Deriving Concept-based User Profiles from Search Engine Logs

Kenneth Wai-Ting Leung, Dik Lun Lee

Abstract—User profiling is a fundamental component of any personalization applications. Most existing user profiling strategies are based on objects that users are interested in (i.e. positive preferences), but not the objects that users dislike (i.e. negative preferences). In this paper, we focus on search engine personalization and develop several concept-based user profiling methods that are based on both positive and negative preferences. We evaluate the proposed methods against our previously proposed personalized query clustering method. Experimental results show that profiles which capture and utilize both of the user's positive and negative preferences perform the best. An important result from the experiments is that profiles with negative preferences can increase the separation between similar and dissimilar queries. The separation provides a clear threshold for an agglomerative clustering algorithm to terminate and improve the overall quality of the resulting query clusters.

Index Terms—Negative preferences, personalization, personalized query clustering, search engine, user profiling.

1 Introduction

Most commercial search engines return roughly the same results for the same query, regardless of the user's real interest. Since queries submitted to search engines tend to be short and ambiguous, they are not likely to be able to express the user's precise needs. For example, a farmer may use the query "apple" to find information about growing delicious apples, while graphic designers may use the same query to find information about Apple Computer.

Personalized search is an important research area that aims to resolve the ambiguity of query terms. To increase the relevance of search results, personalized search engines create user profiles to capture the users' personal preferences and as such identify the actual goal of the input query. Since users are usually reluctant to explicitly provide their preferences due to the extra manual effort involved, recent research has focused on the automatic learning of user preferences from users' search histories or browsed documents and the development of personalized systems based on the learned user preferences.

A good user profiling strategy is an essential and fundamental component in search engine personalization. We studied various user profiling strategies for search engine personalization, and observed the following problems in existing strategies.

- Most personalization methods focused on the creation of one single profile for a user and applied the same profile to all of the user's queries. We believe that different queries from a user should be handled differently because a user's preferences may vary across queries. For example, a user who prefers information about fruit on the query "orange", may prefer the information about Apple Computer for the query "apple". Personalization strategies such as [1], [2], [8], [10], [13], [15], [17], [18] employed a single large user profile for each user in the personalization process.
- Existing clickthrough-based user profiling strategies can be categorized into *document-based* and *concept-based* approaches. They both assume that user clicks can be used to infer users' interests, although their inference methods and the outcomes of the inference are different. Document-based profiling methods try to estimate users' document preferences (i.e., users are interested in some documents more than others) [1], [2], [8], [10], [15], [18].[1] On the other hand, concept-based profiling methods aim to derive topics or concepts that users are highly interested in [13], [17]. These two approaches will be reviewed in Section 2. While there are document-based methods that consider both users' positive and negative preferences, to the best of our knowledge, there are no concept-based methods that considered both positive and negative preferences in deriving user's topical interests.
- Most existing user profiling strategies only consider documents that users are interested in (i.e. users' positive preferences) but ignore documents that users dislike (i.e. users' negative preferences). In reality, positive preferences are not enough to capture the fine-grain interests of a user. For example, if a user is interested in "apple" as a fruit, he/she may be interested specifically in apple recipes, but less interested in information about growing apples, while absolutely not interested in information about the company Apple Computer. In this case, a good user profile should favor information about apple recipes, slightly favor information about growing apple, while downgrade information about Apple Computer. Profiles built on both positive and negative user preferences can

• *K. W.-T. Leung and D. L. Lee are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: kwtleung, dlee@cse.ust.hk.*

Manuscript received April 19, 2005; revised January 11, 2007.

1. In general, document-based profiling methods may also estimate the *properties* of the documents that are likely to arouse users' interest, e.g., whether or not the documents match the queries in their titles, URLs, etc.

represent user interests at finer details. Personalization strategies such as [10], [15], [18] include negative preferences in the personalization process, but they all are document-based and thus cannot reflect users' general topical interests.

In this paper, we address the above problems by proposing and studying seven concept-based user profiling strategies that are capable of deriving both of the user's positive and negative preferences. All of the user profiling strategies are query-oriented, meaning that a profile is created for each of the user's queries. The user profiling strategies are evaluated and compared with our previously proposed personalized query clustering method. Experimental results show that user profiles which capture both the user's positive and negative preferences perform the best among all of the profiling strategies studied. Moreover, we find that negative preferences improve the separation of similar and dissimilar queries, which facilitates an agglomerative clustering algorithm to decide if the optimal clusters have been obtained. We show by experiments that the termination point and the resulting precision and recalls are very close to the optimal results.

The main contributions of this paper are:

- We extend the query-oriented, concept-based user profiling method proposed in [11] to consider both users' positive and negative preferences in building users profiles. We proposed six user profiling methods that exploit a user's positive and negative preferences to produce a profile for the user using a Ranking SVM (RSVM).
- While document-based user profiling methods pioneered by Joachims [10] capture users' document preferences (i.e., users consider some documents to be more relevant than others), our methods are based on users' concept preferences (i.e., users consider some topics/concepts to be more relevant than others).
- Our proposed methods use an RSVM to learn from concept preferences weighted concept vectors representing concept-based user profiles. The weights of the vector elements, which could be positive or negative, represent the interestingness (or uninterestingness) of the user on the concepts. In [11], the weights that represent a user's interests are all positive, meaning that the method can only capture user's positive preferences.
- We conduct experiments to evaluate the proposed user profiling strategies and compare it with a baseline proposed in [11]. We show that profiles which capture both the user's positive and negative preferences perform best among all of the proposed methods. We also find that the query clusters obtained from our methods are very close to the optimal clusters.

The rest of the paper is organized as follows. Section 2 discusses the related works. We classify the existing user profiling strategies into two categories, and review methods among the categories. In Section 3, we review our personalized concept-based clustering strategy to exploit the relationship among ambiguous queries according to the user conceptual preferences recorded in the concept-based user profiles. In Section 4, we present the proposed concept-based user profiling strategies. Experimental results comparing our user profiling strategies are presented in Section 5. Section 6 concludes the paper.

TABLE 1
An Example of Clickthrough for the Query "apple"

Doc	Clicked	Search Results	Extracted Concepts
d_1	$\surd$	**Apple Computer**	**macintosh**
d_2		Apple Support	product
d_3		Apple Inc. Official Downloads	mac os
d_4		Apple Store (U.S.)	apple store, iPod
d_5	$\surd$	**The Apple Store**	**apple store, macintosh**
d_6		Apple Hill Growers	fruit, apple hill
d_7		Apple Corps	fruit
d_8	$\surd$	**Macintosh Products Guide**	**macintosh, catalog**

2 Related Work

User profiling strategies can be broadly classified into two main approaches: *document-based* and *concept-based* approaches. Document-based user profiling methods aim at capturing users' clicking and browsing behaviors. Users' document preferences are first extracted from the clickthrough data and then used to learn the user behavior model which is usually represented as a set of weighted features. On the other hand, concept-based user profiling methods aim at capturing users' conceptual needs. Users' browsed documents and search histories are automatically mapped into a set of topical categories. User profiles are created based on the users' preferences on the extracted topical categories.

2.1 Document-Based Methods

Most document-based methods focus on analyzing users' clicking and browsing behaviors recorded in the users' clickthrough data. On web search engines, clickthrough data is an important implicit feedback mechanism from users. Table 1 is an example of clickthrough data for the query "apple", which contains a list of ranked search results presented to the user, with identification on the results that the user has clicked on. The bolded documents d_1, d_5 and d_8 are the documents that have been clicked by the user. Several personalized systems that employ clickthrough data to capture users' interest have been proposed [1], [2], [10], [15], [18].

Joachims [10] proposed a method which employs preference mining and machine learning to model users' clicking and browsing behavior. Joachims' method assumes that a user would scan the search result list from top to bottom. If a user has skipped a document d_i at rank i before clicking on document d_j at rank j, it is assumed that he/she must have scan the document d_i and decided to skip it. Thus, we can conclude that the user prefers document d_j more than document d_i (i.e. $d_j <_{r'} d_i$, where r' is the user's preference order of the documents in the search result list). Using Joachims'

TABLE 2
Document Preference Pairs obtained using Joachims' Method

Preference Pairs containing d_1	Preference Pairs containing d_5	Preference Pairs containing d_8
Empty Set	$d_5 <_{r'} d_2$	$d_8 <_{r'} d_2$
	$d_5 <_{r'} d_3$	$d_8 <_{r'} d_3$
	$d_5 <_{r'} d_4$	$d_8 <_{r'} d_4$
		$d_8 <_{r'} d_6$
		$d_8 <_{r'} d_7$

TABLE 3
An Example of User Profile as a Set of Weighted Features

Feature	Weight	Feature	Weight
query_abstract_cosine	0.60	top10count_3	0.19
top10_google	0.48	top10_yahoo	0.16
query_url_cosine	0.24	...	...
top1count_1	0.24	url_length	-0.17
top10_msnsearch	0.24	top10count_0	-0.32
host_citeseer	0.22	top1count_0	-0.38

proposition and the example clickthrough data in Table 1, a set of document preference pairs as shown in Table 2 can be obtained. After the document preference pairs are obtained, a Ranking SVM (RSVM) [10] is employed to learn the user behavior model as a set of weighted features. Table 3 shows an example of Joachims' user profile, which consists of a set of weighted features.

Ng et al. [15] proposed an algorithm which combines a spying technique together with a novel voting procedure to determine users' document preferences from the clickthrough data. They also employed the RSVM algorithm to learn the user behavior model as a set of weight features. More recently, Agichtein et al. [1] suggested that explicit feedback (i.e. individual user behavior, clickthrough data, etc) from search engine users is noisy. One major observation is the bias of user click distribution toward top ranked results. To resolve the bias, Agichtein suggested to clean up the clickthrough data with the aggregated "background" distribution. RankNet [6], a scalable implementation of neural networks, is then employed to learn the user behavior model from the cleaned clickthrough data.

2.2 Concept-Based Methods

Most concept-based methods automatically derive users' topical interests by exploring the contents of the users' browsed documents and search histories. Liu et al. [13] proposed a user profiling method based on users' search history and the Open Directory Project (ODP) [16]. The user profile is represented as a set of categories, and for each category, a set of keywords with weights. The categories stored in the user profiles serve as a context to disambiguate user queries. If a profile shows that a user is interested in certain categories, the search can be narrowed down by providing suggested results according to the user's preferred categories.

Gauch et al. [9] proposed a method to create user profiles from user browsed documents. User profiles are created using concepts from the top four levels of the concept hierarchy created by Magellan [14]. A classifier is employed to classify user browsed documents into concepts in the reference ontology. Xu et al. [20] proposed a scalable method which automatically builds user profiles based on users' personal documents (e.g. browsing histories and emails). The user profiles summarize users' interests into hierarchical structures. The method assumes that terms exist frequently in user's browsed documents represent topics that the user is interested in. Frequent terms are extracted from users' browsed documents to build hierarchical user profiles representing users' topical interests.

Liu et al. and Gauch et al. both use a reference ontology (e.g. ODP) to develop the hierarchical user profiles, while Xu et al. automatically extracts possible topics from users' browsed documents and organizes the topics into hierarchical structures. The major advantage of dynamically building a topic hierarchy is that new topics can be easily recognized and extracted from documents and added to the topic hierarchy, whereas a reference ontology such as ODP is not always up-to-date. Thus, all of our proposed user profiling strategies rely on a concept extraction method as described in Section 3.1.1, which extracts concepts from web-snippets[2] to create accurate and up-to-date user profiles.

3 Personalized Concept-Based Query Clustering

Our personalized concept-based clustering method consists of three steps. First, we employ a concept extraction algorithm, which will be described in Section 3.1.1, to extract concepts and their relations from the web-snippets returned by the search engine. Second, seven different concept-based user profiling strategies, which will be introduced in Section 4, are employed to create concept-based user profiles. Finally, the concept-based user profiles are compared with each other and against as baseline our previously proposed personalized concept-based clustering algorithm [11], which is reviewed in Section 3.2.

3.1 Concept Extraction

3.1.1 Extracting Concepts from Web-snippets

After a query is submitted to a search engine, a list of web-snippets are returned to the user. We assume that if a keyword/phrase exists frequently in the web-snippets of a particular query, it represents an important concept related to the query because it co-exists in close proximity with the query in the top documents. Thus, we employ the following support formula, which is inspired by the well-known problem of finding frequent item sets in data mining [7], to measure the interestingness of a particular keyword/phrase c_i extracted from the web-snippets arising from q: *interestingness* of a particular keyword/phrase c_i with respect to the query q:

$$support(c_i) = \frac{sf(c_i)}{n} \cdot |c_i| \tag{1}$$

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

TABLE 4
Example Concepts Extracted for the Query “apple”

Concept c_i	$support(c_i)$	Concept c_i	$support(c_i)$
mac	0.1	apple store	0.06
iPod	0.1	slashdot apple	0.04
iPhone	0.1	picture	0.04
hardware	0.09	music	0.03
mac os	0.06	apple farm	0.02

where $sf(c_i)$ is the snippet frequency of the keyword/phrase c_i (i.e. the number of web-snippets containing c_i), n is the number of web-snippets returned and $|c_i|$ is the number of terms in the keyword/phrase c_i. If the support of a keyword/phrase c_i is greater than the threshold s ($s = 0.03$ in our experiments), we treat c_i as a concept for the query q. Table 4 shows an example set of concepts extracted for the query “apple”. Before concepts are extracted, stopwords, such as “the”, “of”, “we”, etc., are first removed from the snippets. The maximum length of a concept is limited to seven words. These not only reduce the computational time but also avoid extracting meaningless concepts.

3.1.2 Mining Concept Relations

We assume that two concepts from a query q are similar if they co-exist frequently in the web-snippets arising from the query q. According to the assumption, we apply the following well-known signal-to-noise formula from data mining [7] to establish the similarity between terms t_1 and t_2:

$$sim(t_1, t_2) = \log \frac{n \cdot df(t_1 \cup t_2)}{df(t_1) \cdot df(t_2)} / \log n \quad (2)$$

where n is the number of documents in the corpus, $df(t)$ is the document frequency of the term t and $df(t_1 \cup t_2)$ is the joint document frequency of t_1 and t_2. The similarity $sim(t_1, t_2)$ obtained using the above formula always lies between [0,1].

In the search engine context, two concepts c_i and c_j could co-exist in the following situations: 1) c_i and c_j coexist in the title, 2) c_i and c_j co-exist in the summary and 3) c_i exists in the title while c_j exists in the summary (or vice versa). Similarities for the three different cases are computed using the following formulas:

$$sim_{R,title}(c_i, c_j) = \log \frac{n \cdot sf_{title}(c_i \cup c_j)}{sf_{title}(c_i) \cdot sf_{title}(c_j)} / \log n \quad (3)$$

$$sim_{R,sum}(c_i, c_j) = \log \frac{n \cdot sf_{sum}(c_i \cup c_j)}{sf_{sum}(c_i) \cdot sf_{sum}(c_j)} / \log n \quad (4)$$

$$sim_{R,other}(c_i, c_j) = \log \frac{n \cdot sf_{other}(c_i \cup c_j)}{sf_{other}(c_i) \cdot sf_{other}(c_j)} / \log n \quad (5)$$

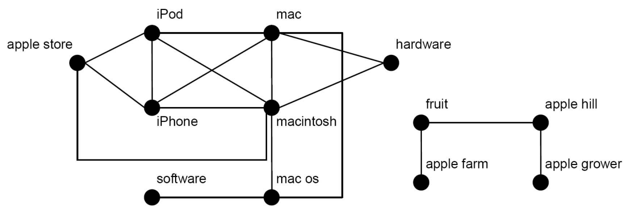
where $sf_{title}(c_i \cup c_j)/sf_{sum}(c_i \cup c_j)$ are the joint snippet frequencies of the concepts c_i and c_j in web-snippets' titles/summaries, $sf_{title}(c)/sf_{sum}(c)$ are the snippet frequencies of the concept c in web-snippets' titles/summaries, $sf_{other}(c_i \cup c_j)$ is the joint snippet frequency of the concepts c_i in a web-snippet's title and c_j in a web-snippet's summary (or vice versa), and $sf_{other}(c)$ is the snippet frequency of concept c in either web-snippets' titles or summaries. The following formula is used to obtain the combined similarity $sim_R(c_i, c_j)$ from the three cases, where $\alpha + \beta + \gamma = 1$ to ensure that $sim_R(c_i, c_j)$ lies between [0,1].

$$sim_R(c_i, c_j) = \alpha \cdot sim_{R,title}(c_i, c_j) + \beta \cdot sim_{R,summary}(c_i, c_j) + \gamma \cdot sim_{R,other}(c_i, c_j) \quad (6)$$

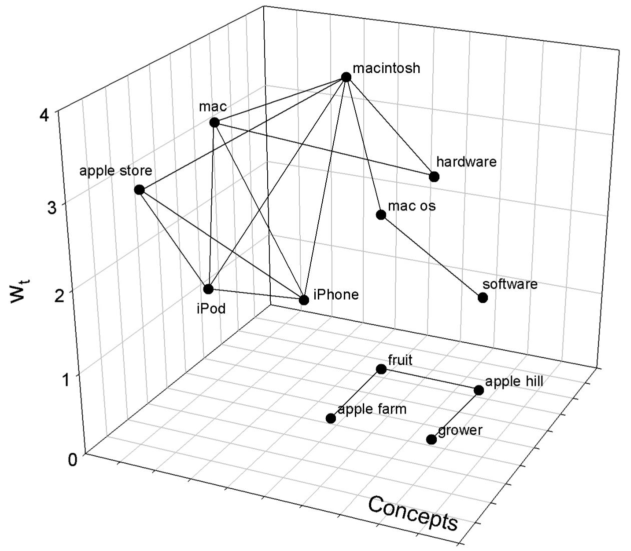
Figure 1(a) shows a concept graph built for the query “apple”. The nodes are the concepts extracted from the query “apple”, and the links are created between concepts having $sim_R(c_i, c_j) > 0$. The graph shows the possible concepts and their relations arising from the query “apple”.

3.2 Query Clustering Algorithm

We now review our personalized concept-based clustering algorithm [11] with which ambiguous queries can be classified into different query clusters. Concept-based user profiles are employed in the clustering process to achieve personalization effect. First, a query-concept bipartite graph G is constructed by the clustering algorithm with one set of nodes corresponds to the set of users' queries, and the other corresponds to the sets of extracted concepts. Each individual query submitted



(a) The concept space derived for the query “apple”.



(b) An example of user profile in which the user is interested in the concept ”macintosh”.

Fig. 1. An example of a concept space and the corresponding user profile.

by each user is treated as an individual node in the bipartite graph by labeling each query with a user identifier. Concepts with interestingness weights (defined in Equation 1) greater than zero in the user profile are linked to the query with the corresponding interestingness weight in G.

Second, a two-step personalized clustering algorithm is applied to the bipartite graph G, to obtain clusters of similar queries and similar concepts. Details of the personalized clustering algorithm is shown in Algorithm 1. The personalized clustering algorithm iteratively merges the most similar pair of query nodes, and then the most similar pair of concept nodes, and then merge the most similar pair of query nodes, and so on. The following cosine similarity function is employed to compute the similarity score $sim(x, y)$ of a pair of query nodes or a pair of concept nodes. The advantages of the cosine similarity are that it can accommodate negative concept weights and produce normalized similarity values in the clustering process.

$$sim(x, y) = \frac{N_x \cdot N_y}{\parallel N_x \parallel \parallel N_y \parallel} \quad (7)$$

where N_x is a weight vector for the set of neighbor nodes of node x in the bipartite graph G, the weight of a neighbor node n_x in the weight vector N_x is the weight of the link connecting x and n_x in G, N_y is a weight vector for the set of neighbor nodes of node y in G, and the weight of a neighbor node n_y in N_y is the weight of the link connecting y and n_y in G.

Algorithm 1 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 query nodes using Equation (7).
2: Merge the pair of most similar query nodes ($q_i$,$q_j$) that does not contain the same query from different users. Assume that a concept node $c$ is connected to both query nodes $q_i$ and $q_j$ with weight $w_i$ and $w_j$, a new link is created between $c$ and $(q_i, q_j)$ with weight $w = w_i + w_j$.
3: Obtain the similarity scores in $G$ for all possible pairs of concept nodes using Equation (7).
4: Merge the pair of concept nodes ($c_i$,$c_j$) having highest similarity score. Assume that a query node $q$ is connected to both concept nodes $c_i$ and $c_j$ with weight $w_i$ and $w_j$, a new link is created between $q$ and $(c_i, c_j)$ with weight $w = w_i + w_j$.
5. Unless termination is reached, repeat Steps 1-4.
// Community Merging
6. Obtain the similarity scores in $G$ for all possible pairs of query nodes using Equation (7).
7. Merge the pair of most similar query nodes ($q_i$,$q_j$) that contains the same query from different users. Assume that a concept node $c$ is connected to both query nodes $q_i$ and $q_j$ with weight $w_i$ and $w_j$, a new link is created between $c$ and $(q_i, q_j)$ with weight $w = w_i + w_j$.
8. Unless termination is reached, repeat Steps 6-7.
```

The algorithm is divided into two steps, **initial clustering** and **community merging**. In initial clustering, queries are grouped within the scope of each user. Community merging is then involved to group queries for the community. A more detailed example is provided in our previous work [11] to explain the purpose of the two steps in our personalized clustering algorithm.

A common requirement of iterative clustering algorithms is to determine when the clustering process should stop to avoid over-merging of the clusters. Likewise, a critical issue in Algorithm 1 is to decide the termination points for initial clustering and community merging. When the termination point for initial clustering is reached, community merging kicks off; when the termination point for community merging is reached, the whole algorithm terminates.

Good timing to stop the two phases is important to the algorithm, since if initial clustering is stopped too early (i.e., not all clusters are well formed), community merging merges all the identical queries from different users , and thus generates a single big cluster without much personalization effect. However, if initial clustering is stopped too late, the clusters are already overly merged before community merging begins. The low precision rate thus resulted would undermine the quality of the whole clustering process.

The determination of the termination points was left open in [11]. Instead, it obtained the optimal termination points by exhaustively searching for the point at which the resulting precision and recall values are maximized. Most existing clustering methods such as [5], [19] and [4] used a fixed criteria which stops the clustering when the intra-cluster similarity drops beyond a threshold. However, since the threshold is either fixed or obtained from a training data set, the method is not suitable in a personalized environment where the behaviors of users are different and change from time to time. In Section 5.4, we will study a simple heuristic that determines the termination points when the intra-cluster similarity shows a sharp drop. Further, we show that methods that exploit negative preferences produce termination points that are very close to the optimal termination points obtained by exhaustive search.

4 User Profiling Strategies

In this section, we propose six user profiling strategies which are both concept-based and utilize users' positive and negative preferences. They are $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$. In addition, we use P_{Click}, which was proposed in [11], as the baseline in the experiments. P_{Click} is concept-based but cannot handle negative preferences.

4.1 Click-Based Method (P_{Click})

The concepts extracted for a query q using the concept extraction method discussed in Section 3.1.1 describe the possible concept space arising from the query q. The concept space may cover more than what the user actually wants. For example, when the user searches for the query "apple", the concept space derived from our concept extraction method contains the concepts “macintosh”, “ipod” and “fruit”. If the user is indeed interested in “apple” as a fruit and clicks on pages containing the concept “fruit”, the user profile represented as a weighted concept vector should record the user interest on the concept “apple” and its neighborhood (i.e., concepts which having similar meaning as “fruit”), while downgrading unrelated concepts such as “macintosh”, “ipod” and their neighborhood. Therefore, we propose the following formulas to capture a user’s degree of interest, w_{c_i}, on the extracted concepts c_i, when a web-snippet s_j is clicked by the user (denoted by $click(s_j)$):

$$click(s_j) \Rightarrow \forall c_i \in s_j, w_{c_i} = w_{c_i} + 1 \tag{8}$$

$$click(s_j) \Rightarrow \forall c_i \in s_j, w_{c_j} = w_{c_j} + sim_R(c_i, c_j) \quad if\ sim_R(c_i, c_j) > 0 \tag{9}$$

where s_j is a web-snippet, w_{c_i} represents the user’s degree of interest on the concept c_i, and c_j is the neighborhood concept of c_i.

When a web-snippet s_j has been clicked by a user, the weight w_{c_i} of concepts c_i appearing in s_j is incremented by 1. For other concepts c_j that are related to c_i on the concept relationship graph, they are incremented according to the similarity score given in Equation (9). Figure 1(b) shows an example of a click-based profile P_{Click} in which the user is interested in information about “macintosh”. Hence, the concept “macintosh” receives the highest weight among all of the concepts extracted for the query “apple”. The weights w_{t_i} of the concepts “mac os”, “software”, “apple store”, “iPod”, “iPhone”, and “hardware” are increased based on Equation (9), because they are related to the concept “macintosh”. The weights w_{c_i} for concepts “fruit”, “apple farm”, “juice”, and “apple grower” remain zero, showing that the user is not interested in information about “apple fruit”.

4.2 Joachims-C Method ($P_{Joachims-C}$)

Joachims [10] assumed that a user would scan the search results from top to bottom. If a user skipped a document d_i before clicking on document d_j (where rank of $d_j >$ rank of d_i), he/she must have scanned d_i and decided not to click on it. According to the Joachims’ original proposition as discussed in Section 2.1, it would extract the user’s document preference as $d_j <_{r'} d_i$.

Joachims’ original method was based on users’ document preferences. If a user has skipped a document d_i at rank i before clicking on document d_j at rank j, he/she must have scanned the document d_i and decided to skip it. Thus, we can conclude that the user prefers document d_j more than document d_i (i.e., $d_j <_{r'} d_i$, where r' is the user’s preference order of the documents in the search result list).

We extended Joachims’ method, which is a document-based method, to a concept based method (Joachims-C). Instead of obtaining the document preferences $d_j <_{r'} d_i$, Joachims-C assumes that the user prefers the concepts $C(d_j)$ associated with document d_j to the concepts $C_(d_i)$ associated with document d_i, and produces the corresponding concept preferences. The idea is captured in the following proposition.

Proposition 1 (Joachims-C *Skip Above*): Given a list of search results for an input query q, if a user clicks on the document d_j at rank j, all the concepts $C(d_i)$ in the *unclicked* documents d_i above rank j are considered as less relevant than the concepts $C(d_j)$ in the document d_j, i.e., ($C(d_j) <_{r'} C(d_i)$, where r' is the user’s preference order of the concepts extracted from the search results of the query q).

Using the example in Table 1, the user did not click on d_2, d_3, and d_4, but clicked on d_5. Thus, according to Proposition 1, we can conclude that the concepts $C(d_5)$ is more relevant to the user than the concepts in the other three unclicked documents (i.e., $C(d_2)$, $C(d_3)$ and $C(d_4)$). The concept preference pairs extracted using Joachims-C method are shown in Table 5.

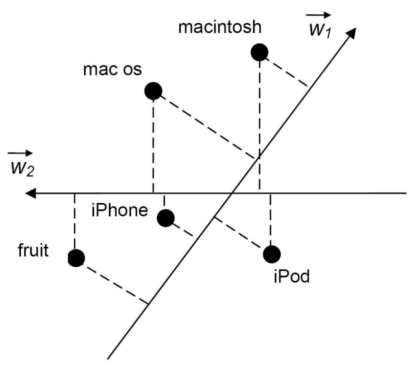


Fig. 2. Ordering of concepts “macintosh”, “mac os”, “iPod”, “iPhone”, and “fruit” using weight vectors $\overrightarrow{w_1}$ and $\overrightarrow{w_2}$.

After the concept preference pairs are identified using Proposition 1, a ranking SVM algorithm [10] is employed to learn the user’s preferences, which is represented as a weighted concept vector. Given a set of concept preference pairs T, ranking SVM aims at finding a linear ranking function $f(q, c)$ to rank the extracted concepts so that as many concept preference pairs in T as possible are satisfied. $f(q, c)$ is defined as the inner product of a weight vector $\overrightarrow{w}$ and a feature vector of query-concept mapping $\phi(q, c)$, which describes how well a concept c matches the user’s interest for a query q.

Figure 2 is an example showing how the weight vector $\overrightarrow{w}$ affects the ordering of the extracted concepts, where the target user concept preferences is (“macintosh” $<_{r^*}$ “mac os” $<_{r^*}$ “iPod” $<_{r^*}$ “iPhone” $<_{r^*}$ “fruit”). We can see that $\overrightarrow{w_1}$ is better than $\overrightarrow{w_2}$, because $\overrightarrow{w_1}$ correctly ranks the concepts as (“macintosh” $<_{w_1}$ “mac os” $<_{w_1}$ “iPod” $<_{w_1}$ “iPhone” $<_{w_1}$ “fruit”), while $\overrightarrow{w_2}$ ranks the concepts as (“fruit” $<_{w_2}$ “mac os” $<_{w_2}$ “iPhone” $<_{w_2}$ “macintosh” $<_{w_2}$ “iPod”).

The feature vector $\phi(q, c) = [Feature\_c_1, Feature\_c_2, ..., Feature\_c_n]$ for the ranking SVM training is composed of all the extracted concepts for a query q. For each concept c_i, we create a feature vector $\phi(q, c_i) = [Feature\_c_1, Feature\_c_2, ..., Feature\_c_n]$ which is defined as follows.

TABLE 5
Concept Preference Pairs obtained using Joachims-C Methods

Concept Preference Pairs for d_1	Concept Preference Pairs for d_5	Concept Preference Pairs for d_8
Empty Set	apple store $<_{r'}$ product macintosh $<_{r'}$ product	macintosh $<_{r'}$ product catalog $<_{r'}$ product
	apple store $<_{r'}$ mac os macintosh $<_{r'}$ mac os	macintosh $<_{r'}$ mac os catalog $<_{r'}$ mac os
	macintosh $<_{r'}$ apple store apple store $<_{r'}$ iPod macintosh $<_{r'}$ iPod	macintosh $<_{r'}$ apple store catalog $<_{r'}$ apple store macintosh $<_{r'}$ iPod catalog $<_{r'}$ iPod
		macintosh $<_{r'}$ fruit catalog $<_{r'}$ fruit macintosh $<_{r'}$ apple hill catalog $<_{r'}$ apple hill
		macintosh $<_{r'}$ fruit catalog $<_{r'}$ fruit

$$Feature\_c_k = \begin{cases} 1 & \text{if } k = i \\ sim_R(c_i, c_j) & \text{if } sim_R(c_i, c_k) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

The concept preference pairs together with the feature vectors serve as the input to the ranking SVM algorithm. The ranking SVM algorithm outputs a weight vector $\overrightarrow{w}$ such that the maximum number of the following inequalities holds:

$$\forall (c_i, c_j) \in r'_k, (1 \leq k \leq n): \quad \overrightarrow{w} \cdot \phi(q_k, c_i) > \overrightarrow{w} \cdot \phi(q_k, c_j) \quad (11)$$

where $(c_i, c_j) \in r'_k$ is a concept pair corresponding to the concept preference pair ($c_i <_{r'_k} c_j$) of the query q_k, which means that c_i should rank higher than c_j in the target concept ordering of r'_k.

The weight vector $\overrightarrow{w}$ = ($w_{Feature\_c_1}$, $w_{Feature\_c_2}$, ..., $w_{Feature\_c_n}$) determines the user preferences on the extracted concepts. For all the concepts c_1, c_2, ..., c_i extracted for the query q, the user preferences are stored in the corresponding weight values $w_{Feature\_c_1}$, $w_{Feature\_c_2}$, ..., $w_{Feature\_c_n}$, creating a concept preference profile $P_{Joachims-C}$ = ($w_{Feature\_c_1}$, $w_{Feature\_c_2}$, ..., $w_{Feature\_c_n}$) for the query q. Table 6 shows an example of feature weights resulted from RSVM Training for the query q = apple (where the user's topical preferences are "fruit" and "farm") using Joachims-C method from our experiments.

4.3 mJoachims-C Method ($P_{mJoachims-C}$)

mJoachims extends Joachims, which only considers unclicked pages above a clicked page, by considering unclicked pages both above and below a clicked page [15]. As with Joachims-C, we extend mJoachims into mJoachims-C by deriving concept-preference pairs from page-preference pairs.

Proposition 2. (mJoachims-C *Skip Above+Skip Next*): Given a set of search results for a query, if documents d_i at rank i is clicked, d_j is the next clicked document right after d_i (no other clicked links between d_i and d_j), and document d_k at rank k between d_i and d_j ($i < k < j$) is not clicked, then concepts $C(d_k)$ in document d_k is considered less relevant than the concepts $C(d_j)$ in document d_j ($C(d_j) <_{r'} C(d_k)$) where r' is the user's preference order of the concepts extracted from the search results of the query q). The predictions obtained are combined with those obtained from Proposition 1 (Joachims-C method) above.

TABLE 6
Example Feature Weights obtained from RSVM Training for the Query q = apple

Feature	Weight (Joachims-C)	Weight (mJoachims-C)	Weight (SpyNB-C)
entertainment	-0.369	-0.275	-0.029
traveler	-0.092	-0.030	-0.022
kid	-0.196	-0.350	-0.228
recipe	-0.333	-0.272	-0.435
program	-0.076	-0.202	-0.188
orchard	-0.939	-0.851	-0.824
directory	-0.335	-0.274	-0.043
fruit	1.941	1.871	1.765
farm	2.048	2.629	1.497
art	0.420	-0.240	-0.247
music	0.243	-0.247	0.243
restaurant	0.212	0.134	-0.005

Table 7 shows the concept preference pairs extracted using mJoachims-C method with the clickthrough in Table 1. The concept preference pairs obtained using Proposition 2 are input to the ranking SVM algorithm, same as in $P_{Joachims-C}$ described in Section 4.2, to create the user profile $P_{mJoachims-C}$ on the concepts c_1, c_2, ..., c_i extracted for the query q. Table 6 shows an example of feature weights resulted from RSVM Training for the query q = apple (where the user's topical preferences are "fruit" and "farm") using mJoachims-C method from our experiments.

4.4 SpyNB-C Method ($P_{SpyNB-C}$)

Both Joachims and mJoachims are based on a rather strong assumption that pages scanned but not clicked by the user are considered uninteresting to the user and hence irrelevant to the user's query. SpyNB does not make this assumption [15], but instead assumes that unclicked pages could be either relevant or irrelevant to the user. Therefore, SpyNB treats clicked

TABLE 7
Concept Preference Pairs obtained using mJoachims-C Method

Concept Preference Pairs for d_1	Concept Preference Pairs for d_5	Concept Preference Pairs for d_8
macintosh $<_{r'}$ product	apple store $<_{r'}$ product macintosh $<_{r'}$ product	macintosh $<_{r'}$ product catalog $<_{r'}$ product
macintosh $<_{r'}$ mac os	apple store $<_{r'}$ mac os macintosh $<_{r'}$ mac os	macintosh $<_{r'}$ mac os catalog $<_{r'}$ mac os
macintosh $<_{r'}$ apple store macintosh $<_{r'}$ iPod	macintosh $<_{r'}$ apple store apple store $<_{r'}$ iPod macintosh $<_{r'}$ iPod	macintosh $<_{r'}$ apple store catalog $<_{r'}$ apple store macintosh $<_{r'}$ iPod catalog $<_{r'}$ iPod
	apple store $<_{r'}$ fruit macintosh $<_{r'}$ fruit apple store $<_{r'}$ apple hill macintosh $<_{r'}$ apple hill	macintosh $<_{r'}$ fruit catalog $<_{r'}$ fruit macintosh $<_{r'}$ apple hill catalog $<_{r'}$ apple hill
	apple store $<_{r'}$ fruit macintosh $<_{r'}$ fruit	macintosh $<_{r'}$ fruit catalog $<_{r'}$ fruit

pages as positive samples and unclicked pages as unlabeled samples in the training process. The problem of finding user preferences becomes one of identifying from the unlabeled set reliable negative documents that are considered irrelevant to the user.

The "Spy" technique incorporates a novel voting procedure into a Naïve Bayes classifier [12] to derive reliable negative examples from the unlabeled set. Let "+" and "-" denote the positive and negative classes, and $D = d_1, d_2, ..., d_n$ a set of N documents in the search result list. For each search result, SpyNB first extracts the words that appear in the title, abstract and URL, creating a word vector $(w_1, w_2, ..., w_M)$. Then, a Naïve Bayes classifier is built by estimating the prior probabilities $(Pr(+)\ \ and\ \ Pr(-))$ and likelihoods $(Pr(w_j|+)\ and\ Pr(w_j|-))$. The detail of the Naïve Bayes Algorithm is presented in [15].

The training data only contains positive and unlabeled examples (without negative examples). Thus, the "Spy" technique is employed to learn a Naïve Bayes classifier. A set of positive examples S is selected from P and moved into U as "spies" to train a classifier using the Naïve Bayes algorithm above. The resulting classifier is then used to assign probabilities $Pr(+|d)$ to each example in $U \cup S$, and an unlabeled example in U is selected as a predicted negative example (PN) if its probability is less than T_s.

Unfortunately, in the search engine context, most users would only click on a few documents (positive examples) that are relevant to them. Thus, only a limited number of positive examples can be used in the classification process, lowering the reliability of the predicted negative examples (PN). To resolve the problem, every positive example p_i in P is used as a spy to train a Naïve Bayes classifier. Consequently, n predicted negative sets (PN_1,PN_2,...,PN_n) are created with the n Naïve Bayes classifiers. Finally, a voting procedure is used to combine the PN_i into the final PN. The detail of the SpyNB algorithm is discussed in [15].

After obtaining the positive and predicted negative samples from the SpyNB, page preferences can be obtained. As with Joachims-C and mJoachims-C, SpyNB-C generalizes page preferences into concept preferences. Specifically, concept preference pairs are obtained by assuming that concepts $C(d_j)$ in the positive sample d_j are more relevant than concept $C(d_i)$ in the predicted negative sample d_j (i.e., $C(d_j) <_{r'} C(d_i)$). Finally, RSVM training, which is similar to the one used in Joachims-C method, is applied on the extracted concept preferences to learn a user profile $P_{SpyNB-C}$ which is represented as a set of weight features. Table 6 shows an example of feature weights obtained from RSVM training in our experiment for the query q = apple (where the user's topical preferences are "fruit" and "farm" using the SpyNB-C method.

4.5 Click+Joachims-C Method ($P_{Click+Joachims-C}$)

In our previous work [11], we observed that P_{Click} is good in capturing user's positive preferences. In this paper, we integrate the click-based method, which captures only positive preferences, with the Joachims-C method, with which negative preferences can be obtained. We found that Joachims-C is good in predicting users' negative preferences. Since both the user profiles P_{Click} and $P_{Joachims-C}$ are represented as weighted concept vectors, the two vectors can be combined using the following formula:

$$\begin{aligned} w(C+J)_{c_i} = w(C)_{c_i} + w(J)_{c_i} \quad & if\ w(J)_{c_i} < 0 \\ w(C+J)_{c_i} = w(C)_{c_i} \quad & otherwise \end{aligned} \tag{12}$$

where $w(C+J)_{c_i} \in P_{Click+Joachims-C}$, $w(C)_{c_i} \in P_{Click}$, and $w(J)_{c_i} \in P_{Joachims-C}$. If a concept c_i has a negative weight in $P_{Joachims-C}$ (i.e., $w(J)_{c_i} < 0$), the negative weight will be added to $w(C)_{c_i}$ in P_{Click} (i.e., $w(J)_{c_i} + w(C)_{c_i}$) forming the weighted concept vector for the hybrid profile $P_{Click+Joachims-C}$.

4.6 Click+mJoachims-C Method ($P_{Click+mJoachims-C}$)

Similar to Click+Joachims-C method, a hybrid method which combines P_{Click} and $P_{mJoachims-C}$ is proposed. The two profiles are combined using the following formula

$$\begin{aligned} w(C+mJ)_{c_i} &= w(C)_{c_i} + w(mJ)_{c_i} \quad if\ w(mJ)_{c_i} < 0 \\ w(C+mJ)_{c_i} &= w(C)_{c_i} \quad otherwise \end{aligned} \tag{13}$$

where $w(C+mJ)_{c_i} \in P_{Click+mJoachims-C}$, $w(C)_{c_i} \in P_{Click}$, and $w(mJ)_{c_i} \in P_{mJoachims-C}$. If a concept c_i has a negative weight in $P_{mJoachims-C}$ (i.e., $w(mJ)_{c_i} < 0$), the negative weight will be added to $w(C)_{c_i}$ in P_{Click} (i.e., $w(mJ)_{c_i} + w(C)_{c_i}$) forming the weighted concept vector for the hybrid profile $P_{Click+mJoachims-C}$.

4.7 Click+SpyNB-C Method ($P_{Click+SpyNB-C}$)

Similar to Click+Joachims-C and Click+mJoachims-C methods, the following formula is used to create a hybrid profile $P_{Click+SpyNB-C}$ that combines P_{Click} and $P_{SpyNB-C}$:

$$\begin{aligned} w(C+sNB)_{c_i} &= w(C)_{c_i} + w(sNB)_{c_i} \quad if\ w(sNB)_{c_i} < 0 \\ w(C+sNB)_{c_i} &= w(C)_{c_i} \quad otherwise \end{aligned} \tag{14}$$

where $w(C+sNB)_{c_i} \in P_{Click+SpyNB-C}$, $w(C)_{c_i} \in P_{Click}$, and $w(sNB)_{c_i} \in P_{SpyNB-C}$. If a concept c_i has a negative weight in $P_{SpyNB-C}$ (i.e., $w(sNB)_{c_i} < 0$), the negative weight will be added to $w(C)_{c_i}$ in P_{Click} (i.e., $w(sNB)_{c_i} + w(C)_{c_i}$) forming the weighted concept vector for the hybrid profile $P_{Click+SpyNB-C}$.

5 Experimental Results

In this section, we evaluate and analyze the seven concept-based user profiling strategies (i.e., P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$). Our previous work had already shown that concept-based profiles are superior to document-based profiles [11]. Thus, the evaluation between concept-based and document-based profiles is skipped in this paper. The seven concept-based user profiling strategies are compared using our personalized concept-based clustering algorithm [11]. In Section 5.1, we first describe the setup for clickthrough collection. The collected clickthrough data are used by the proposed user profiling strategies to create user profiles. We evaluate the concept preference pairs obtained from Joachims-C, mJoachims-C and SpyNB-C methods in Section 5.2. In Section 5.3, the seven concept-based user profiling strategies are compared and evaluated. Finally, in Section 5.4, we study the performance of a heuristic for determining the termination points of initial clustering and community merging based on the change of intra-cluster similarity. We show that user profiling methods that incorporate negative concept weights return termination points that are very close to the optimal points obtained by exhaustive search.

TABLE 8
Topical Categories of the Test Queries

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

TABLE 9
Statistics of the Collected Clickthrough Data

Number of users	100
Number of test queries	500
Number of unique queries	406
Number of queries assigned to each user	5
Number of URLs retrieved	47,543
Number of concepts retrieved	42,328
Number of unique URLs retrieved	36,567
Number of unique concepts retrieved	12,853
Maximum number of retrieved URLs for a query	100
Maximum number of extracted concepts for a query	168

5.1 Experimental Setup

The query and clickthrough data for evaluation are adopted from our previous work [11]. To evaluate the performance of our user profiling strategies, we developed a middleware for Google[3] to collect clickthrough data. We used 500 test queries, which are intentionally designed to have ambiguous meanings (e.g. the query "kodak" can refer to a digital camera or a camera film). We ask human judges to determine a standard cluster for each query. The clusters obtained from the algorithms are compared against the standard clusters to check for their correctness. 100 users are invited to use our middleware to search for the answers of the 500 test queries (accessible at [3]). To avoid any bias, the test queries are randomly selected from 10 different categories. Table 8 shows the topical categories in which the test queries are chosen from. When a query is submitted to the middleware, a list containing the top 100 search results together with the extracted concepts are returned to the users, and the users are required to click on the results they find relevant to their queries. The clickthrough data together with the extracted concepts are used to create the seven concept-based user profiles (i.e., P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$). The concept mining threshold is set to 0.03 and the threshold for creating concept relations is set to zero. We chose these small thresholds so that as many concepts as possible are included in the user profiles. Table 9 shows the statistics of the clickthrough data collected.

The user profiles are employed by the personalized clustering method to group similar queries together according to users' needs. The personalized clustering algorithm is a two-phase algorithm which composes of the initial clustering phase to cluster queries within the scope of each user, and then the

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

community merging phase to group queries for the community.

We define the optimal clusters to be the clusters obtained by the best termination strategies for initial clustering and community merging (i.e., steps 6 and 8 in Algorithm 1). The optimal clusters are compared to the standard clusters using standard precision and recall measures, which are computed using the following formulas:

$$precision(q) = \frac{|Q_{relevant} \bigcap Q_{retrieved}|}{|Q_{retrieved}|} \quad (15)$$

$$recall(q) = \frac{|Q_{relevant} \bigcap Q_{retrieved}|}{|Q_{relevant}|} \quad (16)$$

where q is the input query, $Q_{relevant}$ is the set of queries that exists in the predefined cluster for q, and $Q_{retrieved}$ is the set of queries generated by the clustering algorithm. The precision and recall from all queries are averaged to plot the precision-recall figures, comparing the effectiveness of the user profiles.

5.2 Comparing Concept Preference Pairs Obtained using Joachims-C, mJoachims-C and SpyNB-C Methods

In this Section, we evaluate the pairwise agreement between the concept preferences extracted using Joachims-C, mJoachims-C and SpyNB-C methods. The three methods are employed to learn the concept preference pairs from the collected clickthrough data as described in Section 5.1. The learned concept preference pairs from different methods are manually evaluated by human evaluators to derive the fraction of correct preference pairs. We discard all the ties in the resulted concept preference pairs (i.e., pairs with the same concepts) to avoid ambiguity (i.e., both $c_i > c_j$ and $c_j > c_i$ exist) in the evaluation.

Table 10 shows the precisions of the concept preference pairs obtained using Joachims-C, mJoachims-C and SpyNB-C methods. The precisions obtained from the 10 different users together with the average precisions are shown. We observe that the performance of Joachims-C and mJoachims-C is very close to each other (average precision for Joachims-C method = 0.5965, mJoachims-C method = 0.6130), while SpyNB-C (average precision for SpyNB-C method = 0.6925) outperforms both Joachims-C and mJoachims-C by 13-16%. SpyNB-C performs better mainly because it is able to discover more accurate negative samples (i.e., results that do not contain topics interesting to the user). With more accurate negative samples, a more reliable set of negative concepts can be determined. Since the set of positive samples (i.e., the clicked results) are the same for all of the three methods, the method (i.e., SpyNB-C) with a more reliable set of negative samples/concepts would outperform the others. RSVM is then employed to learn user profiles from the concept preference pairs. The performance of the resulted user profiles will be compared in Section 5.3.

TABLE 10
Average Precisions of Concept Preference Pairs Obtained using Joachims-C, mJoachims-C and SpyNB-C Methods

	Average Precision
Joachims-C	0.5965
mJoachims-C	0.6130
SpyNB-C	0.6925

5.3 Comparing P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$

Figure 3 shows the precision and recall values of $P_{Joachims-C}$ and $P_{Click+Joachims-C}$ with P_{Click} shown as the baseline. Likewise, Figures 4 and 5 compare, respectively, the precision and recall of $P_{mJoachims-C}$ and $P_{Click+mJoachims-C}$, and that of $P_{SpyNB-C}$ and $P_{Click+SpyNB-C}$, with P_{Click} as the baseline.

An important observation from these three figures is that even though $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ are able to capture users' negative preferences, they yield worse precision and recall ratings comparing to P_{Click}. This is attributed to the fact that $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ share a common deficiency in capturing users' *positive* preferences. A few wrong positive predictions would significantly lower the weight of a positive concept. For example, assume that a positive concept c_i has been clicked many times, a preference $c_j <_{r'} c_i$ can still be generated by Joachims/mJoachims propositions, if there ever exists one case in which the user did not click on c_i but clicked on another document that was ranked lower in the result list. Since $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ cannot effectively capture users' positive preferences, they perform worse than the baseline method P_{Click}. On the other hand, P_{Click} captures positive preferences based on user clicks, so

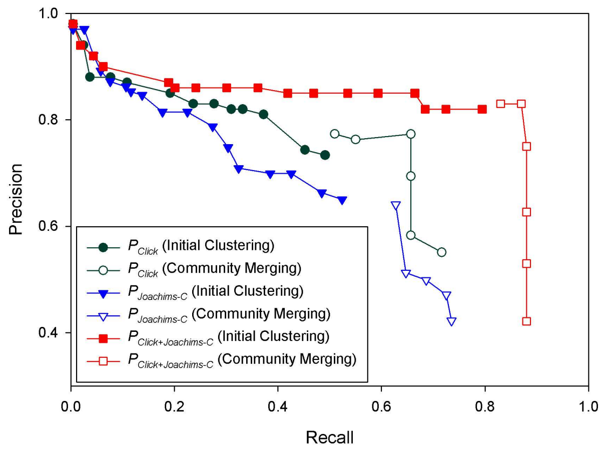


Fig. 3. Precision vs recall when performing personalized clustering using P_{Click}, $P_{Joachims-C}$, and $P_{Click+Joachims-C}$.

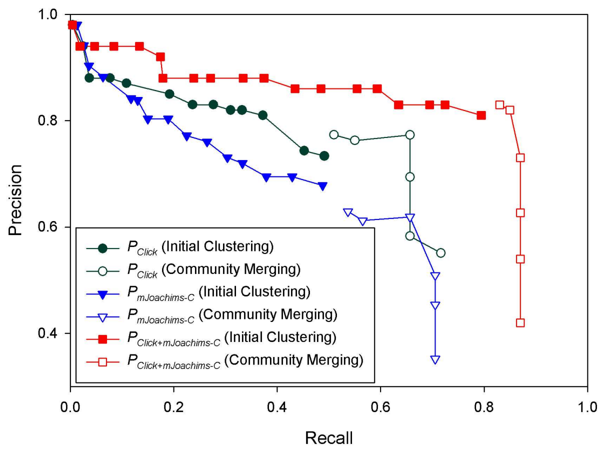


Fig. 4. Precision vs recall when performing personalized clustering using P_{Click}, $P_{mJoachims-C}$, and $P_{Click+mJoachims-C}$.

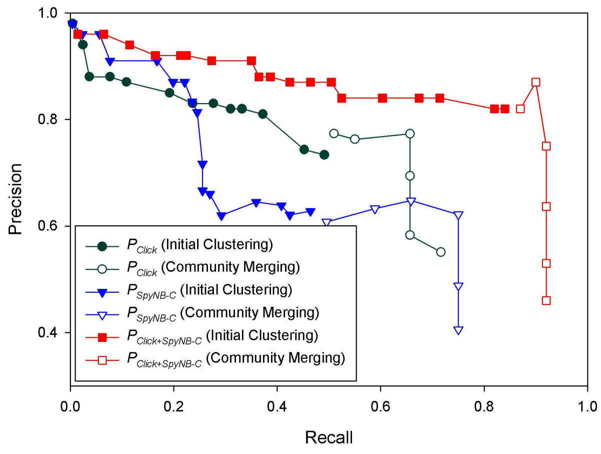


Fig. 5. Precision vs recall when performing personalized clustering using P_{Click}, $P_{SpyNB-C}$, and $P_{Click+SpyNB-C}$.

TABLE 11
Best F-Measure Values when Performing Personalized Clustering using P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$

	Precision	Recall	F-Measure
P_{Click}	0.7726	0.6567	0.7100
$P_{Joachims-C}$	0.6408	0.6276	0.6339
$P_{mJoachims-C}$	0.6191	0.6565	0.6373
$P_{SpyNB-C}$	0.6217	0.75	0.6798
$P_{Click+Joachims-C}$	0.8300	0.8700	0.8495
$P_{Click+mJoachims-C}$	0.8200	0.8500	0.8347
$P_{Click+SpyNB-C}$	0.8700	0.900	0.8847

an erroneous click made by users has little effect on the final outcome as long as the number of erroneous clicks is much less than that of correct clicks.

Although $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ are not ideal for capturing user's positive preferences, they can capture negative preferences from users' clickthroughs very well. For example, assume that a concept c_i has been skipped by a user many times, preferences $c_{k1} <_{r'} c_i$, $c_{k2} <_{r'} c_i$, ..., $c_{kn} <_{r'} c_i$ (where c_{k1}, c_{k2}, ..., c_{kn} are the clicked concepts below c_i) would be generated by these methods. Hence, the concept c_i would be considered less relevant than the clicked concepts c_{k1}, c_{k2}, ..., c_{kn} and assigned a lower or even negative weight.

Since $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ are able to capture negative preferences from users' clickthroughs while P_{Click} is good at capturing positive preferences, we propose three user profiling strategies, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$, to integrate the predicted negative preferences from $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ with the positive preferences from P_{Click}. In Figures 3, 4 and 5, we observe that $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$ produce *significantly* better precision and recall ratings than that of P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$. From the F-measure values in Table 11, we can observe that $P_{Click+SpyNB-C}$ performs the best with an improvement of 25% over the baseline P_{Click}; $P_{Click+Joachims-C}$ and $P_{Click+mJoachims-C}$ tie at the second position, with improvement of 18-20% over the baseline. As discussed in Section 5.2, SpyNB-C produces a more reliable set of negative concepts compared to the others. With a more accurate set of negative preferences, $P_{Click+SpyNB-C}$ achieves better precision and recall results comparing to $P_{Click+Joachims-C}$ and $P_{Click+mJoachims-C}$.

The performance results support our belief that the three integrated user profiles benefit from the positive preferences of P_{Click} that help to group similar queries together and negative preferences derived from Joachims/mJoachims/SpyNB method that help to separate dissimilar queries into different clusters. Thus, they achieve better precision and recall results compared to P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$. Finally, the precisions of all methods drop sharply if community merging is over-performed. Initial clustering is employed to prepare the query clusters within the scope of each individual user. Community merging is then employed to merge the similar clusters resulted from initial clustering across different users. If two big clusters from initial clustering are wrongly merged because over-performing community merging, the precision will drop sharply without improving recall. Thus, a good terminating point is required for community merging to improve the recall, while maintaining good precision. Section 5.4 provides the details on how to obtain such a terminating point.

5.4 Termination Points for Individual Clustering to Community Merging

As initial clustering is run, a tree of clusters will be built along the clustering process. The termination point for initial clustering can be determined by finding the point at which the cluster quality has reached its highest (i.e., further clustering steps would decrease the quality). The same can be done for determining the termination point for community merging. The change in cluster quality can be measured by $\triangle Similarity$, which is the change in the similarity value of the two most similar clusters in two consecutive steps. For efficiency reason, we adopt the single-link approach to measure cluster similarity. As such, the similarity of two cluster is the same as the similarity between the two most similar queries across the two clusters. Formally, $\triangle Similarity$ is defined as:

$$\triangle Similarity(i) = sim_i(P_{q_m}, P_{q_n}) - sim_{i+1}(P_{q_o}, P_{q_p}) \tag{17}$$

where q_m and q_n are the two most similar queries in the i^{th} step of the clustering process, $P(q_m)$ and $P(q_n)$ are the concept-based profiles for q_m and q_n, q_o and q_p are the two most similar queries in the $i+1^{th}$ step of the clustering process, $P(q_o)$ and $P(q_p)$ are the concept-based profiles for q_m and q_n, and $sim()$ is the cosine similarity. Note that a positive $\triangle Similarity$ means that step $i+1$ is producing worse clusters than that of step i.

In our previous work [11], it is not easy to determine where to cut the clustering tree in P_{Click}, because the similarity values decrease uniformly during the clustering process. Figures 6, 7, 8 and 9 show the change in similarity values when performing initial clustering and community merging of the personalized clustering algorithm using P_{Click}, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$,

In Figure 6, we can observe that similarity decreases quite uniformly in P_{Click}. The uniform decrease in similarity values from P_{Click} makes it difficult for the clustering algorithm to determine the termination points for initial clustering and community merging (the triangles are the optimal termination points for initial clustering to community merging).

We observe from the figures that $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$ each exhibits a clear peak in the initial clustering process. It means that at

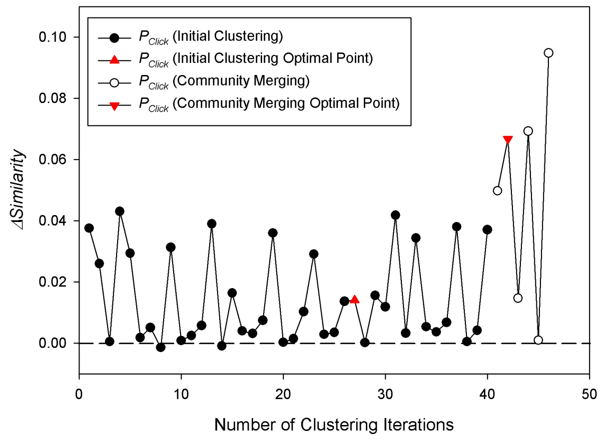


Fig. 6. Change in similarity values when performing personalized clustering using P_{Click}.

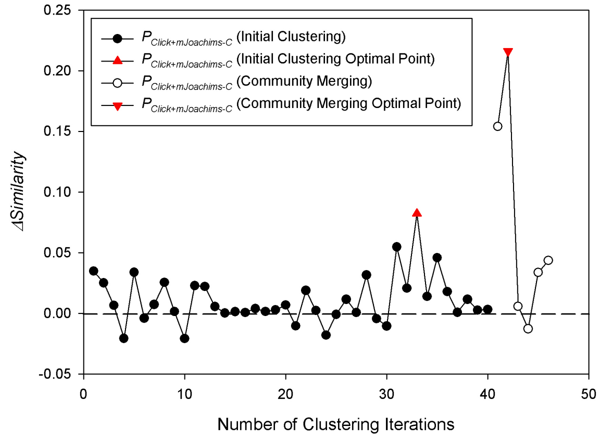


Fig. 8. Change in similarity values when performing personalized clustering using $P_{Click+mJoachims-C}$.

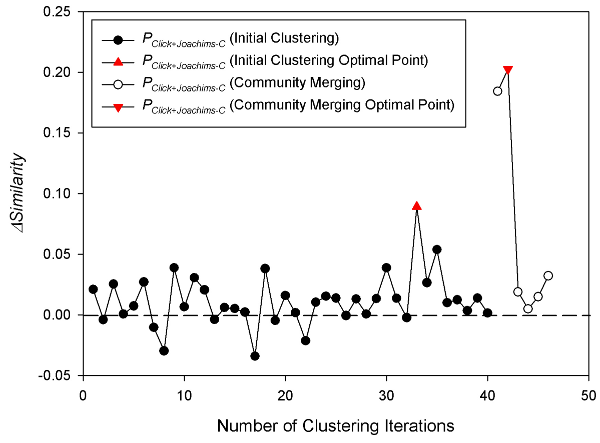


Fig. 7. Change in similarity values when performing personalized clustering using $P_{Click+Joachims-C}$.

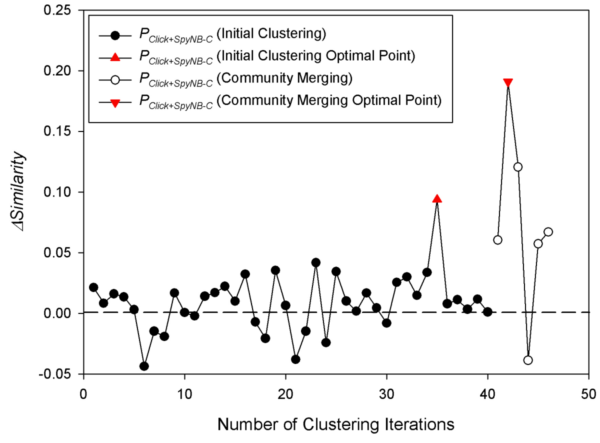


Fig. 9. Change in similarity values when performing personalized clustering using $P_{Click+SpyNB-C}$.

TABLE 12
Comparison of Distances, Precision and Recall Values at Algorithmic and Manually Determined Optimal Points

	Manually Determined			Algorithmic		
	Terminating Step#	Precision	Recall	Terminating Step#	Precision	Recall
$P_{Click+Joachims-C}$ Initial Clustering **(IC)**	34	0.8200	0.7244	33	0.8200	0.6844
$P_{Click+Joachims-C}$ Community Merging **(CM)**	3	0.8300	0.8700	2	0.8300	0.8300
$P_{Click+mJoachims-C}$ **(IC)**	34	0.8300	0.7244	33	0.8300	0.6944
$P_{Click+mJoachims-C}$ **(CM)**	3	0.8200	0.8500	2	0.8300	0.8300
$P_{Click+SpyNB-C}$ **(IC)**	36	0.8200	0.8200	36	0.8400	0.7144
$P_{Click+SpyNB-C}$ **(CM)**	3	0.8700	0.9000	2	0.8200	0.8700

TABLE 13
Example of P_{Click} and $P_{Click+Joachims-C}$ for Two Different Users

	"info"	"computer"	"banana"	"fruit"
P_{Click} apple(u_1)	1	1	0	0
P_{Click} apple(u_2)	1	0	0	1
$P_{Click+Joachims-C}$ apple(u_1)	1	1	-1	-1
$P_{Click+Joachims-C}$ apple(u_2)	1	-1	0	1

TABLE 14
Average Similarity Values for Similar/Dissimilar Queries Computed using P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$

	Similar Queries	Dissimilar Queries
P_{Click}	0.3217	0.0746
$P_{Joachims-C}$	0.1056	-0.0154
$P_{mJoachims-C}$	0.1143	-0.0032
$P_{SpyNB-C}$	0.1044	-0.0059
$P_{Click+Joachims-C}$	0.2546	0.0094
$P_{Click+mJoachims-C}$	0.2487	0.0087
$P_{Click+SpyNB-C}$	0.2673	0.0091

the peak the quality of the clusters is highest but further clustering steps beyond the peak would combine dissimilar clusters together. Compared to P_{Click}, the peaks in these three methods are much more clearly identifiable, making it easier to determine the termination points for initial clustering and community merging.

In Figures 7, 8 and 9, we can see that the similarity values obtained using $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$ decrease sharply at the optimal points (the triangles in Figure 7, 8 and 9. The decrease in similarity values is due to the negative weights in the user profiles, which help to separate the similar and dissimilar queries into distant clusters. Dissimilar queries would get lower similarity values because of the different signed concept weights in the user profiles, while similar queries would get high similarity values as they do in P_{Click}. Table 12 show the distances between the manually determined optimal points and the algorithmic optimal points, and the comparison of the precision and recall values at the two different optimal points. We observe that the algorithmic optimal points for initial clustering and community merging usually are only one step away from the manually determined optimal points. Further, the the precision and recall values obtained at the algorithmic optimal points are only slightly lower than those obtained at the manually determined optimal points.

The example in Table 13 helps illustrate the effect of negative concept weights in the user profiles. Table 13 shows an example of two different profiles for the query "apple" from two different users u_1 and u_2, where u_1 is interested in information about "apple computer" and u_2 is interested in information about "apple fruit". With only positive preferences (i.e., P_{Click}), the similarity values for "apple(u_1)" and "apple(u_2)" is 0.5, showing the rather high similarity of the two queries. However, with both positive and negative preferences (i.e., $P_{Click+Joachims-C}$), the similarity value becomes -0.2886, showing that the two queries are actually different even when they share the common "noise" concept "info". With a larger separation between the similar and dissimilar queries, the cutting point can be determined easily by identifying the place where there is a sharp decrease in similarity values.

To further study the effect of the negative concept weights in the user profiles, we reverse the experiments by first grouping similar queries together according to the predefined clusters, and then compute the average similarity values for pairs of queries within the same cluster (i.e., similar queries) and pairs of queries not in the same cluster (i.e., dissimilar queries) using P_{Click}, $P_{Joachims-C}$, $P_{mJoachims-C}$, $P_{SpyNB-C}$, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$. The results are shown in Table 14. We observe that P_{Click} achieves a high average similarity value (0.3217) for similar queries, showing that the positive preferences alone from P_{Click} are good for identifying similar queries. $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$ achieve negative average similarity values (-0.0154, -0.0032 and -0.0059) for dissimilar queries. They are good in predicting negative preferences to distinguish dissimilar queries. However, as stated in Section 5.3, the wrong positive predictions significantly lower the correct positive preferences in the user profiles, and thus lowering the average similarities (0.1056, 0.1143 and 0.1044) for similar queries. $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$ achieve high average similarity values (0.2546, 0.2487 and 0.2673) for similar queries, but low average similarities (0.0094, 0.0087 and 0.0091) for dissimilar queries. They benefit from both the accurate positive preferences of P_{Click}, and the correctly predicted negative preferences from $P_{Joachims-C}$, $P_{mJoachims-C}$ and $P_{SpyNB-C}$. Thus, $P_{Click+Joachims-C}$, $P_{Click+mJoachims-C}$ and $P_{Click+SpyNB-C}$ perform the best in the personalized clustering algorithm among all the proposed user profiling strategies.

6 Conclusions

An accurate user profile can greatly improve a search engine's performance by identifying the information needs for individual users. In this paper, we proposed and evaluated several user profiling strategies. The techniques make use of clickthrough data to extract from web-snippets to build concept-based user profiles automatically. We applied preference mining rules to infer not only users' positive preferences but their negative preferences, and utilized both kinds of preferences in deriving users profiles. The user profiling strategies were evaluated and compared with the personalized query clustering method that we proposed previously. Our experimental results show that profiles capturing both of the user's positive and negative preferences perform the best among the user profiling strategies studied. Apart from improving the quality of the resulting clusters, the negative preferences in the proposed user profiles also help to separate similar and dissimilar queries into distant clusters, which helps to determine near-optimal terminating points for our clustering algorithm.

We plan to take on the following two directions for future work. First, relationships between users can be mined from the concept-based user profiles to perform collaborative filtering. This allows users with the same interests to share their profiles. Second, the existing user profiles can be used to predict the intent of unseen queries, such that when a user submits a new query, personalization can benefit the unseen query. Finally, the concept-based user profiles can be integrated into the ranking algorithms of a search engine so that search results can be ranked according to individual users' interests.

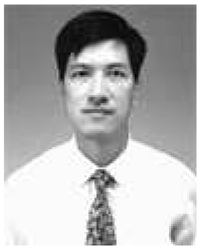
Acknowledgments

This work was supported by grants 615806 and CA05/06.EG03 from Hong Kong Research Grant Council. We would like to express our sincere thanks to the editors and the reviewers for giving very insightful and encouraging comments.

References

[1] E. Agichtein, E. Brill, and S. Dumais, "Improving web search ranking by incorporating user behavior information," in *Proc. of ACM SIGIR Conference*, 2006.
[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning user interaction models for predicting web search result preferences," in *Proc. of ACM SIGIR Conference*, 2006.
[3] Appendix: 500 test queries. [Online]. Available: http://www.cse.ust.hk/~dlee/tkde09/Appendix.pdf
[4] R. Baeza-yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," vol. 3268, pp. 588–596, 2004.
[5] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proc. of ACM SIGKDD Conference*, 2000.
[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proc. of the International Conference on Machine learning (ICML)*, 2005.
[7] 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*, 1991.
[8] Z. Dou, R. Song, and J.-R. Wen, "A largescale evaluation and analysis of personalized search strategies," in *Proc. of WWW Conference*, 2007.
[9] S. Gauch, J. Chaffee, and A. Pretschner, "Ontology-based personalized search and browsing," *ACM WIAS*, vol. 1, no. 3-4, pp. 219–234, 2003.
[10] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. of ACM SIGKDD Conference*, 2002.
[11] K. W.-T. Leung, W. Ng, and D. L. Lee, "Personalized concept-based clustering of search engine queries," *IEEE TKDE*, vol. 20, no. 11, 2008.
[12] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *Proc. of the International Conference on Machine Learning (ICML)*, 2002.
[13] F. Liu, C. Yu, and W. Meng, "Personalized web search by mapping user queries to categories," in *Proc. of the International Conference on Information and Knowledge Management (CIKM)*, 2002.
[14] Magellan. [Online]. Available: http://magellan.mckinley.com/
[15] W. Ng, L. Deng, and D. L. Lee, "Mining user preference using spy voting for search engine personalization," *ACM TOIT*, vol. 7, no. 4, 2007.
[16] Open directory project. [Online]. Available: http://www.dmoz.org/
[17] M. Speretta and S. Gauch, "Personalized search based on user search histories," in *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.
[18] Q. Tan, X. Chai, W. Ng, and D. Lee, "Applying co-training to clickthrough data for search engine adaptation," in *Proc. of DASFAA Conference*, 2004.
[19] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Query clustering using user logs," *ACM TOIS*, vol. 20, no. 1, pp. 59–81, 2002.
[20] Y. Xu, K. Wang, B. Zhang, and Z. Chen, "Privacy-enhancing personalized web search," in *Proc. of WWW Conference*, 2007.

Kenneth Wai-Ting Leung received the BSc degree in computer science from the University of British Columbia, 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 at the Hong Kong University of Science and Technology. His research interests include search engines, Web mining, information retrieval, and ontologies.



Dik Lun Lee received the MS and PhD degrees in computer science from the University of Toronto, Canada, and the B.Sc. degree in Electronics from the Chinese University of Hong Kong. He is currently a professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. He was an associate professor in the Department of Computer Science and Engineering at the Ohio State University, USA. 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.