Towards Automatic Generation of Query Taxonomy: A Hierarchical Query Clustering Approach

Shui-Lung Chuang
Institute of Information Science
Academia Sinica
Taipei 115, Taiwan
slchuang@iis.sinica.edu.tw

Lee-Feng Chien
Institute of Information Science
Academia Sinica
Taipei 115, Taiwan
lfchien@iis.sinica.edu.tw

Abstract

Previous works on automatic query clustering most generate a flat, un-nested partition of query terms. In this work, we are pursuing to organize query terms into a hierarchical structure and construct a query taxonomy in an automatic way. The proposed approach is designed based on a hierarchical agglomerative clustering algorithm to hierarchically group similar queries and generate the cluster hierarchies by a novel cluster partition technique. The search processes of real-world search engines are combined to obtain highly ranked Web documents as the feature source for each query term. Preliminary experiments show that the proposed approach is effective to obtain thesaurus information for query terms, and is also feasible to construct a query taxonomy which provides a basis for in-depth analysis of users' search interests and domain-specific vocabulary on a larger scale.

1. Introduction

As Web searching has grown, research interests on mining search engine logs to discover users' search patterns and information requests have increased. Query clustering is a research task aiming to group similar user queries together. This task is important to discover the common interests among the users and to exploit the experience of previous users for the others [10]. Knowledge obtained with query clustering can be used in the development of thesauri and recommending systems [9]. The discovered clusters of queries can assist users in reformulating refined queries, discovering users' FAQs for question answering systems [10], and performing more effective query expansion and term suggestion routines in search engines [1].

Previous works on automatic query clustering most organize query terms into flat clusters. In this work, we are pursuing to organize users' query terms into a hierarchical structure and construct a query taxonomy in an automatic way. The query taxonomy is defined as the classification tree constituted by the concept hierarchies of users' requests and categorized query terms that are automatically generated. As the example illustrated in Figure 1, a query taxonomy is a classification tree in which similar user queries (as leaf nodes) are grouped to form basic query clusters (as interior nodes), and similar query clusters form super clusters recursively to characterize the associations between composed clusters. Each of the query clusters can represent a certain concept of users' information requests.

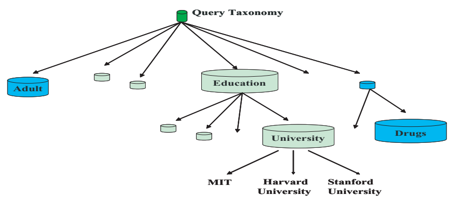

Figure 1. An example query taxonomy.

With the query taxonomy, deeper analysis of domain-specific terminology and further discovery of term relationships can then be performed under the corresponding subject domain of each cluster. Manually collecting and organizing such term vocabularies is cost-ineffective and inapplicable especially for organizing Web queries. For the above purpose, this paper presents a hierarchical query clustering approach. The proposed approach is extended from Hierarchical Agglomerative Clustering algorithm (HAC) to cluster similar queries and to generate appropriate cluster hierarchies. Since a query term is short in length and simple in structure, there is a lack of information to judge the query term's corresponding subject domains. The proposed approach combines with the search processes of real-world search engines to obtain the highly ranked Web documents from which the features for each query term are extracted. With the huge amount of Web pages indexed by the search engines, most of the users' query terms can retrieve relevant documents and create feature sets, and therefore be clustered properly. A hierarchical cluster partitioning technique based on a heuristic quality measure of any particular partition is further applied on the binary-tree hierarchy produced by the HAC algorithm to generate a natural and comprehensive multi-way-tree hierarchy. The preliminary experiments and observations also show the promising results.

In the rest of this paper, we first review some related work and introduce the proposed approach. Then the conducted experiments and their results are presented. Finally, we discuss some applications and conclude our results.

2. Related Work

Some researches on clustering indexed terms are in certain degree related to our research, such as the works on latent semantics, SVD, term relationship analysis, etc [3, 6]. Most of them dealt with the automatic clustering of controlled indexed terms into clusters and used them to assist information retrieval systems in the query expansion process to improve the precision and recall ratios. Query clustering and the construction of concept hierarchies for users' queries were not the main subjects of these investigations.

The characteristics of short Web query and noisy search results have led researchers to investigate the query clustering problem. Beeferman and Berger [1] proposed a query clustering method based on "click-through data," which is a collection of user transactions with queries and their corresponding clicked URLs, to discover correlations between queries and the clicked pages. Query terms with more common clicked URLs were taken as similar and being grouped. Without merely using the clicked URLs, Wen et. al. [10] developed a similar method to combine the indexed terms from the clicked pages to estimate the similarity between queries and achieved better performance. However, the number of distinct URLs is often huge in a Web search service. This might cause many similar queries not to be grouped together due to a lack of common clicked URLs. With fewer clicked URLs as the feature sets, it is more difficult to find similar terms for those new query terms or queries with lower usages. To allow most of users' queries with appropriate features to characterize intended search interests, the proposed approach exploits the highly ranked documents retrieved by a query term as the data source and designs an effective algorithm for query clustering.

3. Overview of the Proposed Approach

Hierarchical query clustering is loosely defined as a problem of automatically grouping similar query terms into disjoint clusters and organizing them into a hierarchical structure with the association strengths between clusters. The diagram depicted in Figure 2 shows the overall concept of the proposed approach, which is mainly composed of three computational processes: relevant document retrieval, feature extraction, and query clustering. The relevant document retrieval process is to retrieve the most relevant document sets for candidate query terms by combining with the search processes of real-world search engines. The retrieved document set is then passed to the feature extraction process to extract the features for each candidate query term. The query clustering process is to cluster similar queries and generate appropriate cluster hierarchies. It consists of two cascaded processing steps: generation of a binary-tree hierarchy and hierarchical cluster partitioning. The former step is to construct an initial binary-tree hierarchy by HAC algorithm to organize query clusters for the input query terms, and the latter step is to partition the hierarchy into a more natural multi-way tree according to the quality of each sub-hierarchy. In the following paragraphs, we will describe the details of these computational processes.

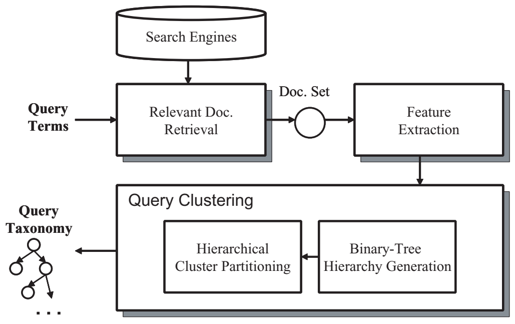

Figure 2. An abstract diagram showing the concept of the proposed approach.

4. Data Set

To instantiate the research, a three-month query term log collected in 1998 from Dreamer search engine[1] in Taiwan is used as the basis for our analysis. This data set contains 228,566 distinct query terms with total frequency 2,184,256. The most-frequent 18,017 queries have been manually categorized into a two-level hierarchy, consisting of fourteen major categories together with one hundred subcategories, by five Library & Information Science students together with a professional reference librarian for three months. This most-frequent query set, which only represented 8% of the distinct queries, totally formed 81% of the

[1] `http://www.dreamer.com.tw/`

search requests in the test log and has been used on analysis of search behavior and several research studies [9].

4.1. Collecting Retrieval Documents

As we mentioned previously, a query term is short in length and simple in structure. In order to judge the relevance of query terms, we take the highly-ranked documents retrieved from on-line search engines by the query terms as their feature source. To collect the contents each query term can retrieve, we adopt Google Chinese[2] as the back-end engine. Each query term is submitted to Google, and then up to 100 search result entries are collected. The title and description of each entry are extracted as the representation of the corresponding document. In the final result, only 107 queries among 18,017 total queries have no retrieved documents. By combining the search processes of real-world search engines, our query clustering hypothesis can be intuitively stated as: That two queries are clustered together is due to that they can retrieve similar contents.

4.2. Feature Set

It is a crucial step to determine a set of feature terms for characterizing the feature space of users' queries. The feature term set should be both as broad in coverage and as modest in size as possible. Query terms from search engine logs or the document terms extracted from the retrieved Web documents are two alternatives that can be taken as feature terms.

Since many queries may be related to ephemeral interests, such as a new movie or some recent events, the feature terms are pursuing to be sustainable. Another two-week query term log collected in 1999 from GAIS search engine[3] is used to filter out those ephemeral queries. Only 9,751 queries among our 18,017 queries still appear in the new query log. It is also found that query terms affected by time are mostly proper nouns. On the other hand, terms not affected by time, except for some proper nouns like the names of famous websites and people, are mostly subject terms like "movie," "baseball," or "flight ticket." These terms are considered to be core terms because they are long-lasting, modest in size, and rich in content. From our observations, core terms are more comprehensive in meaning and are often used by Web users to express popular search interests. Using them as features is believed to be more effective.

In order to reveal the effects of different feature sets, we prepare four kinds of features: (1) the 9,751 core terms (coreterm); (2) 10,000 most-frequent query terms in our data set (freqterm); (3) randomly-selected 10,000 terms from the 18,017 queries (randterm); (4) the Chinese character bi-, tri-grams and English words extracted from the retrieved documents (2,3-gram). The comparison will be made in a later section.

[2]`http://www.google.com/`
[3]`http://www.gais.cs.ccu.edu.tw`

5. The Query Clustering Algorithm

Clustering can be loosely defined as the process of organizing objects into groups whose members are closely associated in some way. The problem has been studied extensively in the literature, and there exist many different clustering algorithms. They are mainly in two major styles: partitioning and hierarchical clustering. K-means and hierarchical agglomerative clustering (HAC) are representatives of these two styles [4].

According to the literature, the HAC approach is more common and well-performed on the problems dealing with text-represented data such as clustering documents or Web search results [11, 12]. Hence we adopt HAC as the basic mechanism to our on-hand clustering problem. Perhaps one of the most important criteria to choose a clustering algorithm is the nature of the data and the anticipated clusters. Except common and extensive adoption of the algorithm, there are other considerations that lead us to this decision, and they will be briefly mentioned furthermost in the next subsection.

5.1. HAC Algorithm

An HAC algorithm operates on a set of objects with a matrix of inter-object distances and builds a binary-tree hierarchy where each node is a cluster and the clusters corresponding to the node's immediate children form a complete partition of that cluster [8]. First, the objects are placed into a list of singleton clusters C_1, C_2, ..., C_n. Then the closest pair of clusters $\{C_i, C_j\}$ from the list is chosen to merge. Finally, C_i and C_j are removed from the list and replaced with a new cluster $\{C_i \cup C_j\}$ (here we treat clusters as sets). This process is repeated until there is only one cluster remaining.

Inter-object distance
For the clustering approach, we have to measure the distances between instances to be grouped. We adopt the vector-space model as our data representation. As stated previously, each candidate query term is converted to a bag of feature terms via the content retrieved from on-line search engines by the query term. Let T be the feature term vocabulary, i.e., the set of all distinct terms in the whole data collection, and t_j be the j-th term in T. With simple processing, the i-th query term can be represented as a term vector v_i in a $|T|$-dimensional space. Let V be the set of all distinct query term vectors $v_1, v_2, \ldots, v_n$ and $v_{i,j}$ be the

weight t_j in v_i. The term weights in this work are determined according to the conventional *tf-idf* term weighting scheme,in which each term weight $v_{i,j}$ is defined as:

$$v_{i,j} = \left(0.5 + 0.5\frac{tf_{i,j}}{\max_{t_k \in T} tf_{i,k}}\right) \log \frac{n}{n_j}$$

where $tf_{i,j}$, the term frequency, is the number of occurrences of term t_j in the v_i's corresponding feature term bag, n is the total number of query terms, and n_j is the number of query terms which contain t_j in their corresponding bags of feature terms. The similarity between a pair of query terms is computed as the cosine of the angle between the corresponding vector ($cos\theta$), i.e.,

$$sim(v_a, v_b) = \frac{\sum_{t_j \in T} v_{a,j} v_{b,j}}{\sqrt{\sum_{t_j \in T} v_{a,j}^2}\sqrt{\sum_{tj \in T} v_{b,j}^2}}.$$

In this study, the distance between a pair of candidate query terms is defined as one minus the cosine measure:

$$dist(v_a, v_b) = 1 - sim(v_a, v_b).$$

Inter-cluster distance

The core of an HAC algorithm is to choose a specific distance function for clusters. Table 1 lists three most well-known inter-cluster distance functions.

Table 1. Three well-known cluster distance functions.

Method	Distance function
Single-linkage (SL)	$\min\limits_{v_a \in C_i, v_b \in C_j} dist(v_a, v_b)$
Average-linkage (AL)	$\frac{1}{\lvert C_i \rvert \lvert C_j \rvert} \sum\limits_{v_a \in C_i} \sum\limits_{v_b \in C_j} dist(v_a, v_b)$
Complete-linkage (CL)	$\max\limits_{v_a \in C_i, v_b \in C_j} dist(v_a, v_b)$

Intuitively speaking, the single-linkage method defines the distance between two clusters as the smallest distance between two objects in both clusters, and the complete-linkage uses the largest distance instead. Usually, the clusters produced by the single-linkage method are isolated but not cohesive, and there may be some undesirably "elongated" clusters. On the other extreme, the complete-linkage method produces cohesive clusters that may not be isolated at all. The average-linkage method represents a compromise between the two extremes.

To choose a feasible distance measure for our term-clustering problem, a major consideration is the heterogeneous and diverse nature of Web contents. Many traditional information retrieval problems such as text categorization meet big challenges when dealing with Web contents by directly applying their traditional well-performed methods. This is because Web in particular encourages diverse authorship, navigational and citation links, and short, fragmented documents with objects in various media types. All of these make the content data in Web noisy. In our approach, we extract the features from the snippets returned from search engines, whose quality may be worse and less trustful. Besides, term-vector-based data representation makes our clustering problem naturally a data-sparseness problem with high dimensional feature space. Some noisy features may make the distance measure not so reliable. So the method with nature to produce cohesive clusters is preferred, and the complete- and average-linkage methods are preferred mainly based on this consideration. The comparison of these methods will be made in a later section.

Before we move ahead, let's define some notations and formalization of the HAC algorithm for further illustration. In the HAC clustering process, at each iteration step, two clusters are merged as a new one, and the whole process halts when there exists only one un-merged cluster, i.e., the root node in the binary-tree hierarchy. Let v_1, v_2, ..., v_n be the initial input object vectors, and C_1, C_2, ..., C_n be the corresponding singleton clusters. Also let C_{n+i} be the new cluster created at the i-th step. The output binary-tree cluster hierarchy of the HAC algorithm can be undoubtedly and unambiguously expressed as a list C_1, C_2, ..., C_n, C_{n+1}, ..., C_{2n-1} with two functions $left(C_{n+i})$ and $right(C_{n+i})$, $1 \leq i < n$, indicating the left and right child of internal cluster node C_{n+i}, respectively. Figure 3 shows this detailed algorithmic procedure.

```
**HAC**$(v_1, v_2, \ldots, v_n)$
$v_i$, $1 \leq i \leq n$: the vectors of the objects
 1: for all $v_i, 1 \leq i \leq n$ do
 2:     $C_i \leftarrow \{v_i\}$
 3:     $f(i) \leftarrow true$ {$f$: whether a cluster can be merged}
 4: calculate the pairwise cluster distance matrix
 5: for all $1 \leq i < n$ do
 6:     choose the closest pair $\{C_a, C_b\}$ with $f(a) \wedge f(b) \equiv true$
 7:     $C_{n+i} \leftarrow C_a \cup C_b, left(C_{n+i}) \leftarrow C_a, right(C_{n+i}) \leftarrow C_b$
 8:     $f(n+i) \leftarrow true, f(a) \leftarrow false, f(b) \leftarrow false$
 9:     update the distance matrix with new cluster $C_{n+i}$
10: return $C_1, C_2, \ldots, C_{2n-1}$ together with functions $left$ and $right$
```

Figure 3. The hierarchical agglomerative clustering procedure.

5.2. Hierarchical Cluster Partitioning

The HAC algorithm produces a binary-tree hierarchy of clusters. However, we are more interested in an approach to producing a natural and comprehensive hierarchical organization such as Yahoo!, in which there are 13-15 major categories and each sub-category also contains an appropriate number of sub-categories. This multi-way-tree representation, instead of the binary-tree one, is believed to be more natural, easier, and more suitable for human to browse, interpret, and do some deeper analysis.

To generate a multi-way-tree hierarchy from a binary-tree one, a top-down approach is to decompose the hierarchy into several major sub-hierarchies first, and then, recursively apply the same procedure to each sub-hierarchy. To create a particular major sub-hierarchy partition from the binary-tree hierarchy, our approach is to determine a suitable level to cut the hierarchy at it. Before we describe our cutting criterion, let's illustrate some notations and the formalization of the problem. The problem of cutting at a suitable level can be taken as to determine which pair of adjacent clusters $\{C_{n+i-1}, C_{n+i}\}$, $1 \leq i < n$, in the binary-tree hierarchy $C_1, C_2, \ldots, C_n, C_{n+1}, \ldots, C_{2n-1}$ to put the partition point in between. Let the cut level l between $\{C_{n+i-1}, C_{n+i}\}$, $1 \leq i < n$, be indexed as number $n-i$, e.g., $\{C_{2n-2}, C_{2n-1}\}$ means cut level $l = 1$, and so on (referred to Figure 4). To simplify the further illustration, we also let $clusters(l)$ be the clusters produced at cut level l, which is the set of remaining unmerged clusters after $n-l-1$ iterations of the HAC procedure, and $CH(C_i)$ be the cluster hierarchy rooted at node C_i, i.e., $CH(C_i) = C_1^i, C_2^i, \ldots, C_{n_i}^i, C_{n_i+1}^i, \ldots, C_{2n_i-1}^i$ where C_1^i, …, $C_{n_i}^i$ are the leaf, singleton clusters, $C_{n_i+1}^i, \ldots, C_{2n_i-1}^i$ are the internal, merged clusters, and $C_{2n_i-1}^i = C_i$,. For example, in Figure 4, $clusters(2)$ is $\{C_5, C_6, C_7\}$, and $CH(C_8)$ is $\{C_3, C_4, C_5, C_6, C_8\}$. Note that all the above information could be obiter collected while the HAC clustering process is proceeding, so they are available without too much extra computational efforts.

The idea of our approach to this cluster partitioning problem is to find a proper cut level whose corresponding unmerged clusters are most qualified. Let $QC(C)$ be a function to measure the quality of a set of clusters C. Then, to determine a proper partition level is transfered to the problem of finding a cut level l with the best quality measure of clusters at that cut level, i.e., with the maximum value of $QC(clusters(l))$. This problem can be easily determined if the QC function is defined, and next, we will describe our defintion of QC.

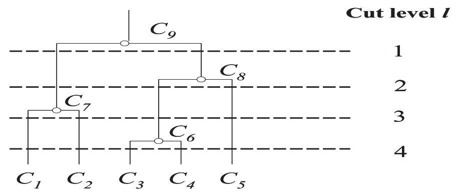

Figure 4. An illustrative example for cluster partitioning.

The generally accepted requirement of "natural" clusters is that they must be cohesive and isolated from the other clusters [8]. Our criterion to determine the proper cut level given a binary-tree cluster hierarchy is to heuristically realize this intuition. Our definition of $QC(C)$ is a product of three components: (a) $F(C)$: A function to measure the cohesion of the clusters; (b) $S(C)$: A function to measure the isolation of the clusters; And (c) $M(C)$: A function to measure whether the number of clusters are proper, i.e., the number of clusters should be neither too few nor too many. Thus the formula of $QC(C)$ is defined as

$$QC(C) = F(C)S(C)M(C).$$

Next, we will give the definition to each component.

Given a cluster C_i with n_i objects, we define the cohesion measure of the cluster as the average similarity measure of all its object pairs. For a singleton cluster, we define its cohesion measure as one because a singleton cluster is undoubtedly most cohesive. The overall cohesion measure of a set of clusters is defined as the weighted average cohesion measure and its formal definition is given by

$$F(C) = \frac{1}{n} \sum_{C_i \in C} n_i f(C_i)$$

$$f(C_i) = \begin{cases} \frac{2}{n_i(n_i-1)} \sum\limits_{\substack{v_a, v_b \in C_i \\ v_a \neq v_b}} sim(v_a, v_b), & \text{if } n_i > 1; \\ 1 & \text{otherwise.} \end{cases}$$

where n_i is the number of objects contained in C_i and n is the total number of objects in cluster set C.

Given two clusters C_i and C_j, we define the isolation measure between them as the smallest distance between two objects in both clusters. Notice that this definition is equivalent to their single-linkage distance measure, and its formula is shown in Table 1. Let it be notated as $mindist(C_i, C_j)$. The overall isolation measure of a cluster set $C = \{C_1, C_2, \ldots, C_k\}$ is defined as the average of this distance measure of all cluster pairs in C:

$$S(C) = \frac{2}{k(k-1)} \sum_{1 \leq i < k} \sum_{i < j \leq k} mindist(C_i, C_j)$$

Usually, a partition with neither too few nor too many clusters are preferred. Given n objects, there are at least one cluster and at most n clusters . In a hierarchical partitioning approach, we expect that the number of top-level clusters should be small, but a proper number is really hard to anticipate automatically because we have no idea of how many meaningful groups exist among the objects. Here, we take the square root of n as the expected cluster number. And then, an ellipse function is used to measure the degree of preference on the number of the given clusters. Its definition is given as follows:

$$M(C) = \sqrt{1 - \frac{(|C| - en)^2}{n^2}}$$

where n is the total number of objects contained in C and en is set as $\sqrt{n}$.

Now, all three components of our QC measure are defined. To choose a suitable cut level, we just need to compute QC value for each cut level and then select the one with maximum QC value. Figure 5 shows the detailed algorithmic procedure of the whole hierarchical clustering process, which first constructs a binary-tree cluster hierarchy of given objects and then, recursively applies the partitioning procedure to determine most appropriate cut level on each sub-hierarchy. To avoid performing the partitioning procedure on the cluster with too few objects, a constant ϵ is provided to restrict the size of a cluster to be further processed.

```
**HierarchicalClustering**($v_1, v_2, \ldots, v_n$)
$v_i, 1 \leq i \leq n$: the vectors of the objects
 1: $C_1, C_2, \ldots, C_{2n-1} \leftarrow$ **HAC**($v_1, v_2, \ldots, v_n$)
 2: return **HierarchicalPartitioning**($C_1, C_2, \ldots, C_{2n-1}$)
**HierarchicalPartitioning**($C_1, C_2, \ldots, C_n, C_{n+1}, \ldots, C_{2n-1}$)
$C_i, 1 \leq i \leq 2n-1$: the binary-tree hierarchy
 1: if $n < \epsilon$ then
 2:     return $C_1, C_2, \ldots, C_n$
 3: $maxqc \leftarrow 0, bestcut \leftarrow 0$
 4: for all cut level $l, 1 \leq l < n$ do
 5:     $qc \leftarrow QC(clusters(l))$
 6:     if $maxqc < qc$ then
 7:         $maxqc \leftarrow qc, bestcut \leftarrow l$
 8: for all $C_i \in clusters(bestcut)$ do
 9:     $children(C_i) \leftarrow$ **HierarchicalPartitioning**($CH(C_i)$)
10: return $clusters(bestcut)$
```

Figure 5. Hierarchical clustering algorithm.

In the literature, several criteria to determine the number of clusters for HAC algorithms have been suggested [7], but they are typically based on predetermined constants, e.g., the number of final clusters or a threshold for distance measure. Relying on predefined constants is practically harmful in applying the clustering algorithm because these criteria are very sensitive to the data set on-hand and hard to be properly determined. Our approach is automatic to determine the proper level to cut only based on the data set itself.

5.3. Cluster Naming

It is not easy to determine an appropriate name for a cluster. There are different alternative methods. In our current stage, we take the most high-frequency co-occurred feature terms from the composed query terms to name the cluster.

6. Experiment

To assess the performance of the proposed approach, two categories of experiments have been conducted. The first one was performed to test the accuracy of query clustering compared with human analysis under various feature sets and distance measure strategies, and the second one is to examine the quality of the hierarchical structure generation.

Two test query term sets are prepared from our data set: (1) the most-frequent 1,000 query terms (HF) and (2) randomly selected 1,000 query terms from the most-frequent 10,000 queries (RD). Notice that all 18,017 query terms in our data set (ref. Section 4) have been manually categorized into a two-level subject hierarchy, and this class information will be treated as the external information for us to evaluate the clustering results.

With the available external class information, we adopt F-measure [5] as the evaluation metric for the generated clusters. The F-measure of cluster j with respect to class i is defined as:

$$F_{i,j} = \frac{2R_{i,j}P_{i,j}}{R_{i,j} + P_{i,j}}$$

where $R_{i,j}$ and $P_{i,j}$ are recall and precision and been defined as $n_{i,j}/n_i$ and $n_{i,j}/n_j$, respectively, in which $n_{i,j}$ is the number of members of class i in cluster j, n_j is the number of members in cluster j, and n_i is the number of members of class i. For an entire hierarchical clustering, the F-measure of any class is the maximum value it attains at any node in the tree, and an overall F-measure is computed by taking the weighted average of all values for the F-measure as given by the following:

$$F = \sum_i \frac{n_i}{n} \max\{F_{i,j}\}$$

where the maximum is taken over all clusters at all levels, n is the total number of query terms, and n_i is the number of query terms in class i.

Table 2 shows the experimental results on the two test query sets by variant feature term sets and variant distance measure strategies. This category of experiments was performed by fixing the final cluster number to 100, and the performance measures were achieved based on the human-assigned 100 second-level category information. Besides, in order to reveal the effects of variant size of feature set, we ran another experiment on HF test set with complete-linkage method and different core term sets: top 100, 500, 1,000, 3,000, 5,000, and 9,751 core terms in accordance with the term frequency.

From the experimental results, we found that the average- and complete-linkage methods perform much better than the single-linkage one under the F-measure metric, and the average-linkage method is even slightly better. Also using core terms as features is stabler in the two

Table 2. The experimental results.

(A) With variant feature sets and distance measures.

HF	randterm	freqterm	coreterm	2,3-gram
SL	.1277	.1181	.1221	.1421
AL	.5217	.4955	.4921	.4977
CL	.4757	.4882	.4794	.4390
RD	randterm	freqterm	coreterm	2,3-gram
SL	.1092	.1049	.1200	.1295
AL	.4370	.3795	.4394	.3732
CL	.4097	.2288	.4043	.3747

(B) With variant size of core-term feature set.

HF	100	500	1,000	3,000	5,000	9,751
CL	.3443	.4547	.4562	.4634	.4851	.4767

Cluster 44 (Airlines class, Precision: 1, Recall: 1)	**Cluster 28** (Security software class, Precision: 0.78, Recall 0.55)	**Cluster 38** (Job class, Precision: 0.8; Recall: 0.75)
Eva (Airline name) tp	病毒 (Virus) cd	履歷表 (Resume) lj
長榮航空 (Airline) tp	norton cd	自傳 (Bio) lj
長榮 (EVA) tp	掃毒 (Virus Scanning) cd	就業 (Employment) lj
機票 (Flight Ticket) tp	病毒碼 (Virus Code) cd	徵才 (Job Opportunities) lj
遠東航空 (FAT Airline) tp	防毒軟體 (Anti -virus SW) cd	千里馬 (Taiwan Job Online) lj
復興航空 (Formosa Airline) tp	防毒 (Anti -virus) cd	工作 (Job) lj
華航 (China Airline ab.) tp	電腦病毒 (Computer Virus) cd	找工作 (Job Finding) lj
中華航空 (China Airline) tp	趨勢科技 (Trends, Inc.) cg	人力 (Human Resource) lj
飛機 (Airplane) tp	pc-cillin cd	104 (HR site) lj
航空公司 (Airway Co.) tp	cih cd	104 人力銀行 (HR site) lj
航空 (Airway) tp	sscan cd	人力銀行 (HR site) lj
	搞怪 cd	人力資源 (HR) lj
	駭客 (Hacker) ck	job lj
	駭客任務 (The Matrix) en	求職 (Job Hunting) lj
		求才 (Head Hunting) lj
		青輔會 (National Youth Commission) pl
		勞委會 (Bureau of Labor) pl
		職訓局 (Vocational Training Department) pl

Major category	/	Sub-category	:	ID
Computer&Network	/	Security Software	:	`cd`
	/	Software Company	:	`cg`
	/	Hacker/Crack	:	`ck`
Entertainment	/	Movie	:	`en`
Life Information	/	Job	:	`lj`
Politics	/	Local Government	:	`pl`
Travel	/	Airlines	:	`tp`

Figure 6. Several example clusters of query terms.

test sets, which confirms our consideration on feature selection stated in Section 4.2. To provide the readers a more comprehensive and clearer clustering result, Figure 6 lists three selected query clusters generated in the experiment. The headlines of the columns are the cluster IDs which are attached with the corresponding manually-assigned class names, achieved precision and recall rates respectively. The second lines of the columns are clustered query terms. The class ID assigned by humans is attached after each query term, and English translations are provided for the Chinese queries. The meanings of the class IDs could refer to the bottom part of the figure, in which it lists some of manual classes with their names of major classes, sub-classes, and symbols, respectively. It would be easy to see that the terms grouped in the same clusters are most highly relevant. Although some of the terms whose corresponding classes are not matched with the generated classes, e.g., “Trend, Inc.” in Cluster 28 and “Vocational Training Department” in Cluster 38, their classes ”Software Company” and “Local Government” are related to the automatically-assigned classes “Security Software” and “Job” respectively, and might not be considered as incorrect clustering.

On the other hand, the second category of experiments was performed to test the quality of the hierarchical structure generation. Table 3 shows some statistics on the hierarchies generated from the two test sets based on core-term feature set and average-linkage distance measure. The depth of the hierarchy and the number of clusters at each level seem appropriate. Notice that the F-measure values are better than the ones in the previous experiments. The achieved high F-measures reveal that hierarchically grouping the query terms seem more suitable to capture hidden associations among the query terms than the flat approach. Although it’s hard to have a quantitative approach to measure the goodness of the generated hierarchy structure, we believe this multi-way-tree representation is more natural and easier for human to browse and interpret than a deep binary-tree hierarchy.

7. Discussions and Applications

Query clustering serves as a first step toward the construction of thesaurus for Web search. For those query terms grouped in the same clusters, we found they contain many abbreviations, synonyms, related terms, and even translations that are hard to be manually organized. In fact, the proposed approach is very effective and can compete with conventional document-based methods to obtain thesaurus information for query terms. Conventional approaches rely on extracting co-occurring terms from documents and suffer from term segmentation difficulty. However, more accurate analysis on the relationships between the clustered terms need to be further discovered.

Table 3. Results of hierarchical structure generation.

	HF	**RD**
Hierarchy depth	6	5
# 1st-level clusters	34	35
# 2nd-level clusters	164	132
# 3rd-level clusters	206	216
# 4th-level clusters	79	93
# 5th-level clusters	25	40
# 6th-level clusters	14	
F-measure	.6048	.5532

One of the reasons that increases the robustness of the proposed approach is the representative of the selected feature sets and the co-occurring feature terms extracted from the highly ranked documents. It is worthy to note that the F-measure value can achieve 0.3443 by only using the top 100 core terms as the feature set. This reveals that these most-frequent core terms might represent some major search interests and are effective features in determining the query terms into certain clusters. In fact, a small set of core terms, e.g., 500, is useful to obtain an acceptable performance. This size is relatively much smaller than the total number of query terms.

However, there exist weaknesses in our initial study. In fact, some other factors may affect the accuracy of query clustering and need to be further studied, such as the relevance and the numbers of the retrieved documents, and the sufficiency of the surrogates of these documents as described in the previous paragraphs etc. Another challenge is the ambiguous nature of terms. Query terms are usually very short. This means that some terms might have multiple information requests. The proposed query clustering approach only groups such ambiguous query terms into one appropriate cluster. Clustering ambiguous query terms into multiple clusters is still unexplored. There are other challenges regarding the class size of query terms. With the automatic approach, we found both too large classes and too small classes are difficult to be successfully grouped. A larger class is easier to be clustered into separate clusters, such as adult classes. A small class is easier to be merged with other larger classes, such as academic-related classes. These classes will decrease the obtained F-measure value compared with the manual classes.

By the proposed query clustering approach for real-world query terms, there are some applications that can benefit from. Our work provides a good startup for constructing Web thesauri. With such thesaurus information, several applications, such as term suggestion for interactive Web search, can be applied in the search process. On the other hand, our approach is very helpful to know more about the information needs of Web users. With our approach to subject clustering of query terms, it's straightforward to construct an automatic system for Web search engines or digital library systems to monitor the changes of users' search requests, such as the distributions of users' search subject categories, and up-to-date frequencies of query terms in each class. The proposed approach has been successful applied to some of the above applications [2].

8. Concluding Remarks

In this work, we have proposed a hierarchical query clustering approach to organizing users' query terms into a hierarchical structure and construct a query taxonomy in an automatic way. To assess the performance of the proposed approach, two categories of experiments have been conducted. The first one was performed to test the accuracy of the query clustering compared with human analysis, and the second one is to test the quality of the hierarchical structure generation. The obtained experimental results have shown the possibility of the automatic approach to grouping similar query terms and generate concept hierarchies of users' search interests. The approach was also proven useful in various Web information retrieval applications.

References

[1] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 407–416, New York, NY, 2000. ACM Press.

[2] S.-L. Chuang and L.-F. Chien. Enriching web taxonomies through subject categorization of query terms from search engine logs. To appear in *Decision Support Systems, Special Issue on Web Retrieval and Mining*, 2002.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41(6):391–407, 1990.

[4] R. C. Dubes and A. K. Jain. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ,, 1988.

[5] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceeding of KDD-99*, San Diego, California, 1999.

[6] R. Mandata, T. Tokunaga, and H. Tanaka. Combining multiple evidence from different types of thesaurus for query expansion. In *Proceeding of the 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 191–197, New York, NY, 1999. ACM Press.

[7] G. W. Milligan and M. C. Cooper. An examination of procedures for detecting the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[8] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer, 1996.

[9] H.-T. Pu, S.-L. Chuang, and C. Yang. Subject categorization of query terms for exploring web users' search interests. *Journal of the American Society for Information Science and Technology*, 53(8):617–630, June 1 2002.

[10] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th International World Wide Web Conference*, pages 162–168, 2001.

[11] P. Willet. Recent trends in hierarchical document clustering: a critical review. *Information Processing and Management*, 24:577–597, 1988.

[12] O. Zamir, O. Etzioni, O. Madani, and R. M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 287–290, 1997.