Web Image Semantic Clustering

Zhiguo Gong, Leong Hou U, and Chan Wa Cheang

Faculty of Science and Technology, University of Macau, Macao, PRC
{zggong, ma36575, ma36600}@umac.mo

Abstract. This paper provides a novel Web image clustering methodology based on their associated texts. In our approach, the semantics of Web images are firstly represented into vectors of term-weight pairs. In order to correctly correlate terms to a Web image, the associated text of the Web image is partitioned into semantic blocks according to the semantic structure of the text with respect to the Web images. The weight of a term in the vector of an embedded Web image is calculated with respect to both its local occurrence in semantic blocks and the distances of the blocks to the image. With this method, 'Web image clustering' is transformed into 'term vector clustering'. And a feature based solution is employed in our solution. To reach this objective, we define the associate relations between two terms based on their co-occurrence in the associated text of the Web images. Thus, a term semantic network (*TSN*) is constructed with terms as the nodes and associate relationships as the edges. To cluster terms in *TSN*, *CHAMELEON* algorithm is utilized. In order to determine the significances of terms in each cluster, *HITS* algorithm is applied. Finally, web images are assigned to different clusters based on the similarity between image term vectors and the term vector of the clusters.

1 Introduction

Currently, Web images have been becoming one of the most important information types on the Web. Thus, how to effectively gather, manage and reuse this valuable resource are among the most attractive research topics in the area of Web information retrievals. However, many differences exist in comparison with traditional images in their creation purposes, amount, and semantic coherence. Firstly, Web images are generally created or selected by Web authors as visually supports for their Web page presentations, and no standard exists for the image size and visual qualities. Some images are highly qualified and much valuable for reuse while many others may not have any reuse values. Secondly, the amount of Web images is huge, therefore, manually semantic annotations for Web images, as for many traditional image management systems, is too hard or even impossible. Thirdly, the traditional image management systems are usually dedicated for specific domains which have limited scopes of users and image topics. And domain-based image processing techniques can be exploited for automatic feature extracting and indexing of the images. Web images, however, are produced by authors from different domains. And domain-independent algorithms for feature extracting and indexing of Web images can only be limited to some special features, such as color histogram or textures. As a result, it

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3761, pp. 1416 – 1431, 2005.

is hard to couple such lower features with higher semantic descriptions of the Web images. In order to overcome such shortages caused by the heterogeneous and semantic diverse intrinsic of Web images, an effective and efficient algorithm for Web image automatic clustering or classification is definitely desirable.

In [2], we have addressed our techniques on Web image indexing and searching. The semantic extraction of Web images in [2] is based on the HTML structure (DOM tree), the user feedback and Web page segmentations. In this paper, we provide our categorization method of web images based on the term clustering techniques.

In general, topic directory system suffers from lower performance of manual classification of newly collected documents. For example, Yahoo, the largest directory system on the Internet, has more than 1.5 million links in its topic hierarchy and needs as more as 100 editors in its classifying work of the Web documents. Up to now, most commercial categorization work of directory systems is performed manually. Therefore, to find an effectively automatic solution for Web document classification is much desirable and draws many attentions from both academic and industrial communities.

There are two ways for automatic document categorization. One is called *supervised learning techniques*--classification. Such kind of techniques consists of two steps. First, categories are predefined and described by experts. Then, all images are assigned to corresponding categories based on automatic similarity measurements between Web images' subject and the predefined category descriptions. The obvious weakness of such classification methods lies in that large percentage of the heterogeneous Web documents may not be fit to any given predefined category. In classification models, a training set should be manually defined at first, and the size of it may directly affect the precision of the classification result. So it is not suitable for huge amount of Web documents. The other method for categorization is called *unsupervised learning techniques* – Clustering. The main difference between the two is that clustering does not need to predefine the category beforehand, thus reduce manual labors dramatically. With such methods, subject experts can only need to define the label for categorization.

In this paper, we provide a solution for Web image clustering based on the associated text. The semantics of Web images are extracted by partitioning the textual parts of the Web pages with respect to the images. Then, Web images are represented into vectors of term-weight pairs. We suppose that each cluster of Web images can be described with a feature set of term-weight pairs too. The principle behind our approach is based on the observation that co--occurrence words or terms are often closely related to each other in describing the same topics for the Web images. To formalize our discussions, for any term pair t_1 and t_2, we define term relations with $Sup(t_1, t_2)$ (*Support*) and $Conf(t_1, t_2)$ (*Confidence*) which are extracted from our Web document/image database [2]. $Sup(t_1, t_2)$ indicates the absolute co-occurrence of term t_1 and t_2 in the collection, and $Conf(t_1, t_2)$ reflects their co-occurrence relative to t_1's appearances in the collection. Then, a term semantic network (*TSN*) is created with the terms as the nodes and $Sup(t_i, t_j)*Conf(t_i, t_j)$ as the directed edges(t_i, t_j). With this definition for *TSN*, it is obvious that the more value of edges(t_i, t_j), the more possible for t_i and t_j in the same cluster. In our approach, we suppose that terms with strong edges connected in *TSN* are likely to be used together in describing the same topics. In order to cluster Web image topics, we then partition *TSN* into several subgraphs

using algorithm *CHAMELEON* [1]. And we consider each subgraph of *TSN* as one feature cluster for one subject. As a matter of the fact, terms in each cluster are weighed differently in support the cluster topic. For example, the more edges linked to term *t*, the more significance of term *t* as one of the clustering features. We assign weights to terms by using algorithm *HITS* [21]. Finally, we assign Web images to corresponding clusters by the similarity between term vectors of the images and the feature vectors of the clustering. Our experimental results show that retrieval precisions can be improved in comparison with the methodology without using *HITS* or K-Mean partitioning approach.

The reminder of the paper is organized as follows: Section 2 provides a review on related works. Section 3 gives Web image representations. Section 4 addresses our techniques in constructing the term semantic network (*TSN*). And we provide our approaches for Web image clustering in section 5. In section 6, we show our experimental result. Finally in section 7, we conclude this paper.

2 Related Work

In this section, we introduce some past works on general clustering methodologies, as well as on document clustering.

2.1 Works on Fundamental Clustering Techniques

Clustering, which is the process of grouping the data into classes or clusters so that objects within a cluster have high similarity in comparison to one another, is one kind of method in data mining technology. The idea of clustering is that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized. Typical pattern clustering activity involves the following steps [5]:

- pattern representation (optionally including feature extraction and/or selection),
- definition of a pattern proximity measure appropriate to the data domain,
- clustering or grouping,

Feature selection is the process of identifying the most effective subset of the original features to use in clustering. And *Feature extraction* is the use of one or more transformations of the input features to produce new salient features. Either or both of these techniques can be used to obtain an appropriate set of features to use in clustering. In this paper, we consider all terms, which appear in the associated text of the Web image, as the semantic features of the image.

Pattern proximity is usually measured by a distance function defined on pairs of patterns. A variety of distance measures can be found in use in the different communities [5][6][7]. And a simple distance measure like Euclidean distance can always be effective to reflect dissimilarity between two patterns.

The *grouping* step can be performed in a number of ways. The output clustering (or clusterings) can be hard (a partition of the data into groups) or fuzzy (where each pattern has a variable degree of membership in each of the output clusters). Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters by similarity. In general, a partitional clustering algorithm is trying to find one partition that optimizes (usually locally) a clustering criterion.

Actually, based on these three fundamental steps, clustering can be implemented in different models, such as Single Link, Complete Link, Minimal spanning tree, and K-means. In the following, we will introduce some typical methods.

2.1.1 Single Link

Single Link is a simple one of the Hierarchical clustering algorithms. In the single-link method, the distance between two clusters is the *minimum* of the distances between all pairs of patterns drawn from the two clusters respectively (one pattern from the first cluster, the other from the second). The single-link algorithm suffers from a chaining effect [8]. It has a tendency to produce clusters that are straggly or elongated.

2.1.2 Complete Link

In the complete-link algorithm, the distance between two clusters is the *maximum* of all pairwise distances between patterns in the two clusters. And two clusters are merged to form a large cluster if the distance below some predefined threshold. The complete-link algorithm often produces tightly bound or compact cluster [9].

2.1.3 Minimal Spanning Tree (MST)

Most clustering models make all object link together such that it looks like a graph. So graph-theories can be used to solve the clustering problem. One typical technique in graph-theoretic is called Minimal spanning tree (MST). To cluster the nodes in the graph, MST is firstly obtained from the original graph, then, edges with the largest lengths are deleted recursively until ideal clusters are reached. The problem of MST is that the clusters may be quite unbalanced since the algorithm always chose the most weakness edge to cut. It often finds the large cluster group only.

2.1.4 K-Means

K-means is one kind of important partitional clustering algorithms which obtain a single partition of the data instead of a clustering structure. K-means uses the following step to get the cluster:

I. Choose k cluster centers to coincide with k randomly-chosen patterns or k randomly defined points.
II. Assign each pattern (object) to the closest cluster center
III. Re-compute the cluster centers using the new cluster membership
IV. If a convergence criterion is not met, go to step II. Typical convergence criteria are: no (or minimal) reassignment of patterns to new cluster centers, or minimal decrease in squared error.

The weakness of K-means is that it just cut the cluster into simple shape. It is not suitable for many situations.

2.1.5 CHAMELEON

CHAMELEON belongs to the kinds of hierarchical clustering algorithm. It modeling data items as graph and its sparse graph representation of the data items is based on the commonly used *k*-nearest neighbor graph approach. (Figure 1)

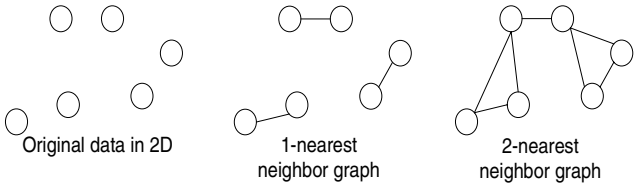

Fig. 1. *k*-nearest graphs from an original data in 2D

It measures the similarity of two clusters based on a dynamic model. In the clustering process, two clusters are merged only if the inter-connectivity and closeness (proximity) between two clusters are high enough relatively to the internal ones of these two individual clusters. The methodology of dynamic cluster modeling used in *CHAMELEON* is applicable to all types of data as long as a similarity metric can be defined. *CHAMELEON* is a successful algorithm to overcome many limitations of the existing hierarchical schemes [14][15][16].

On the other hand, *CHAMELEON* can also be carried out reversely to partition a data set or graph. In fact, *CHAMELEON* utilizes multilevel graph partitioning algorithms to find the initial sub-clusters. In particular, it uses the graph partitioning algorithm which is provided as programming API called *hMETIS* library [17]. *hMETIS* has been shown to quickly produce highly-qualified partitions for a wide range of unstructured graphs and hypergraphs [18][19][20]. *CHAMELEON* uses *hMETIS* to split a cluster C_i into two sub-clusters C^A_i and C^B_i such that the edge-cut between C^A_i and C^B_i is minimized and each one of these sub-clusters contains at least 25% of the nodes in C_i. This requirement is often referred to as the *balance constraint* 1.

2.2 Some Works on Document Clustering Model

Document clustering can be implemented either with object cluster model or word cluster model.

In the former model, documents are represented into vectors of term-weight pairs. Therefore, document clustering is transformed into vector clustering in the multi-dimension space. And many research works existed in this model. For example, [11] provided a clustering algorithm called CBC (Clustering By Committee). This work initially discovers a set of tight clusters (high intra-group similarity), called committees, which are well scattered in the similarity space (low inter-group similarity). The union of the committees is however a subset of all elements. The algorithm proceeds by assigning elements to their most similar committee.

In the latter model, features (words) of the cluster are firstly extracted with respect to their semantic relationships. Then, documents are assigned to the corresponding clusters based on the similarity between features of the cluster and the document. In this respect, the documents clustering is actually topic oriented other than document oriented. That mean, if a document contains multiple topics, it will probably be assigned into several clusters. Work [10], for instance, use the information bottleneck method to obtain a better result by using word cluster model. It first finds *word-clusters* that capture most of the mutual information about the set of documents, and then find *document clusters*, that reserve the information. It is much less sparse and noisy than the object cluster model.

Besides, document clustering, there are also many works on document classification. ACIRD: Intelligent Internet Document Organization and Retrieval 3 is a typical work in this model. *Term Semantic Network* (*TSN*) is one of the main concepts used in ACIRD system. It builds the *TSN* based on the term association rule and supports the data representation model.

3 Web Image Semantic Representation

In our system, Web pages are gathered and converted into Document object model (*DOM*) trees by the system crawler. Then, Web images are filtered out by removing meaningless images (or stop images, such as Web site logo, button) before further processing. In order to extract the semantics of a Web image, we partition the Web page of the image into a sequence of semantic blocks with respect to the distances to the image. And we suppose that different blocks have different semantic relevance to the Web image [2]. A term's semantic relevance to an embedded image is calculated both based on its local occurrences in the individual semantic blocks and the semantic distances of the corresponding blocks to the image.

In our approach, we firstly partition the whole associated text of a Web image into three parts, which include *TM* (the texts from <TITLE> and <META> element), *LT* (the text attached to the image display command), and *BT* (the text of <BODY> element), based on their functions in the text. *TM* is often used as the summary of the page content, and it may provide valuable semantic implications to the images especially in an image containers; *LT* is the closest part of the associated text to the image, and often highlight the semantics of the local Web image; *BT* is big in size, but can provide explanations for the contents of the image in most of cases.

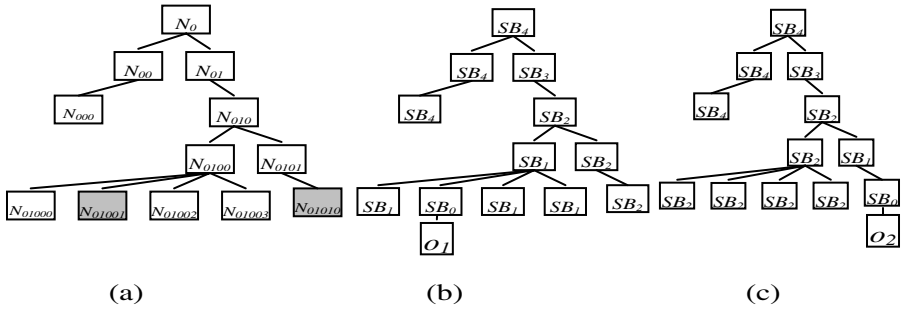

Fig. 2. Semantic Fragmentation

Since the big size of *BT*, it may contain diverse semantics. Therefore, we further partition it into a sequence of semantic blocks with respect to the nested structure of tag elements. Figure 2 is an example for partitioning *BT* blocks. Figure 2(a) represents the original tree structure of a Web page which contains two images o_1 and o_2 in nodes N_{01001} and N_{01010} respectively. Then, Figure 2(b) and (c) shows the semantic fragmentations for image o_1 and o_2 respectively. Formally, for any image o, there exist a sequence of nested nodes E_0, E_1, …E_N of *BT*, and $E_N \supset E_{N-1} \supset \cdots \supset E_0$ such that

E_N=<*Body*> (the root node) and $E_0 \ni o$. And furthermore, no other HTML element exists between E_{j+1} and E_j for any j. Obviously, value N (the number of the nested elements) may be different for different web pages. To simplify our notations, we set N big enough. And if $E_j=BT$, then $E_n=BT$ for any $N \geq n \geq j$. Therefore, semantic blocks are defined as $SB_j=E_{j+1}-E_j$ for $N-1 \geq j \geq 0$ (Figure 2 (b), (c)). Then, we have $BT=\cup_{0 \leq j \leq N-1} SB_j$ with $SB_k \cap SB_l = \Phi$ for any $k \neq l$. Figure 2 illustrates the principle of our Web page fragmentation. In this example, we suppose Web page p has been partitioned into 4 layers of SB_0, SB_1, SB_2, and SB_3, with SB_0 containing Web image o. Then, the semantic relevance of SB_k to o reduces with k increasing from 0 to 3. That is, t_i is more relevant to image o than t_j because of their locations relative to image o. Actually, a term t may appear in several semantic blocks (other than only one). In such a case, t's overall relevance to o should be taken into account of both its importance degrees in all blocks and block's relevances to image o.

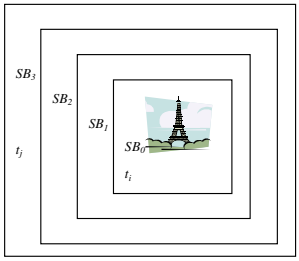

Fig. 3. Semantic Fragmentation for a Web Image

In general, we suppose SB_j ,$0 \leq i \leq N$, are semantic blocks with respect to Web image o, the local occurrence of a term t in block SB_j is calculated as

$$ntf(t)|_{SB_j} = \frac{tf(t)|_{SB_j}}{|SB_j|} \tag{1}$$

where $ntf(t)|_{SB_j}$ is called normalized frequency of term t over SB_j, $tf(t)|_{SB_j}$ is the frequency of term t in SB_j, and $|SB_j|$ is the size of SB_j. Thus, term t's overall semantic relevance to image o through page p is obtained as

$$ttf(t)|_p = \sum_{1 \leq i \leq N-1} w_j * ntf(t)|_{SB_j} \tag{2}$$

where N is the total number of the semantic blocks, and w_j is the weight of SB_j's semantic relevant to the embedded image o. Without loss of generality, we suppose w_j's are normalized, such that $\sum_{1 \leq i \leq N-1} w_j = 1$. In order to determine the values of those w_j's, a greedy algorithm, called *Two-Way-Merging*, is used. In our approach, a binary tree is constructed bottom upward, with SB_j's as all the leaves (Figure 4). Then, merging is carried out recursively, from bottom to top. For each step, a pair of factors

(α^l_i,β^l_i) is obtained, where l stands for the level of merging, α^l_i and β^l_i are merging factors from left and right sub-branches of the processing node respectively, and they are normalized as $\alpha^l_i+\beta^l_i = 1$. In the calculation, we choose average precision of retrievals [9] as the objective function for determining (α^1_i,β^1_i). In other words, the values should maximize retrieval precisions of Web images on average. Then, w_j is finally gotten by multiplying all corresponding level factors of SB_j in the tree. For example, in Figure 4, $w_1 = \alpha^1_1 * \alpha^2_1$, $w_2 = \beta^1_1 * \alpha^2_1$, $w_3 = \alpha^1_2 * \alpha^2_1$, $w_2 = \beta^1_2 * \beta^2_1$.

With above processing, in our approach, a Web image o, which is embedded in page p, can be semantically represented as a vector

$$o = <(t_1, ttf\downarrow_p(t_1)), (t_2, ttf\downarrow_p(t_2)), \ldots, (t_n, ttf\downarrow_p(t_n))>. \quad (1)$$

Therefore, the problem of Web image clustering is transformed into that of term vector clustering.

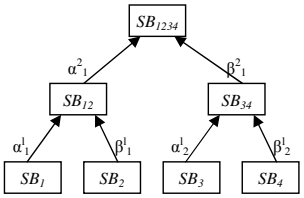

Fig. 4. Two-Way Merging Processing

4 Building Term Semantic Network

To extract topics of Web images, term co-occurrences with respect to Web images are important resource to use. Semantic relationships between terms can be described by the co-occurrences of terms. For example, if two terms often appear together in association with images, they are most likely to be related to the same topics. Term Semantic Network is often used in representing the overall semantic relevances among terms. According to [12], the association rule is formally defined as follows:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. T is a database of transactions. Each transaction t in T is a set of items such that $t \subseteq 1$. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \Phi$. The rule $X \rightarrow Y$ holds in the transaction set T with confidence c if c percent of transactions that contain X also contain Y. The rule $X \rightarrow Y$ has support s in the transaction set T if s percent of transactions contain $X \cup Y$.

In this paper, we use a simple mining association algorithm – Apriori [13] -- to mine out the association rules of the words. And we only consider one-to-one term relationship. We define *confidence* (*conf*) and *support* (*sup*) of term association $t_i \rightarrow t_j$ as follows: let

$$D(t_i, t_j) = D(t_i) \cap D(t_j) \quad (2)$$

where $D(t_i)$ stands for the documents include term t_i, $D(t_i) \cap D(t_j)$ stands for the documents that include both t_i and t_j, then, confidence from t_i to t_j is:

$$Conf(t_i, t_j) = \frac{\| D(t_i, t_j) \|}{\| D(t_i) \|} \tag{3}$$

where $\| D(t_i, t_j) \|$ stands for the total number Web documents which contain both t_i and t_j, and $\| D(t_i) \|$ is the total number of Web documents containing term t_i. And

$$Sup(t_i, t_j) = \frac{\| D(t_i, t_j) \|}{\| D \|} \tag{4}$$

where $\|D\|$ stands for the number of document in our Web document database. Above definitions for *Confidence* and *Support* between two words reflect their co-occurrences in the collection. In detail, $Conf(t_i,t_j)$ indicates t_i and t_j's co-occurrence relative to t_i, and $Sup(t_i,t_j)$ indicates their absolute co-occurrences in the collection. In other words, these two functions can get higher values if t_i and t_j are often used together in Web documents. Therefore, in this paper we suppose that two terms are likely to be features for the same topic if they have higher values of *Confidence* and *Support*. And a term semantic network (*TSN*) is created with terms as nodes and $Sup(t_i,t_j)*Conf(t_i,t_j)$as the weighted edge from t_i to t_j.

The initial *TSN* is quite complex because of the huge amount of edges. In order to simplify the graph, we set the minimum value of *support* and *confidence* and try to remove the edges whose *support* and *confidence* values below the thresholds. Currently, in this paper, we empirically set the minimum support to 0.05 and minimum confidence to 0.1. With this filtering, the edges of *TSN* reduce to 4 millions in number.

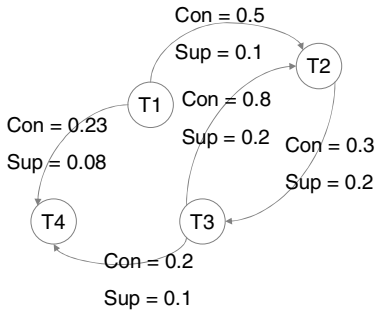

Fig. 5. *TSN*: *Con* and *Sup* Annotated to the edges

Figure 5 shows our *TSN*. In this figure, each node represents a term, and each edge is annotated with two values (*support* and *confidence*) to represent its weight. In our implementation, we use a single value *Rank* = *support* * *confidence* to represent the edge weight. We will discuss how to use this graph for term clustering in following section.

5 Web Image Clustering

To cluster Web images, we firstly mine out term clusters using *TSN*. And in our system, we suppose each term cluster is on the same topics. And we split terms of *TSN* into sub-graphs using *CHAMLEON* algorithm [1] and extract the weights of terms in each cluster by using *HITS* algorithm [21]. And finally we assign Web images to corresponding topic clusters by calculating the similarities between Web image features and topic features.

5.1 Term Clustering

CHAMELEON is an attractive algorithm due to its adaptivity to different clustering applications. And the algorithm includes two steps- splitting and merging.

In splitting phase, *TSN* is partitioned into a set of sub-graphs. Actually, it splits the graph based on the hMETIS algorithm. hMETIS can split a cluster into two sub-clusters such that the edge-cut between these two clusters is minimized and each one of these sub-clusters contains at least 25% of the nodes in the original cluster (balance constraint) [17]. This balance constraint in *CHAMELEON* algorithm is critical to prevent outliers when splitting *TSN* and generate natural clusters.

And the second phase recursively merge some sub-clusters with respect to two indicators *RI* (Relative Inter-connectivity) and *RC* (Relative Closeness). Two clusters are merged if the *RI* and *RC* are over some thresholds. The relative inter-connectivity between a pair of clusters C_i and C_j is given by

$$RI\ (C_i, C_j) = \frac{\left|EC_{\{Ci,Cj\}}\right|}{\frac{\left|EC_{Ci}\right| + \left|EC_{Cj}\right|}{2}} \tag{5}$$

Where $EC_{\{Ci,Cj\}}$ is the set of edges crossing cluster C_i and C_j. And EC_{Ci} and EC_{Cj} are the size of its min-cut bisector (i.e., the weighted sum of edges that partition the graph into two roughly equal parts) in C_i and C_j respectively.

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{Ci,Cj\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{ECCi} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{ECCj}} \tag{6}$$

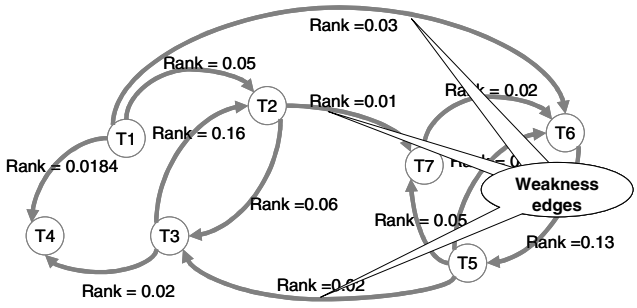

Fig. 6. *TSN*: Choosing Weak Edges to Split

where $\overline{S}_{ECCi}$ and $\overline{S}_{ECCj}$ are the average weights of the edges that belong in the min-cut bisector of clusters *Ci* and *Cj* , respectively, and $\overline{S}_{EC_{\{Ci,Cj\}}}$ is the average weight of the edges that connect vertices in *Ci* to vertices in *Cj* .

Figure 6 shows an example of how to calculate the *RI* and *RC*.

In splitting phase, *CHAMELEON* utilizes *hMETIS* algorithm to split *TSN* based on the weight of each edge. In this example, it chooses to cut the edge $E_{T1\rightarrow T6}$, $E_{T2\rightarrow T7}$ and $E_{T5\rightarrow T3}$ due to their weaknesses comparing with others. After this process, terms of *TSN* are categorized into two clusters: Term Cluster 1 (*T1, T2, T3, T4*) and Term Cluster 2 (*T5, T6, T7*) (Figure 7).

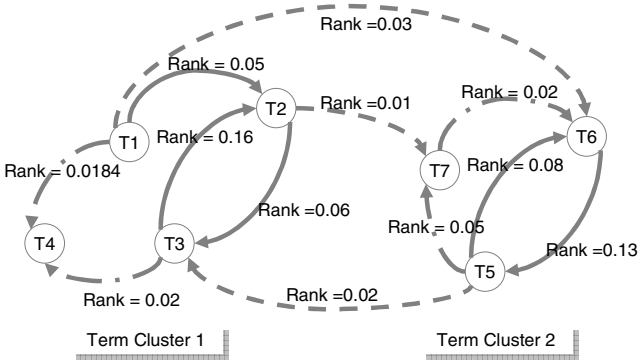

Fig. 7. *TSN*: $EC_{\{Ci,Cjj}$ and $\overline{S}_{EC_{\{Ci,Cj\}}}$ Calculation

Figure 7 provides an illustration on how to calculate equation (4) and (5) in *CHAMELEON* algorithm. In above figure, we represent edges with different line types. And lines of (⟶) represents edges between two terms from the same cluster, lines of (- - -▶) represent edges between two terms across different clusters, and lines of (—·▶) stand for edges between two terms from the cluster's min-cut bisector. Table 1 shows the calculations and corresponding values.

Table 1. Paramiter Calculations in *CHAMELEON* Algorithm

$EC_{\{Ci,Cj\}}$	$\lvert E_{T1\rightarrow T6}\rvert+\lvert E_{T2\rightarrow T7}\rvert+\lvert E_{T5\rightarrow T3}\rvert$	0.06
EC_{Ci}	$\lvert E_{T1\rightarrow T4}\rvert+\lvert E_{T3\rightarrow T4}\rvert$	0.0384
EC_{Cj}	$\lvert E_{T7\rightarrow T6}\rvert+\lvert E_{T5\rightarrow T7}\rvert$	0.07
$\overline{S}_{EC_{\{Ci,Cj\}}}$	$(\lvert E_{T1\rightarrow T6}\rvert+\lvert E_{T2\rightarrow T7}\rvert+\lvert E_{T5\rightarrow T3}\rvert)/3$	(0.03+0.02+0.01)/3=0.02
$\overline{S}_{ECCi}$	$(\lvert E_{T1\rightarrow T4}\rvert+\lvert E_{T3\rightarrow T4}\rvert)/2$	(0.0184+0.02)/2=0.0192
$\overline{S}_{ECCj}$	$(\lvert E_{T7\rightarrow T6}\rvert+\lvert E_{T5\rightarrow T7}\rvert)/2$	(0.05+0.02)/2=0.035
$\lvert C_i\rvert$	Number of terms in *Ci*	4
$\lvert C_j\rvert$	Number of terms in *Cj*	3

With all the values in table 1, we can easily calculate out the values of *RI* and *RC* as 1.107 and 0.488 respectively. With these two functions, two different merging schemes can be implemented in *CHAMELEON*. The first scheme merges only those pairs of clusters whose relative inter-connectivity and relative closeness are both above some user specified thresholds. *CHAMELEON* visits each cluster C_i , and checks to see if any one of its adjacent clusters C_j satisfy the following two conditions:

$$RI(C_i, C_j) \geq T_{RI} \cap RC(C_i, C_j) \geq T_{RC} \tag{7}$$

where T_{RI} and T_{RC} are the user specified thresholds.

In the second scheme, *CHAMELEON* uses one function as the objective indicator which is actually a combination of *RI* and *RC* as

$$RI(C_i, C_j) * RC(C_i, C_j)^{\alpha} \tag{8}$$

where α is a user specified parameter which can be tailored with respect to different applications. If α > 1, then *CHAMELEON* gives a higher importance to the relative closeness, and when α < 1, it gives a higher importance on the relative inter-connectivity. In this paper, we use the second scheme with α =1. And the term set *C* of *TSN* is partitioned into several clusters C_0, C_1, …, C_n, such that

$$C = \bigcup_{0 \leq i \leq n} C_i \tag{9}$$

And each term cluster *Ci* is assumed to dedicate on one topic.

5.2 Extracting Semantic Significances of Terms

For each cluster *Ci*, all terms in *Ci* is taken as its features which well reflect the topic of the cluster. However, their significances to the cluster may not be the same in general. In this paper, we also want to determine term's weights based on *TSN*. To reach this objective, we calculate the weights of terms in cluster (C_i) by using *HITS* algorithm [21]. Actually, *HITS* is a popularly used link analytic algorithm, it extracts the degrees of node importance from the link structure of hyperlinked environment (Web). In this paper, we use it to extract the significances of terms with respect to the semantic links among them. In our context, terms in each cluster *Ci* are linked with direct edges. The ideas behind *HITS* algorithm is that a node is important in the directed graph if more links are either from or to this node. In fact, for each node *HITS* uses two concepts—*Authority* and *Hub*—in its analysis. *Authority* indicates the degree of links coming to the node, while *Hub* indicates the degree of links going from this node. A good *Authority* means it is pointed by many good nodes and a good *Hub* means it point to many good nodes. The relationship between these two factors is shown in Figure 8.

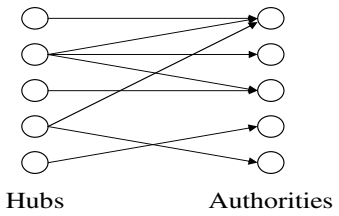

Fig. 8. A densely linked set of hubs and authorities

Because we have already obtained the values of support and confidence between different terms, we define the relationship between term t_i and term t_j as follows:

$$R(t_i, t_j) = Sup(t_i, t_j) \times Conf(t_i, t_j) \tag{10}$$

We initially assign values for *Authority* and *Hub* as follows:

$$Authority(t_i) = \sum_{t_j:(t_j,t_i)\in E} R(t_j, t_i) \tag{11}$$

$$Hub(t_i) = \sum_{t_j:(t_i,t_j)\in E} R(t_i, t_j) \tag{12}$$

Then, *HITS* uses an Interative Algorithm to get the final values for them with:

$$Authority(t_i) = \sum_{t_j:(t_j,t_i)\in E} Hub(t_j) \tag{13}$$

$$Hub(t_i) = \sum_{t_j:(t_i,t_j)\in E} Authority(t_j) \tag{14}$$

Because *HITS* has already shown that the nodes with higher values of *Authority* and *Hubs* are important in the semantic network, we define the weight of each term in cluster *C* by the following formula:

$$Rank_c(t_i) = Authority(t_i) + Hub(t_i) \tag{15}$$

Then, cluster C can be described with vector as

$$C = \langle Rank_C(t1), Rank_C(t2), \ldots, Rank_C(tn) \rangle. \tag{16}$$

5.3 Assigning Web Images to Corresponding Clusters

In the previous discussions, both Web images and topic clusters are represented into vectors of term-weigh pairs (equation 1 and 16). Then, Web image assignments to the clusters are based on similarity measurement between the image vectors and the cluster vectors. The following equation shows our similarity measurement:

$$Sim(o,c) = \frac{\sum \left(Rank_c(t_i) \times ttf(t_i)\right)}{\|o\|_2 + \|c\|_2} \tag{17}$$

where $o = \langle (t_1, ttf|_p(t_1)), (t_2, ttf|_p(t_2)), \ldots, (t_n, ttf|_p(t_n)) \rangle$ is a Web image and c is a cluster, $\|o\|^2$ is the norm of the image object, and $\|c\|^2$ is the norm of the cluster feature. These two values are defined as:

$$\|c\|^2 = \sqrt{(rank_c(t_1))^2 + (rank_c(t_2))^2 + \ldots + (rank_c(t_n))^2} \tag{18}$$

$$\|o\|^2 = \sqrt{ttf(t_1)^2 + ttf(t_2)^2 + \ldots + ttf(t_n)^2} \tag{19}$$

With this method, a Web image may be conceptually related to several clusters. So if the $Sim(o,c) > \gamma$, we assign the object o to cluster C.

6 Experiments

In this section, we compare our proposed approach with other two feature based clustering methods: (1) *CHAMELEON* algorithm without using *HITS* and, (2) a variant of K-Means feature clustering algorithm.

The web crawler in our system crawled more than 1000000 web pages from internet and only about 50000 pages are filtered out for the experiment. Here, all useful pages are the ones which contain image link(s). Our crawler guarantees that Web page are fully crawled (all child links from the page are crawled) in order to support our semantic extractor's [2]. To carry out the experiment, we defined labels for each testing image. And Recall/Precision is used as the objective function.

K-Means is one of the most popularly used clustering algorithms. The basic idea of traditional K-Means algorithm is to recursively re-compute the cluster centers using new cluster membership until no more change. The distance between two objects is symmetric. However, this traditional method can not be directly used to slit our *TSN*. In our situation, firstly, *TSN* is a directed graph. Therefore, the distance implied in *TSN* between two terms is asymmetric. We modify *TSN* into an undirected graph by simply defining the distance between two terms as:

$$Dist(t_i, t_j) = R(t_i, t_j) + R(t_j, t_i) \tag{18}$$

Where $R(t_i,t_j)$ and $R(t_j,t_i)$ are as in equation (10). Secondly, if t_i does not have a direct edge to t_j, we need to use *Prefect Term Support* (*PTS*) algorithm to uniquely define the distance between them [3]. After above two preprocessing steps, we apply K-Means algorithm to split all terms into clusters. Then, we use the same similarity measurement function to assign all images to corresponding clusters. We call this clustering method as 'Variant of K-Means'.

In Section 5.2, we used *CHAMELEON* algorithm to split the *TSN*. Then, *HITS* algorithm is employed to extract the weights of terms in the clusters. In our experiment, we try to compare it with the one without using *HITS* algorithm processing.

In this experiment, we start with 32 seed points for Variant of K-Means algorithm and use *CHAMELEON* to split the *TSN* network into 32 clusters too. As a result, we find only 17 out of 32 of the clusters are meaningful. The other 15 clusters are hard to be identified as coherence clusters. Then we manually combine some clusters that have the same meaning together, after this adjustment, we got 13 meaningful clusters from the K-Means algorithm (table 2).

With *CHAMELEON* algorithm, we can get about 23 meaningful (coherence) clusters. After manually merging clusters which have the same semantics, we got 17 meaningful clusters. This result reveals that *CHAMELEON* algorithm can always produce much finer clusters that the Variant of K-Means.

Table 2. The meaningful clusters number for each method

Method Name	**Meaningful Clusters Num**	
	After running the algorithm	**After human adjust**
K-Means	17	13
Without HITS	23	17
With HITS	23	17

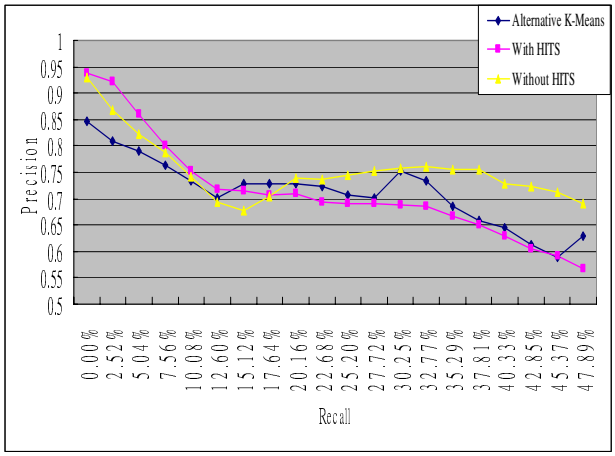

Fig. 9. The statistical result of our system

In Figure 9, we show performances of all the three methods. As we see in the figure, K-Means has the lowest performance in these three methods.

When we used *CHAMELEON* algorithm to split the *TSN*, the result is better than K-Means. If we used *HITS* to weigh cluster features, the query performance is better within the first 17.64% of recall rate than that of the one without running *HITS*. Although the improvement is not very high, we still think it is very useful since the large amount of the Web query result. As a matter of the fact, most Web users can only browse the result within top 10% of the query result. Therefore, it is worth of running *HITS* for weighing terms in clusters.

7 Conclusions and Future Work

In this paper, we have addressed our approach for Web image clustering. Our model is feature based (other than object based). Therefore, one image may be assigned into multiple clusters if it semantically related to more topics. This property is more effective in the Web information retrieval systems since the topic comprehensive of the Web information. Our model is based on the combination of several existing machine learning and data mining techniques with some necessary modifications.

We firstly represent Web images with vectors of term-weight pairs. Then a term semantic network (*TSN*) is constructed by using semantic associations between terms. The semantic associations in our system is described with two functions—*Confidence* and *Support*. Thirdly, we cluster terms by using *CHAMELEON* algorithm. Finally, in order to extract the weights of terms in each cluster, *HITS* algorithm is used. We assign Web images into clusters with respect to the similarity measurements between image vectors and cluster features. Our experimental result shows that our approach can rank relevant result earlier in the query result. That is necessary in the Web query environment.

References

1. George Karypis, Eui-Hong (Sam) Han, Vipin Kumar, CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling, computer, Volume 32 , Issue 8 (August 1999) 68-75
2. Zhiguo Gong, Leong Hou U and Chan Wa Cheang, An Implementation of Web Image Search Engine, In Proceedings of the 7th International Conference on Asian Digital Libraries, ICADL, (2004)
3. Lin, S. H., Chen, M. C., Ho, J. M., and Huang, Y. M., ACIRD: Intelligent Internet Documents Organization and Retrieval, IEEE Transactions on Knowledge and Data Engineering (2002)
4. K.S. Jones and D.M. Jackson, The Use of Automatically-Obtained Classifications for Information Retrieval, Information Processing and Management (IP&M), vol. 5, (1970) 175-201
5. JAIN, A. K. AND DUBES, R. C. Algorithms for Clustering Data. Prentice-Hall advanced reference series. Prentice-Hall, Inc., Upper Saddle River, NJ. (1988)
6. ANDERBERG, M. R. Cluster Analysis for Applications. Academic Press, Inc., New York, NY. (1973)
7. DIDAY, E. AND SIMON, J. C. Clustering analysis. In Digital Pattern Recognition, K.S. Fu, Ed. Springer-Verlag, Secaucus, NJ, (1976) 47–94
8. NAGY, G. State of the art in pattern recognition. Proc. IEEE 56, (1968) 836–862
9. BAEZA-YATES, R. A. Introduction to data structures and algorithms related to information retrieval. In Information Retrieval: Data Structures and Algorithms, W. B. Frakes and R. Baeza-Yates, Eds. Prentice-Hall, Inc., Upper Saddle River, NJ, (1992) 13–27
10. N Slonim, N Tishby, Document Clustering using Word Clusters via the Information Bottleneck Method, SIGIR, (2000)
11. P Pantel, D Lin, Efficiently Clustering Documents with Committees, PRICAI, 2002
12. R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules between Sets of Items in Large Databaseds, Proc. ACM SIGMOD Int'l Conf. Management of Data, (May 1993)
13. R. Agrawaland R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases, (VLDB), (Sept. 1994)
14. A.K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, (1988)
15. Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In Proc. of the 15th Int'l Conf. on Data Eng., (1999)
16. Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In Proc. of 1998 ACM-SIGMOD Int. Conf. on Management of Data, (1998)
17. G. Karypis and V. Kumar. hMETIS 1.5: A hypergraph partitioning package. Technical report, Department of Computer Science, University of Minnesota, (1998)
18. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, 48(1) (1998) 96–129
19. G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In Proceedings of the Design and Automation Conference, (1999)
20. C. J. Alpert. The ISPD98 circuit benchmark suite. In Proc. of the Intl. Symposium of Physical Design, (1998) 80–85
21. J.M. Kleinberg. Authorative sources in a Hyperlinked Environment. J. ACM 46:5, (1999) 604-632