=1,\ldots,l} \{prec(i,j)\}. \quad (7)$$

The third measure is Entropy, which provides a measure of "goodness" for unnested clusters or the clusters at one level of a hierarchical clustering. The entropy tells us how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. Like F-measure and Purity, we also use weighted average Entropy as an overall clustering metric:

$$Entropy = -\frac{1}{\log k} \sum_{j=1}^{k} \frac{|C_j|}{N} \sum_{i=1}^{l} p_{ij} \log p_{ij}, \quad (8)$$

where p_{ij} is the probability that a member of cluster C_j belongs to class C_i^*.

To sum up, we would like to maximize the F-measure and Purity scores, and minimize the Entropy score of the clusters to achieve a high-quality clustering.

4.2 Document Collections

4.2.1 OHSUMED Document Collection

The *OHSUMED* medical abstract corpus was created to assist information retrieval research [14]. It is a clinically oriented MEDLINE subset consisting of 348,566 references (out of a total of over seven 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 use a subset of *OHSUMED* corpus, which is very similar to the data collection for evaluating the STC algorithm in [11]. We selected the documents having at least one MeSH index term in the "C14-Cardiovascular Diseases (C14)" subbranch of MeSH hierarchy. The corpus provided us a total of 293,856 documents. We selected eight disjoint groups from these documents, each group related to a specific topic. The eight groups of documents are created as described below.

There are 494 index terms under the "C14" term in the MeSH hierarchy. For each term, we collect 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 discard the document groups with less than 100 documents as well as the document groups whose term is an ancestor (in the MeSH hierarchy) of another selected term. Finally, we get eight groups of document sets, each group has 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 has a total of 800 documents, containing 6,281 keywords after the document preprocessing. The average length of the documents is about 110 words.

4.2.2 RCV1 Document Collection

We also generate a document collection from *RCV1* corpus. RCV1 is a corpus published by the Reuters Corporation for information retrieval research purposes [13]. It contains 806,792 documents, each consisting of hundreds up to thousands words. The documents have been manually enriched by metainformation like category (also called topic), geographic region, or industry sector. RCV1 has 103 different categories, arranged within a hierarchy of four top-level categories.

Because the *OHSUMED* document collection that we have created already has eight disjoint groups of documents, it is not necessary to build a new document collection under such a strict condition again. The purpose of building the *RCV1* document collection is to test the effectiveness of the clustering algorithms in a more complicated condition close 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*. Table 1 lists the number of documents in each category. We build a group of documents with regard to each category of *C11*, *C12*, *C21*, *C41*, *E11*, and *M11*: First, 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*, and *GWEA*, because of the 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 keywords. After the document preprocessing, the average length of documents is about 150 words.

4.2.3 20-Newsgroups Document Collection

The 20-newsgroups document collection is a collection of approximately 20,000 newsgroup documents, partitioned evenly across 20 different newsgroups. It was originally collected by Ken Lang, probably for his paper [29]. Now, it

TABLE 2
Overview of the Document Sets (Corpus Type: O—OHSUMED, R—RCV1, N—20-Newsgroups)

Document Set	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Corpus Type	O	O	O	R	R	R	N
#categories	3	5	8	4	6	10	20
#documents	300	500	800	400	600	2,000	2,000
#nodes	40,416	68,430	102,784	69,507	104,083	345,638	301,922
#overlap nodes	5,342	8,914	13,259	12,536	16,765	57,855	35,834

has become a popular data set for the experiments in text applications of machine learning techniques such as text classification and clustering. In this document collection, each news group constitutes a different category, with varying overlaps between them; some news groups are very related and others are not related at all. The main purpose of choosing this collection is to test the capacity of the phrase-based document clustering approach against noise. In our experiments, we directly used the minisubset provided by UCI KDD Archive. This data set contains 2,000 documents, 100 documents for each news group. For each document in the data set, the text of the message headers and e-mail addresses are ignored in our experiments. After the document preprocessing, the average length of documents is about 131 words.

4.3 Document Preprocessing

Before the document clustering, a document "cleaning" procedure is executed for all documents in the data sets: First, all nonword tokens are stripped off. Second, the text is parsed into words. Third, all stopwords are identified and removed. Fourth, the *Porter's* suffix-stripping algorithm [26] is used to stem the words. Finally, all stemmed words are concatenated into a new document.

Since the length of a word is variable, it is quite difficult to implement a suffix tree based on words directly. To solve the problem, we build a wordlist to store all keywords in alphabetical order. The similar ideas are often used in some text retrieval approaches for simplifying the computation complexity, such as the inverted index systems. In the wordlist, a unique integer number (called a *word_id*) is assigned to each keyword so that we can use the *word_id* to replace the corresponding word in the "cleaned" document. Finally, each document becomes an array of *word_ids* for the suffix tree construction.

4.4 Comparison of Four Clustering Algorithms

Document clustering has been investigated for a long time. A number of document clustering approaches have been developed based on different document models and various clustering algorithms for decades. In general, the approaches can be categorized into agglomerative solutions (e.g., hierarchical clustering) and partitional solutions (e.g., K-means clustering).

HAC algorithm [15] starts with each instance representing a cluster. The algorithm recursively merges clusters until a stop criterion is met. Each iteration step results in a certain level of clustering. The final results depend on the threshold of granulation. The key of HAC algorithm is the method used to determine which pair of clusters will be the most similar pair for merging at each iteration. The building method is quite simple but needs to specify how to compute the similarity of two clusters. In our work, the *group-average* similarity is chosen for the HAC algorithm, which measures the similarity of two clusters with the average of pairwise similarities of the documents from each cluster.

K-NN algorithm is well known for text-based document classification. It has also been used for document clustering. For each document, the K-NN algorithm calculates its pairwise similarities to all other documents in the data set. The top K documents are selected as its nearest neighbors, then the document is assigned to the cluster where the majority of the K documents belong to. Thus, the K-NN clustering algorithm uses the parameter K to refine the resolution of the clusters, in order to obtain a given bound m on the number of the clusters that are generated for representing the data set.

In this paper, the performance of four clustering algorithms are evaluated in the comparison experiment: the original STC algorithm, the GHAC with the phrase-based document similarity, the GHAC with the traditional single-word *tf-idf cosine* similarity, and the K-NN clustering algorithm with the phrase-based document similarity.

The original STC algorithm selects the 500 highest scoring base clusters for further cluster merging, but only chooses the top 10 clusters as the final result of the clustering [11]. However, we still record all clusters generated by the cluster merging of the STC algorithm, and compute the three kinds of measure scores for the clustering result as well as the other clustering algorithms.

In the comparison experiments, we constructed three data sets from the *OHSUMED* collection, three data sets from the *RCV1* collection, and one data set from the *mini 20-newsgroups* collection. The overview of the seven document data sets is illustrated in Table 2, 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 that are shared by at least two different documents.

Figs. 8, 9, and 10, respectively, illustrate the *F-measure*, *Purity*, and *Entropy* scores computed from the clustering results of four clustering algorithms on the seven document data sets, where STC designates the results of all clusters generated by the STC algorithm, and STC-10 designates the results of the top 10 clusters selected by the original STC algorithm; ST-GHAC designates the results of GHAC with the phrase-based document similarity; GHAC designates the results of the same GHAC with traditional keyword *tf-idf cosine* document similarity; ST-KNN designates the results of K-NN clustering algorithm with the phrase-based document similarity. Since the STC algorithm allows a document to appear at more than one cluster, the *Purity* and

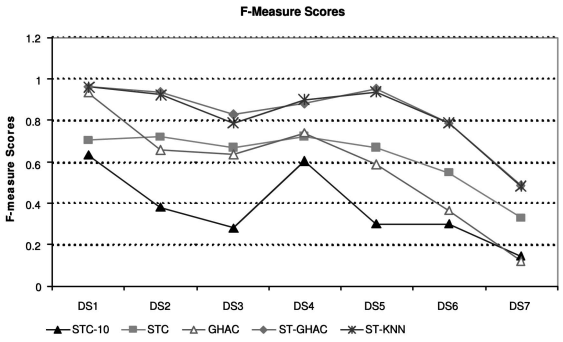

Fig. 8. The *F-measure* scores of seven data sets for four clustering algorithms.

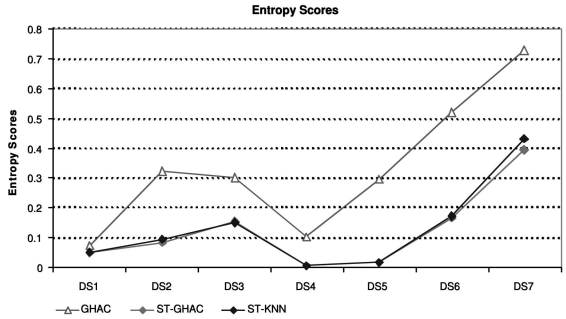

Fig. 10. The *Entropy* scores of seven data sets for four clustering algorithms.

Entropy measures are not suitable for evaluating such clustering results; thus, we exclude the corresponding scores for STC and STC-10 in Figs. 9 and 10.

In comparison with the results of STC-10, the ST-GHAC has a performance improvement of 0.456 on the average *F-measure* scores of the seven document sets (DS1-7). Comparing with the results of the single-word *tf-idf cosine* similarity measure in the same GHAC, the ST-GHAC also achieved an increment of 0.256 on the average *F-measure* score, an increment of 0.309 on the average *Purity* score, and a decrement of 0.210 on the average *Entropy* score. The larger improvements on the *Purity* and *Entropy* scores indicate that the new phrase-based document clustering approach tends to be a highly accurate document clustering approach. In particular, the data set DS7 is a subset of *20-newsgroups*, which is known as a corpus containing some overlapped categories (e.g., talk.politics.misc and talk.politics.mideast). The GHAC gets rather poor scores on this data set (*F-measure*: 0.119, *Purity*: 0.097, *Entropy*: 0.729), whereas the ST-GHAC obtains a significant improvement (*F-measure*: 0.482, *Purity*: 0.481, *Entropy*: 0.394). This comparison indicates that the phrase-based document similarity can efficiently improve the ability of GHAC against the noise.

The results of STC also discover a potential improvement in the original STC algorithm, because the STC can obtain fairly high *F-measure* scores (0.623 on average) in the seven document sets when all the clusters are taken into account. However, there often exist some large-sized clusters of poor quality in the clustering results. The experimental results indicate that the major reason affecting the performance of STC algorithm is the lack of an effective quality measure to evaluate the clusters, either the base clusters designated by the overlap nodes (phrases) in the suffix tree or the clusters generated by the *single-linkage* cluster merging. The STC algorithm uses a quite simple method to compute the score for selecting a base cluster (as seen in (3)). The score is mainly decided by two factors: the number of documents in the cluster and the length of the labeling phrase. A cluster that is labeled by a long phrase or contains many documents will get a higher score and be selected as a base cluster. The same method is also used to rank the merged clusters after the cluster merging; finally, the top 10 clusters with the largest scores are selected as the final clustering result. In fact, this evaluation method does not concern whether the documents in the cluster have similar content in semantics, even though they are sharing a common phrase now. In other words, the coherence of a cluster has not been evaluated. Thus, the STC algorithm seldom generated large-sized clusters of high quality in our experiments.

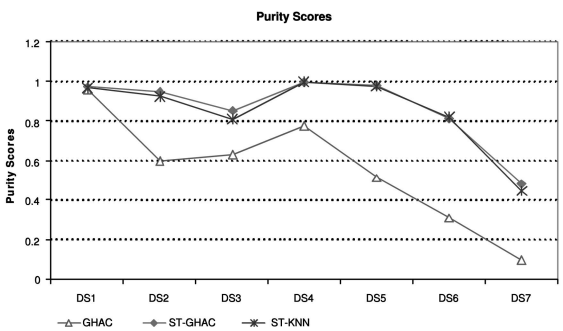

Fig. 9. The *Purity* scores of seven data sets for four clustering algorithms.

The phrase-based document similarity is a general definition which is independent of any clustering algorithm. Thus, we have considered using the new document similarity in other clustering algorithms, such as K-means, single pass [16] in the experiments. Unfortunately, the K-means clustering did not work well with the document similarity, because it is difficult to select suitable k centroids on the document similarity matrix that had been computed in our approach. In fact, the computation cost of the K-means clustering algorithm should become extremely high if we let k-means clustering algorithm directly work on the high-dimensional feature vectors of the nodes' *tf-idf* weights. Moreover, we found that the single pass clustering algorithm could be considered as a simplified version of the GHAC when we investigated the mechanism and source code of the algorithm. Thus, we finally selected two clustering algorithms (HAC and K-NN) for evaluating the effectiveness of phrase-based document similarity in the evaluation experiments. The experimental results indicate that, with the phrase-based document similarity, the GHAC achieves the best clustering results. The K-NN clustering algorithm also obtains quite good results. When K is set to 10, the quality of clustering results (ST-KNN) is very close to the results of the ST-GHAC.

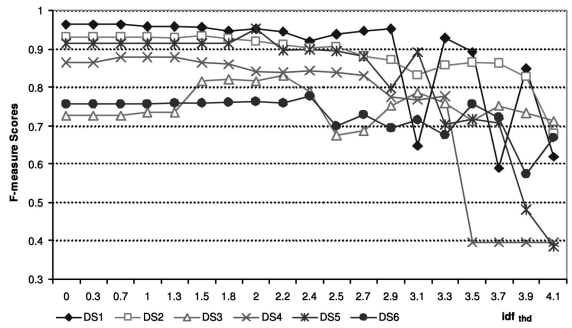

Fig. 11. The *F-measure* scores for different idf_{thd} values on six data sets.

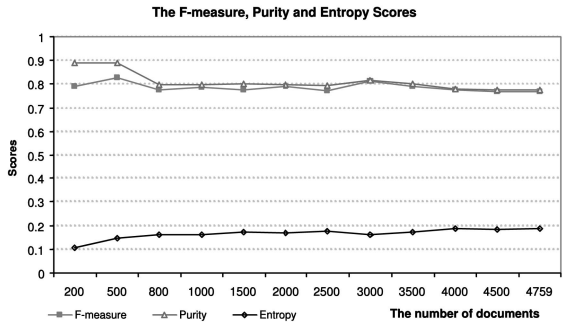

Fig. 13. The *F-measure*, Purity, and Entropy measure scores achieved on different number of documents in data set DS8.

4.5 The Effect of Threshold idf_{thd} in the Phrase-Based Document Clustering Approach

Fig. 11 shows the *F-measure* scores on the six data sets (DS1-DS6) that are effected by various idf_{thd} values from 0.0 to 4.1. It is easy to observe that there is no stopnodes problem in the small document data sets. However, along with the increasing size of the document data set, the idf_{thd} values can effect the *F-measure* scores within an offset of 0.149. The document clustering algorithm often obtains stable and higher *F-measure* scores when the idf_{thd} value is set to be 2.0-2.4 in the experimental data sets with over 800 documents. Thus, in our following experiments, the idf_{thd} value is empirically set to be 2.3.

4.6 The Performance Evaluation on Large Document Data Sets

To evaluate the performance of the new document similarity, we conducted a set of experiments on a large data set (DS8) that are generated from the *RCV1* document collection. The data set DS8 contains 500 documents of category *GSPO*, *M11*, respectively, and all documents of other eight categories. The total number of documents is 4,759.

To collect the variable length of the phrases denoted by the *first-level nodes*, we count the numbers of different *first-level nodes* by their phrase lengths. We find that the numbers of all different *first-level nodes* increase linearly to the size of the document data set. The total number of *first-level nodes* is equal to the number of keywords extracted from the corresponding data set. The length of the phrases denoted by the *first-level nodes* is variable from 1 to 9. In Fig. 12, the numbers of the phrases with a different length (1-5) scale along with the number of documents, where the curve *#1* designates the numbers of single-word phrases, *#2* designates the numbers of two-words phrases, *#3* designates the numbers of three-words phrases, and so on. The total number of five lengths of phrases (1-5) accounts for about 54 percent of the amount of all *first-level nodes* (with variable length phrases from 1 to 9) in the suffix tree generated from the document data set. Note especially that, the number of single-word phrases particularly accounts for over 92.6 percent in the amount of five lengths of phrases in Fig. 12.

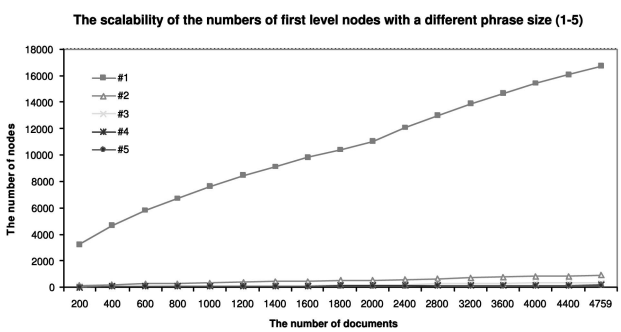

Fig. 12. The scalability of the numbers of different *first-level nodes*.

Fig. 13 illustrates that the *F-measure*, *Purity*, and *Entropy* scores vary along with the size of document data sets. The three measure scores indicate that the quality of the clusters is not affected by the size of document data set. To validate the observation, we increase the number of documents in the data set DS8 by adding 1,500 new documents from the category *GSPO* and *M11*, respectively. The document clustering algorithm (ST-GHAC) takes 24,213 seconds (about 6.73 hours) for clustering 7,858 documents of the new data set. The *F-measure* score is 0.8094, which is very close to the average of *F-measure* scores (0.7857) in Fig. 13. Thus, we can say that the phrase-based document clustering algorithm can work well on the large document data set. To the best of our knowledge, this is a distinctive advantage of the new clustering approach, which differs from the current document clustering approaches based on the VSD model.

Figs. 14 and 15 illustrate the time cost and memory cost for clustering different number of documents in the data set DS8.[3] Actually, the exact memory cost for building the suffix tree of various documents and computing the similarity matrix is hard to be calculated. We use a process monitoring tool called *top* to continuously record the memory usage status of the clustering program in the LINUX Server, and the top amount of the memory usage is recorded as the memory cost for the clustering algorithm. The results declare that the new phrase-based document clustering algorithm is a quite efficient practical approach.

4.7 Counting All Nodes versus Counting the Overlap Nodes Only

In this paper, we define the overlap nodes in the suffix tree as the internal nodes that are traversed by at least two different documents. That is to say, only the internal nodes with a *df* no less than 2 are selected in computing the document similarities, all other nodes are ignored including

3. Since the Pentium D 840 is a dual-core processor, and we have not used any parallel technique in the implementation, the clustering program can only occupy half of the CPU power in LINUX system.

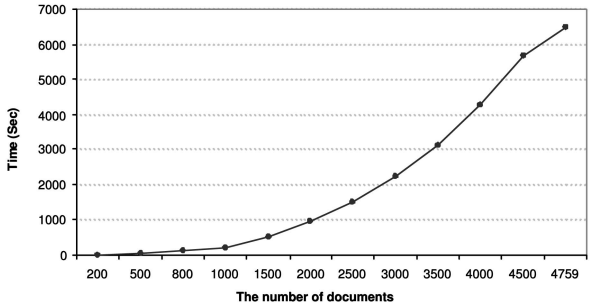

Fig. 14. The time cost for clustering the documents in data set DS8.

all leaf nodes and internal nodes that are traversed by only one document. With the overlap nodes only, we computed the document similarity matrixes of six data sets and used them in the GHAC. The clustering results are compared with the results of original algorithm (ST-GHAC) in Fig. 16, where ST-GHAC $(df \geq 1)$ designate the results of the document similarities with all nodes in the suffix tree, and ST-GHAC $(df \geq 2)$ designate the results of the document similarities with the overlap nodes only.

Comparing to the ST-GHAC $(df \geq 1)$, the *F-measure* score of ST-GHAC $(df \geq 2)$ just drops by 0.037 on average, *Purity* score drops by 0.006 on average, and the *Entropy* score increases by 0.015 on average. The comparison result indicates an optimized way to compute the phrase-based document similarity: By taking account of the overlap nodes only, the computation complexity of computing the document similarity will be reduced without compromising the quality of clustering result. Moreover, the high-quality clustering results based on the overlap nodes only also indicates that we can possibly get rid of the suffix tree data structure if we can find a more efficient way to identify and extract the LCPs in the documents. The suffix tree is a full text index data structure with many powerful functions that we do not need in the current approach; the high memory redundancy of the suffix tree data structure has been a concern since it was developed. Actually, the meaningful feature terms used in computing the document similarities are the common prefix phrases in the documents, not the nodes used to denote phrases.

5 Conclusions

Both the traditional VSD model and STD 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 the VSD model ignore the occurring position of words in the documents and the different semantic meanings of a word in different sentences are unavoidably discarded. The STD model keeps all sequential characteristics of the sentences in each document, the phrases consisting of one or more words are used to designate the similarity of two documents. Through analyzing the original STC algorithm, we become interested in the STD model. As a result of studying the property of STD model, we propose a phrase-based document similarity for document clustering. By mapping all nodes (or phrases) in the suffix tree into a M-dimensional term space of the VSD model, the new phrase-based document similarity successfully connects the two document models and inherits their advantages. The significant improvement of the clustering quality in our experiments clearly indicates that the word order preservation is critical to document clustering.

The concept of the suffix tree and the new document similarity are quite simple, but the implementation is complicated. To improve the performance of the phrase-based document similarity, we investigated the STD model in both the theoretical data structure analysis and the clustering algorithmic optimization. As a result, the efficiency of the new document clustering approach has been proven in our experiments on large document data sets. Obviously, the *group-average* hierarchical clustering algorithm with the phrase-based document similarity is a

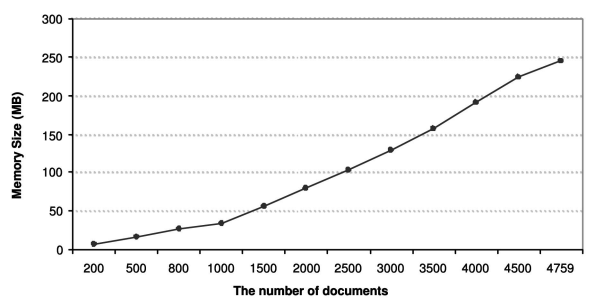

Fig. 15. The memory cost for clustering the documents in data set DS8.

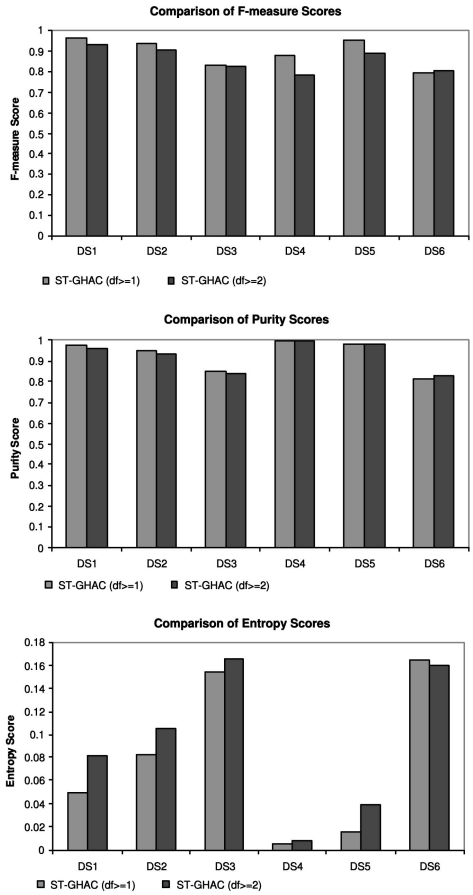

Fig. 16. The comparison of *F-measure*, *Purity*, and *Entropy* measures on six data sets (DS1-DS6).

highly accurate and efficient practical document clustering solution.

In our current approach, the suffix tree data structure is used as an n-gram technique to identify and extract phrases in the documents. We also believe that there are other efficient ways to identify and extract phrases in the documents. In fact, the phrases in documents are independent to the extraction methods and tools. For the first time, feature vectors of phrases' *tf-idf* weights are used in computing document similarities and are proven to be very effective in clustering documents. Our work has presented a successful approach to extend the usage of *tf-idf* weighting scheme: the term *tf-idf* weighting scheme is suitable for evaluating the importance of not only the keywords but also the phrases in document clustering.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their careful and insightful reviews. The work described in this paper was supported by a grant from CityU (7001975) and an NNSFC Grant (60496327).

REFERENCES

[1] P.O.R. Allen and M. Littman, "An Interface for Navigating Clustered Document Sets Returned by Queries," *Proc. ACM Conf. Organizational Computing Systems (COCS '93)*, pp. 166-171, 1993.
[2] W.B. Croft, "Organizing and Searching Large Files of Documents," PhD dissertation, Univ. of Cambridge, 1978.
[3] G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," *Comm. ACM*, vol. 18, no. 11, pp. 613-620, 1975.
[4] C.J. van Rijsbergen, *Information Retrieval*. Butterworths, 1975.
[5] O. Zamir and O. Etzioni, "Grouper: A Dynamic Clustering Interface to Web Search Results," *Computer Networks*, vol. 31, nos. 11-16, pp. 1361-1374, 1999.
[6] E. Charniak, *Statistical Language Learning*. MIT Press, 1993.
[7] M. Yamamoto and K.W. Church, "Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a Corpus," *Computational Linguistics*, vol. 27, no. 1, pp. 1-30, 2001.
[8] U. Manber and G. Myers, "Suffix Arrays: A New Method for On-Line String Searches," *SIAM J. Computing*, vol. 22, no. 5, pp. 935-948, 1993.
[9] K.M. Hammouda and M.S. Kamel, "Efficient Phrase-Based Document Indexing for Web Document Clustering," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 10, pp. 1279-1296, Oct. 2004.
[10] O.M. Oren Zamir, O. Etzioni, and R.M. Karp, "Fast and Intuitive Clustering of Web Documents," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, 1997.
[11] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Proc. 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, 1998.
[12] D.S. Sven Meyer zu Eissen and M. Potthast, "The Suffix Tree Document Model Revisited," *Proc. Fifth Int'l Conf. Knowledge Management (I-Know '05)*, pp. 596-603, 2005.
[13] D.D. Lewis, Y. Yang, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *J. Machine Learning Research*, vol. 5, pp. 361-397, 2004.
[14] W. Hersh, C. Buckley, and D. Hickam, "Ohsumed: An Interactive Retrieval Evaluation and New Large Test Collection for Research," *Proc. 17th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '94)*, pp. 192-201, 1994.
[15] P. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing and Management*, vol. 24, no. 5, pp. 577-597, 1988.
[16] K. Cios and W. Pedrycz, *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Publishers, 1998.
[17] J.J. Carlson, M.R. Muguira, J.B. Jordan, G.M. Flachs, and A.K. Peterson, "Final Report: Weighted Neighbor Data Mining," *SANDIA Report*, vol. SAND2000-312, 2000.
[18] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1997.
[19] J.R. Paul Bieganski and J.V. Carlis, "Generalized Suffix Trees for Biological Sequence Data: Application and Implementation," *Proc. 27th Ann. Hawaii Int'l Conf. System Sciences (HICSS '94)*, pp. 35-44, 1994.
[20] A. Ehrenfeucht and D. Haussler, "A New Distance Metric on Strings Computable in Linear Time," *Discrete Applied Math.*, vol. 40, 1988.
[21] B.M. Rajesh Pampapathi and M. Levene, "A Suffix Tree Approach to Anti-Spam Email Filtering," *Machine Learning*, vol. 65, 2006.
[22] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999.
[23] G. Salton and C. Buckley, "On the Use of Spreading Activation Methods in Automatic Information Retrieval," *Proc. 11th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '88)*, pp. 147-160, 1988.
[24] E. Ukkonen, "On-Line Construction of Suffix Trees," *Algorithmica*, vol. 14, no. 3, pp. 249-260, 1995.
[25] R. Giegerich and S. Kurtz, "From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction," *Algorithmica*, vol. 19, no. 3, pp. 331-353, 1997.
[26] M. Porter, "New Models in Probabilistic Information Retrieval," British Library Research and Development Report, no. 5587, 1980.
[27] M. Eisen, *Cluster 3.0*. Stanford Univ.
[28] B. Larsen and C. Aone, "Fast and Effective Text Mining Using Linear-Time Document Clustering," *Proc. KDD Workshop Web Usage Analysis and User Profiling (WebKDD)*, 1999.
[29] K. Lang, "Newsweeder: Learning to Filter Netnews," *Proc. 12th Int'l Conf. Machine Learning (ICML '95)*, pp. 331-339, 1995.

Hung Chim received the BEng degree from Huazhong University of Science and Technology, Wuhan, China, in 1988 and the MS degree from the City University of Hong Kong, in 2001. He is a PhD student at the City University of Hong Kong. His current research interests include information retrieval, distributed system, and Web service.

Xiaotie Deng received the BS degree from Tsinghua University, Beijing, in 1982, the MS degree from the Chinese Academy of Sciences, Beijing, in 1984, and the PhD degree from Stanford University, California, in 1989. He is a chair professor at the City University of Hong Kong. His current research interests include algorithms and applications in Internet economics and Internet computing. After finishing his PhD, he received an International Research Fellowship from the Natural Science and Engineering Council of Canada at Simon Fraser University, Burnaby, British Columbia, Canada. In 1991, he joined York University, Toronto as an assistant professor and then tenured as an associate professor. He joined the Department of Computer Science, City University of Hong Kong in 1997. He is a senior member of the IEEE.