Journal of Computational Information Systems
Available at http://www.JofCI.org

Shingles-based Structural Clustering of Web Documents

Tian XIA[1,2 †]

[1] *Key Laboratory of Data Engineering and Knowledge Engineering(Renmin University of China), MOE, Beijing 100872, China*
[2] *School of Information Resource Management, Renmin University of China. Beijing 100872, China*

Abstract

Web document structural clustering is a useful task for many web intelligent applications, however, processing based on the structure of web documents have not yet received strong attention. In this paper, we propose a shingles-based approach to clustering web documents by structure. Firstly, semi-structured web documents are converted into structured tree which composed of a set of limited nodes, and structural features are extracted by shingles. Secondly, we define document distance and structural distance matrix, and then structural similarity is calculated according to this matrix. Finally, we cluster the document structure based on modified k-means algorithm. Different from existing methods, we construct shingles not only including real vertical paths, but also virtual horizontal paths. Weight factors are also considered to optimize the algorithm. Experimental results show the effectiveness of the new shingles-based similarity measurement and the structural clustering. The proposed document similarity, as well as the structural shingles analysis, could be applied to other web-based research issues.

Keywords: Structural Clustering; Structural Similarity; Shingles

1. Introduction

Clustering web documents by structure can be a useful task for many web mining applications. For example, the detection of structural similarities among documents can help in solving the problem of recognizing different sources providing the same kind of information, or the structural analysis of a website. While the processing and management of web documents are popular research issues, operations based on the structure of web documents have not yet received strong attention, previous work about clustering has focused on document contents, and current structural similarity has focused on XML[1-3] Several researchers have exploited partial structure information such as hyper-links to improve content clustering results[4, 5], but its main purpose is still to achieve satisfied accuracy according to the content distribution and importance.

In this paper, we introduce a measure of web document structural clustering applies the shingles technique. Different from the simple path shingle method, we construct the shingles not only from real vertical path, but also consider the horizontal structure of document. Weight information such as node depth and type are also taken into account for better performance. Furthermore, we define the structural distance between any two web documents based on weighted shingles, and propose the shingles distance matrix for document similarity calculation.

The rest of this paper is organized as follows: In Section 2, we give a brief overview of shingles technique, and discuss our approach to shingles features representation and extraction, and define the shingles distance matrix and document similarity algorithm. In section 3, we establish a structural clustering algorithm based on document shingles similarity measure. Experimental results compared with

[†] Corresponding author.
Email addresses: xiat@ruc.edu.cn (Tian XIA)

simple path shingles method are presented in section 4. In section 5, we briefly review related work. Finally, we conclude and discuss future work in the last section.

2. Structural similarity measurement

Document structural similarity can be calculated by shingles, but common shingles method takes the whole document as the basic similarity unit, clustering speed will be very slow when applies the shingles directly [8]. In the following, we first give an overview of shingles technique, and propose our approach in detail.

2.1. Overview of shingles technique

Shingles has been introduced several years ago by Andrei Broder as a technique to compare two text documents for resemblance and containment [6]. In the shingles algorithm, resemblance is expressed as a set intersection problem. A contiguous subsequence of ω tokens contained in D is called a shingle. A shingle of length n is also known as a *n-gram*, particularly when the tokens are alphabet letters.

Once defined the shingles, the resemblance r and containment c of two documents D_i and D_j can be computed by the intersection and the union of the documents shingles:

$$r(D_i, D_j) = \frac{|S(D_i,\omega) \cap S(D_j,\omega)|}{|S(D_i,\omega) \cup S(D_j,\omega)|} \qquad c(D_i, D_j) = \frac{|S(D_i,\omega) \cap S(D_j,\omega)|}{|S(D_i,\omega)|} \tag{1}$$

Where ω is the shingle size and $S(D_i, \omega)$ is the set of shingles in D_i considering the shingle's size of ω. These two metrics define how similar and how contained document D_i is to document D_j , so, strong resemblance and strong containment can capture the informal notion of "roughly the same" and "roughly contained".

2.2. Pre-processing

In order to represent the structured information of web documents, we design a simple tree model. Unlike the HTML DOM tree, we only reserve the element nodes in this model, and add special properties to each node, text contents and comments are removed for space saving purpose. In our implementation, each node includes the following items:

N ={*label, weight, parent, children, <key1, value1>,<key2,value2>…*}

Where *label* is the tag name, and *weight* is a positive real number which indicates the importance of current node. Items of *parent* and *children* are pointers to maintain the relationship of the tree. Each pair of <*key, value*> represents a property of this node. Before document tree constructed, we also capitalize each token, and then use NekoHTML to fix up many common mistakes that human authors make in writing HTML documents such as unclosed tag scope, overlapping tag scopes, etc [7].

Once we have converted each semi-structured web document into a structured tree, as we have done above, the next step is to find a mechanism to generate a bag of shingles that represent the web document structure. We do this by creating shingles from real vertical path and virtual horizontal path that composed of tree nodes, assigning each shingle a weight, merging all shingles with the same tag sequences at different tree positions to a weighted shingle, and finally clustering the documents based on this shingles features.

2.3. Real vertical paths and virtual horizontal paths

Tag paths represent the natural basis for extracting structural features from web documents, D. Buttler uses path similarity to measure the similarity of paths between two different web documents [8]. A path is defined as a list of connected nodes starting at the root and terminating in a leaf node. In this paper, we call it a "*real vertical path*", because each pair of adjacent nodes has real parent-child relationships in the structured tree. Real vertical path captures the structural information of web document from top-bottom view, it has strong descriptive ability. However, real vertical path can not locate document block's horizontal position; it has no ability to distinguish different content blocks composed by same tags, which is very important for document template design. For example, in figure 1, there are three different html tree blocks marked by A, B and C, but they have the same two real vertical paths completely: "/TABLE/TR/TD/[TEXT]".

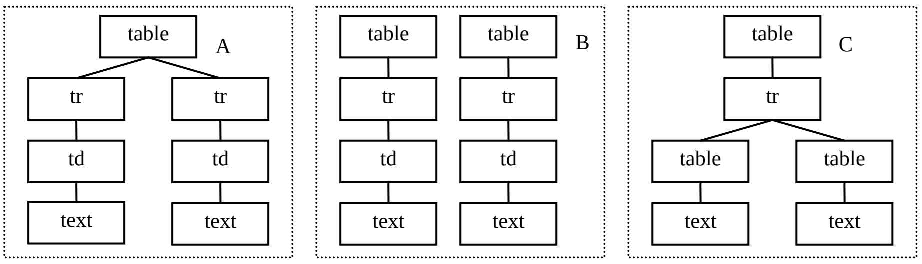

Fig. 1 Three different html blocks with the same real vertical paths

For solving the problem above, we propose a new type of path named "virtual horizontal path". Here, a virtual horizontal path is defined as a list of nodes which has the same neighbor relationships connected by the common parent node, and element order in the path is specified by the tag occurrence of the document. Because all the nodes of this type of path have no direct link, we call the path "virtual". Virtual horizontal path intents to describe the web document structure from every horizontal cross section, different content blocks of document have different virtual horizontal paths, though their real vertical paths may be the same. In figure 1, the virtual horizontal path of block A, B and C at the first level is "/TABLE", "/TABLE/TABLE" and "/TABLE" separately. At the second level, block A has one path: "/TR/TR", block B has two paths, both are "/TR", and block C has one: "/TR".

The above analysis shows that, combining real vertical path and virtual horizontal path together is a more reasonable approach to represent the web document structure.

2.4. Shingle-Features extracting and weight definition

Given all the real vertical paths and virtual horizontal paths of a document *D*, we can construct every *ω-shingling* as any partial path that includes *ω* contiguous nodes. For instance, the 3-Shingling of path "/TABLE/TR/TD/ [TEXT]" appeared in Figure 1 contains 2 shingles as follows:

{/TABLE/TR/TD, /TR/TD/[TEXT]}

Another simple method to generate shingles is taking each path of document *D* as a whole shingle. So the document tree illustrated in Figure 1(a) contains 5 different shingles:

{/TABLE/TR/TD/[TEXT], /TABLE, /TR/TR, /TD, /[TEXT]}

Different tag name and tag position occurred in web document have different contributions to describe document structure. In order to represent this feature, we assign a weight to every shingle. Weight can be assigned by following principles:

(1) Important nodes such as "<TABLE>", "<DIV>" should be assigned higher weights, while unimportant nodes such as "<BR>", "<FONT>" should be assigned lower weights;

(2) Nodes with more important attributes should be assigned higher weights;

(3) Nodes which have bigger node depth in document tree should be assigned lower weights.

Suppose shingle *s* is composed by *m* nodes in document *D*, i.e. $s=n_1,n_2, \ldots,n_m$. According to above principles, we design the weight-assignment function as follows:

$$w(s \mid D) = \sum_{i=1}^{m} \frac{\prod_{j=1}^{3} w_j(n_i)}{m} \qquad \text{Where } w_3(n_i) = \frac{\alpha}{Depth(n_i)+\alpha} \tag{2}$$

In the above formula, n_i is the *i-th* node of shingle *s*, w_1 and w_2 represent the weight of current node type and attributes respectively, by default, we assign 1 to w_1 and w_2, this means different type of nodes are equally important, and all attributes are ignored. Of course, these values can be changed by manual assignment or some other functions later, and tag style should also be considered in our future work. w_3 represents the weight of node depth, in the formula, α is an adjustable factor, the bigger value of α means the little influence among different depth of nodes. In our experiments, the default value of *α* is 10. Furthermore, there are many shingles with the same tag sequences, to simplify the calculation, we merge all these same labeled shingles to one shingle, and assign the sum of the weights to this merged shingle.

2.5 Shingles distance matrix and document similarity

According to the above analysis and shingles definition, each web document can be represented by a bag of weighted shingles. For measuring the correlation of documents, we define the shingles distance matrix, and then use this matrix to calculate document similarity for clustering.

Let D_i and D_j be two shingles set that represent two document *i* and *j*, the *distance* from document *i* to document *j*, denoted as $l(D_i, D_j)$, is the set difference between D_i and D_j as follows:

$$l(D_i, D_j) = \sum_{s \in \{D_j - D_i\}} w(s \mid D_j) + \sum_{s \in D_j \cap D_i} | w(s \mid D_j) - w(s \mid D_i) | \quad (3)$$

This definition of distance represents the minimal cost for document *i* to add the missing shingle features appeared in document *j*. It can be inferred from this definition that $l(D_i, D_j)$ does not equal to $l(D_j, D_i)$, the length of distance from a document to itself is zero, i.e. $l(D_i, D_i)=0$.

For the given document set *S*, we suppose its size is *n*, then the distance matrix of all the web documents in *S* can be expressed in a $n \times n$ matrix $M=(m_{ij})_{n\times n}$, called shingles distance matrix. This matrix *M* is a numerical format that converts the document structure information into shingles set difference, incorporating the two different shingle paths and its importance. Obviously, each element value m_{ij} in matrix is the distance $l(D_j, D_i)$.

In the shingles distance matrix *M*, the row vector that corresponds to each document *i* in *S* is in the form of

$$row_i = (m_{i,1}, m_{i,2}, \cdots, m_{i,n}), \quad i = 1, 2, \cdots, n$$

From the construction of matrix *M*, it is known that row_i represents shingles difference relationship of document *i* with all the documents in *S*, and element values in this row vector indicate the distance of this document to the specified document. For any two documents *i* and *j* in *S*, their similarity can be defined as

$$sim_{i,j} = \frac{(row_i, row_j)}{| row_i | \bullet | row_j |} \quad \text{Where } (row_i, row_j) = \sum_{k=1}^{n} (m_{ik} \times m_{jk}), \; | row_i | = \sqrt{\sum_{k=1}^{n} m_{ik}^2} \quad (4)$$

3. Web document structural clustering

With the structural similarity measurement (4) and the shingles distance matrix *M*, a structural web page clustering algorithm can be established. In our experiment, we clustering the documents based on modified k-means algorithm. Unlike standard k-means algorithm, we take the structural similarity measure instead of Euclidian Distance to adjust the centroid of each cluster. The algorithm are described as follows.

Input: a set of web documents $S=\{d_1, d_2, ..., d_n\}$, clustering threshold *T*. maximum number of clusters *K*.
Output: a set of clusters $C=\{c_i\}$
Algorithm:

Step 1: Convert every document into weighted shingles.

Step 2: Select the first document d_1 as the initial cluster c_1 and the centroid of this cluster, $c_1=\{d_1\}$, $centroid_1=d_1$

Step 3: For each document p_i in *S*, calculate the similarity between p_i and the *centroid* of each existing cluster c_j: $sim(p_i, centroid_j)$

Step 4: Let $sim(p_i, centroid_k) = \max_j (sim(p_i, centroid_j))$,

if $sim(p_i, centroid_k) > T$ or $| C | \geq K$, then add p_i to the cluster c_k and recalculate the $centroid_k$ of cluster c_k:

$$centroid_k = \frac{1}{| c_k |} \sum_{j \in c_k} row_j$$

Where $|c_k|$ is the number of documents in c_k. $|C|$ is the number of clusters in C. If p_i has already belonged to cluster c_i before adjustment, then remove p_i from c_i and recalculate the *centroid*$_i$ of cluster c_i too.

Otherwise, p_i itself initiates a new cluster and is the cenroid of this new cluster.

Step 5: Select another document p_i in S, go back to step 3.

Setp 6: Iterate from step 2 to step 5, until all cluster centroids no longer change or the number of iterations has achieved specified value.

Step 7: return clusters $C=\{c_i\}$

Jingyu Hou pointed out that the clustering threshold T in the algorithm should be chosen such that the documents are clustered into a reasonable number of clusters [4]. Furthermore, every document in S only belongs to one cluster, but the requirement of multiple clusters can be met easily. In our clustering algorithm, we add a parameter K to specify the number of clusters, and iterate the main steps for better clustering results. This can meet the fixed-size clustering or dynamic-size clustering requirements simultaneously.

4. Experimental Results

The algorithm presented in this paper has been implemented in java language, and executed on the Sun 1.6.0 JVM platform. The quality measure we use in the experiments is the F-measure, which combines the precision and recall ideas from information retrieval [9]. Given a predefined partitioning of the document collection into a set of classes, the precision and recall of a cluster j with respect to a predefined class i are defined as follows:

$$P(i,j) = \frac{N_{ij}}{N_i} \qquad R(i,j) = \frac{N_{ij}}{N_j} \tag{5}$$

Where N_{ij} is the number of members of class i appeared in cluster j; N_i is the number of members of class i, and N_j is the number of members of cluster j.

The F-measure of a class i with respect to cluster j is given by evaluating the relevance of the cluster:

$$F(i,j) = \frac{(\beta^2+1)\times P(i,j)\times R(i,j)}{\beta^2\times P(i,j)+R(i,j)} \quad 0<\beta\le 1 \tag{6}$$

In our experiments, we select β=1. Furthermore, for each cluster j, we select the maximal value of $F(i, j)$ as the F-measure of class i, marked as $F(i)$. Let $|class_i|$ represents the number of members in class i, the total F-measure can be computed by the weighted average values of each class:

$$F = \frac{\sum_i (|class_i| \times F(i))}{\sum_i |class_i|} \tag{7}$$

D. Buttler's experiments show that the simple path shingles method outperforms the existing metrics such as FFT (Flesca et al., 2002), Path, Tree Edit Distance and Weighted Tag, so, we compared our method with the simple path shingles method proposed by D. Buttler in the following experiments.

The first document collection consists of 600 HTML pages fetched from the News Website of Renmin University (http://news.ruc.edu.cn). All these pages are divided into 3 big classes: 65 special column pages, 200 list pages and 335 articles, and all 335 articles are divided into 2 small classes: 280 text-based articles and 55 image-based articles. The experimental results are shown in table 1.

Tab.1 The experimental results for single website.

Classes	Method	F(1)/%	F(2)/%	F(3)/%	F(4)/%	Total F-measure /%
3	Simple shingles	89.71	95.63	99.26	—	97.02
3	Our method	91.04	96.43	99.41	—	97.51
4	Simple shingles	89.71	95.63	95.87	82.05	94.27
4	Our method	91.04	96.43	96.98	86.49	95.19

The second document collection consists of 300 news HTML pages fetched from three portals: SOHU (http://www.sohu.com), SINA (http://www.sina.com.cn) and Yahoo! (http://cn.yahoo.com), all these pages are divided into three equal classes according to their websites, and can be clustered correctly with both algorithms. Furthermore, we add 51 column pages to the dataset, and divide all pages into two classes: column pages and news pages. The corresponded experimental results are shown in table 2.

Tab.2 The experimental results for multiple websites.

Classes	Method	F(1)/%	F(2)/%	Total F-measure /%
3	Simple shingles	55.15	92.79	86.74
3	Our method	60.78	93.75	88.96

The experimental results show that both algorithms are more effective. For the first dataset, articles structure is much different from special column pages and list pages, both algorithms can cluster all the articles correctly. Special column pages and list pages have some common blocks, and the F-measures for them are correspondingly lower than articles class. Furthermore, both text-based articles and image-based articles are articles, and they have little difference on inner structures except the number of IMG tags, so the F-measure of clustering results for class 4 is lower. For the second dataset, because different websites have different HTML templates, clustering the websites by news pages can achieve high accuracy, but the F-measure is lower when clustering the different kind of pages among multiple websites. Since horizontal structure and weight information are taken into account in our method, it achieves better accuracy than simple shingles method. It is obvious that structural clustering from the same website can achieve high accuracy, for different websites document clustering, multiple features should be taken into account, and there is still a lot of work to do in our future work.

5. Related work

There are many ways to clustering web documents, such as using link-content analysis [10], hyperlink analysis [11], or text content analysis [12]. Here, we present and discuss some representative work that has relative with the structural aspect of web documents.

Previous work for web structural clustering was almost focused on hyperlink analysis. The early representative work of hyperlink analysis can be found in [13, 14]. In paper [4], the authors proposed an approach to measure web page similarity, and take hyperlink transitivity and page importance from page structure into consideration. However, its main purpose was to clustering the page by link distance, page structural similarity factor was not taken into account.

The main step for clustering web documents by structure is document similarity calculation. Algorithms for structural similarity include tree edit distance [15], structural grapth similarity [16], Fourier transform technique [17, 18], and entropy-based algorithm [19]. Most of these algorithms have applied to XML documents, and some algorithms are more sophisticated and slowly. Buttler presented a brief survey of document structural similarity algorithm, compared their accuracy and performance, and proposed a new similarity metric based on simple path shingles, his research showed that path shingles may be the most effective and efficient mechanism for many structure processing applications [8], but the clustering speed would be very slow when applying document shingles directly. In our work, we propose two typical path shingles: real vertical path shingles and virtual horizontal path shingles, and weight are assigned to each shingles. Furthermore, we define the numeric document distance and structural shingles matrix, and then document similarity is calculated quickly based on the values in row vectors.

6. Conclusions and future work

This work studied methods to cluster web documents by structure, exploiting shingles technique and improve it. Firstly, structured tree of document is build from the HTML source code. Secondly, document structural features represented by real vertical paths and virtual horizontal paths are extracted from the tree and convert into weighted shingles. Thirdly, we define the shingles distances and construct structural distance matrix, then similarity between any documents is computed according to this matrix. Finally, we establish structural clustering algorithm based on k-means. Experimental results show that our approach performs better than the simple path shingles metric.

Several research directions to extend and improve this work in the future include:

(1) Dealing with sophisticated web pages from different websites which may adopt scripting technique for silently loading new data from the server, especially appeared in web 2.0 websites.

(2) Using and comparing different algorithms for structural clustering based on our shingles approach.

Acknowledgments

This work has been supported by the open project of Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education (Project Name: Study on the Knowledge platform construction on the Web). The helpful comments from anonymous reviewers are also gratefully acknowledged.

References

[1] Gianni Costa, Giuseppe Manco, Riccardo Ortale, Andrea Tagarelli. (2004): A Tree-based Approach to Clustering XML Documents by Structure. in Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, 137-148.

[2] T. Dalamagas, T. Cheng, K. J. Winkel, T. Sellis. (2006): A Methodology for Clustering XML Documents by Structure. Information Systems. 31(3):187-228.

[3] Massih R. Amini, Anastasios Tombros, Nicolas Usunier et.al. (2005): Learning to Summarise XML Documents Using Content and Structure. in Proceedings of the 14th ACM international conference on Information and knowledge management. Bremen, Germany. 297-298.

[4] Jingyu Hou, Yanchun Zhang. (2003): Utilizing Hyperlink Transitivity to Improve Web Page Clustering. In Proceedings of the 14th Australasian database conference. Adelaide, Australia. 49-57.

[5] Campos, R. and Dias, G. (2005): Automatic Hierarchical Clustering of Web Pages. In Proceedings of the ELECTRA Workshop associated to 28th Annual International ACM SIGIR Conference (SIGIR 20005). August 19, Salvador, Brazil. 83-85.

[6] Broder, A. Z. (1997): On the Resemblance and Containment of Documents. In Compression and Complexity of Sequences(SEQUENCES'97): 21-29.

[7] CyberNeko HTML Parser, http://sourceforge.net/projects/nekohtml.

[8] Buttler, D. (2004): A Short Survey of Document Structure Similarity Algorithm. The 5th International Conference on Internet Computing. Las Vegas, NV, United States.

[9] K. M. Hammouda, M. S. Kamel. (2002): Phrase-based Document Similarity Based on an Index Graph Model. in Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02).

[10] Marchiori, M. (1997): The quest for correct information on the web: Hyper Search Engines. In Proceedings of the 6th International World Wide Web Conference.

[11] Wang, Y. and Kitsuregawa, M. (2001): Use Link-based Clustering to Improve Web Search Results. In Proceedings of the Second International Conference on Web Information Systems Engineering (WISE 2001). Kyoto, Japan. 119-128.

[12] WEN, C. W., LIU, H., WEN, W. X. and ZHENG, J. (2001): A Distributed Hierarchical Clustering System for Web Mining. In Proceedings of the Second International Conference on Web-Age Information Management (WAIM 2001), Xi'an, China. 103-113.

[13] Kleinberg, J. (1998): Authoritative Sources in a Hyperlinked Environment. In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA).

[14] Dean, J. and Henzinger, M. (1999): Finding Related Pages in the World Wide Web. In Proceedings of the 8th International World Wide Web Conference. 389-401.

[15] A. Nierman and H. V. Jagadish. (2002): Evaluating Structural Similarity in XML Documents. In Proceedings of the 5th International Workshop on the Web and Databases (WebDB), Madison, Wisconsin. 61-66.

[16] M. Dehmer, F.E. Streib, A. Mehler and J. Kilian. (2006): Measuring the Structural Similarity of Web-based Documents: A Novel Approach. International Journal of Computational Intelligence Vol. 3 (1): 1-7

[17] S. Flesca, G. Manco, E. Masciari, L. Pontieri and A. Pugliese. (2007): Exploiting Structural Similarity for Effective Web Information Extraction. Data & Knowledge Engineering, v.60 n.1, 222-234.

[18] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. (2002): Detecting structural similarities between XML documents, Fifth International Workshop on the Web and Databases. Madison, Wisconish, United States.

[19] Helmer, S. (2007): Measuring the Structural Similarity of Semistructured Documents Using Entropy. In Proceedings of the 33th International Conference on Very Large Databases. Vienna, Austria. 1022-1032.