Personalized Concept-Based Clustering of Search Engine Queries

Kenneth Wai-Ting Leung, Wilfred Ng, and Dik Lun Lee

Abstract—A major problem of current Web search is that search queries are usually short and ambiguous, and thus are insufficient for specifying the precise user needs. To alleviate this problem, some search engines suggest terms that are semantically related to the submitted queries so that users can choose from the suggestions the ones that reflect their information needs. In this paper, we introduce an effective approach that captures the user's conceptual preferences in order to provide personalized query suggestions. We achieve this goal with two new strategies. First, we develop online techniques that extract concepts from the web-snippets of the search result returned from a query and use the concepts to identify related queries for that query. Second, we propose a new two-phase personalized agglomerative clustering algorithm that is able to generate personalized query clusters. To the best of the authors' knowledge, no previous work has addressed personalization for query suggestions. To evaluate the effectiveness of our technique, a Google middleware was developed for collecting clickthrough data to conduct experimental evaluation. Experimental results show that our approach has better precision and recall than the existing query clustering methods.

Index Terms—Clickthrough, concept-based clustering, personalization, query clustering, search engine.

1 Introduction

As the Web keeps expanding, the number of pages indexed in a search engine increases correspondingly. With such a large volume of data, finding relevant information satisfying user needs based on simple search queries becomes an increasingly difficult task. Queries submitted by search engine users tend to be short and ambiguous. A study by Jansen et al. [20] found that the average query length on a popular search engine was only 2.35 terms. These short queries are not likely to be able to precisely express what the user really needs. As a result, lots of pages retrieved may be irrelevant to the user needs because of the ambiguous queries. On the other hand, users may not want to reformulate their queries using more search terms, since it imposes additional burden on them during searching.

To improve user's search experience, most major commercial search engines provide query suggestions to help users formulate more effective queries. When a user submits a query, a list of terms that are semantically related to the submitted query is provided to help the user identify terms that he/she really wants, hence improving the retrieval effectiveness. Yahoo's "Also Try" [6] and Google's "Searches related to" features provide related queries for narrowing search, while Ask Jeeves [1] suggests both more specific and more general queries to the user. Unfortunately, these systems provide the same suggestions to the same query without considering users' specific interests.

• *The authors are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.*
E-mail: {kwtleung, wilfred, dlee}@cse.ust.hk.

Manuscript received 19 Sept. 2007; revised 19 Mar. 2008; accepted 24 Apr. 2008; published online 25 Apr. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-09-0468.
Digital Object Identifier no. 10.1109/TKDE.2008.84.

In this paper, we propose a method that provides personalized query suggestions based on a personalized concept-based clustering technique. In contrast to existing methods that provide the same suggestions to all users, our approach uses clickthrough data to estimate user's conceptual preferences and then provides personalized query suggestions for each individual user according to his/her conceptual needs. The motivation of our research is that queries submitted to a search engine may have multiple meanings. For example, depending on the user, the query "apple" may refer to a fruit, the company Apple Computer or the name of a person, and so forth. Thus, providing personalized query suggestion (e.g., users interested in "apple" as a fruit get suggestions about fruit, while users interested in "apple" as a company get suggestions about the company's products) certainly helps users formulate more effective queries according to their needs.

The underlying idea of our proposed technique is based on concepts and their relations extracted from the submitted user queries, the web-snippets,[1] and the clickthrough data. Clickthrough data was exploited in the personalized clustering process to identify user preferences: A user clicks on a search result mainly because the web-snippet contains a relevant topic that the user is interested in. Moreover, clickthrough data can be collected easily without imposing extra burden on users, thus providing a low-cost means to capture user's interest.

Our approach consists of the following four major steps. First, when a user submits a query, concepts (i.e., important terms or phrases in web-snippets) and their relations are mined online from web-snippets to build a concept relationship graph. Second, clickthroughs are collected to predict user's conceptual preferences. Third, the concept

1. "Web-snippet" denotes the title, summary, and URL of a web page returned by search engines.

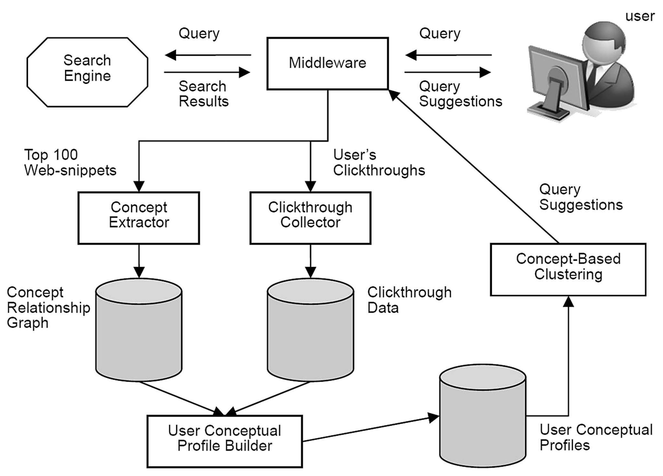


Fig. 1. The general process of concept-based clustering.

relationship graph together with the user's conceptual preferences is used as input to a concept-based clustering algorithm that finds conceptually close queries. Finally, the most similar queries are suggested to the user for search refinement. Fig. 1 shows the general process of our approach.

To evaluate the performance of our approach, we developed a Google middleware for clickthrough data collection.[2] Users were invited to test our middleware with test queries selected from a spectrum of topical categories. We evaluate the performance of our approach using the standard recall-precision measures. Beeferman and Berger's agglomerative clustering algorithm [11] (or simply called BB's algorithm in this paper) is used as the baseline to compare with our concept-based approach. Our experimental results show that the average precision at any recall level is better than the baseline method.

The main contributions of this paper are given as follows:

1. Most of the previous approaches on query clustering consider two different queries to be semantically similar if they lead to common clicks on the same pages. However, the chance for different queries leading to common clicks on the same URLs are rare in Web search engines (see Section 2 for more discussion). Based on this important observation, we propose to use concepts, not pages, as the common ground for relating semantically similar queries. That is, two queries are considered related if they lead to clicks on pages that share some common concepts, which are mined from the web-snippets in the search results.
2. To our knowledge, there is no previous study on the personalization of query suggestions. We propose a two-phase clustering method to cluster queries first within the scope of each user and then for the community.
3. We conduct experiments to evaluate different methods and show that our concept-based two-phase clustering method yields the best precision and recall.

2. The middleware approach is for facilitating experimentation. The techniques developed in this paper can be directly integrated into any search engine to provide personalized query suggestions.

The rest of this paper is organized as follows: In Section 2, we compare our method with other similar approaches. We also discuss some works related to concept mining. In Section 3, we review BB's algorithm, which is also an effective technique in personalized query clustering. In Section 4, our concept mining method for extracting concepts from web-snippets is presented. In Section 5, we adapt BB's algorithm to our concept-based approach. We further extend the concept-based BB's algorithm to a personalized clustering algorithm by utilizing the user concept preference profiles. Experimental results comparing BB's algorithm with our methods are presented in Section 6. Section 7 concludes this paper.

2 Related Work

Query clustering techniques have been developed in diversified ways. The very first query clustering technique comes from information retrieval studies [26]. Similarity between queries was measured based on overlapping keywords or phrases in the queries. Each query is represented as a keyword vector. Similarity functions such as cosine similarity or Jaccard similarity [26] were used to measure the distance between two queries. One major limitation of the approach is that common keywords also exist in unrelated queries. For example, the queries, "apple iPod" (an MP3 player) and "apple pie" (a dessert), are very similar since they both contain the keyword "apple." However, the queries are actually expressing two different search needs.

Chuang and Chien [14] proposed to cluster and organize users' queries into a hierarchical structure of topic classes. A Hierarchical Agglomerative Clustering (HAC) [25] algorithm is first employed to construct a binary-tree cluster hierarchy. The binary-tree hierarchy is then partitioned in order to create subhierarchies forming a multiway-tree cluster hierarchy like the hierarchical organization of Yahoo [6] and DMOZ [3].

Baeza-Yates et al. [10] proposed a query clustering method that groups similar queries according to their semantics. The method creates a vector representation Q for a query q, and the vector Q is composed of terms from the clicked documents of q. Cosine similarity is applied to the query vectors to discover similar queries. More recently, Zhang and Nasraoui [33] presented a method that discovers similar queries by analyzing users' sequential search behavior. The method assumes that consecutive queries submitted by a user are related to each other. The sequential search behavior is combined with a traditional content-based similarity method to compensate for the high sparsity of real query log data.

Recently, Beitzel et al. [12] proposed a query classification method that combines multiple classifiers. The method combines techniques from machine learning and computational linguistics. Their results were compared to those from the 2005 KDD Cup [5], showing that their combined

TABLE 1
The Clickthrough Data for the Query "Apple"

Links	Clicked	Web-Snippets for the Search Results
l_1	√	**Apple Hong Kong (http://www.appleclub.com.hk/)**
l_2		Apple Hong Kong - iPod + iTunes (http://www.appleclub.com.hk/ipod/)
l_3		Apple Daily (http://www.atnext.com)
l_4	√	**Apple (http://www.apple.com/)**
l_5		Apple - iPod + iTunes (http://www.apple.com/itunes/)
l_6		Apple Inc. - Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Apple_Computer)
l_7		Apple II series - Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Apple_II)
l_8		Apple .Mac (http://www.mac.com/)
l_9	√	**The Apple Store (US) (http://store.apple.com/)**
l_{10}		Apple - Support (http://www.info.apple.com/)

TABLE 2
Frequently Used Symbols

Symbol	Description
G	A bipartite graph
m	The number of iterations (i.e. merges) required for agglomerative clustering
n_b	The number of black vertices in G
n_w	The number of white vertices in G
$\|N\|_{max}$	The maximum number of neighbors of any vertex in G
$sim(x,y)$	Similarity between vertices x and y in G
$sim_R(t_i,t_j)$	Similarity between concepts t_i and t_j
$sf(t_i)$	Snippet frequency of the keyword/phrase t_i
$support(t_i)$	Interestingness of a particular keyword/phrase t_i with respect to the returned web-snippets arising from a query
$\|t_i\|$	The number of terms in the keyword/phrase t_i
upper bound	The upper bound for the number of operations required for agglomerative clustering

approach produced higher recall and smoother tradeoffs between recall and precision than any of the component approaches.

On Web search engines, clickthrough data is a kind of implicit feedback from users. Table 1 is an example clickthrough data for the query "apple," which shows the URLs returned from the search engine for the query and the URLs clicked on by the user. Clearly, it is a valuable resource for capturing the user's interest for building personalized Web search systems [7], [8], [17], [18], [21], [22], [24], [27], [28], [29]. Joachims [21] proposed a method that employs preference mining and machine learning to rerank search results according to user's personal preferences. Later on, Smyth et al. [27] suggested that user search behavior is repetitive and regular. They proposed to rerank search results such that the results that have been previously selected for a given query are promoted ahead of other search results. More recently, Deng et al. [17] proposed an algorithm that combines a spying technique together with a novel voting procedure to determine user preferences from the clickthrough data. Dou et al. [18] also performed a large-scale evaluation on different personalized search strategies, including clickthrough-based and profile-based personalization. They suggested that click-based personalization strategies perform consistently and considerably well when compared to profile-based methods (Table 2).

To resolve the disadvantage of keyword-based clustering methods, clickthrough data has been used to cluster queries based on common clicks on URLs. Beeferman and Burger [11] proposed an agglomerative clustering algorithm (i.e., BB's algorithm) to exploit query-document relationships from clickthrough data. Given a search engine log, BB's algorithm first constructs a bipartite graph with one set of vertices corresponding to queries, and another corresponding to documents. If a user clicks on a document, a link between the corresponding query and document is created on the bipartite graph. After the bipartite graph is obtained, the agglomerative clustering algorithm is used to obtain the clusters. The algorithm is content-independent in the sense that it exploits only the query-document links on the bipartite graph to discover similar queries and similar documents without examining the keywords in the queries or the documents. The details of the algorithm will be described in Section 3.

Wen et al. [31] proposed a clustering algorithm combining both query contents and URL clicks. They suggested that two queries should be clustered together, if they contain the same or similar terms, and lead to the selection of the same documents. However, since Web search queries are usually short and common clicks on documents are rare (see discussion below), Wen et al.'s method may not be effective for disambiguating Web queries. In contrast, our approach relates the queries with a set of extracted concepts in order to identify the precise semantics of the search queries.

One major problem with the clickthrough-based method is that the number of common clicks on URLs for different queries is limited. This is because different queries will likely retrieve very different result sets in very different ranking orders. Thus, the chance for the users to see the same results would be small, let alone clicking on them. It was reported that in a large clickthrough data set from a commercial search engine the chance for two random queries to have a common click is merely 6.38×10^{-5} [11]. The small number of common clicks leads to low recall.

To alleviate this problem, we introduce the notion of concept-based graphs by considering concepts extracted from web-snippets and adapt BB's method to this new context. In contrast to the existing methods, our approach provides effective personalization effect by using the concept preference profiles that are built upon the extracted concepts and clickthroughs. The use of concepts helps reduce the size of the resulted profiles, while retaining the accuracy and capability to capture user's interests.

Along the line of concept extraction from web-snippets, Koester [23] combined Web mining techniques and formal concept analysis to extract concepts from web-snippets and build a concept lattice capturing user's conceptual needs. However, it was not concerned with personalization. Xu et al. [32] proposed a method to extract concepts from users' browsed documents to create hierarchical concept profiles

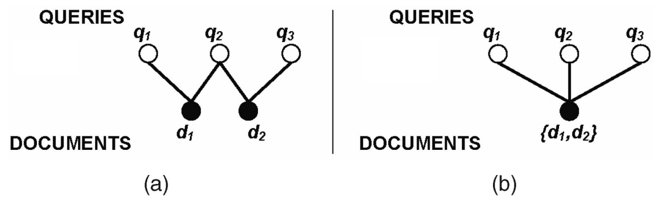


Fig. 2. (a) Queries q_1 and q_3 seem unrelated before document clustering. (b) After document clustering, queries q_1 and q_3 are then related to each other because they are both linked to the document cluster $\{d_1, d_2\}$.

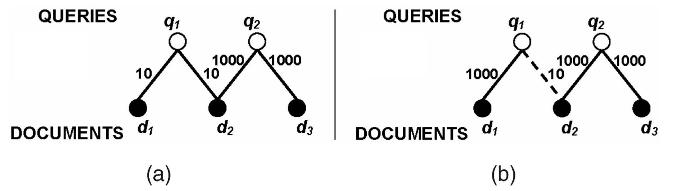


Fig. 3. (a) A bipartite graph without "noise." (b) A bipartite graph with a "noise" link, where the solid edges represent "real" links and the dash edge represents a "noise" edge.

for personalized search in a privacy-enhanced environment. Their method assumes that the system knows the documents that the user is interested in, instead of using clickthrough. Thus, their method is quite different from ours.

Another technique to discover related queries is query expansion. The aim of query expansion is to improve retrieval effectiveness by expanding the query with words or phrases to match additional documents. Cui et al. [15] proposed a query expansion method based on user interactions recorded in the clickthrough data. The method focuses on mining correlations between query terms and document terms by analyzing user's clickthroughs. Document terms that are strongly related to the input query are used together to narrow down the search.

3 BB's Graph-Based Clustering Algorithm

In BB's graph-based clustering [11], a query-page bipartite graph is first constructed with one set of the nodes corresponding to the set of submitted queries, and the other corresponding to the sets of clicked pages. If a user clicks on a page, a link between the query and the page is created on the bipartite graph. After obtaining the bipartite graph, an agglomerative clustering algorithm is used to discover similar queries and similar pages. During the clustering process, the algorithm iteratively combines the two most similar queries into one query node, then the two most similar pages into one page node, and the process of alternative combination of queries and pages is repeated until a termination condition is satisfied. The main reason for not clustering *all* the queries first and then *all* the pages next is that two queries may seem unrelated prior to page clustering because they link to two different pages but they may become similar to each other if the two pages have a high enough similarity to each other and are merged later. The example in Fig. 2 helps illustrate this scenario.

To compute the similarity between queries or documents on a bipartite graph, the algorithm considers the overlap of their neighboring vertices as defined in the following equation:

$$sim(x,y) = \begin{cases} \frac{|N(x)\cap N(y)|}{|N(x)\cup N(y)|}, & \text{if } |N(x)\cup N(y)| > 0, \\ 0, & \text{otherwise,} \end{cases} \qquad (1)$$

where $N(x)$ is the set of neighboring vertices of x, and $N(y)$ is the set of neighboring vertices of y. Intuitively, the similarity function formalizes the idea that x and y are similar if their respective neighboring vertices largely overlap and vice versa.

As discussed in Section 2, a problem of the BB's method is its low recall rate since the number of common clicks on the URLs is rather small. Another problem of the similarity function proposed by BB is that it cannot identify "noise" links in the clustering process. Consider the example shown in Fig. 3, where the number attached to a link is the total number of clicks on the document. In Fig. 3a, q_2 is a hot query, which generates 1,000 clicks for each of the documents d_2 and d_3, while q_1 is a cold query, which only generates 10 clicks for each of the documents d_1 and d_2. Even though the click distributions for q_1 and q_2 are different, we can see that d_1 and d_2 are both relevant to q_1 because the number of clicks on d_1 and the number of clicks on d_2 are roughly the same for q_1 (i.e., 10 clicks). Similarly, we can see that d_2 and d_3 are both relevant to q_2 because the number of clicks on d_2 and the number of clicks on d_3 are roughly the same for q_2 (i.e., 1,000 clicks). Thus, we conclude that q_1 and q_2 are similar queries because they share the common relevant document d_2. However, in Fig. 3b, d_2 cannot be considered relevant to q_1 because only a small fraction of the clicks (10 out of 1,010) supports that conclusion. Consequently, we cannot conclude that q_1 and q_2 are similar queries. BB's similarity function does not detect the "noise" link as shown Fig. 3b. It gives the same similarity score of 1/3 in both cases. To solve the problem, the following similarity function was proposed in our earlier work [13]:

$$sim(x,y) = \begin{cases} \frac{|L(x,y)|}{|L(x)\cup L(y)|}, & \text{if } |L(x)\cup L(y)| > 0, \\ 0, & \text{otherwise,} \end{cases} \qquad (2)$$

where $L(x,y)$ is the set of links connecting x and y to the same vertices, $L(x)$ and $L(y)$ are all the links connecting to x and y, respectively, and $|L(\cdot)|$ is the cardinality of $L(\cdot)$.

Applying the similarity function, we get a similarity score of $1{,}010/2{,}020 = 1/2$ for $sim(q_1, q_2)$ in Fig. 3a, and similarity score of $1{,}010/3{,}010 = 1/3$ for $sim(q_1, q_2)$ in Fig. 3b. Note that the score for $sim(q_1, q_2)$ in Fig. 3a is higher than that in Fig. 3b, because most people are selecting document d_1 in Fig. 3b, and the links between q_1 and d_2 can be considered as "noise." Therefore, it is reasonable to assign a lower score to $sim(q_1, q_2)$ in Fig. 3b. Using the noise-tolerant similarity function, the similarity between two vertices always lies between [0, 1]. The similarity for two vertices is 0, if they share no common neighbor, and the similarity between two vertices is 1, if they have exactly the same neighbor vertices.

It is noted that noise elimination by itself is a difficult problem since it requires complex inference rules to distinguish the informative from the erroneous clicks. Since the noise-tolerant version has been shown to be superior to the original version [13] and we are not aware of any better methods, in the rest of this paper, BB's algorithm refers to this improved version of similarity function.

4 Concept Extraction

Before explaining our concept-based clustering method, we first describe our concept extraction method, which is composed of the following three basic steps: 1) extracting concepts using the web-snippets returned from the search engine, 2) mining concept relations, and 3) creating a user concept preference profile using the extracted concepts, concept relations, and user's clickthroughs.

4.1 Concept Extraction Using Web-Snippets

Our concept extraction method is inspired by the well-known problem of finding frequent item sets in data mining [9], [19]. When a user submits a query to the search engine, a set of web-snippets are returned to the user for identifying the relevant items. We assume that if a keyword or a phrase appears frequently in the web-snippets of a particular query, it represents an important concept related to the query because it coexists in close proximity with the query in the top documents. We use the following support formula for measuring the interestingness of a particular keyword/phrase t_i with respect to the returned web-snippets arising from a query q:

$$support(t_i) = \frac{sf(t_i)}{n} \cdot |t_i|, \tag{3}$$

where n is the total number of web-snippets returned, $sf(t_i)$ is the snippet frequency of the keyword/phrase t_i (i.e., the number of web-snippets containing t_i), and $|t_i|$ is the number of terms in the keyword/phrase t_i. For simplicity, we omit q in the above expression if no ambiguity arises.

To extract concepts for a query q, we first extract all the keywords and phrases from the web-snippets returned by the query. After obtaining a set of keywords/phrases (t_i), we compute the support for all t_i ($support(t_i)$). If the support of a keyword/phrase t_i is bigger than the threshold s ($support(t_i) > s$), we would treat t_i as a concept for the query q. Table 3 illustrates the extracted concepts for the query $q =$ "apple".

4.2 Mining Concept Relations

To find relations between concepts, we apply a well-known signal-to-noise ratio formula from data mining [16] to establish similarity between terms t_1 and t_2. The similarity value of Church and Hanks' formula always lies between [0, 1] and thus can be used directly in step 3:

$$sim(t_1, t_2) = \frac{n \cdot df(t_1 \cup t_2)}{df(t_1) \cdot df(t_2)} \Big/ \log n, \tag{4}$$

where n is the number of documents in the corpus, $df(t_1 \cup t_2)$ is the joint document frequency of t_1 and t_2, and $df(t)$ is the document frequency of the term t.

In our context, two concepts t_i, t_j could coexist in a web-snippet in the following situations: 1) t_i and t_j coexist in the title, 2) t_i and t_j coexist in the summary, or 3) t_i exists in the title, while t_j exists in the summary (or vice versa).

TABLE 3
Extracted Concepts for the Query "Apple"

Concept t_i	$support(t_i)$	Concept t_i	$support(t_i)$
mac	0.1	macintosh	0.05
ipod	0.1	tour	0.05
iphone	0.1	slashdot apple	0.04
hardware	0.09	picture	0.04
software	0.09	apple ii	0.04
big apple	0.08	apple variety	0.04
apple store	0.06	music	0.04
mac os	0.06	farm market	0.04
apple orchard	0.06	apple grower	0.04
apple valley	0.06	gift shop	0.04
apple and macintosh	0.06	apple farm	0.04
apple blossom festival	0.06		

Therefore, we modify Church and Hanks' formula for the three different cases in our context as follows:

$$sim_R(t_i, t_j) = sim_{R,title}(t_i, t_j) + sim_{R,summary}(t_i, t_j) + sim_{R,other}(t_i, t_j), \tag{5}$$

where $sim_R(t_i, t_j)$ is the similarity between concepts t_i and t_j, which is composed of $sim_{R,title}(t_i, t_j)$, $sim_{R,summary}(t_i, t_j)$, and $sim_{R,other}(t_i, t_j)$ as follows:

$$sim_{R,title}(t_i, t_j) = \alpha \cdot \log\left(\frac{n \cdot sf_{title}(t_i \cup t_j)}{sf_{title}(t_i) \cdot sf_{title}(t_j)}\right) \Big/ \log n, \tag{6}$$

$$sim_{R,summary}(t_i, t_j) = \alpha \cdot \log\left(\frac{n \cdot sf_{summary}(t_i \cup t_j)}{sf_{summary}(t_i) \cdot sf_{summary}(t_j)}\right) \Big/ \log n, \tag{7}$$

$$sim_{R,other}(t_i, t_j) = \left(\alpha \cdot \log \frac{n \cdot sf_{other}(t_i \cup t_j)}{sf_{other}(t_i) \cdot sf_{other}(t_j)}\right) \Big/ \log n, \tag{8}$$

where n is the total number of web-snippets returned, $sf_{title}(t_i \cup t_j)$ is the joint snippet frequency of concepts t_i and t_j in document titles, $sf_{title}(t)$ is the snippet frequency of concept t in document titles, $sf_{summary}(t_i \cup t_j)$ is the joint snippet frequency of t_i and t_j in document summaries, $sf_{summary}(t)$ is the snippet frequency of concept t in document summaries, $sf_{other}(t_i \cup t_j)$ is the joint snippet frequency of concept t_i in a document title and t_j in the document's summary (or vice versa), and $sf_{other}(t)$ is the snippet frequency of concept t in either document summaries or document titles.

Using the extracted concepts and concept relations, we can create a concept relationship graph with the extracted concepts as nodes and mined concept relations as links. Fig. 4a shows a concept preference graph for the query $q =$ "apple". A link is created between concept t_i and t_j, if their similarity, $sim_R(t_i, t_j)$, is greater than zero. The strength of each link is determined by $sim_R(t_i, t_j)$, which is the similarity between concepts t_i and t_j.

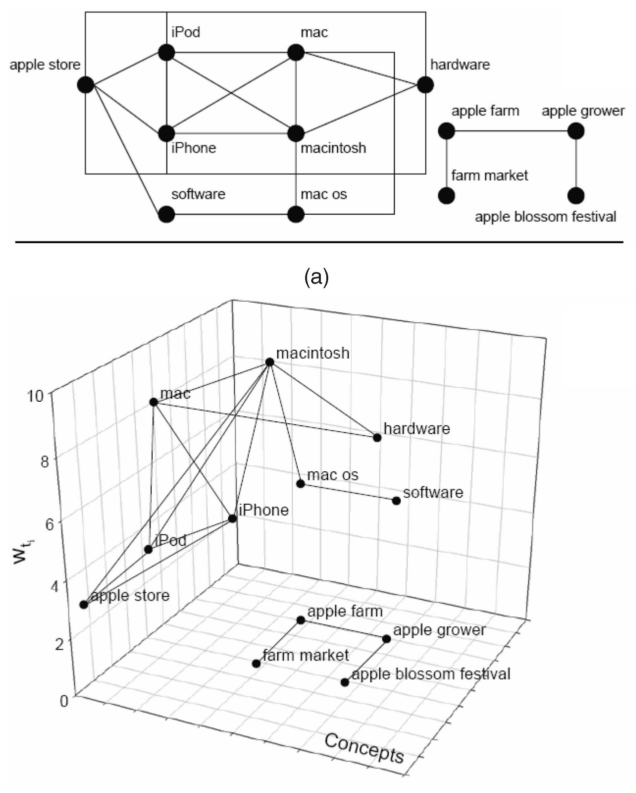


Fig. 4. (a) A concept relationship graph for the query "apple" derived without incorporating user clickthroughs. (b) A concept preference profile constructed using the user clickthroughs and the concept relationship graph in (a). w_{ti} is the interestingness of the concept t_i to the user. More clicks on a concept gradually increase the interestingness w_{ti} of the concept.

4.3 Creating User Concept Preference Profile

The concept relationship graph is first derived without taking user clickthroughs into account. Intuitively, the graph shows the possible concept space arising from user's queries. The concept space, in general, covers more than what the user actually wants. For example, when the user searches for the query "apple," the concept space derived from the web-snippets contains concepts such as "ipod," "iphone," and "recipe." If the user is indeed interested in the concept "recipe" and clicks on pages containing the concept "recipe," the clickthroughs should gradually favor the concept "recipe" and its neighborhood (by assigning higher weights to the nodes), but the weights of the unrelated concepts such as "iphone," "ipod," and their neighborhood should remain zero. Therefore, we propose the following formulas to capture user's interestingness w_{ti} on the extracted concepts t_i when a clicked web-snippet s_j, denoted by $\mathrm{click}(s_j)$, is found as follows:

$$click(s_j) \Rightarrow \forall t_i \in s_j, w_{t_i} = w_{t_i} + 1, \tag{9}$$

$$click(s_j) \Rightarrow \forall t_i \in s_j, w_{t_j} = w_{t_j} + sim_R(t_i, t_j) \text{ if } sim_R(t_i, t_j) > 0, \tag{10}$$

where s_j is a web-snippet, w_{ti} is the interestingness weight of the concept t_i, and t_j is the neighborhood concept of t_i.

When a user clicks on s_j, the weight of concepts t_i appearing in s_j is incremented by 1 to reflect the user's interestingness on the concepts embedded in the clicked page s_j. For other concepts that are related to the clicked concepts on the concept relationship graph, they are incremented according to the similarity score given in (4), which is normalized to the range [0, 1]. Therefore, if a concept is closely related to the clicked concept, it is incremented to a higher value (which could be as close to 1 as the clicked concepts). Otherwise, it is only incremented by a small fraction (close to 0). By imposing user's interestingness on the concepts, a concept preference profile with respect to the input query is created. Fig. 4b shows an example of concept preference profile in which the user is interested in information about "apple macintosh." w_{ti} in Fig. 4b represents the interestingness of the concepts to the user. The values of w_{ti} for "macintosh" and "mac" are highest because the users have interest in them (and the values of w_{ti} are incremented using (9)). Indirectly, the values of w_{ti} for "mac os," "software," "apple store," "iPod," "iPhone," and "hardware" are increased because they are related to "apple macintosh" and thus incremented using (10). Finally, the weights of the concepts about "apple" as fruit are not changed. As a result, the concepts formed two clusters representing the user concept preference profile.

5 Concept-Based Clustering

Using the concepts extracted from web-snippets, we propose two concept-based clustering methods. We first extend BB's algorithm to a concept-based algorithm in Section 5.1. In Section 5.2, the concept-based algorithm is further enhanced to achieve effective personalized clustering.

5.1 Clustering on Query-Concept Bipartite Graph

We now describe our concept-based algorithm (i.e., BB's algorithm using query-concept bipartite graph) for clustering similar queries. Similar to BB's algorithm, our technique is composed of two steps: 1) Bipartite graph construction using the extracted concepts and 2) agglomerative clustering using the bipartite graph constructed in step 1.

Using the extracted concepts and clickthrough data, the first step of our method is to construct a query-concept bipartite graph, in which one side of the vertices correspond to unique queries, and the other corresponds to unique concepts. If a user clicks on a search result, concepts appearing in the web-snippet of the search result are linked to the corresponding query on the bipartite graph. Algorithm 1 shows the first step of our method.

```
Algorithm 1 Bipartite Graph Construction
Input: Clickthrough data $CT$, Extracted Concepts $E$
Output: A Query-Concept Bipartite Graph $G$

1: Obtain the set of unique queries $Q = \{q_1, q_2, q_3 \ldots\}$ from $CT$
2: Obtain the set of unique concepts $C = \{c_1, c_2, c_3 \ldots\}$ from $E$
3: $Nodes(G) = Q \cup C$ where $Q$ and $C$ are the two sides in $G$
4: If the web-snippet $s$ retrieved using $q_i \in Q$ is clicked by
```

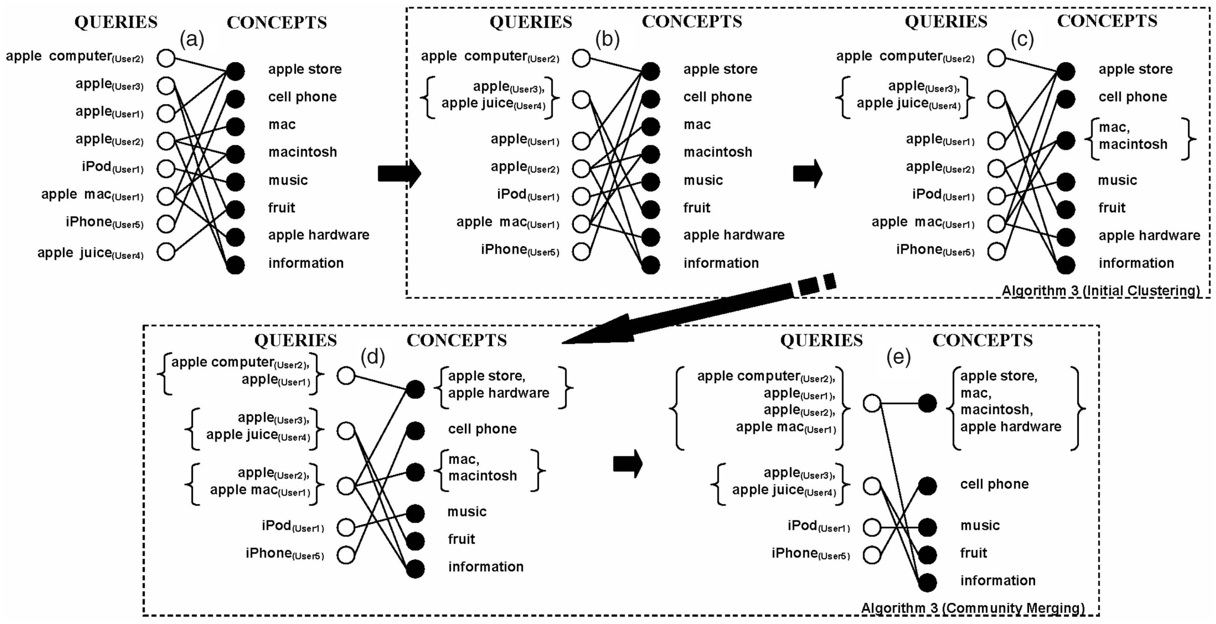


Fig. 5. Performing personalized concept-based clustering algorithm on a small set of clickthrough data. Starting from top left: (a) The original bipartite graph. (b), (c) Initial clustering. (d), (e) Community merging.

a user, create an edge $e = (q_i, c_j)$ in G, where c_j is a concept appearing in s.

After the bipartite graph is constructed, the agglomerative clustering algorithm is applied to obtain clusters of similar queries and similar concepts. The noise-tolerant similarity function (recall (2)) is used for finding similar vertices on the bipartite graph G. The agglomerative clustering algorithm would iteratively merge the most similar pair of white vertices and then merge the most similar pair of black vertices and so on. We present the details in Algorithm 2.

```
Algorithm 2 Agglomerative Clustering
Input: A Query-Concept Bipartite Graph $G$
Output: A Clustered Query-Concept Bipartite Graph $G^c$

1: Obtain the similarity scores for all possible pairs of queries in $G$ using the noise-tolerant similarity function given in (2).
2: Merge the pair of queries $(q_i, q_j)$ that has the highest similarity score.
3: Obtain the similarity scores for all possible pairs of concepts in $G$ using the noise-tolerant similarity function given in (2).
4: Merge the pair of concepts $(c_i, c_j)$ that has the highest similarity score.
5. Unless termination is reached, repeat steps 1-4.
```

The terminating condition for BB's algorithm is when all connected components in G^c satisfy the following conditions:

$$\underset{q_i,q_j\in Q}{max}\ sim(q_i, q_j) = 0 \quad \text{and} \quad \underset{c_i,c_j\in C}{max}\ sim(c_i, c_j) = 0.$$

However, this terminating condition possibly generates a single big cluster of queries and a single big cluster of concepts because having the similarity threshold set to zero means that two queries (concepts) would be assigned to the same cluster even if they have only a tiny fraction of overlapping concepts (queries). To resolve this problem, we apply higher similarity thresholds, which have been observed from our experiments to yield high precision and recall:

$$\underset{q_i,q_j\in Q}{max}\ sim(q_i, q_j) = 0.18 \quad \text{and} \quad \underset{c_i,c_j\in C}{max}\ sim(c_i, c_j) = 0.18.$$

5.2 Personalized Concept-Based Clustering

We now explain the essential idea of our personalized concept-based clustering algorithm with which ambiguous queries can be clustered into different query clusters. Personalized effect is achieved by manipulating the user concept preference profiles in the clustering process.

In contrast to BB's agglomerative clustering algorithm, which represents the same queries submitted from different users by one query node, we need to consider the same queries submitted by different users separately to achieve personalization effect. In other words, if two given queries, whether they are identical or not, mean different things to two different users, they should not be merged together because they refer to two different sets of concepts for the two users.

Therefore, we treat each individual query submitted by each user as an individual vertex in the bipartite graph by labeling each query with a user identifier. Moreover, concepts appearing in the web-snippet of the search result with interestingness weights greater than zero in the concept preference profile are linked to the corresponding query on the bipartite graph. An example is shown in Fig. 5a. We can see that the query "apple" submitted by users User1 and User3 become two vertices "apple$_{(\text{User1})}$"

and "apple$_{(\text{User3})}$." If User1 is interested in the concept "apple store," as recorded in the concept preference profile, a link between the concept "apple store" and the query "apple$_{(\text{User1})}$" would be created. On the other hand, if User3 is interested in the concept "fruit," a link between the concept "fruit" and "apple$_{(\text{User3})}$" would be created.

After the personalized bipartite graph is created, our initial experiments revealed that if we apply BB's algorithm directly on the bipartite graph, the query clusters generated will quickly merge queries from different users together, thus losing the personalization effect. We found that identical queries, though issued by different users and having different meanings, tend to have some generic concept nodes such as "information" in common, e.g., "apple$_{(\text{User1})}$" and "apple$_{(\text{User3})}$" both connect to the "information" concept node in Fig. 5a. Thus, these query nodes will likely be merged in the first few iterations and cause more queries from different users to be merged together in subsequent iterations. Considering Fig. 5a again, if "apple$_{(\text{User1})}$" and "apple$_{(\text{User3})}$" are merged, the next iteration will merge the concept nodes "apple store," "fruit," and "information." When the clustering algorithm goes further, queries across users will be further clustered together. At the end, the resulting query clusters have no personalization effect at all.

To resolve the problem, we divide clustering into two steps. In the initial clustering step, an algorithm similar to BB's algorithm is employed to cluster all the queries, but it would not merge identical queries from different users. After obtaining all the clusters from the initial clustering step, the community merging step is employed to merge query clusters containing identical queries from different users. We can see from Fig. 5d that "apple$_{(\text{User1})}$" and "apple$_{(\text{User3})}$" belong, correctly, to different clusters. We will see further in Section 6.3 that the initial clustering step is able to generate high precision rate because it preserves the preference of each user, while the community merging step is able to improve the recall rate because of the collaborative filtering effect.

Algorithm 3 shows the details of the personalized clustering algorithm. Similar to the BB's algorithm, a query-concept bipartite graph is created as input for the clustering algorithm. The bipartite graph construction algorithm is similar to Algorithm 1, except each individual query submitted by each user is treated as an individual vertex in the bipartite graph.

```
Algorithm 3 Personalized Agglomerative Clustering
Input: A Query-Concept Bipartite Graph $G$
Output: A Personalized Clustered Query-Concept
Bipartite Graph $G^p$

// Initial Clustering
1: Obtain the similarity scores in $G$ for all possible pairs of queries using the noise-tolerant similarity function given in (2).
2: Merge the pair of most similar queries $(q_i, q_j)$ that does not contain the same queries from different users.
3: Obtain the similarity scores in $G$ for all possible pairs of concepts using the noise-tolerant similarity function given in (2).
4: Merge the pair of concepts $(c_i, c_j)$ having highest similarity score.
5. Unless termination is reached, repeat steps 1-4.
// Community Merging
6. Obtain the similarity scores in $G$ for all possible pairs of queries using the noise-tolerant similarity function given in (2).
7. Merge the pair of most similar queries $(q_i, q_j)$ that contains the same queries from different users.
8. Unless termination is reached, repeat steps 6 and 7.
```

Initial clustering (i.e., steps 1-5 of Algorithm 3) is similar to BB's agglomerative algorithm as already discussed in Section 5.1. However, queries from different users are not allowed to be merged in initial clustering. Figs. 5b and 5c show examples of query and concept merging, respectively. Fig. 5d illustrates the result of initial clustering. In community merging (i.e., steps 6-8 of Algorithm 3), query clusters containing identical queries from different users are compared for merging. Figs. 5d and 5e show an example of query cluster merging. The query clusters $\{\text{apple computer}_{(\text{User2})}, \text{apple}_{(\text{User1})}\}$ and $\{\text{apple}_{(\text{User2})}$ and $\text{apple mac}_{(\text{User1})}\}$ both contain the query "apple" and are leading to the same concept "apple store." Therefore, they are merged in community merging as one big cluster.

Good timing to start community merging is important for the success of the algorithm. If we stop initial clustering too early (i.e., not all clusters are well formed), community merging merges all the identical queries from different users first and thus generates a single big cluster without much personalization effect. However, if we stop initial clustering too late (i.e., clusters are being overly merged in this case), the low precision rate generated by initial clustering would not be improved by community merging. To obtain the optimal results in our experiments, we use the following terminating conditions for initial clustering (*i-clustering*) and community merging (*c-merging*) in Algorithm 3. These parameters are empirically investigated in our experiment. We will further justify our choice using Table 10 in Section 6.3:

$$\underset{q_i,q_j \in Q}{max_{i-clustering}} sim(q_i, q_j) = 0.29 \quad \text{and}$$
$$\underset{c_i,c_j \in C}{max_{i-clustering}} sim(c_i, c_j) = 0.29,$$
$$\underset{q_i,q_j \in Q}{max_{c-merging}} sim(q_i, q_j) = 0.39 \quad \text{and}$$
$$\underset{c_i,c_j \in C}{max_{c-merging}} sim(c_i, c_j) = 0.39.$$

The query clusters outputted by the algorithm are shown in Fig. 5e. We assume in this example that the links between the generic concept nodes, "information," and the two query clusters are weak and the terminating similarity is able to prevent the merging of the query clusters about "apple computer" and "apple juice." We can see in the resulting clusters that User1 and User2 both submit the query "apple" in order to seek information about "apple computer," while User3 submits the query "apple" to look for information about "apple juice." In this example, even the query "apple" submitted by User1, User2, and User3 appear to be the same, the algorithm can successfully

TABLE 4
Categories of the Test Queries

Category	Description	Category	Description
1	Cooking	6	Computer Programming
2	Dining	7	Computer Gaming
3	Internet Shopping	8	Music
4	Traveling	9	Computer Science Research
5	Automobile Repairing	10	Computer Hardware

differentiate them to archive personalization effect according to individual user conceptual preferences. Finally, we can see that queries about "apple computer" (e.g., "apple mac," "apple computer") are suggested to User1 and User2, while queries about "apple juice" (e.g., "apple juice") are suggested to User3.

6 Experimental Results

In this section, we evaluate the performance of the proposed clustering methods for obtaining related queries using user clickthroughs. In Section 6.1, we first describe the experimental setup for collecting the required clickthrough data. In Section 6.2, we compare the performance of BB's algorithm using query-URL, query-word, and query-concept bipartite graphs (or simply called the QU, QW, and QC methods). In Section 6.3, we evaluate the effectiveness of our proposed personalized concept-based clustering (or simply called the P-QC method). In Section 6.4, we discuss the algorithmic complexities based on the related parameters.

6.1 Experimental Setup

To collect the clickthrough data to evaluate our proposed methods, we implemented a Google middleware to track user clicks. Google[3] was chosen as a common basis for comparing the performance of the methods under evaluation.

We invited 40 students from our department to use the middleware to search 200 given test queries, which are accessible in [2]. To avoid any bias, the test queries are randomly selected from 10 different categories and submitted to Google without any modification by the middleware. Table 4 shows the topical categories in which the queries we have chosen. When a query is submitted to the middleware, the top 100 search results from Google are retrieved, and the web-snippets of the search results are displayed to the users. Since most users would examine only the top 10 results, our concept extraction method, digging deep into the first 100 results, will discover concepts related to the query that would otherwise be missed by the users.

The extracted concept relationship graph is then stored in our database. If a user clicks on one of the web-snippets of the returned results, the user's clickthrough together with his/her concept preference profile are updated as discussed in Section 4.3. The threshold s for concept mining was set to 0.03 and the threshold for establishing concept relations (as specified in (10)) is set to zero. We chose these small thresholds so that as many concepts as possible are mined. The quality of the query suggestions is then relied more on the clustering algorithms, which are the main focus of this paper.

3. Google is one of the most popular commercial search engines. If a different search engine is used, we expect the absolute performances of the methods under evaluation to be different but their relative performances remain the same.

TABLE 5
Statistics of the Clickthrough Data Collected in the First Experiment

Statistics	
Number of users	30
Number of queries assigned to each user	5
Number of test queries	150
Number of unique queries	150
Maximum number of retrieved URLs for a query	100
Maximum number of extracted concepts for a query	217
Maximum number of extracted words for a query	1,093
Number of URLs retrieved	14,880
Number of unique URLs retrieved	12,430
Number of concepts retrieved	13,321
Number of unique concepts retrieved	6,008
Number of words retrieved	117,924
Number of unique words retrieved	21,920

In the first experiment (which will be described in Section 6.2), 30 students were asked to search the 150 test queries, all of which have unambiguous meanings (e.g., "apple pie" and "cheese cake"). The 150 test queries are separated into 10 predefined clusters (e.g., the queries "apple pie," "cheese cake," and "brownies" belong to the cluster about dessert recipes). The users were asked to click on the web-snippets of the returned results that are relevant to the queries. The clickthrough data collected are used to measure the performance of the concept-based clustering method as discussed in Section 5.1. Table 5 shows the statistics of our collected clickthrough data for this experiment.

In the second experiment (which will be described in Section 6.3), 10 students were asked to search using another 50 test queries. Some of the test queries are intentionally designed to have ambiguous meanings (e.g., the query "Canon" could mean a digital camera or a printer). The 50 test queries are separated into eight predefined clusters. Some of the queries could possibly exist in more than one cluster (e.g., the query "Canon" could belong to the cluster about digital cameras or the cluster about printers). Each user is assigned with one of the information seeking tasks shown in Table 6. The users are then asked to click on the web-snippets of the returned results that are both relevant to the queries and their information needs. The clickthrough data collected are used to measure the performance of the personalized concept-based clustering method as discussed in Section 5.2. Table 7 shows the statistics of our collected clickthrough data for this experiment.

6.2 Comparing QU, QW, and QC Methods

We now discuss the result of the first experiment, which compares the performance of QU, QW, and QC methods.

TABLE 6
User's Information Needs for the Second Experiment

User Group	Information Needs
1	Purchase of digital cameras
2	Purchase of printers
3	Information on camera films
4	Information on dessert cooking recipes
5	Purchase of clothes
6	Download of Mac software
7	Purchase of Macintosh
8	Purchase of iPod

TABLE 7
Statistics of the Clickthrough Data Collected for the Second Experiment

Statistics	
Number of users	10
Number of queries assigned to each user	5
Number of test queries	50
Number of unique queries	38
Maximum number of retrieved URLs for a query	100
Maximum number of extracted concepts for a query	168
Maximum number of extracted words for a query	938
Number of URLs retrieved	4,962
Number of unique URLs retrieved	3,239
Number of concepts retrieved	4,130
Number of unique concepts retrieved	1,971
Number of words retrieved	38,831
Number of unique words retrieved	8,891

QU method is the original input of BB's algorithm, which serves as a baseline for comparison. QW method uses query-word bipartite graph, which is similar to the query-concept bipartite graph in that they are both constructed using Algorithm 1. The difference is that the former contains all words (excluding stopwords) from the web-snippets and the latter contains the extracted concepts. QW and QC methods are necessary, since they allow us to study the benefits of concept extraction. The three methods are also employed to cluster the collected data. The results are compared to our predefined clusters for precision and recall. Given a query q and its corresponding query cluster $\{q_1, q_2, q_3 \ldots\}$ generated by a clustering algorithm, the precision and recall are computed using the following formulas:

$$precision(q) = \frac{|Q\_relevant \cap Q\_retrieved|}{|Q\_retrieved|}, \tag{11}$$

$$recall(q) = \frac{|Q\_relevant \cap Q\_retrieved|}{|Q\_relevant|}, \tag{12}$$

where $Q\_relevant$ is the set of queries that exist in the predefined cluster for q, and $Q\_retrieved$ is set of the related queries $\{q_1, q_2, q_3 \ldots\}$ generated by the algorithm. The precision and recall values from all queries are averaged

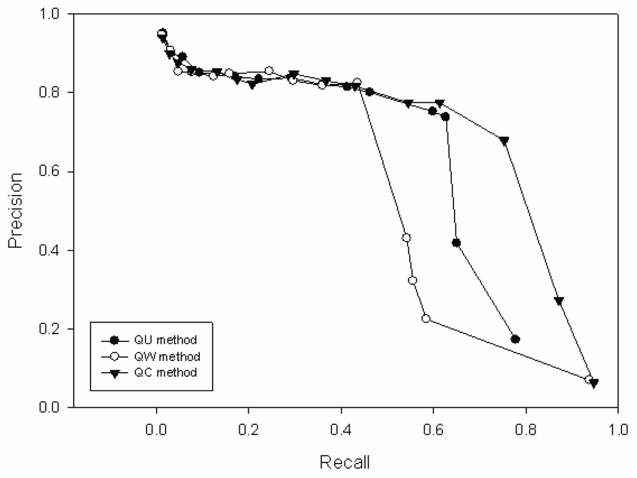


Fig. 6. Precision versus recall when performing QU, QW, and QC methods.

for plotting the precision-recall figures. The performance of the three methods is compared using precision-recall figures and best F-measure values.

Fig. 6 shows the precision-recall figures for QU, QW, and QC methods. We observe that QC method yields better recall rate than QU method (i.e., the original BB's algorithm), while preserving high precision rates. This can be attributed to the fact that the average number of overlapping URLs between queries is only 16.3 according to the statistics in Table 5, whereas the average number of overlapping concepts between the queries is 48.8, which is much higher than the URL overlap rate. As a result, related queries that cannot be discovered by URL overlap can be brought together by our QC method, thus improving the recall rate. The effect of high concept overlap rate is also apparent in Fig. 6, which shows that the recall of QU method can only go up to around 0.8, while QW and QC methods can go beyond 0.9. Note that QU method can yield high precision rate because of the valuable URL overlaps between queries. However, QC method benefits both precision and recall compared to QU method, showing that the use of extracted concepts is much better for finding similar queries.

We also observe that QW method performs the worst among the three methods because common nonstopwords such as "discussion," "information," and "news" bring unrelated queries together, thus lowering both the precision and recall rate. The main difference between QW and QC methods is the availability of concept extraction. Intuitively, QC method outperforms QW method because the concept extraction process can successfully eliminate unrelated common words within web-snippets.

Figs. 7 and 8 show the change of precision and recall, respectively, for the three clustering methods. In Fig. 7, when the cutoff similarity score is around 0.3, the precision obtained using QU method is very close to that of QC method, which is much better than the precision obtained using QW method. In Fig. 8, at the same cutoff similarity score, the recall obtained using QU method is close to zero, which is much lower compared to the recalls obtained using QW and QC methods. We can

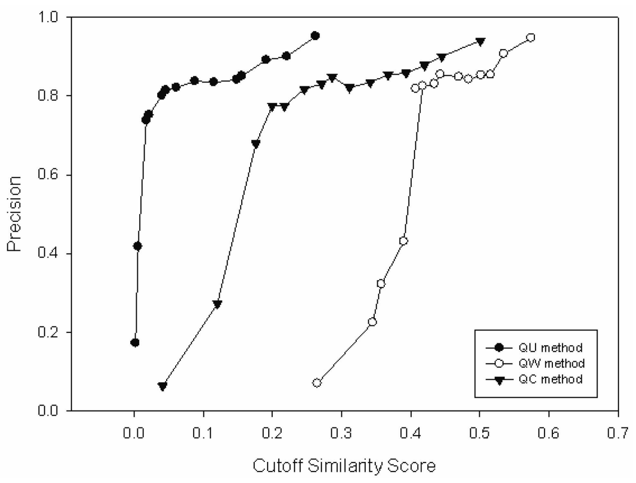


Fig. 7. Change of precision when performing QU, QW, and QC methods.

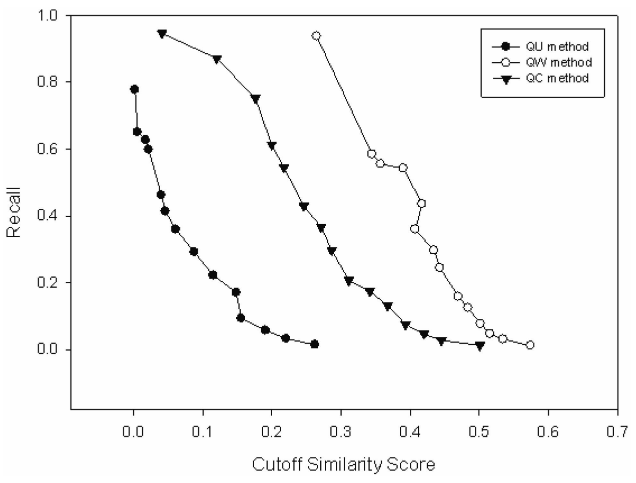


Fig. 8. Change of recall when performing QU, QW, and QC methods.

easily see from Figs. 7 and 8 that QC method is able to generate good recall, while achieving a precision comparable to that of QU method.

We observe that the three methods are able to achieve their optimal precision/recall at different cutoff similarity scores. To obtain and compare the best F-measures [30] (i.e., evenly weighted harmonic means of precisions and recalls) for the three different methods, the following three terminating strategies are used:

$$max_{URL,\ q_i,q_j \in Q}\ sim(q_i,q_j) = 0.017 \text{ and } max_{URL,\ c_i,c_j \in C}\ sim(c_i,c_j) = 0.017,$$

$$max_{word,\ q_i,q_j \in Q}\ sim(q_i,q_j) = 0.39 \text{ and } max_{word,\ c_i,c_j \in C}\ sim(c_i,c_j) = 0.39,$$

$$max_{concept,\ q_i,q_j \in Q}\ sim(q_i,q_j) = 0.18 \text{ and } max_{concept,\ c_i,c_j \in C}\ sim(c_i,c_j) = 0.18.$$

The F-measure, F, is defined by the following formula:

$$F = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)}. \tag{13}$$

Table 8 shows the best F-measure values for the QU, QW, and QC methods. From the results, we can conclude that query clusters obtained using QC method are much more accurate compared to those obtained from QU and QW methods.

TABLE 8
Best F-Measure Values of QU, QW, and QC Methods for the First Experiment

	Best F-Measure Values		
	Precision	*Recall*	*F-measure*
QU method	0.737	0.640	0.685
QW method	0.775	0.493	0.603
QC method	0.768	0.727	0.747

6.3 Personalized Concept-Based Clustering

In the second experiment, QU, QW, QC, and P-QC methods are employed to cluster queries that are intentionally designed to have ambiguous meanings. Again, the results are compared to our predefined clusters in terms of precision and recall. We analyze the performance of P-QC method using precision-recall figures and best F-measure values.

Fig. 9 shows the precision-recall figures of P-QC methods. The solid line is the precision-recall graph if only initial clustering is performed. We can observe that recall is max out at 0.62. The other three lines illustrate how community merging can further improve recall beyond the limit of initial clustering. We observe that the timing for switching from initial clustering to community merging is very important to the precision and recall of the final query clusters. When initial clustering is stopped too early (see the dark-triangle and white-triangle graphs in Fig. 9), initial clustering achieves high precision and low recall, as can be expected, but community merging fails to improve the recall—it drags down precision without improving recall. The drop of precision is due to easy merging of identical queries from different users, thus generating a single big cluster without personalization benefit.

When initial clustering is switched to community merging at the optimal point (see the white-circle graph in Fig. 9), community merging clearly boosts up the precision-recall envelop, which means that both precision and recall achieved in initial clustering are improved. This

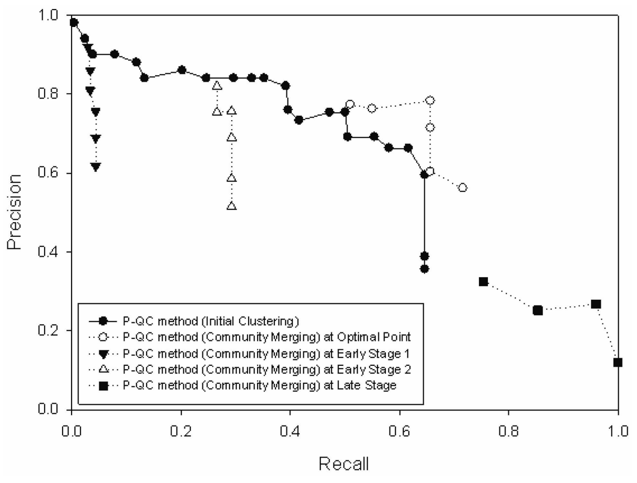


Fig. 9. Precision versus recall when performing P-QC method. The solid line represents the results obtained from initial clustering of Algorithm 3, and the dash lines represent the results obtained from community merging of Algorithm 3.

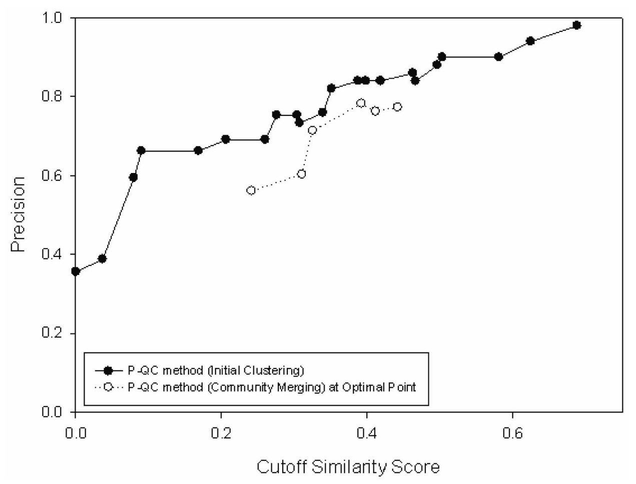


Fig. 10. Change of precision when performing P-QC method. The solid line represents the results obtained from initial clustering of Algorithm 3, and the dash lines represent the results obtained from community merging of Algorithm 3.

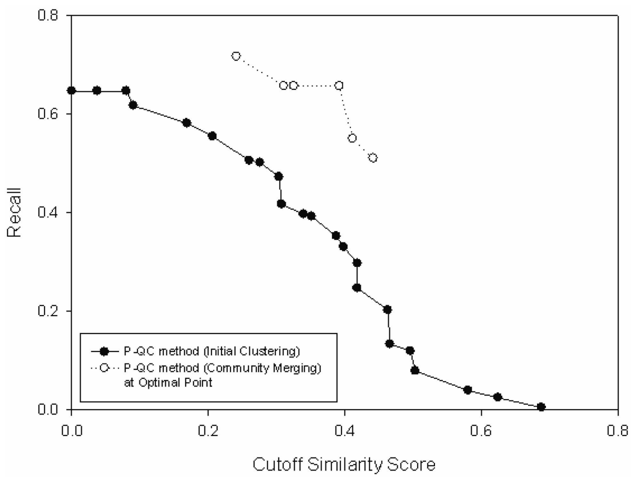


Fig. 11. Change of recall when performing P-QC method. The solid line represents the results obtained from initial clustering of Algorithm 3, and the dash lines represent the results obtained from community merging of Algorithm 3.

indicates that community merging is successful in choosing query clusters with identical queries from different users for merging.

Finally, when the switching from initial clustering to community merging is performed later than the optimal point, we can observe that recall is increased but precision is lowered, which is a typical phenomenon resulted from the conflicting nature of precision and recall. The behavior is due to the fact that overly merged clusters from initial clustering are further merged in community merging (see the dark-box graph in Fig. 9), thus further lowering the low precision generated in initial clustering. Although community merging at late stage generates low precision, it extends the recall from 0.65 obtained by initial clustering to 1.0 (i.e., at $\text{precision} = 0.14$ in Fig. 9).

Figs. 10 and 11 show the change of precision and recall when performing P-QC method. In Fig. 10, we observe that the precisions generated by community merging are slightly lower than those generated by initial clustering because some unrelated queries can be wrongly merged in community merging. In Fig. 11, we observe that the recalls generated by community merging are much higher than those generated by initial clustering because community merging can successfully merge conceptually related clusters together. We can easily see from Figs. 10 and 11 that only a small fraction of precision is used to trade for a much better recall in community merging.

In order to further justify our choice of the parameters used in P-QC, we show in Table 10 different terminating values near the optimal point for initial clustering and community merging in the second experiment. Two best cutoff values listed in the fifth row of the table (approximately 0.29 and 0.39) are used for defining the terminating conditions of initial clustering ($\textit{i-clustering}$) and community merging ($\textit{c-merging}$) in Algorithm 3 in order to obtain the best results. (Recall the terminating conditions for personalized agglomerative clustering given in Section 5.2.)

The best F-measure value obtained is shown in Table 9. We observe that the best F-measure value for P-QC method is better than those obtained using QU, QW, and QC methods. Therefore, we conclude that query clusters obtained from P-QC method are more accurate compared to those obtained from QU, QW, and QC methods, and that P-QC method can effectively group similar queries together even when the queries are ambiguous.

Table 11 shows some of the query clusters generated by Algorithm 3 on the collected data. In Table 11, User7, User8, and User9 have submitted the query "apple" to our middleware. User7 gets query suggestions about "macintosh's software" (Cluster12) because he/she is interested in concepts on "macintosh's software." User8 gets query suggestions about "macintosh hardware" (Cluster11), and User9 gets query suggestions about "iPod" (Cluster10). By using P-QC method, query suggestions according to individual user's conceptual preferences can be found effectively. Moreover, the algorithm yields high precision accuracy and better recall rate, clearly outperforming BB's algorithm.

6.4 Data Size

Clustering web pages by content requires manipulating a staggeringly large amount of data. An advantage of BB's algorithm is that it is content-independent, which is important for Web-scale data size. The sizes of the bipartite

TABLE 9
Best F-Measure Values of QU, QW, QC, and P-QC Methods for the Second Experiment

	Best F-Measure Values		
	Precision	*Recall*	*F-measure*
QU method	0.521	0.685	0.592
QW method	0.469	0.694	0.559
QC method	0.541	0.768	0.635
P-QC method	0.783	0.657	0.715

TABLE 10
Cutoff Values for Initial Clustering and Community Merging in the Second Experiment

Initial Clustering				Community Merging			
Cutoff	Precision	Recall	F-measure	Cutoff	Precision	Recall	F-Measure
0.3104	0.74	0.406667	0.524884	0.442	0.786667	0.535556	0.637266
0.3075	0.733333	0.416667	0.531401	0.4093	0.762857	0.545556	0.636162
0.3073	0.753333	0.452222	0.565174	0.3922	0.782857	0.616667	0.689895
0.3038	0.753333	0.472222	0.580538	0.3922	0.782857	0.636667	0.702234
0.2901	**0.753333**	**0.49222**	**0.59541**	**0.3922**	**0.78286**	**0.65667**	**0.71423**
0.276	0.753333	0.501111	0.601866	0.3922	0.762857	0.647778	0.700623
0.2758	0.733333	0.505556	0.598505	0.3922	0.742857	0.652222	0.694596
0.2602	0.691111	0.505556	0.583947	0.3995	0.701538	0.612222	0.653844
0.2483	0.691111	0.514444	0.589833	0.3995	0.701538	0.621111	0.658879

TABLE 11
Sample Query Clusters Generated by P-QC Method

Query Clusters	Query Clusters
hp$_{(User2)}$ + canon$_{(User2)}$ + epson$_{(User2)}$	macintosh$_{(User8)}$ + apple$_{(User8)}$ + imac$_{(User8)}$ + mac mini$_{(User8)}$ + macbook$_{(User8)}$
gap$_{(User6)}$ + calvin klein$_{(User6)}$ + banana republic$_{(User6)}$	ipod mini$_{(User9)}$ + ipod nano$_{(User9)}$ + apple$_{(User9)}$ + ipod$_{(User9)}$ + itunes$_{(User9)}$
apple$_{(User5)}$ + banana$_{(User4)}$ + fruit$_{(User4)}$ + apple$_{(User4)}$	mac os$_{(User7)}$ + apple$_{(User7)}$ + apple software$_{(User7)}$ + mac games$_{(User7)}$
fuji$_{(User3)}$ + kodak$_{(User3)}$ + photo film$_{(User3)}$ + camera film$_{(User3)}$ + xerox$_{(User2)}$ + fuji$_{(User2)}$ + konica$_{(User3)}$	fuji$_{(User1)}$ + kodak$_{(User1)}$ + canon$_{(User10)}$ + konica$_{(User1)}$ + hp digital camera$_{(User10)}$ + canon$_{(User1)}$ + minolta$_{(User10)}$ + pentax$_{(User10)}$ + hp$_{(User1)}$ + olympus$_{(User10)}$

graphs in the two experiments are shown in Tables 12 and 13, where *upper bound* is the upper bound for the number of operations required for agglomerative clustering, n_b is the number of black vertices in the bipartite graph G, n_w is the number of white vertices in the bipartite graph G (i.e., corresponding to the sets of queries and concepts, respectively, in our setting), $|N|_{max}$ is the maximum number of neighbors of any vertex in the bipartite graph G, and m is the number of iterations (i.e., merges) required for agglomerative clustering.

The bipartite graphs constructed using QC and P-QC methods are even smaller than that of the original BB's algorithm, because the number of concepts extracted from the web-snippets is small and the number of concepts resulting from web-snippets clicked by users is even smaller. The bipartite graph containing all words from the web-snippets (i.e., QW method) is the largest among the four methods, resulting in low clustering performance. From the experimental results, we can conclude that our concept-based clustering method is efficient because of the significant reduction of the size of the bipartite graph but at the same time effective as evident from the high precision and recall achieved.

7 Conclusions

As search queries are ambiguous, we have studied effective methods for search engines to provide query suggestions on semantically related queries in order to help users formulate more effective queries to meet their diversified needs. In this paper, we have proposed a new personalized concept-based clustering technique that is able to obtain personalized query suggestions for individual users based on their conceptual profiles. The technique makes use of clickthrough data and the concept relationship graph mined from web-snippets, both of which can be captured at the back end and as such do not add extra burden to users. An adapted agglomerative clustering algorithm is employed for finding queries that are conceptually close to one another. Our experimental results confirm that our approach can successfully generate personalized query suggestions according to individual user conceptual needs. Moreover, it improves prediction accuracy and computational cost compared to BB's algorithm, which is the state-of-the-art technique of

TABLE 12
Parameter Values Obtained from QU, QW, and QC Methods in the First Experiment

	Values of Clustering Parameters				
	n_b	n_w	$\|N\|_{max}$	m	*Upper Bound*
QU method	12,430	150	100	513	126,005,200
QW method	21,920	150	1,093	972	26,370,153,014
QC method	6,008	150	217	416	290,335,150

TABLE 13
Parameter Values Obtained from QU, QW, QC, and PQ-C Methods in the Second Experiment

	Values of Clustering Parameters				
	n_b	n_w	$\|N\|_{max}$	m	*Upper Bound*
QU method	3239	38	100	184	32,843,600
QW method	8891	38	938	572	7,858,273,220
QC method	1917	38	168	97	55,243,104
P-QC method	1917	50	152	113	55,592,544

query clustering using clickthroughs for the similar objective.

There are several directions for extending the work in the future. First, instead of considering only query-concept pairs in the clickthrough data, we can consider the relationships between users, queries, and concepts to obtain more personalized and accurate query suggestions. Second, clickthrough data and concept relationship graphs can be directly integrated into the ranking algorithms of a search engine so that it can rank results adapted to individual users' interests.

ACKNOWLEDGMENTS

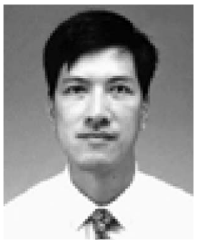
This work was supported by Grants 615806 and 615707 from Hong Kong RGC. The authors would like to thank the editors and reviewers, who gave very insightful and encouraging comments.

REFERENCES

[1] http://www.ask.com/, 2008.
[2] http://www.cse.ust.hk/~dlee/tkde08/query.html, 2008.
[3] http://www.dmoz.org/, 2008.
[4] http://www.google.com/, 2008.
[5] http://www.sigkdd.org/kdd2005/kddcup.html, 2008.
[6] http://www.yahoo.com/, 2008.
[7] E. Agichtein, E. Brill, and S. Dumais, "Learning User Interaction Models for Predicting Web Search Result Preferences," *Proc. 29th Ann. Int'l ACM SIGIR Conf. (SIGIR),* 2006.
[8] E. Agichtein, E. Brill, S. Dumais, and R. Rango, "Improving Web Search Ranking by Incorporating User Behavior Information," *Proc. 29th Ann. Int'l ACM SIGIR Conf. (SIGIR),* 2006.
[9] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD,* 1993.
[10] R.A. Baeza-Yates, C.A. Hurtado, and M. Mendoza, "Query Recommendation Using Query Logs in Search Engines," *Proc. EDBT Workshop,* vol. 3268, pp. 588-596, 2004.
[11] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," *Proc. ACM SIGKDD,* 2000.
[12] S.M. Beitzel, E.C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder, "Hourly Analysis of a Very Large Topically Categorized Web Query Log," *Proc. 27th Ann. Int'l ACM SIGIR Conf. (SIGIR),* 2004.
[13] V.W. Chan, K.W. Leung, and D.L. Lee, "Clustering Search Engine Query Log Containing Noisy Clickthroughs," *Proc. Symp. Applications and the Internet (SAINT),* 2004.
[14] S. Chuang and L. Chien, "Automatic Query Taxonomy Generation for Information Retrieval Applications," *Online Information Rev.,* vol. 27, no. 4, pp. 243-255, 2003.
[15] H. Cui, J. Wen, J. Nie, and W. Ma, "Query Expansion by Mining User Logs," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 4, pp. 829-839, July/Aug. 2003.
[16] K.W. Church, W. Gale, P. Hanks, and D. Hindle, "Using Statistics in Lexical Analysis," *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon,* U. Zernik, ed., Lawrence Erlbaum, 1991.
[17] L. Deng, W. Ng, X. Chai, and D.L. Lee, "Spying Out Accurate User Preferences for Search Engine Adaptation," *Advances in Web Mining and Web Usage Analysis,* LNCS 3932, pp. 87-103, 2006.
[18] Z. Dou, R. Song, and J.R. Wen, "A Large-Scale Evaluation and Analysis of Personalized Search Strategies," *Proc. 16th Int'l World Wide Web Conf. (WWW),* 2007.
[19] B. Goethals and M. Zaki, "Frequent Itemset Mining Implementations," *Proc. ICDM Workshop Frequent Itemset Mining Implementations (FIMI),* 2003.
[20] M. Jansen, A. Spink, J. Bateman, and T. Saracevic, "Real Life Information Retrieval: A Study of User Queries on the Web," *Proc. ACM SIGIR Forum,* vol. 32, pp. 5-17, 1998.
[21] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," *Proc. ACM SIGKDD,* 2002.
[22] T. Joachims and F. Radlinski, "Search Engines That Learn from Implicit Feedback," *Computer,* vol. 40, no. 8, pp. 34-40, 2007.
[23] B. Koester, "Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies," *Proc. Sixth IEEE Int'l Conf. Data Mining (ICDM),* 2006.
[24] F. Liu, C. Yu, and W. Meng, "Personalized Web Search for Improving Retrieval Effectiveness," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 1, pp. 28-40, Jan. 2004.
[25] B. Mirkin, *Mathematical Classification and Clustering.* Kluwer, 1996.
[26] G. Salton and M.J. Mcgill, *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.
[27] B. Smyth et al., "Exploiting Query Repetition and Regularity in an Adaptive Community-Based Web Search Engine," *User Modeling and User-Adapted Interaction,* vol. 14, no. 5, pp. 383-423, 2005.
[28] M. Speretta and S. Gauch, "Personalized Search Based on User Search Histories," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI),* 2005.
[29] Q. Tan, X. Chai, W. Ng, and D.L. Lee, "Applying Co-Training to Clickthrough Data for Search Engine Adaptation," *Proc. Ninth Int'l Conf. Database Systems for Advanced Applications (DASFAA),* 2004.
[30] C.J. Van Rijsbergen, *Information Retrieval.* Butterworths, 1979.
[31] J. Wen, J. Nie, and H. Zhang, "Query Clustering Using User Logs," *ACM Trans. Information Systems,* vol. 20, no. 1, pp. 59-81, 2002.
[32] Y. Xu, B. Zhang, Z. Chen, and K. Wang, "Privacy-Enhancing Personalized Web Search," *Proc. 16th Int'l World Wide Web Conf. (WWW),* 2007.
[33] Z. Zhang and O. Nasraoui, "Mining Search Engine Query Logs for Query Recommendation," *Proc. 15th Int'l World Wide Web Conf. (WWW),* 2006.

Kenneth Wai-Ting Leung received the BSc degree in computer science from the University of British Columbia, Vancouver, Canada, in 2002 and the MSc degree in computer science from the Hong Kong University of Science and Technology in 2004. He is currently a PhD candidate in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include search engines, Web mining, information retrieval, and ontologies.

Wilfred Ng received the MSc (with distinction) and PhD degrees in computer science from the University of London. He is currently an assistant professor of computer science and engineering at the Hong Kong University of Science and Technology (HKUST), where he is a member of the database research group. His research interests are in the areas of databases, data mining, and information systems, which include Web data management and XML searching. Further information can be found at http://www.cs.ust.hk/faculty/wilfred/index.html.



Dik Lun Lee received the BSc degree in electronics from the Chinese University of Hong Kong and the MS and PhD degrees in computer science from the University of Toronto, Canada. He is currently a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He was an associate professor in the Department of Computer Science and Engineering, Ohio State University, Columbus. He was the founding conference chair for the International Conference on Mobile Data Management and served as the chairman of the ACM Hong Kong Chapter in 1997. His research interests include information retrieval, search engines, mobile computing, and pervasive computing.