A Level-wise Clustering Algorithm on Structured Documents

Ching-Yao Wang, Ying-Chieh Lei, Pei-Chi Cheng and Shian-Shyong Tseng[*]
National Chiao-Tung University, Taiwan
{cywang, gis90529, gis91516, sstseng}@cis.nctu.edu.tw

Abstract

Document clustering is the process of applying clustering technique for document management [4][5]. Similar documents are grouped together so that both managing and searching the documents is efficient. However, since traditional document clustering algorithms do not take the structure information of documents into consideration, the clustering results can not reflect the characteristics of the documents fully. As the result, we represent each document as a tree structure and propose a *level-wise clustering algorithm* to solve this issue. The clustering process applies the level property of the tree and run level by level by the concept generalization operation. In order to store the clustering results and search interesting clusters efficiently, a multistage graph called *Level-wise Document Clustering Graph* (*LDC-Graph*) is proposed. Based on LDC-Graph, three search strategies are provided to meet the different requirements for uses. Finally, the experimental results show that the similarity search is efficient and the accuracy of the search is acceptable

Keywords: document clustering, structured document, clustering, tree structure

1. Introduction

Since more and more digital documents interchange on Internet, how to manage these documents becomes a very important issue. In recent years, many document clustering methods have been thus proposed to manage massive documents [9][14][18]. In general, these algorithms only represent each document by a flat feature vector consisting of significant keywords, and do not take the inherent structure behind the document into consideration. This way seems rather simple and efficient, but may cause the following two drawbacks:

(1). Inaccuracy: Traditional document clustering algorithms use a finite set of features to represent documents. However, it is difficult to select representative features [8].

(2). Inflexibility: When users are only interested in parts of a document, traditional document clustering algorithms can not return these ones, since they treat whole document as a unit.

In order to overcome the drawbacks, in this paper we will propose a document clustering algorithm by taking the structure information of documents into consideration. With the structure information, each document can be decomposed of several logical components and represented as a tree-like structure, where the upper component represents a higher concept that covering all the concepts beneath it. Figure 1 exemplifies a document made up of several components, such as title, abstract, chapters, sections, and paragraphs.

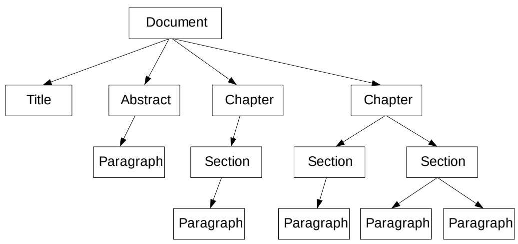

Figure 1: An example of structured document

We first represent each document as a tree structure of feature vector in XML (eXtend Markup Language) [22] instead of a flat feature vector. Then a novel algorithm called *level-wise clustering algorithm* is proposed to cluster all nodes in the document trees by a level-wise approach. The key idea is to subdivide the document trees into several clustering populations according to the number of levels in the tree structure. The clustering process will start from the bottom document level to the top document level with the same similarity measure. Moreover, for concept generalization, the

[*] Corresponding author

clustering information of lower document level will simultaneously reflect to the higher document level. To store the clustering results and search interesting clusters efficiently, a multistage graph called *Level-wise Document Clustering Graph* (*LDC-Graph*) is proposed. After clustering, three search strategies based on the multistage graph are proposed so that user can get not only general search results but also specific search results.

2. Background and Related Work

2.1 Document Clustering

Document clustering manages massive documents by grouping similar documents into the same cluster. It has been extensively used for efficiently finding the nearest neighbors of documents and browsing a collection of documents in many areas, such as text mining, information retrieval, etc. [2][13][10][15]. The most common steps of document clustering are shown in Figure 2.

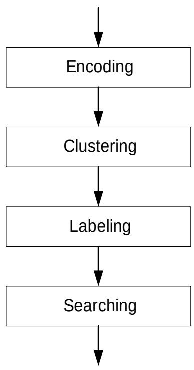

Figure 2: The flowchart of document clustering

In the encoding phase, the common approach is to represent each document by a finite set of keywords [12][16]. The selected keywords are treated as descriptive features and represented by a vector. This way is so-called *vector space model* method [3], and the popular weighting scheme for the vectors is based on the term frequency (TF) or the term frequency combined with the inverse document frequency (TF-IDF) [1][4]. A document can be thus represented as $d_{idf} = (tf_1 * idf_1, tf_2 * idf_2, ..., tf_n * idf_n)$, where tf_i is the frequency of the ***i***-th term in the document, and idf_i is the inverse document frequency of the ***i***-th term in the document and it can be calculated by $\log(n/df)$ where n is the number of documents and *df* is the number of documents that contains the term.

In the clustering phase, similar documents are then grouped together according to a similarity function and the *cosine function* is the most commonly used in the vector space model. It can by calculated by the following formula:

$$Similarity = \text{cosine}(d_1, d_2) = \frac{d_1 \bullet d_2}{|d_1||d_2|},$$

where d_1 and d_2 are the vectors of two documents, $\bullet$ is the vector dot product, and $|d_1|$ and $|d_2|$ are the lengths of the vector d_1 and d_2, respectively. If the cosine value is larger than the user-specified similarity threshold, two documents are considered as similar to each other.

In the labeling phase, the generated clusters are labeled according to a criterion function. The common labeling method is to treat the most frequent keywords or the cluster centers in the cluster as the label. The cluster center can be computed by averaging all data vectors in the cluster.

In the searching phase, according to a user-specified vector and a similarity threshold in the query, similarity search will find the interesting clusters by a similarity function. Moreover, the clustering performance is usually evaluated by comparing the searching results with the correct answers.

2.2 BIRCH

BIRCH (Balance Iterative Reducing and Clustering using Hierarchies) is a hierarchical clustering algorithm introduced in [20]. The authors employed the concepts of *Clustering Feature* and *CF tree* to implement the clustering. Clustering feature in BIRCH is a triple summarizing the information about a cluster. CF tree is a balance tree with two parameters, branching factor *B* and threshold *T*, to store the clustering features. Each non-leaf node in CF tree will contain at most *B* entries recording the cluster feature of subclusters and pointing to these subclusters. When new data objects are inserted, the closest cluster is searched from the root of CF tree descending to the leaf nodes by the similarity function and threshold *T*.

3. Level-wise Clustering Algorithm

In order to take structure information of documents into consideration, each document can be decomposed of several logical components and represented by a depth-fixed tree structure called a *document tree* according to a prior-known document structure. Based on the representation, we then propose a novel algorithm called *level-wise clustering algorithm* to cluster the nodes in the document trees by a level-wise approach. The key idea is to subdivide the document trees into several clustering populations according to the number of levels in the tree structure. The clustering process will start from the bottom document level to the top document level with the same similarity measure. Moreover, for concept generalization, the clustering information of lower document level will simultaneously reflect to the higher document level by *roll-up* operation. After level-wise clustering, each level will have its own clusters and the results will be stored in a *Level-wise Document Clustering Graph* (*LDC-Graph*). For detail, we will follow the sequential steps, *document encoding* phase, *clustering* phase and *concept generalization* phase, to describe the proposed level-wise clustering algorithm.

Algorithm 1: Level-wise Clustering Algorithm

Denotation:
D: is the depth of the document tree.
L_0~L_{D-1}: denote the document levels of document tree descending from the top level of document tree.
S_0~S_{D-1}: denote the stages of LDC-Graph
Input: N document trees with the same depth D, similarity threshold T_0~T_{D-1} for clustering the document nodes in the document level L_0~L_{D-1} respectively.
Output: LDC-Graph which holds the clustering results of every document level.
Step 1: Group the document nodes in the document trees with the same document levels.
Step 2: For i=L_{D-1}down to L_0 do
 Step 2.1: Run ***single-level clustering algorithm*** for document nodes in document level i with the threshold T_i.
 Step 2.2: Store the clustering result in the stage S_i of LDC-Graph.
 Step 2.3: If i<>L_0 then
  Run roll-up operation to set the value of document nodes in the document level L_{i-1}

Algorithm 2: Single-level Clustering Algorithm

Input: N document nodes in the same document level, similarity threshold T for clustering.
Output: The set of LDC-Nodes for representing the clusters of N document nodes.
Step 1: Extract a document node from N document nodes and place it into a cluster of its own. The cluster is represented by the LDC-Node.
Step 2: For each document node, find the most similar cluster by the similarity measure.
Step 3: If the similarity measure> T then
  Place the document node into the LDC-Node of most similar cluster and the LDC-Node is updated.
 Else
  Place the document node into a LDC-Node of its own.
Step 4: Return the set of the LDC-Nodes.

3.1 Document Encoding Phase: Document Tree and Similarity Measure

All documents are represented as *document trees* for document representation. A document tree is a tree structure where the depth of the tree is the same. Each node in the document tree is called *document node* which contains a vector consisting of the features of its corresponding component in the document. The level where a document node belongs is called *document level*. The document levels are labeled as L_0, L_1, …, L_{D-1} from the top level to bottom level, where D is the depth of a tree and the node in L_{i-1} is the parent node of L_i. Notice that in a document tree only the value of vectors of leaf nodes need to be assigned values by feature extracting since the values of the vectors in the internal nodes can be generated from the sub-tree it holds. The detail will be described in the concept generalization phase.

Example 1: Given a book shown in the left part of Figure 3, we take *TF-IDF* as weighting schema for feature extracting. The corresponding document tree is shown in the right part of Figure 3.

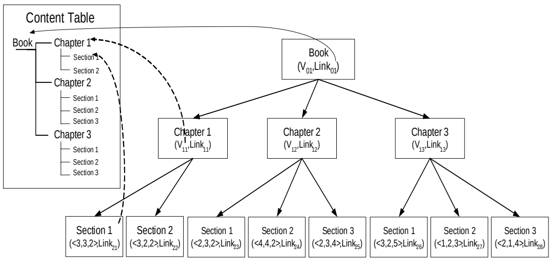

Figure 3: An example of a document tree

■

For clustering purpose, the cosine function, the most common similarity measure for document clustering [14][19], can be used to measure the similarity between two document nodes is defined as follow:

$$Similarity = \text{cosine}(V_A, V_B) = \frac{V_A \bullet V_B}{|V_A||V_B|},$$

where V_A and V_B are the vectors of document nodes A and B, respectively. The larger the value is the more similar two vectors are.

3.2 Clustering Phase: Level-wise Document Clustering Graph (LDC-Graph)

The clustering process will start from the bottom document level to the top document level with the same similarity measure. After level-wise clustering, each level will have its own clusters and the results will be stored in a *Level-wise Document Clustering Graph* (LDC-Graph).

Definition 1: Level-wise Document Clustering Graph (LDC-Graph)

LDC-Graph is a multistage graph comprising of several stages. The vertex represents a cluster, denoted as an *LDC-Node* = (*CF*, *DDL*), where *CF* (*Cluster Feature*) is used to store the summarized information of a cluster and *DDL* (*Drill-Down List*) is a list containing several entries. Each of which is represented as the form (CF_i, $Pointer_i$), where $Pointer_i$ is the *i*-th pointer connecting to the *i*-th related *LDC-Node* and CF_i is the *CF* of the subcluster connected by this pointer. ■

Definition 2: Cluster Feature (CF)

Cluster Feature (*CF*) of a cluster is defined as a triple: $CF = (N, \overrightarrow{VS}, CS)$, where N is the number of nodes in the cluster, $\overrightarrow{VS}$ is the sum of feature vectors for the N nodes, i.e. $\sum_{i=1}^{N} \vec{V}_i$, and CS is the cluster center or the average of the vector sum, i.e. $|\sum_{i=1}^{N} \vec{V}_i / N| = |\overrightarrow{VS} / N|$. When combining two clusters $CF_1 = (N_1, \overrightarrow{VS}_1, CS_1)$ and $CF_2 = (N_2, \overrightarrow{VS}_2, CS_2)$ into one new cluster, the new cluster feature CF_{new} can be calculated by $(N_1+N_2, \overrightarrow{VS}_1+\overrightarrow{VS}_2, |(\overrightarrow{VS}_1+\overrightarrow{VS}_2)/(N_1+N_2)|)$. ■

Example 2: Assume there are two document trees DT_1 and DT_2. After level-wise clustering, the results are shown in Figure 4(a). Then, the corresponding LDC-Graph is shown in Figure 4(b). The *DDL* of *LDC-node* C_{01} will contain three entries which point to the LDC-node C_{11}, C_{12} and C_{13}, respectively.

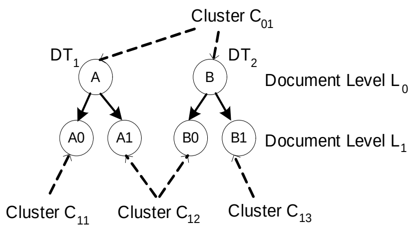

Figure 4(a): The clusters for two document trees DT_1 and DT_2.

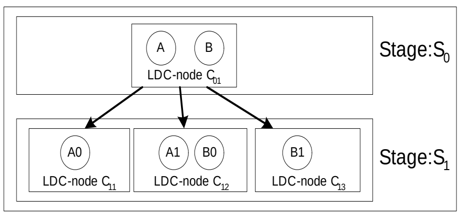

Figure 4(b): The corresponding *LDC-Graph* for Figure 4(a)

■

Example 3: Assume the cluster C is represented by the LDC-Node $N_C = (CF_C, DLL_C)$, where CF_C = (4, <8, 8, 16>, 4.899) and DLL_C = <(CF_1, $Pointer_1$), (CF_2, $Pointer_2$)>. When a new document node A with the vector V = <7, 2, 4> is inserted to the cluster C and its child nodes are belonging to the clusters 3 and 4 respectively, then the updated CF_C = (5, <15, 10, 20>, 5.385) and DLL_C = <(CF_1, $Pointer_1$), (CF_2, $Pointer_2$), (CF_3, $Pointer_3$), (CF_4, $Pointer_4$)>.

3.3 Concept Generalization Phase: Roll-up Operation

The concept generalization phase is used to generate the values of document nodes in the upper document level from the clustering results of document nodes in the lower document level. It will make the value of document nodes in the upper level more objective and representative by generalizing the detailed information in the lower level. Therefore, we define a *roll-up* operation for the non-leaf document nodes by averaging the cluster centers of the clusters which the lower document nodes belong to.

Example 4: Assume a document node *A* contains three child nodes A_1, A_2 and A_3, where A_1 and A_2 belong to cluster C_A and A_3 belongs to cluster C_B. If the cluster center of C_A is <3, 3, 2> and the cluster center of C_B is <3, 2, 4>, then after running roll-up operation the vector of the document node *A* will be: Average (<3, 3, 2>, <3, 3, 2>, <3, 2, 4>) = <3, 8/3, 8/3> ■

4. Similarity Search by LDC-Graph

Similarity search for document clustering is to find the interesting clusters for fulfilling user requirements. An interesting cluster is defined as a cluster which has higher similarity value than the user-specified threshold in the query. By the LDC-Graph, the similarity search opposite to the clustering process starts from the top stage (top document level) to the bottom stage (bottom document level). Since the clusters in the upper stage contain more general information than the clusters in the lower stage, the search from the top stage finds the general clustering result first and gets the specific clustering result when descending to the lower stage. The key operation for descending search is called drill-down operation. That is, the drill-down operation can return a set of LDC-Nodes in the next lower stage which are pointed by the present DDL. In the following, we propose three search strategies including *single stage search*, *top-down search* and *heuristic search*.

4.1 Single Stage Search Strategy

The single stage search strategy is used to find interesting clusters in a specific level. The algorithm of single stage search strategy is described as follows.

Algorithm 3: Similarity Search Algorithm for Single Stage of the LDC-Graph

Denotation:
ClusterSet: a set of LDC-Nodes.
Input: The query vector Q whose dimension is the same as the vector of each document node, the desired destination stage S_{DES} and search threshold S.
Output: The set of similar clusters.

Step 1: ClusterSet= ϕ

Step 2: For each LDC-Node ***N*** in the stage S_{DES} of an LDC-Graph.

Step 2.1: Compute the similarity LDC-Node ***N*** with query Q.

Step 2.2: If the similarity $\geq$ S then
ClusterSet=ClusterSet $\cup$ ***N***

Step 3: Return ClusterSet.

4.2 Top-down Search Strategy

The top-down search strategy is used to find interesting clusters by the drill-down operation. If the cluster is considered as similar one with the query, the drill-down operation will be executed to get the specific clusters of the next lower stage. With executing the drill-down operation repeatedly, users can get the similar clusters in the specified stage they want. The algorithm of top-down search strategy is described as follows.

Algorithm 4: Top-down Search Strategy

Denotation:
D: is the number of the stages in an LDC-Graph.
$S_0 \sim S_{D-1}$: denote the stages of an LDC-Graph from the top stage to the lowest stage.
ResultSet, DataSet: the sets of LDC-Nodes.
Input: The query vector Q whose dimension is the same as the vector of each document node, search threshold S and the destination stage S_{DES} where $S_0 \leq S_{DES} \leq S_{D-1}$.
Output: The set of similar clusters represented by LDC-Nodes
Step 1: Let DataSet be the set of LDC-Nodes in the stage S_0.
Step 2: ResultSet= ϕ.
For each LDC-Node ***N*** $\in$ DataSet,
If the similarity measure with Q $\geq$ S then ResultSet=ResultSet $\cup$ ***N***.
Step 3: If the stage of the node in ResultSet< S_{DES} then
DataSet= ϕ.
For each LDC-Node ***N*** $\in$ ResultSet
DataSet=DataSet $\cup$ LDC-Nodes returned by drill-down operation.
Go to Step 2.
Step 4: Return ResultSet.

4.3 Heuristic Search Strategy

Each cluster returned by the top-down search strategy belonged to some user-specified stage of the LDC-Graph. However, if the clusters in the higher stage are similar enough to the query, the clusters may be the desirable ones. It is thus not necessary to execute the drill-down operation. Based on this idea, we define a *full similarity* measure to evaluate the degree and propose a corresponding heuristic search strategy. Figure 5 illustrates the concept of full similarity.

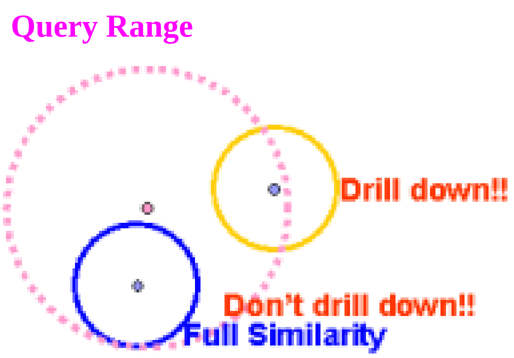

Figure 5: The concept of full similarity

Definition 3: Full Similarity

Assume that the similarity threshold for clustering is *T* and the similarity threshold for searching in the query is *S*, where *S* < *T*. Since similarity function is cosine function, the threshold can be represented as the form of the angle. The angle of *T* is denoted as $\theta_T = \cos^{-1} T$ and the angle of *S* is denoted as $\theta_S = \cos^{-1} S$. When the angle between the input and the cluster is lower than $\theta_S - \theta_T$, we say the cluster is full similar to the query. The full similarity can be formally defined by the following formula.

$$\text{Full Similarity} > Cos(\theta_S - \theta_T)$$
$$= Cos\theta_S Cos\theta_T + Sin\theta_S Sin\theta_T$$
$$= S * T + \left(\sqrt{1-S^2}\right)\left(\sqrt{1-T^2}\right)$$

■

The algorithm of heuristic search strategy is described as follows.

Algorithm 5: Heuristic Search Strategy

Denotation:
D: is the number of the stage in an LDC-Graph.
$S_0 \sim S_{D-1}$: denotes the stage of an LDC-Graph from the top stage to the lowest stage.
ResultSet, DataSet, FullSimilaritySet: the sets of LDC-Nodes.
Input: The query vector Q whose dimension is the same as the vector of each document node, search threshold S and the destination stage S_{DES} where $S_0 \leq S_{DES} \leq S_{D-1}$.
Output: The set of similar clusters represented by LDC-Nodes
Step 1: Let DataSet be the set of LDC-Nodes in the stage S_0 and FullSimilaritySet= ϕ.
Step 2: ResultSet= ϕ.
For each LDC-Node ***N*** $\in$ DataSet,
If N is full similar with Q then
FullSimilaritySet=FullSimilaritySet $\cup$ ***N***.
Else if the similarity measure with Q $\geq$ S then
ResultSet=ResultSet $\cup$ ***N***.
Step 3: If the stage of the node in ResultSet< S_{DES} then
DataSet= ϕ.
For each LDC-Node ***N*** $\in$ ResultSet
DataSet=DataSet $\cup$ LDC-Nodes returned by drill-down operation.
Go to Step 2.
Step 4: Return ResultSet $\cup$ FullSimilaritySet.

5. Discussion

As mentioned above, there are four salient features in our proposed level-wise clustering algorithm:

(1). Complete: Each structured document is represented as tree structure, so the inherent structure can be hold and the characteristic of structured document can be retained. This way may reduce the loss of information and since the document content can be represented more completely in a limited number of features.

(2). Representative: All features which used to represent the document node are generated from the base feature by concept generalization. The concept generalization phase makes document representation more objective and representative, and based on this representation manner, the clustering result is also more representative than other document clustering algorithms.

(3). Flexible: The level-wise clustering algorithm executes clustering based on the tree structure of each document. Each node of the tree stores the features extracted from one part of the document. Based on the inputs and clustering structure, we can get more flexible application than traditional algorithm. For example, we can find the documents with the similar hierarchical structure.

(4). Efficient: Since the LDC-Graph structure can effectively store the information of clusters, we can enhance the performance either search or clustering.

6. Experiments

All experiments are run on AMD Athlon 1.13GHz processor with 512MB DDR RAM. All programs are implemented in Borland C++ Builder 6 under Windows XP operating system.

6.1 Synthetic Data Generation

We use synthetic data generated by a synthetic system for evaluating the performance of our proposed algorithms. The synthetic generator controlled by the following four parameters is developed: the dimension of the vector of each document node, the depth of the document tree, the upper bound and lower bound branching factor for each document node, and the number of the document trees. The value of each entry in the vector is then randomly assigned in the range of [0, 1]. For reality, two transformation functions, $f(x)=\sqrt{1-(1-x)^2}$ and $f(x)=1-\sqrt{1-x^2}$, are used to amplifies and diminishes the assigned value. Moreover, an additional parameter called vector tendency need to be given for deciding the number of the entries in the vector should be amplified even if the entries are selected randomly. Other entries without amplifying are diminished by the diminution function.

6.2 Experimental Design

To evaluate the performance, we will compare the clustering quality and the searching time of a traditional document clustering algorithm (i.e. single level clustering algorithm) with our proposed level-wise clustering algorithm associated with top-down search strategy. In the traditional document clustering algorithm, each leaf node of document trees is considered as the input, and the clustering result is required without any concept generalization. The cluster quality can be evaluated by the *F-measure* [11] and can be calculated by the following formula:

$$F=\frac{2*P*R}{P+R},$$

where P and R are precision and recall, respectively. The range of F-measure is [0,1]. The higher the F-measure is the better the clustering result is.

6.3 Experimental Results

By synthetic data generator, 500 document trees are generated. The related parameters is that the dimension of the vector is 15, the depth of the document tree is 3, the range of the branching factor for each document node is [5, 10], and the vector tendency is 3. The clustering thresholds for level-wise clustering algorithm and the traditional document clustering algorithm are both set by 0.92. After clustering, there are 101, 104 and 2529 clusters generated from 500, 3664 and 27456 document nodes in the document level L_0, L_1 and L_2, respectively. Then, 30 queries generated randomly are used to compare the performance of two clustering algorithms. Figures 6 and 7 show the F-measure and the execution time for each query with the search threshold is set by 0.85.

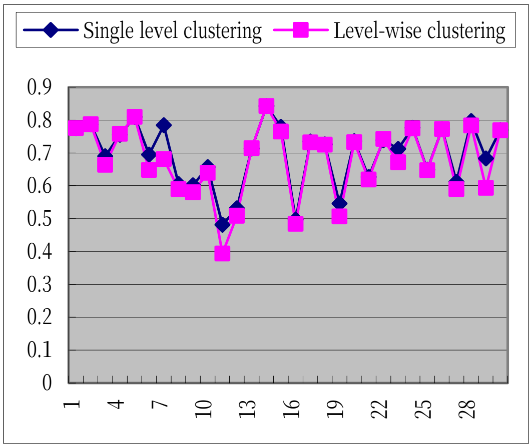

Figure 6: The F-measure of each query

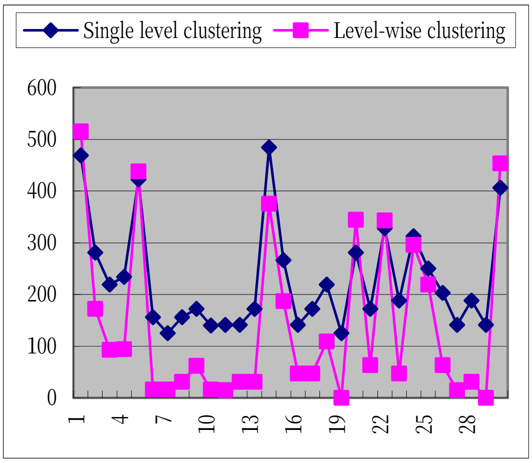

Figure 7: The executing time when similarity search

Since the concept generalization in the level-wise clustering algorithm will results in little information loss, the similarity search by drill-down operation in the LDC-Graph will decrease the accuracy. However, as shown in Figure 6, the differences of the F-measures are small in most cases. Moreover, for most cases illustrated in Figure 7, the searching time of level-wise clustering algorithm is far less than the ones of the traditional document clustering algorithm.

7. Concluding Remarks

In this paper, a level-wise clustering algorithm on structured documents has been proposed. The level-wise clustering algorithm represents each document as a tree structure and clusters the nodes of the trees according to the level of the tree. Besides, a multistage graph and cluster features are used to store the clustering results. Finally, three search strategies are proposed to utilize the multistage graph to get the similarity search efficiently. Our experimental results show that the level-wise clustering algorithm speeds up the searching time of each query without losing much information. Moreover, with three search strategies, users can not only get the general search results but also get the specific search results. In the future, experiments with real data will be implemented to analyze the performance and check if the proposed algorithm can really meet the needs of different users.

8. Acknowledgment

This research was supported by MOE Program for Promoting Academic Excellence of Universities under the grant number 89-E-FA04-1-4 and National Science Council of the Republic of China under Grand No. 91-2213-E-009-007-.

Reference

[1]. H, Avancini, A, Lavelli, B, Magnini, F, Sebastiani, and R, Zanoli, "Expanding domain-specific lexicons by term categorization," Proc. of ACM Symposium on Applied Computing, pp. 793 -797, 2003.

[2]. C. Buckley and A. F. Lewit, "Optimizations of inverted vector searches," SIGIR '85, pp. 97-110, 1985.

[3]. D. R. Cutting, D. R. Karger, J. O. Predersen, J. W. Tukey, "Scatter/Gather: A cluster-based approach to browsing large document collections," Proc. of the Fifteenth Interntional Conference on Research and Development in Information Retrieval, 318-329, 1992.

[4]. F. Debole and F. Sebastiani, "Supervised term weighting for automated text categorization," Proc. of ACM Symposium on Applied Computing, pp. 784 -788, 2003.

[5]. R. Freeman and H. Yin, "Self-organising maps for tree view based hierarchical document clustering," Proc. Int. Joint Conf. Neural Networks, vol. 2, pp. 1906-1911, 2002.

[6]. W. Fu, B. Wu, Q. He, Z. Shi, Info-tech and Info-net, "Text document clustering and the space of concept on text document automatically generated," Proc. ICII 2001 - Beijing. 2001 Int. Conf. on, Vol. 3, pp.107 -112, 2001.

[7]. S. Iiritano, M. Ruffolo, "Managing the knowledge contained in electronic documents: a clustering method for text mining," Database and Expert Systems Applications, Proc. 12th Int. Workshop on, pp. 454 -458, 2001.

[8]. G. Karypis and E. H. Sam, "Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval ," Proc. of the ninth international conference on Information and knowledge management, pp. 12 -19, 2000.

[9]. R. Kondadadi, R. Kozma, "A modified fuzzy ART for soft document clustering," Neural Networks, Proc. of the 2002 Int. Joint Conf. on, Vo. 3 , pp. 2545 -2549, 2002.

[10]. G. Kowalski, Information Retrieval Systems-Theory and Implementation, Kluwer Academic Publishers, 1997.

[11]. B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," Proc. of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 16-22, 1999.

[12]. Y.K. Lee, S.J. Yoo, K. Yoon, B. Berra, "Index structures for structured documents," Proc. Digital Library, pp. 91-99, 1996.

[13]. X. Long and T. Suel, "Optimized query execution in large search engines with global page ordering," Proc. of the 29th VLDB Conference, 2003

[14]. F. Sebastiani, "Machine learning in automated text categorization," ACM Computing Surveys, Vol. 34, No. 1, pp. 1-47, 2002.

[15]. S. Shankar and G. Karyp, "A feature weight adjustment algorithm for

document categorization," Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[16]. D.W. Shin, H.C. Jane, H.L. Jin, "BUS: An effective indexing and retrieval scheme in structured documents," Proc. of Digital Libraries, pp. 235-243, 1998.

[17]. M. Steinbach, G. Karypis and V. Kumar, "A comparison of document clustering techniques," Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[18]. P. Willett., "Document clustering using an inverted file approach," Journal of Information Science, pp. 2:223-231, 1990.

[19]. W.C. Wong and A. Fu, "Incremental document clustering for web page classification," IEEE Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges (IS2000), 2000.

[20]. T. Zhang, R. Ramakrishnan, and M. Livny., "BIRCH: an efficient data clustering method for very large databases," Proc. ACM-SIGMOD Int. Conf. Management of Data, pp. 103-114, 1996.

[21]. Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," Techical Report #02-022.

[22]. http://www.w3.org/XML/