ISSN:0975-9646

V.K.Priyanka Kolluri et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (5) , 2012,5116 - 5122

Efficient Personalized Search using Ranking SVM

V.K.Priyanka Kolluri, A.Bala Ram

Department of CSE,
CMR Institute of Technology,
Hyderabad,Andhra Pradesh,
India.

***Abstract* -The main problem that the web search currently faces is that search queries are short and ambiguous and thus are unable to meet user needs. To avoid such problems, some search engines suggest terms that are meaningfully 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. Experimental results show that our approach has better precision and recall than the existing query clustering methods.**

***Keywords*—Click-through , ranking , personalization , concept-based clustering , query clustering**

I. INTRODUCTION

Now-a-days the amount of information on the web is increasing rapidly, which has become increasingly harder for the web search engines to get the information that satisfies the user's individual interests. In general, the queries given by the user are shorter in length. This may lead to ambiguity for the web search engine to retrieve the results for a particular user query. Therefore, lots of retrieved results may not match the interests of the users.

Most major commercial search engines provide query suggestions to help users make more effective queries which improve user's search experience. Whenever a user gives a query, a list of terms that are meaningfully related to the submitted query is provided to help the user identify terms that they really wants, hence improving the effectiveness of retrieval. Yahoo's "Also Try" and Google's "Searches related to" features provide related queries for narrowing search, while Ask Jeeves 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.

Personalized search is a promising way to improve search quality by customizing search results for people with different requirements. Many recent research efforts have focused on this area. Most of them could be categorized into two general approaches: Re-ranking query results returned by search engines locally using personal information; or sending personal information and queries together to the search engine. 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 user's 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.

In this paper, a method that provides personalized query suggestions based on a personalized concept-based clustering was proposed. Our methods use the click through data to estimate user's conceptual preferences and then provide 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 certainly helps users formulate more effective queries according to their needs.

The identified problems in the existing system are as follows:

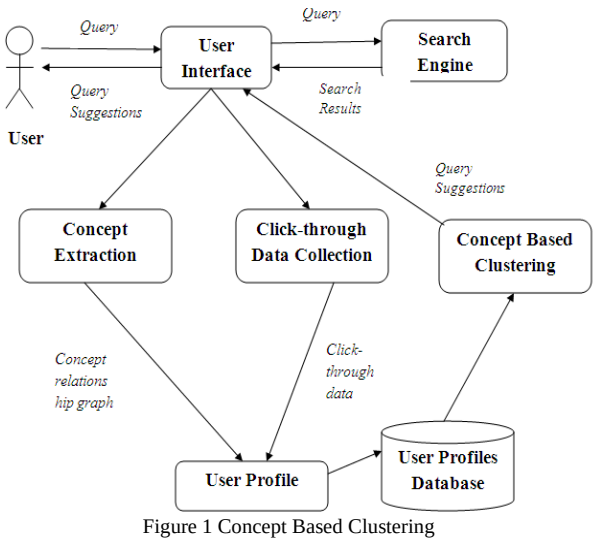

Figure 1 Concept Based Clustering

- Many personalization methods present now are based on creation of one single user profile for a user and use the same profile for all his/her searches which does not give relevant results for all queries. Personalization strategies such as [1],[2],[4],[6] employed a single large user profile for each user in the process.

- Existing click through-based strategies can be categorized into document-based and concept based approaches. Document-based profiling methods try to estimate users' document preferences[1],[2],[4].On the other hand, concept based profiling methods aim to derive topics or concepts that users are highly interested in[6].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 user's dislike (i.e., users' negative preferences).

The approach we followed consists of four steps. First, when a user submits a query, concepts and their relations are mined online from web-snippets to build a concept relationship graph. Second, click-through data are collected to predict user's conceptual preferences. Third, the concept relationship graph together with the user's conceptual preferences is used as input to a concept-based clustering algorithm that finds conceptually close queries based on the user profiles. Finally, the most similar queries are suggested to the user for search refinement. Fig. 1 shows the general process of our approach.
The main contributions of this paper are as follows:

- In order to consider both users' positive and negative preferences in building users profiles, we extend the query-oriented, concept-based user profiling method proposed [5].
- While document-based user profiling methods pioneered by Joachim's capture users' document preferences, our methods are based on users' concept preferences[3].
- Our proposed methods use an RSVM to learn from concept preferences weighted concept vectors representing concept-based user profiles [5]. The weights of the vector elements, which could be positive or negative, represent the interestingness (or uninterestingness) of the user on the concepts.

II. RELATED WORK

User Profiling methods are basically categorized into two types. They are as follows

1) Document Based Method
2) Concept Based Method
3)

Document based user profiling method mainly depends on the user's past searches and also the previous clicks that he has made on the web snippets. User's document preferences are first extracted from the click through data and then user behavior model for that particular user will be created [1],[2],[3],. Table 1 is an example for the query "apple" given to a search engine which gave a result set with ranking as shown below. According to Joachim's theory user scans the search result and clicks some of the results. Consider the results from Table 1where the bolded columns are clicked by the user.

Documents	***Results***	***Concepts Extracted***
A	Apple Computer	Macintosh
B	Apple Support	Product
C	Apple Inc. official downloads	Mac OS
D	Macintosh Product Guide	Macintosh catalog
E	The Apple Store	Apple store, macintosh
F	Apple Hill Growers	Fruit, apple hill
G	Apple Crops	Fruit
H	Apple Store (U.S)	Apple store, ipod

TABLE 1 EXAMPLE OF CLICK-THROUGH FOR QUERY "APPLE"

Preference Pairs of A	***Preference Pairs of D***	***Preference Pairs of H***
Empty set	D has more preference than B	H has more preference than B
	D has more preference than C	H has more preference than C
		H has more preference than E
		H has more preference than F
		H has more preference than G

TABLE 2 DOCUMENT PREFERENCE PAIRS OBTAINED USING JOACHIMS' METHOD

If user does not click on document C and user clicks on document D then the conclusion can be drawn saying user prefers document D more than document C. Using Joachim's method and the example click-through 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, an RSVM [3] is employed to learn the user behavior model as a set of weighted features.
Concept-based user profiling methods aim at capturing users' conceptual needs. Concept based methods basically derive user's major interests by exploring user's browsed documents and search histories. 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.
In the method proposed by Liu et al [6] 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.
Liu et al. use reference ontology(e.g., ODP) [8] to develop the hierarchical user profiles, while Xuet al. automatically extract possible topics from users' browsed documents and organize the topics into hierarchical structures. The major advantage of dynamically building atopic hierarchy is that new topics can be easily recognized and extracted from documents and added to the topic hierarchy, whereas reference ontology such as ODP is not always up-to-date.

III. BASIC IDEA OF RSVM

Support Vector Machines (SVMs) have been extensively researched in the data mining and machine learning communities for the last decade and actively applied to applications in various domains. SVMs are typically used for learning classification, regression, or ranking functions, for which they are called classifying SVM, support vector regression (SVR), or ranking SVM (or RankSVM) respectively.

Ranking SVM is one of the pair-wise ranking methods, which is used to adaptively sort the web-pages by their relationships (how relevant) to a specific query. A mapping function is required to define such relationship. The mapping function projects each data pair (inquire and clicked web-page) onto a feature space. These features combined with user's click-through data (which implies page ranks for a specific query) can be considered as the training data for machine learning algorithms.

Generally, Ranking SVM includes three steps in the training period:

1. It maps the similarities between queries and the clicked pages onto certain feature space.
2. It calculates the distances between any two of the vectors obtained in step 1.
3. It forms optimization problem which is similar to SVM classification and solve such problem with the regular SVM solver.

IV. CLICK BASED METHOD

The concepts that are retrieved for a particular query using the concept extraction method described below gives the possible concept space arising from the query. This concept space covers more than what the user actually wants. For example, when the user searches for the query "apple," the concept space shown in Figure 2 derived from our concept extraction method contains the concepts "macintosh," "ipod," and "fruit." If the user is interested in "apple" 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 concepts which have similar meaning, 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_{ci} on the extracted concepts c_i, when a Web-snippet s_j is clicked by the user (denoted by click(s_j)):

$$click(s_j) => \forall c_j \in s_j, w_{c_i} = w_{c_i} + 1 \quad \text{..... (1)}$$

$$click(s_j) => \forall c_j \in s_j, w_{c_i} = w_{c_i} + sim_R(c_i, c_j)\ if\ sim_R(c_i, c_j) > 0 \quad \text{.....(2)}$$

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 by1. 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 (2). Fig. 3 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} ofthe concepts "mac os," "software," "apple store," "iPod,""iPhone," and "hardware" are increased based on (2),because they are related to the concept "macintosh." The weights w_{c_i} for concepts "fruit," "apple farm," "juice,"

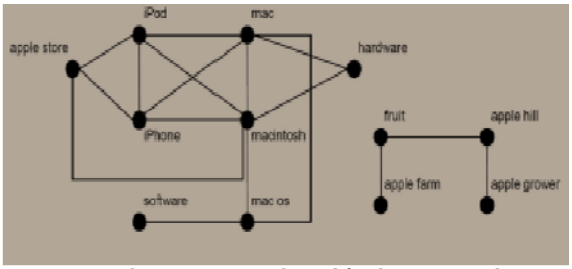

Figure 2 The concept space derived for the query "apple."

and "apple grower" remain zero, showing that the user is not interested in information about "apple fruit."

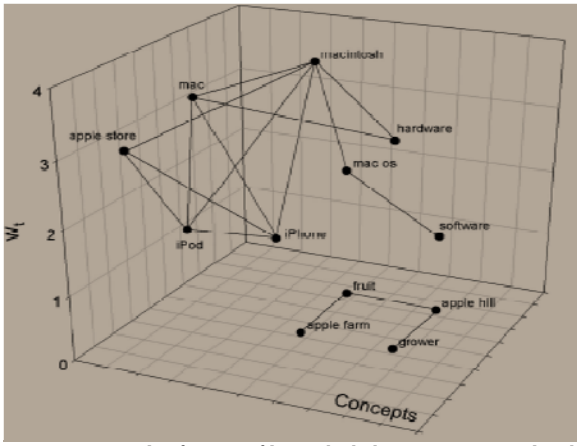

Figure 3 An example of user profile in which the user is interested in the concept "macintosh."

V. EXTRACTING CONCEPTS

Concept extraction is mainly divided into three steps. 1) Concept extraction from the results of a search query by search engine. 2) Finding concept relations 3) creating a user concept preference profile using the extracted concepts, concept relations, and user's click-throughs.

A. Concept Extraction with RSVM using web-snippets

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. The web-snippets that are returned uses RSVM which is one of the best pair-wise ranking methods, which is used to adaptively sort the web-pages by their relationships (how relevant) to a specific query. 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 ti with respect to the returned web-snippets arising from a query q:

$$support(t_i) = \frac{sf(t_i)}{n}.|t_i| \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

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 thresholds (support(t_i)> s), we would treat t_i as a concept for the query q.

B. Finding concept relations

To find relations between concepts, we apply a well-known signal-to-noise ratio formula from data mining [7] to establish similarity between terms t_1 and t_2. The similarity value of Church and Hanks' formula always lies between[0, 1]

$$sim(t_1, t_2) = \frac{\frac{n.df(t_1 \cup t_2)}{df(t_1).df(t_2)}}{\log n} \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

where n is the number of documents in the corpus, df(t_1U 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).

C. Creating user concept preference profile

The concept relationship graph is first derived without taking user click-throughs into account. Initially, 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 click-throughs 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_{t_i}on the extracted concepts t_i when a clicked web-snippet s_j, denoted by click(s_j), is found as follows:

$$click(s_j) => \forall t_i \in s_j, w_{t_i} = w_{t_i} + 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$click(s_j) => \forall t_i \in s_j, w_{t_i} = w_{t_i} + sim_R(t_i, t_j)\ if\ sim_R(t_i, t_j) > 0 \quad \ldots\ldots\ldots\ldots\ldots (6)$$

Where s_j is a web-snippet, w_{t_i} 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_{t_i} in Fig. 3 represents the interestingness of the concepts to the user. The values of wti for "macintosh" and "mac" are highest because the users have interest in them (and the values of w_{t_i} are incremented using (5)). Indirectly, the values of w_{t_i} for "mac os," "software," "apple store," "iPod," "iPhone," and "hardware" are increased because they are related to "apple macintosh" and thus incremented using (6). 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.

D. Query Clustering Algorithm

We now review our personalized concept-based clustering algorithm 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 in which 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 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 (3)) 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 are shown in Algorithm 1 [10]. 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:

$$sim(x,y) = N_x.N_y / ||N_x||\ ||N_y|| \quad \ldots\ldots\ldots (7)$$

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 Gp
// Initial Clustering
1: Obtain the similarity scores in G for all possible pairs of query nodes using Equ. (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 Equ. (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 Equ. (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.

VI. RESULTS AND DISCUSSIONS

In this section, we evaluate the performance of the proposed clustering methods for obtaining related queries using user click-throughs. In Section A, we first describe the experimental setup for collecting the required click-through data. In Section B, we evaluate the effectiveness of our proposed personalized concept-based clustering (or simply called the P-QC method).

A. Experimental Setup

To collect the click-through data to evaluate our proposed methods, we implemented a Google middleware to track user clicks. Google was chosen as a common basis for comparing the performance of the methods under evaluation because it 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.
We invited 20 people to use the middleware to search 100 given test queries, which are accessible as in [9]. 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 3 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.

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

TABLE 3 CATEGORIES OF TEST QUERIES

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 click-through together with his/her concept preference profile are updated.
In the experiment (which will be described in Section A), 10 people 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 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 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 click-

through data collected are used to measure the performance of the personalized concept-based clustering method.

B. ***Personalized Concept-Based Clustering***

In the methods were 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 analyzed the performance of P-QC method using precision-recall figures and best F-measure values. Fig. 4 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. 4), 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. 4), community merging clearly boosts up the precision-recall envelop, which means that both precision and recall achieved in initial clustering are improved. This 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. 4), 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 precision ¼ 0:14 in Fig. 4).

Figs. 5 and 6 show the change of precision and recall when performing P-QC method. In Fig. 5, 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. 6, 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. 5 and 6 that only a small fraction of precision is used to trade for a much better recall in community merging.

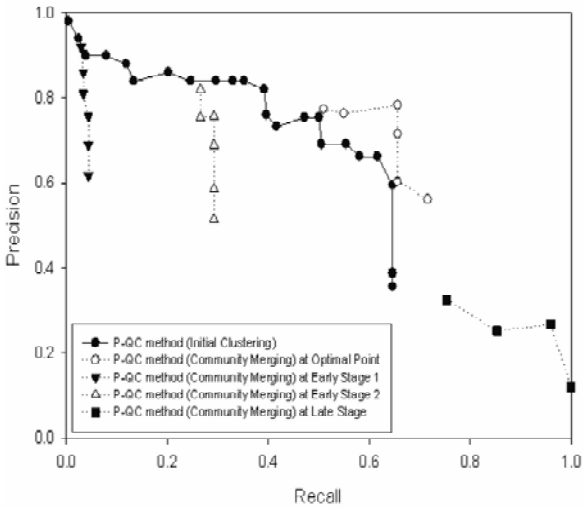

Figure 4 Precision versus recall while performing P-QC method

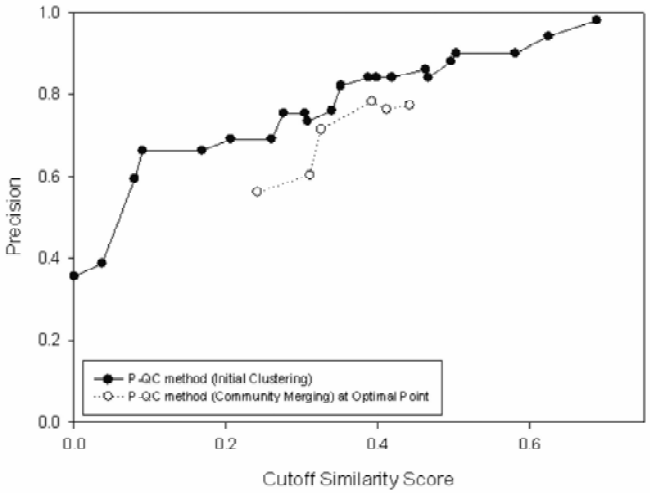

Figure 5 Change of precision when performing P-QC method

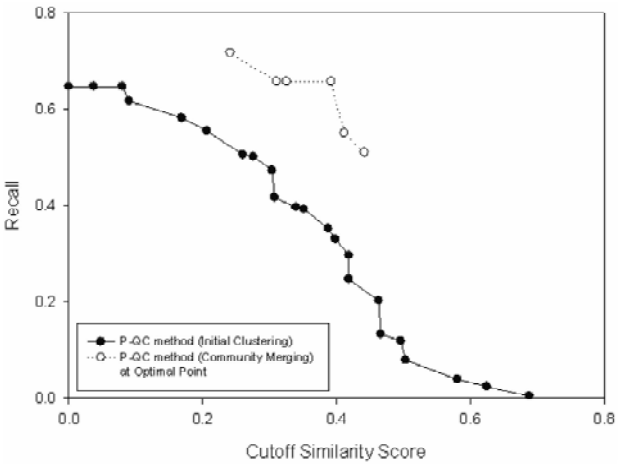

Figure 6 Change of recall when performing P-QC method

VII. CONCLUSION AND FUTURE WOK

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 click-through 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.

There are several directions for extending the work in the future. First, instead of considering only query-concept pairs in the click-through data, we can consider the relationships between users, queries, and concepts to obtain more personalized and accurate query suggestions. Second, click-through 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.

REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais, "Improving Web SearchRanking by Incorporating User Behavior Information," Proc. ACMSIGIR, 2006.

[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning UserInteraction Models for Predicting Web Search Result Preferences,"Proc. ACM SIGIR, 2006.

[3] T. Joachims, "Optimizing Search Engines Using ClickthroughData," Proc. ACM SIGKDD, 2002.

[4] Z. Dou, R. Song, and J.-R. Wen, "A Largescale Evaluation andAnalysis of Personalized Search Strategies," Proc. World Wide Web (WWW) Conf., 2007.

[5] K.W.-T. Leung, W. Ng, and D.L. Lee, "Personalized Concept-Based Clustering of Search Engine Queries," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 11, pp. 1505-1518, Nov. 2008.

[6] F. Liu, C. Yu, and W. Meng, "Personalized Web Search byMapping User Queries to Categories," Proc. Int'l Conf. Information and Knowledge Management (CIKM), 2002.

[7] K.W. Church, W. Gale, P. Hanks, and D. Hindle, "Using Statisticsin Lexical Analysis," Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon, U. Zernik, ed., Lawrence Erlbaum,1991.

[8] Open Directory Project, http://www.dmoz.org/, 2009.

[9] http://www.cse.ust.hk/~dlee/tkde08/query.html, 2008.

[10] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," Proc. ACM SIGKDD, 2000.