A Novel Weighted Phrase-Based Similarity for Web Documents Clustering

Ruilong Yang
College of Computer Science, Chongqing University, Chongqing, China
Email: ruilong.yangrl@gmail.com

Qingsheng Zhu and Yunni Xia
College of Computer Science, Chongqing University, Chongqing, China
Email: qszhu@cqu.edu.cn, xiayunni@yahoo.com.cn

***Abstract*—Phrase has been considered as a more informative feature term for improving the effectiveness of document clustering. In this paper, a weighted phrase-based document similarity is proposed to compute the pairwise similarities of documents based on the Weighted Suffix Tree Document (WSTD) model. The weighted phrase-based document similarity is applied to the Group-average Hierarchical Agglomerative Clustering (GHAC) algorithm to develop a new Web document clustering approach. According to the structures of the Web documents, different document parts are assigned different levels of significance as structure weights stored in the nodes of the weighted suffix tree which is constructed with sentences instead of documents. By mapping each node and its weights in WSTD model into a unique feature term in the Vector Space Document (VSD) model, the new weighted phrase-based document similarity naturally inherits the term TF-IDF weighting scheme in computing the document similarity with weighted phrases. The evaluation experiments indicate that the new clustering approach is very effective on clustering the Web documents. Its quality greatly surpasses the traditional phrase-based approach in which the Web documents structures are ignored. In conclusion, the weighted phrase-based similarity works much better than ordinary phrase-based similarity.**

***Index Terms*—suffix tree, web document clustering, weight computing, phrase-based similarity, document structure**

I. INTRODUCTION

In an effort to keep up with the exponential growth of the WWW, many researches are targeted on how to organize such information in a way that will make it easier for the end users to find the information efficiently and accurately. One of these is document clustering [1-4] which tries to identify inherent groupings of the text documents so that a set of clusters is produced in which clusters exhibit high intracluster similarity and low intercluster similarity. Text document clustering methods attempt to segregate the documents into groups where each group represents a topic that is different than those topics represented by the other groups[4]. Applications of document clustering include: clustering of retrieved documents to present organized and understandable results to the user, clustering documents in a collection, automated creation of document taxonomies, and efficient information retrieval by focusing on relevant subsets rather than whole collections[1].

In general, the clustering techniques are based on four concepts [4, 5]: data representation model, similarity measure, clustering model, and clustering algorithm. Most of the current documents clustering methods are based on the Vector Space Document (VSD) model. 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. In particularly, the term weights, usually TF-IDF, of the words are also contained in each feature vector. 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[5].

To achieve a more accurate document clustering, phrase has been considered as a more informative feature term in recent research work and literature [3-5]. A phrase of a document is an ordered sequence of one or more words. Hammouda[4] proposed a phrase-based document index model called Document Index Graph (DIG), which allows for the incremental construction of a phrase–based index for a document set. The quality of clustering based on this model significantly surpassed the traditional VSD model-based approaches in the experiments of clustering Web documents. Another important phrase-based clustering, Suffix Tree Clustering (STC), is based on the Suffix Tree Document (STD) model which was proposed by Zamir and Etzioni[6]. The STC algorithm was used in their meta-searching engine to cluster the document snippets returned from other search engine in realtime. They claim to achieve $N\log(N)$ performance with the size of the documents set and produce clusters of high quality. The algorithm is based on identifying the phrases that are common to groups of documents. However, the STD model and STC algorithm

Supported by National Key Technology R&D Program of China (No.2007BAH08B04), Chongqing Key Technology R&D Program of China (No.2008AC20084), CSTC Research Program of Chongqing of China (No.2009BB2203) and Post-doctorial Science Foundation of China (No. 20090450091).

doi:10.4304/jsw.6.8.1521-1528

have not been analyzed in their work as pointed out by Zu Eissen [7], The STC algorithm got poor results in some case of documents sets.

By studying the STD model, Chim and Deng [5] 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 substring are selected as phrases to label the edges or nodes of a suffix tree. They combine the advantages of STD and VSD models to develop an approach for document clustering. By mapping each node of a suffix tree except 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. With these feature vectors, the *cosine* similarity measure is employed to compute the pairwise similarities of documents which are applied to the Group-average Hierarchical Agglomerative Clustering (GHAC) algorithm [8].

However, when applying these clustering methods to Web documents, the characteristics of the Web documents are ignored. Web documents are known to be semi-structured. HTML tags are used to designate different parts of the document and identify key parts based on this structure[4]. Some parts of the Web document are more informative than other parts, thus having different level of significance based on where they appear in the document and the tags that surround them. For example, it is inappropriate to treat the title and the body text equally.

Most of the Web documents in data sets have some hundreds words. The suffix tree constructed with documents has many long paths. Some phrases extracted from the suffix tree would be much longer than needed length and cross the boundaries of sentences. So, the different parts of documents can be partitioned into a set of sentences. Phrases can be extracted in boundaries of the sentences from the suffix tree constructed with sentences instead of documents.

A Weighted Suffix Tree Document (WSTD) model is proposed to develop a novel approach for clustering Web documents. This approach consists of four parts: (1) Analyze the HTML document and assign different levels of significance to different document parts as structure weights. (2) Represent different parts of the documents as some sentences with levels of significance. The sentence is represented as a sequence of words, not characters. These sentences instead of documents are used to build WSTD model. Except the root node, each node of the WSTD model contains document ID, sentence ID with its level of significance, and the traversed times of each sentence. (3) Map each node into a unique dimension of an *M*-dimensional term space. Each document is represented by a feature vector of *M* nodes. The statistical features and significance levels of all nodes are taken into account of the feature term weights and similarity measures. (4) Apply the documents similarities to GHAC algorithm to cluster the Web documents.

The rest of this paper is organized as follows: Web document structure analysis is presented in Section 2; Section 3 starts with the definition of the weighted suffix tree and WSTD model, and then presents the detailed design of the new weighted phrase-based document similarity; Section 4 illustrates the experimental results of evaluation for the new WSTD model, with the comparison of STD model; Section 5 summarizes the work.

II. Web Document Representation

For Web documents, HTML tags are used to designate different parts of the document and also to identify key parts of the document based on this structure. One of three levels of significance is assigned to the different parts[4]; HIGH (H), MEDIUM (M), and LOW (L). Examples of HIGH significance parts are the title, meta keywords, meta description, and section headings. Examples of MEDIUM significance parts are texts that appear in bold, italic, colored, hyper-linked text. LOW significance parts are usually comprised of the document body text that was not assigned any of the other levels. This structuring scheme is exploited in measuring the similarity between two documents. For example, a phrase in title is much more informative than in body text.

A formal model is presented that the document is represented as a vector of sentences, which in turn are composed of a vector of terms[4]:

$$d_i=\{s_{ij}: j=1,\ldots,p_i\} , \quad (1a)$$

$$s_{ij}=\{t_{ijk}:k=1,\ldots,l_{ij}; w_{ij}\} , \quad (1b)$$

where

d_i: is document i in documents set D,
s_{ij}: is sentence j in document i,
p_i: is the number of sentences in document i,
t_{ijk}: is term k of sentence s_{ij},
l_{ij}: is the length of sentence s_{ij}, and
w_{ij}: is the level of the significance associated with sentence s_{ij}.

III. Clustering Method Based on WSTD Model

The main steps of the new approach are constructing WSTD model, mapping each node into a unique dimension of an *M*-dimensional vector space, computing feature term weights, finally applying GHAC to clustering Web documents.

A. Definition of Weighted Suffix Tree

For a sentence *s* of *n* words, in order to build a suffix tree[6, 9], a nonempty character $ is appended to the end of the sentence *s* with Ukkonen's algorithm[9]. The length of *s* is changed to *n*+1. When constructing the suffix tree, the sentences *s* is represented as a sequence of words, not characters. The *k*th suffix of a sentence *s* is the substring of *s* that starts with word *k* to the end of *s*.

Definition 1 The suffix tree *T* of the sentence *s* is a rooted, compact trie containing all the suffixes of *s*. Each internal node has at least 2 children. Each edge is labeled with a nonempty substring of *s*. The label (phrase from

the node) of a node is defined to be the concatenation of the edge-labels on the path from the root to that node. No two edges out of the same node can have edge-labels that begin with the same word. For each suffix $s[k…n+1]$ of s, there exists a suffix-node whose label equals it.

Definition 2 The weighted suffix tree T_s of the sentence s is based on a suffix tree of s, each node v of which records the sentence s, the level of significance w of s, and the times r that s traverses the node v. Its information is represented as a triple (s, r, w).

Definition 3 The weighted suffix tree T_d of document d_i is based on a suffix tree which p_i sentences are inserted into. Each node records a list of triple (s_{ij}, r_{ij}, w_{ij}). Each internal node has at least 2 children. Each edge is labeled with a nonempty substring of s_{ij} of d_i. For any leaf node, the concatenation of the edge-labels on the path from root to itself is a suffix of s_{ij} of d_i, where $0 \le j < p_i$.

Definition 4 The weighted suffix tree T_D of documents set D is based on a suffix tree which N documents are inserted into. Each document d_i contains p_i sentences. Thus, $\sum p_i$ sentences are inserted into T_D. Each node records a list of triple (s_{ij}, r_{ij}, w_{ij}). Each internal node has at least 2 children. Each edge is labeled with a nonempty substring of s_{ij} of d_i in D. For any leaf node, the concatenation of the edge-labels on the path from root to itself is a suffix of s_{ij} of d_i, where $0 \le j < p_i$, $0 \le i < N$.

Four properties are concluded from T_D.

Property 1 Any node v of the weighted suffix tree T_D has its depth level L_v. Each phrase P_v denoted by any internal node v at a higher level ($L_v \ge 2$) in T_D contains at least two words. The length of the phrase $|P_v| \ge 2$. Phrases of various lengths can be extracted.

Property 2 The levels of significance (structure weights) of any node appearing in different sentences or documents can be obtained from T_D.

Property 3 Summing up all times which the sentences of a document traverse a node v, the node frequency, i.e., phrase frequency in a document, can be obtained. Phrase frequency is denoted by *tf*.

Property 4 Summing up all times which different documents traverse a node, the document frequency of a node can be obtained . Document frequency is denoted by *df*.

Level of significance, *tf*, and *df* of each node are applied to modified TF-IDF weighting scheme for computing the feature terms weights.

Fig.1 is an example of a suffix tree composed from two Web documents.

d_0: <title>cat ate cheese. </title><body>mouse ate cheese too. </body>

d_1: <bold>cat ate mouse too. </bold>

The nodes in the suffix tree are drawn in circles. There are three kinds of nodes: the root node, internal node, and leaf node. One kind of leaf nodes is called terminal node which is labeled with a special single phrase ”$”. Each leaf node designates a suffix substring of a document; each internal node represents a common phrase shared by at least two suffix substrings. 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 designate a document identifier, the number below designates the traversed times of the document.

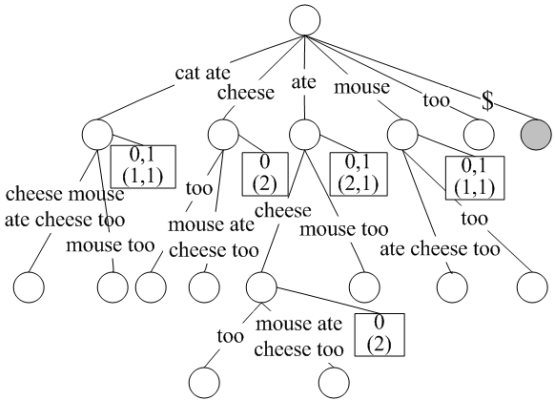

Figure 1. Suffix tree of documents set.

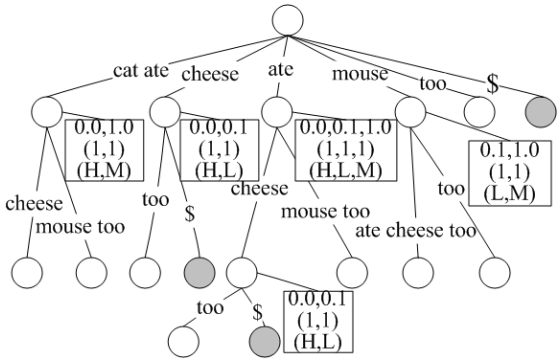

Figure 2. Weighted suffix tree of documents set.

After inserting many documents to the suffix tree, it is found that the longest suffix is the length of a document and the deepest level would be at worst the length of the longest document. The longer phrases make up a large percentage of all phrases.

Fig.2 is an example of a weighted suffix tree composed of three sentences with levels of significance which originate from the two documents d_0 and d_1 partitioned by HTML tags.

Each internal node is attached to an individual box. Each upper number designates a sentence identifier which contains document identifier in the left of the period mark, the middle number designates the traversed times of the corresponding sentence, the number below designates the level of significance of corresponding sentence.

Generally, the sentence is treated as a unit to represent complete semantic concept in most natural language including English. The deepest level of the weighted suffix tree is length of the longest sentence which is generally bounded by a constant. The length of phrases extracted from weighted suffix tree is much less than from ordinary suffix tree. From the weighted suffix tree, structure weight *w*, *tf*, and *df* of each node can be obtained.

B. Construction of WSTD Model

Before constructing the WSTD model, a document preprocessing procedure is executed for all Web documents in data sets. First, all Web documents are

divided into some different parts with significance level according to the HTML tags, such as <title>, <body>, and <bold>. Second, the parts are partitioned into many sentences by punctuation marks, such as periods, semicolon, and so on. Third, these sentences are parsed into words. All stop words are identified and removed. The Porter's suffix-stripping algorithm is used to stem the words. All stemmed words are concatenated into a new sentence. Finally, all the sentences and their significance levels are put into two 2-dimensional arrays respectively, one for sentences contents, the other for significance level. The first dimension is for document identifier, the second for sentence identifier.

All contents of the two arrays are sequentially passed into the weighted suffix tree as described in previous section. Each node records a list of (s_{ij}, r_{ij}, w_{ij}). Thus, the WSTD model of Web documents set D is constructed.

Each node v of WSTD model has its level of depth, L_v. The children of the root node are in first-level, i.e., L_v=1. The phrase of each internal node represents a LCP (Longest Common Phrase) of Web documents set D. The Phrase of each leaf node represents a suffix substring of a sentence. The total number of first-level nodes is approximately equal to the number of keywords extracted from the corresponding data set [5]. The phrases of nodes with its L_v equal to or greater than 2 have at least 2 words.

C. Similarity Based on the Weighted Phrases

In this paper, the symbols N, M, and q are used to denote the number of documents, the number of feature terms, and the number of clusters, respectively. The C_1, C_2,…,C_q are used to denote each one of the q clusters of result.

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 uses the VSD model to represent documents. Each document d is considered to be a vector in the M-dimensional feature term space. In this paper, the TF-IDF weighting scheme is employed, in which each document d can be represented as[5]:

$$\vec{d} = \{w(1,d), w(2,d), ..., w(M,d)\} , \quad (2a)$$

$$w(i,d)=(0.1+\log tf(i,d))\times\log(0.1+N/df(i)) , \quad (2b)$$

where $w(i,d)$ is ith feature term weight in document d, $tf(i,d)$ is the frequency of the ith term in the document d, and $df(i)$ is the number of documents containing the ith term.

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

$$sim(d_i, d_j) = \frac{\vec{d}_i \bullet \vec{d}_j}{\left|\vec{d}_i\right| \times \left|\vec{d}_j\right|} , \quad (3)$$

As mentioned earlier, there are three different kinds of nodes in WSTD model. Each internal node represents a nonempty phrase that appears in at least two documents in the Web data set.

By mapping each internal node v labeled by a nonempty phrase into a feature of M-dimensional term space, each Web document d can be represented as a feature vector of the weights of M node terms in the VSD model [5]. 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.

With respect to WSTD model, a node v can appear in different sentences with its level of significance. When mapping a node v of WSTD model to VSD model, besides $df(v)$ and $tf(v,d)$, the feature term weight depends on the level of significance of v.

The symbols $tf^H(v,d)$, $tf^M(v,d)$, and $tf^L(v,d)$ are used to denote the term frequency of the node v in HIGH, MEDIUM and LOW significance parts of a document respectively. The term frequency $tf(v,d)$ of node v is updated to:

$$tf(v,d)=w^H\times tf^H(v,d)+w^M\times tf^M(v,d)+w^L\times tf^L(v,d) , \quad (4)$$

where w^H, w^M, w^L are used to denote different level of significance of a node v, w^L is assigned 1, and the others are greater than w^L. The feature term weight of node v is represented as.

$$w(v,d)=(0.1+\log tf(v,d))\times\log(0.1+N/df(v)) . \quad (5)$$

The effect of the denominator of (3) is to length-normalize the vectors of d_i and d_j to unit vectors. Let unit vector $\vec{u}_x = \{x_1, x_2, ..., x_M\}$ and $\vec{u}_y = \{y_1, y_2, ..., y_M\}$ denote two documents d_x and d_y, where x_i and y_i are the normalized weights of corresponding node term respectively.

The similarity of two documents is calculated by the following formula:

$$sim(d_x, d_y) = \vec{u}_x \bullet \vec{u}_y . \quad (6)$$

D. Computation Complexity of the Novel Approach

The naïve and straightforward method of building suffix tree needs comparing each suffix substring of the document d with all suffix substrings which already exist in the tree. The time complexity for a document of m words is $O(m^2)$. Ukkonen[9] provides an algorithm with time complexity of $O(m)$. Zamir[6] declared that the time cost for building a suffix tree of N documents is $O(N)$, where the length of a document m is assumed to be bounded by a constant. In WSTD model, a document d is partitioned into n_d sentences. The length of sentence s is assumed to be bounded by a constant. Thus, n_d is bounded by a constant. Therefore, the time complexity of building WSTD model is also $O(N)$.

The time complexity of using GHAC algorithm to cluster Web documents is $O(N^2\log N)$[8]. For the dimensionality M of features vector usually is large, similarity computations are time consuming.

As a result, the overall time complexity of the proposed approach is $O(N^2\log N)$.

For the space complexity, Ukkonen's algorithm is criticized for the large memory redundancy. 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[5, 9]. The memory cost of a suffix tree can be estimated by counting the total numbers of nodes which increase proportionally to the number of documents.

For features vectors, the memory complexity is $O(NM)$; the memory complexity of similarities matrix is $O(N^2)$.

The total memory cost can not be calculated precisely on Java platform. In experiments, *System* class of Java is used to estimate the memory cost because of the Java GC is out of control.

E. Removing Stopnodes

Stopwords are frequently occurring, insignificant words that appear in documents. A standard stopwords list is used to process the document. Similar to stopwords, some nodes are frequently occurring in documents and insignificant. These nodes are treated as stopnodes[5]. A node with a high document frequency *df* can be ignored in computing similarities of documents. In the new approach, a threshold df_{thd} is introduced to identify whether a node is a stopnode. In experiments, df_{thd} is determined by the statistical features of *df* histogram of nodes. All the stopnodes with *df* greater than df_{thd} are excluded in computing the document similarities.

IV. EXPERIMENTAL RESULTS

In order to test the effectiveness and efficiency of the WSTD model, a series of experiments are conducted to compare the new weighted phrase-based document similarity based on WSTD model with the traditional phrase-based similarity measure based on STD model in the same GHAC algorithm. Without loss of generality, first some Web document data sets without any bias must be provided, and then some standard clustering quality metrics shall be examined.

A. Experimental Setup

Three data sets are used, two of which are Web document data sets, and the third is a collection of the search engine results called AMBIENT[10] in which each document includes a title and a snippet. Table 1 describes the data sets. The first data set (DS1) is a collection of 314 Web documents manually collected and labeled from various University of Waterloo and Canadian Web sites. It is categorized manually based on topic description. The second data set (DS2) is a collection of 2340 Reuters news articles posted on Yahoo! News. The categories of the data set come from the Yahoo categories of Reuters news feed. Data set DS1 and DS2 are used in[4].

The proposed approach is implemented in Java and all experiments are conducted on an Intel core2 2GHz Windows XP computer with 2G memory and JDK 6. For the weighted suffix tree construction, an open source library of the Carrot2[11] is used. But there exist some errors and defects for counting the times which documents traverse nodes. These errors are corrected in the work and Carrot2 is extended with levels of significance of nodes for WSTD model. The GHAC algorithm is implemented in Java by following the description in [8].

TABLE II. DATA SETS DESCRIPTIONS

Data Sets	Name	Type	#Docs	Categories	Avg. words
DS1	UW-CAN	HTML	314	10	469
DS2	Yahoo news	HTML	2340	20	289
DS3	AMBIENT	TEXT	4400	44	41

TABLE I. DATA SETS NODES AND TIME

Data Sets	#Cates	#Docs	#Nodes	#All (ms)	#GHAC (ms)
DS1-1	5	100	11162	18344	16219
DS1-2	5	142	20969	64671	60515
DS2-1	5	100	6422	10313	9172
DS2-2	5	200	13894	115656	112563
DS3-1	5	100	439	985	844
DS3-2	5	220	1109	11641	11454

B. Evaluation Metrics

In order to evaluate the quality of the clustering, two quality measures are adopted, which are widely used in the text mining literature for the purpose of document clustering [2-5]. The first is the *F*-measure, which combines the *Precision* and *recall* ideas in information retrieval. Recalling $C=\{C_1,C_2,\ldots,C_q\}$ is a clustering of data set D of N documents, let $C'=\{C'_1,C'_2,\ldots,C'_l\}$ designate the original class set of D. The *recall* and *precision* of cluster j with respect to class i are defined as:

$$P = \text{precison}(i, j) = \frac{\left|C_j \cap C_i'\right|}{\left|C_j\right|} , \tag{7a}$$

$$R = \text{recall}(i, j) = \frac{\left|C_j \cap C_i'\right|}{\left|C_i'\right|} . \tag{7b}$$

The *F*-measure is defined as:

$$F(i, j) = \frac{2PR}{P + R} . \tag{8}$$

Based on (8), the *F*-measure for overall quality of cluster set C is defined as:

$$F = \sum_{i=1}^{l} \frac{\left|C_i'\right|}{N} \cdot \max_{j=1,\ldots,q} \{F(i, j)\} . \tag{9}$$

The second measure is the *Entropy*, which provides a measure of "goodness" for unnested clusters or for the clusters at one level of a hierarchical clustering. *Entropy* tells how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. For every cluster C_j in the clustering result C, the computed p_{ij} is the probability that a member of cluster C_j belongs to class C'_i. The *entropy* of each cluster

C_j is calculated using the standard formula $E_j = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy of a set of clusters is calculated as the sum of entropies of each cluster weighted by size of that cluster:

$$E = \sum_{j=1}^{q} (\frac{|C_j|}{N} \times E_j) \quad . \tag{10}$$

To sum up, the approach maximizes the *F*-measure and minimizes the *entropy* score of the clusters to achieve a high-quality clustering.

C. Comparison of STD and WSTD Model

A number of document clustering approaches have been developed based on different documents models and various clustering algorithms for decades. In general, the approaches can be categorized into agglomerative solutions and partitional solutions [5, 8]. Hierarchical Agglomerative Clustering(HAC) algorithm[8] starts with each instance representing a cluster. The algorithm recursively merges clusters until a stop criterion is met. 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. In experiments, the group-average similarity is chosen for the HAC algorithm for some reasons in [5], which measures the similarity of two clusters with the average of pairwise similarities of the documents from each cluster. The effectiveness of two document models is evaluated with GHAC algorithm in the comparison experiments: STD model and WSTD model.

In the comparison experiments, two subsets of each original data set are selected to be six small data sets: DS1-1, DS1-2, DS2-1, DS2-2, DS3-1 and DS3-2, as illustrated in Table 2, where #Cates denotes the number of categories, #Docs denotes the number of documents. For WSTD model, coefficient w^H is 2 times w^L, w^M is 1.5 times w^L. The stop criterion of GHAC is adjusted so that the number of clusters is equal to the number of the original class set of each data set.

Table 3 and Table 4, respectively, illustrates the *F*-measure and *Entropy* scores computed from the clustering results of GHAC algorithm with STD and WSTD model. Table 3 illustrates the improvement on the *F*-measure score, and Table 4 illustrates the improvement on the *Entropy* score.

In order to better illustrate the improvements of the weighted phrase-based similarity on the clustering quality, the experiments results are shown in Fig.3 and Fig.4.

Compared with the results of the GHAC algorithm based on STD model, the GHAC algorithm based on WSTD model achieves an increment of 0.02 to 0.16 on the average *F*-measure score, or 2.1 to 29.4 percent. The new approach achieves a decrement of 0.01 to 0.1 on the average *Entropy* score, or 2.6 to 42.2 percent. The improvements on the *F*-measure and *Entropy* scores indicate that the new weighted phrase-based document clustering approach based on WSTD model tends to be a highly accurate documents clustering approach.

D. The Performance Evaluation

The time cost and the nodes mapped into the vector space are illustrated in Table 2, where #Nodes denotes the number of nodes which appear in at least two documents and are mapped into vector space, #All denotes the total time of the approach executed, and #GHAC denotes the time of GHAC algorithm except the time of building WSTD model. The unit of time is millisecond (ms). Comparing #All and #GHAC, it is concluded that the main time cost is caused by GHAC

TABLE III. IMPROVEMENT ON F-MEASURE

Data Sets	STD	WSTD	Improvement
DS1-1	0.899	0.96	6.7%
DS1-2	0.957	0.977	2.1%
DS2-1	0.542	0.702	29.4%
DS2-2	0.569	0.614	7.8%
DS3-1	0.817	0.909	11.3%
DS3-2	0.821	0.915	11.5%

TABLE IV. IMPROVEMENT ON ENTROPY

Data Sets	STD	WSTD	Improvement
DS1-1	0.112	0.065	42.2%
DS1-2	0.064	0.040	37.4%
DS2-1	0.419	0.318	24.2%
DS2-2	0.368	0.358	2.6%
DS3-1	0.254	0.178	29.8%
DS3-2	0.185	0.157	15.2%

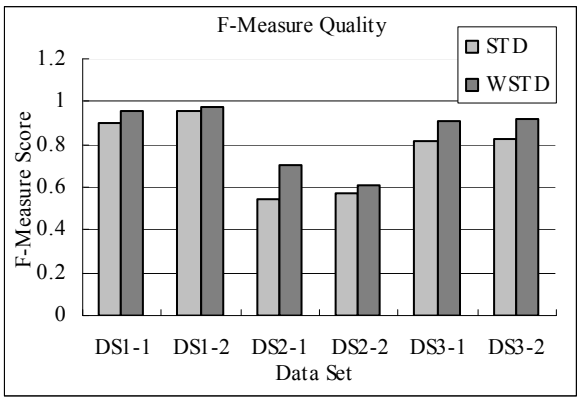

Figure 3. The *F*-measure scores of six data sets.

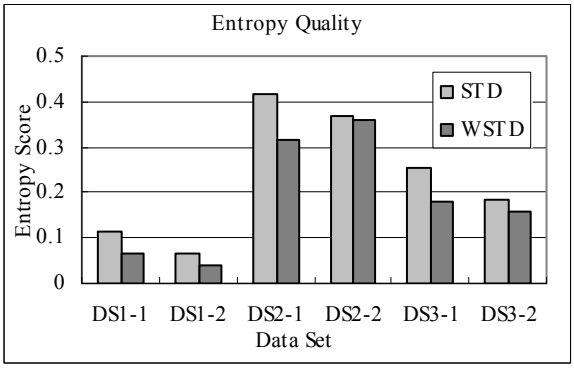

Figure 4. The entropy scores of six data sets.

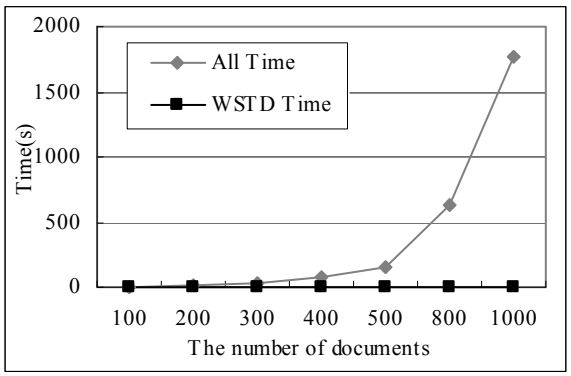

Figure 5. The time cost for clustering in data set DS3.

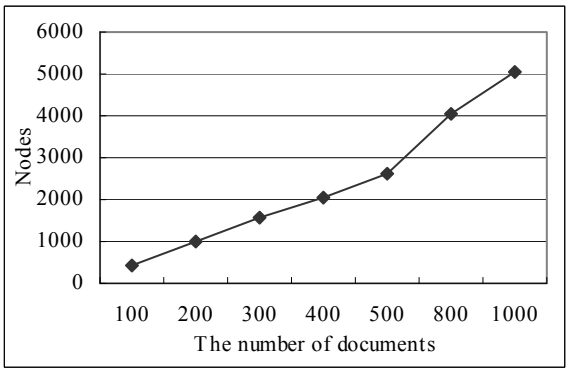

Figure 6. The scalability of the numbers of nodes.

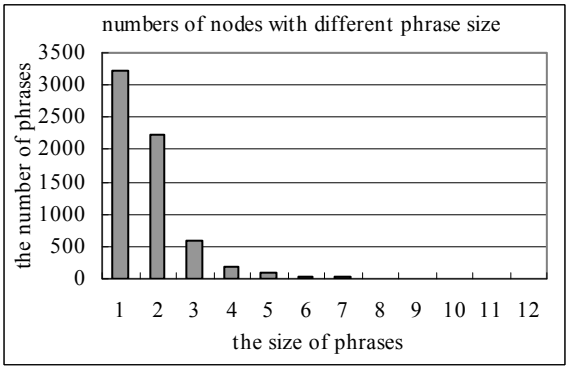

Figure 7. Distribution of phrases of DS2-1.

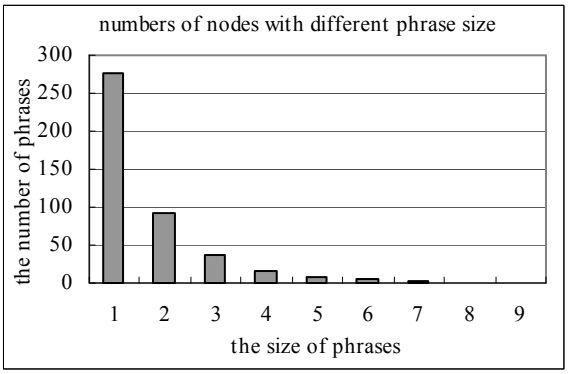

Figure 8. Distribution of phrases of DS3-1.

algorithm, and constructing the WSTD model needs a little time.

In order to better illustrate the time cost for clustering different number of documents, an experiment is conducted on data set DS3. Fig.5 shows that the all time of the proposed approach increases quadraticly with the size of data set, whereas time of building WSTD model increases a little. The unit of time is second.

From Table 2, it is found that the #Nodes increase with the size of documents set and the average length of documents. With the increment of nodes, the memory cost increase significantly. With respect to data set DS2-1 and DS3-1, there is a same size of data set; the memory cost and time cost increase significantly with the average length of documents. In Fig.6, #Nodes scales along with the number of documents.

Actually, the exact memory cost for building the WSTD model and computing similarities is hard to be calculated. Java class *System* is used to record the total memory and free memory, and to estimate the memory usage status. When the JVM argument -Xmx is assigned 1024M, all experiments are conducted successfully. Because of the Java GC out of control, the accurate memory cost can't be obtained.

E. The Variable Length of Phrases

In WSTD model, a document is partitioned into some sentences. The length of a sentence is assumed to be bounded by a constant. The Phrases extracted from WSTD model are also bounded a constant. As described in [4, 6], the length of phrases less than 6 is appropriate and efficient.

To collect the variable length of the phrases of nodes and count the numbers by their phrases length, some experiments are conducted on data sets DS2-1, DS3-1. The distribution of the phrases by the length is illustrated in Fig.7 and Fig.8. The length of the phrases is variable from 1 to 12 on data set DS 2-1. The total number of six lengths of phrases (from 1 to 6) accounts for about 99 percent of the amount of all nodes mapped into VSD model on DS2-1. The length of the phrases is variable from 1 to 9 on data set DS3-1. The total number of six lengths of phrases (from 1 to 6) accounts for about 99 percent of the amount of all nodes mapped into VSD model on DS3-1.

The distributions of six lengths of phrases (from 1 to 6) indicate that the most efficient phrases can be extracted from the WSTD model.

V. CONCLUSIONS

The traditional VSD model ignores 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 words in documents; the phrase consisting of one or more words are used to designate the similarity of two documents. By mapping all internal nodes in the suffix tree into VSD model, phrase-based similarity successfully connects the two models.

For Web documents, HTML tags can identify key parts of document. Some parts are more informative than other parts. Different parts are assigned a level of significance.

A document is partitioned into some sentences with significance; a sentence is represented as a sequence of words, not characters. These sentences are used to build the weighted suffix tree instead of documents. The STD model is extended to the WSTD model with the weighted nodes. Thus, the weighted phrase-based document similarity is proposed. The significant improvements of the clustering quality in experiments clearly indicate that the new clustering approach with the weighted phrase-based document similarity is more effective on clustering the Web documents and the Web documents structure is critical to Web document clustering.

The new weighted phrase-based document similarity is a general definition which is independent of any clustering algorithm. Using sentences to build WSTD model, the length of the phrase is bounded by a constant. Most efficient phrases can be extracted from the WSTD model which is used as an n-gram technique to identify and extract weighted phrases in Web documents. The work has presented a successful approach to extend the usage of TF-IDF weighting scheme.

ACKNOWLEDGMENT

The authors are grateful to reviewers for their careful and insightful reviews. This work was partly supported by the National Key Technology R&D Program of China (No.2007BAH08B04), the Chongqing Key Technology R&D Program of China (No.2008AC20084), CSTC Research Program of Chongqing of China (No.2009BB2203) and Post-doctorial Science Foundation of China (No. 20090450091).

REFERENCES

[1] N. Oikonomakou, and M. Vazirgiannis, "A Review of Web Document Clustering Approaches," Data Mining and Knowledge Discovery Handbook, pp. 921-943: Springer US, 2005.

[2] L. Yanjun, “Text Clustering with Feature Selection by Using Statistical Data,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, pp. 641-652, 2007.

[3] Y. Li, S. M. Chung, and J. D. Holt, “Text Document Clustering Based on Frequent Word Meaning Sequences,” Data & Knowledge Engineering, vol. 64, no. 1, pp. 381-404, 2008.

[4] K. M. Hammouda, and M. S. Kamel, “Efficient Phrase-Based Document Indexing for Web Document Clustering,” IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, pp. 1279-1296, 2004.

[5] H. Chim, and X. Deng, “Efficient Phrase-Based Document Similarity for Clustering,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 9, pp. 1217-1229, 2008.

[6] O. Zamir, and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 46-54, 1998.

[7] S. Zu Eissen, B. Stein, and M. Potthast, “The Suffix Tree Document Model Revisited,” in Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05), Graz, Austria, 2005, pp. 596-603.

[8] C. Manning, P. Raghavan, and H. Schütze, "An introduction to information retrieval," p. 377~400, Cambridge,England: Cambridge University Press, 2009.

[9] E. Ukkonen, “On-Line Construction of Suffix Trees,” Algorithmica, vol. 14, no. 3, pp. 249-260, 1995.

[10] C. Carpineto, and G. Romano. "Ambient Dataset," 2008; http://credo.fub.it/ambient/.

[11] S. Osiński, and D. Weiss, “Carrot 2: Design of a Flexible and Efficient Web Information Retrieval Framework,” Advances in Web Intelligence, vol. 3528, pp. 439-444, 2005.

Ruilong Yang is a PhD student of College of Computer Science, Chongqing University, China. His research interests are in fields of Web Data Mining, Web Service, Mobile Computation and new Web development technology.

Qingsheng Zhu is a PhD. Professor and PhD supervisor, College of Computer Science, Chongqing University, Chongqing, China. He was a Visiting Scholar (1993-1994) at Dept. of Computer, Birkbeck College, University of London, UK. He was a Visiting Professor (2001-2002) at Dept. of Computer Science, University of Illinois at Chicago, USA. His research areas are Data Mining, Web Service and Image Processing.

Yunni Xia is a PhD. associated Professor of College of Computer Science, Chongqing University, China. His research areas are Web Service and Web intelligence.