ISSN 2229-5518

Deriving Personalized Concept and Fuzzy Based User Profile from Search Engine Queries

Abstract — Personalized search is an important research area that aims to resolve the ambiguity of query terms. 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. To alleviate this problem, some search engines suggest terms that are semantically related to the submitted queries so that users can choose from the suggestions the ones that reflect their information needs. 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. A new two phase personalized agglomerative clustering algorithm and fuzzy clustering algorithm, which is able to generate personalized query clusters. Experimental results show that our approach has better precision and recall than the existing query clustering methods. So we can find an effective method 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. All above analysis is done on the basis of mathematical calculations.

Index Terms — Query log analysis, Clustering URL, Clustering queries, Clickthrough, Concept based clustering, Personalization, Fuzzy clustering, Search engine.

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

To improve user's search experience, most major commercial search engines provide query suggestions to help users formulate more effective queries. When a user submits a query, a list of terms that are semantically related to the submitted query is provided to help the user identify terms that he/she really wants, hence improving the retrieval effectiveness. Yahoo's "Also Try" 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.

It propose a method that provides personalized query suggestions based on a personalized concept-based clustering technique. In contrast to existing methods that provide the same suggestions to all users, our approach uses click through data to estimate user's conceptual

- *Rahul Verma is currently pursuing masters degree program in Information Technology engineering in M.I.T. Ujjain, R.G.P.V Universit., Bhopal, India PH-9926621928. E-mail: rahulv.8617@gmail.com*
- *Prof. Kshitij Pathak has ben working as faculty in the Information Technology,at MIT Ujjain (M.P.), E-mail: er.k.pathak @gmail.com*

preferences and then provides personalized query suggestions for each individual user according to his/her conceptual needs. The motivation of our research is that queries submitted to a search engine may have multiple meanings. For example, depending on the user, the query "apple" may refer to a fruit, the company Apple Computer or the name of a person, and so forth. Thus, providing personalized query suggestion (e.g., users interested in "apple" as a fruit get suggestions about fruit, while users interested in "apple" as a company get suggestions about the company's products) certainly helps users formulate more effective queries according to their needs. The underlying idea of our proposed technique is based on concepts and their relations extracted from the submitted user queries, the web-snippets,1 and the click through data. Click through data was exploited in the personalized clustering process to identify user preferences: A user clicks on a search result mainly because the web snippet contains a relevant topic that the user is interested in. Moreover, clickthrough data can be collected easily without imposing extra burden on users, thus providing alow-cost means to capture user's interest.

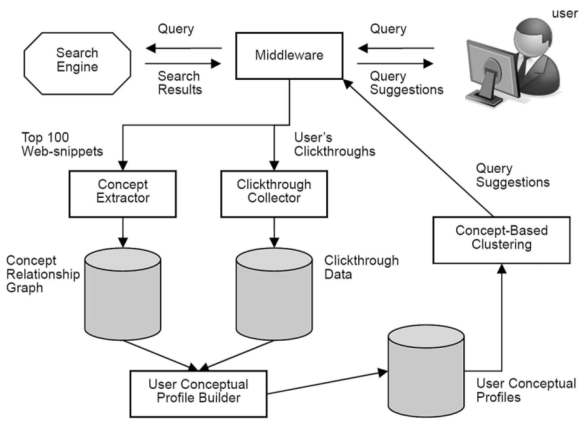

Figure 1: The general process of concept-based clustering.

It approach consists of the following four major steps. First, when a user submits a query, concepts (i.e., important terms or phrases in web-snippets) and their relations are mined online from web-snippets to build a concept relationship graph. Second, clickthroughs are collected to predict user's conceptual preferences. Third, the concept relationship graph together with the user's conceptual preferences is used as input to a concept-based clustering algorithm that finds conceptually close queries. Finally, the most similar queries are suggested to the user for search refinement. Figure 1 shows the general process of our approach. To evaluate the performance of our approach, we developed a Google middleware for clickthrough data collection.2 Users were invited to test our middleware with test queries selected from a spectrum of topical categories. We evaluate the performance of our approach using the standard recall-precision measures. Beeferman and Berger's agglomerative clustering algorithm (or simply called BB's algorithm in this paper) is used as the baseline to compare with our concept-based approach. Our experimental results show that the average precision at any recall level is better than the baseline method.

2. Related Work

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

Dik Lun Lee [2] proposed to 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). It focuses on search engine personalization and develops 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.

Eric Brill and Susan Dumais [3] have proposed that incorporating user behavior data can significantly improve ordering of top results in real web search setting. We examine alternatives for incorporating feedback into the ranking process and explore the contributions of user feedback compared to other common web search features. A large scale evaluation

over 3,000 queries and 12 million user interactions with a popular web search engine. We show that incorporating implicit feedback can augment other features, improving the accuracy of a competitive web search ranking algorithms by as much as 31% relative to the original performance.

Doug Beeferman and Adam Berger [5] has proposed a technique for mining a collection of user transactions with an Internet search engine to discover clusters of similar queries an similar URLs. The information we exploit is "Clickthrough data": each record consists of a user's query to a search engine along with the URL which the user selected from among the candidates offered by the search engine. By viewing this dataset as a bipartite graph, with the vertices on one side corresponding to queries and the other side to URLs, one can apply an agglomerative clustering algorithm to the graph's vertices to identify related queries and URLs. One noteworthy feature of the proposed algorithm is that it is "content-ignorant" the algorithm makes no use of the actual content of the queries or URLs, but only how they co-occur within the clickthrough data. It describes how to enlist the discovered clusters to assist users in web search, and measure the effectiveness of the discovered clusters in the Lycos search engine.

We know that most of the previous approaches on query clustering consider two different queries to be semantically similar if they lead to common clicks on the same pages. However, the chances for different queries leading to common clicks on the same URLs are rare in Web search engines. Based on this important observation, we propose to use concepts, not pages, as the common ground for relating semantically similar queries. That is, two queries are considered related if they lead to clicks on pages that share some common concepts, which are mined from the web-snippets in the search results.

To our knowledge, there is no previous study on the personalization of query suggestions. We propose a two-phase clustering method to cluster queries first within the scope of each user and then for the community. We conduct experiments to evaluate different methods and show that our concept-based two-phase clustering method yields the best precision and recall.

3. QUERY LOG ANALYSIS

Because of the proprietary nature of the information, the scientific literature contains very little analysis of data collected from large-scale commercial search engines. An exception is Silverstein et. al, who reported statistics accumulated from a billion entry set of user queries to Altavista.

One of the central aims of the present work was to develop a rapid turnaround clustering module capable of identifying and adapting too late-breaking or ephemeral trends in web usage. It involved a query log accumulated over six weeks; we are more interested in discovering information from a single day's records. After having processed the day's queries, a clustered could provide "fresh" clusters for deployment in the search engine the following day. This is critical for rapid response to news events. Where sprint and MCI/WorldCom are two unrelated telecommunication companies one day, they might announce a merger the very next day. As soon as possible after the announced merger, a search engine with a rapid –turnaround clustering module could, in response to the query Sprint, propose www.mci.com as a possibly relevant URL, and suggest MCI/WorldCom as a related query.

4. CLUSTERING URLS

A high quantity URL clustering algorithm could help in automating the process of ontology generation (the task which Yahoo and Open Directory human employees perform), organizing bookmark files into categories, constructing a user specific profile of "pages this person finds interesting" and grouping the results of a web search by category. Not surprisingly the past years have witnessed a great deal of interest in this area.

Researchers have been investigating the more general problem of document clustering

algorithms for decades, and it makes sense to consider how well these approaches-which do not exploit the correlations between documents and queries inherent in a search engine transaction database, would fare on a subset of the internet. One popular technique is k-means, whose running time is linear in the member of documents lie in a spherical configuration with respect to underlying metric. Also popular are hierarchical agglomerative clustering (HAC) techniques, typically at least quadratic in n, which repeatedly find the closest two documents and merge them.

Our approach falls into the HAC category, but differs from traditional techniques in one significant way. Namely, the distance between two documents can be evaluated without examining the content of those documents, a property we shall refer to as "content ignorance." This stands in contrast to traditional "content-aware" clustering algorithms, which typically use as a distance between documents some function of the fraction of tokens they have in common.

Clustering web pages by content may require storing and manipulating a staggeringly large amount of data: by early 2000, there were atleast a billion pages on the Internet accessible by commercial search engines. Content ignorance can obviously be a valuable property when handling a dataset of this scale. It can also be applicable, at least in principle, in setting where content-aware clustering is not, including:

- *Text-free pages:* A distance function calculated from the text of a web page isn't capable of recognizing the correspondence between a web page containing just a picture of an emu and another describing the appearance and behavior of an emu.
- *Pages with restricted-access:* URLs may be password protected or temporarily unavailable, rendering them unavailable to a clustering procedure which relies on the content of the page.
- *Pages with dynamic content:* A URL might always point to a company's web page, but the contents of that page are likely to change regularly, updated with news about the company. A content-aware clustered is more susceptible to placing a URL in different clusters as the page is modified.

The last and perhaps most important advantage is that content-ignorant clustering can be implemented relatively more efficiently than standard agglomerative techniques.

5. CLUSTERING QUERIES

Clustering queries submitted to search engine appears to be a rather less explored problem than clustering web pages, through there are practical, commercial applications for a high quality query clustered. For instance, if a user submits to a search engine a query q which is a member of a cluster C, the search engine could suggest as alternate, related queries the other member of C. Such a "related query" tool is deployed in the Lycos search engine; the component appears in Figure 1.

Figure 2: Lycos Search Engine.

Many commercial search engines offer users the opportunity to rephrase their information need by suggesting alternate queries. Shown is the top of a page generated by Lycos is response to the query American Airlines. A query clustering algorithm could provide such a list of suggestions by offering, in response to a query q, the other members of the cluster containing q.

6. CLICKTHROUGH DATA

The http protocol allows commercial search engines the ability to record a great deal of information about their users the name and IP address of the machine which sent the request, the type of web browser running on the machine, and so on. Here we are interested only in the sequence of character comprising the query submitted by the user from among the choices presented by the search engine. It applied a simple filter to detect and remove those records containing objectionable content such as pornography and hate speech. This eliminated about 20% of the records gathered, including an undetermined number of false positives and negatives. It mapped the query characters to lowercase and converted a sequence of one or more spaces to positive, but otherwise performed no processing of the queries submitted to Lycos.

It avoids significant processing of the search engine log file to underscore the "pushbutton" nature of the proposed clustering based algorithms. Clearly, however, even a minimal amount of preprocessing could go a long way in helping the clustering.

7. CONCEPT BASED CLUSTERING

Using the concepts extracted from web-snippets, we propose concept-based clustering methods.

We first extend BB's algorithm to a concept-based algorithm. After that personalized concept-based algorithm is further enhanced to achieve effective personalized clustering.

A. Clustering on Query-Concept Bipartite Graph

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

B. Personalized Concept-Based Clustering

We now explain the essential idea of our personalized concept-based clustering algorithm with which ambiguous queries can be clustered into different query clusters. Personalized effect is achieved by manipulating the user concept preference profiles in the clustering process. In contrast to BB's agglomerative clustering algorithm, which represents the same queries submitted from different users by one query node, we need to consider the same queries submitted by different users separately to achieve personalization effect. In other words, if two given queries, whether they are identical or not, mean different things to two different users, they should not be merged together because they refer to two different sets of concepts for the two users.

Therefore, we treat each individual query submitted by each user as an individual vertex in the bipartite graph by labeling each query with a user identifier. Moreover, concepts appearing in the web-snippet of the search result with interestingness weights greater than zero in the concept preference profile are linked to the corresponding query on the bipartite graph. An example is shown in Figure 3a. We can see that the query "apple" submitted by users User1 and User3 become two vertices "$\text{apple}_{(\text{User1})}$" and "$\text{apple}_{(\text{User3})}$." If User1 is interested in the concept "apple store," as recorded in the concept preference profile, a link

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

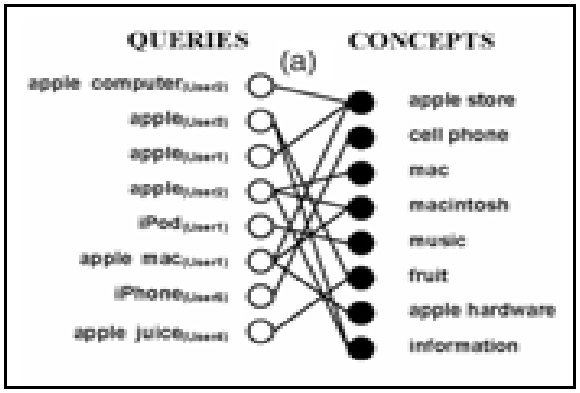

Figure 3a: Original Bipartite Graph.

After the personalized bipartite graph is created, our initial experiments revealed that if we apply BB's algorithm directly on the bipartite graph, the query clusters generated will quickly merge queries from different users together, thus losing the personalization effect We found that identical queries, though issued by different users and having different meanings, tend to have some generic concept nodes such as "information" in common, e.g., "$\text{apple}_{(\text{User1})}$" and "$\text{apple}_{(\text{User3})}$" both connect to the "information" concept node in Figure 3a. Thus, these query nodes will likely be merged in the first few iterations and cause more queries from different users to be merged together in subsequent iterations. Considering Figure 3a again, if "$\text{apple}_{(\text{User1})}$" and "$\text{apple}_{(\text{User3})}$" are merged, the next iteration will merge the concept nodes "apple store," "fruit," and "information." When the clustering algorithm goes further, queries across users will be further clustered together. At the end, the resulting query clusters have no personalization effect at all. To resolve the problem, we divide clustering into two steps. In the initial clustering step, an algorithm similar to BB's algorithm is employed to cluster all the queries, but it would not merge identical queries from different users

.

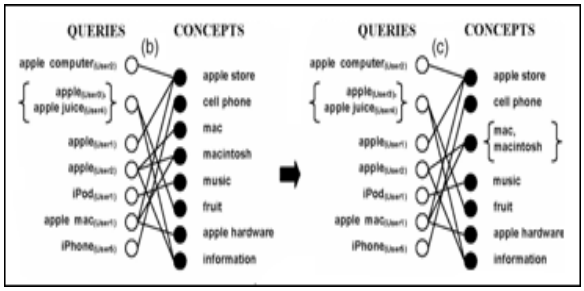

Figure 3b: Initial Clustering.

After obtaining all the clusters from the initial clustering step, the community merging step is employed to merge query clusters containing identical queries from different users. We can see from Figure 3d that "$\text{apple}_{(\text{User1})}$" and "$\text{apple}_{(\text{User3})}$" belong, correctly, to different clusters. The initial clustering step is able to generate high precision rate because it preserves the preference of each user, while the community merging step is able to improve the recall rate because of the collaborative filtering effect.

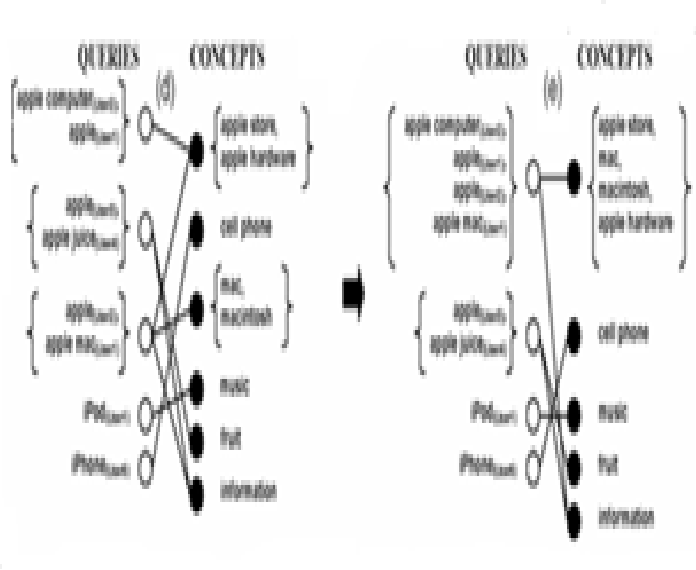

Figure 3c: Community merging

Initial clustering is similar to BB's agglomerative algorithm as already discuss. However, queries from different users are not allowed to be merged in initial clustering Figures 3b and 3c show examples of query and concept merging, respectively. Figure 5d illustrates the result of initial clustering. In community merging, query clusters containing identical queries from different users are compared for merging. Figures 3d and 3e show an example of query cluster merging.

The query clusters both contain the query "apple" and are leading to the same concept "apple store." Therefore, they are merged in community merging as one big cluster. Good timing to start community merging is important for the success of the algorithm. If we stop initial clustering too early (i.e., not all clusters are well formed), community merging merges all the identical queries from different users first and thus generates a single big cluster without much personalization effect. However, if we stop initial clustering too late (i.e., clusters are being overly merged in this case), the low precision rate generated by initial clustering would not be improved by community merging.

8. FUZZY CLUSTERING USING CONCEPTS

The first step in generating the concept-based fuzzy clusters is to obtain a set of concepts associated with the users queries. The source of the conceptual information is a concept knowledge base that was originally devised for query expansion. This concept knowledge base contains relationships between concepts and the terms have been used to describe them. The ACM Computing Classification System was used as the source of the conceptual knowledge for the prototype tool, resulting in a concept knowledge base specifically for the computer science domain. The process for obtaining the concepts that are related to the users queries is similar to the process for generating the query space as described. The query terms are first processed using Porter's stemming algorithm, which removes the prefixes and suffixes from terms to generate the root words, called stems. These stems are matched to the stems in the concept knowledge base, and the nearest concepts are selected. For each of these concepts, the set of stems that are nearest to the concept are selected from the knowledge base. Each of these sets will contain one or more of the original query term stems, plus additional stems that are not present in the query. The first step in generating the concept-based fuzzy clusters is to obtain a set of concepts associated with the users' queries. The source of the conceptual information is a concept knowledge base that was originally devised for query. This concept knowledge base contains relationships between concepts and the terms have been used to describe them. The ACM Computing Classification System was used as the source of the conceptual knowledge for the prototype tool, resulting in a concept knowledge base specifically for the computer science domain. The process for obtaining the concepts that are related to the users queries is similar to the process for generating the query space as described. The query terms are first processed using Porter's stemming algorithm, which removes the prefixes and suffixes from terms to generate the root words, called stems. These stems are matched to the stems in the concept knowledge base, and the nearest concepts are selected. For each of these concepts, the set of stems that are nearest to the concept are selected from the knowledge base. Each of these sets will contain one or more of the original query term stems, plus additional stems that are not present in the query. Therefore, as a result of this query space generation, a set of concept vectors C = $\{c_1, c_2, \ldots, c_m\}$ are generated. If the total number of unique stems that were selected from the concept knowledge base is p, then the dimension of all vectors c_1 (i = 1 . . .m) is p. Further, the magnitude of the vector c_1 (i = 1 . . .m) on dimension j (j = 1 . . . p) is given by the concept knowledge base weight between concept i and term j.

After the concepts have been obtained from the concept knowledge base, and the concept vectors have been created, the users' queries are sent to the Google API. As each of the document surrogates are retrieved, a single-pass fuzzy c-means clustering algorithm is performed. The title and snippet from the document surrogate are processed using Porter's stemming algorithm, and the frequency of each unique stem is calculated. These frequencies are used to generate vectors for each of the document surrogates. Although some argue against using term frequencies (TF) as the sole source of information in a text retrieval system, using other global information such as the inverse document frequency (IDF) is not feasible when the document surrogate vectors need to be generated as each document surrogate is retrieved (to achieve a near real-time web information retrieval system). Given a set of concept vectors C = $\{c_1, c_2, \ldots, c_m\}$ and a document surrogate vectord_i, the fuzzy membership of document surrogate d_i with respect to concept c_j is given by:

$$u_{i,j} = \frac{1}{\sum_{k=1}^{m} \left(\frac{sim(d_i, c_j)}{sim(d_i, c_k)} \right)^2}$$

In this calculation, the similarity between a document surrogate vector and a concept vector is given by the Euclidean distance metric.

$$sim(x_i, x_j) = (\sum_{k=1}^{p} (x_{i,k} - x_{j,k})^2)^{1/2} \quad (1)$$

Normally, when evaluating the document surrogates, all unique stems would contribute to the construction of the document surrogate vector. However, since the distance calculations in this single-pass fuzzy clustering algorithm are always between concept vectors and document surrogate vectors, we only need to consider the stems that are already present in the concept vectors. This reduction in the dimension of the document surrogate vectors results in an increase in the speed at which the fuzzy clusters are generated.

9. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed clustering methods for obtaining related queries using user clickthrough. Firstly describe the experimental setup for collecting the required clickthrough data. To collect the clickthrough 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. After that to compare the performance of BB's algorithm using query URL, query-word, and query-concept bipartite graphs (or simply called the QU, QW, and QC methods). In which to evaluate the effectiveness of our proposed personalized concept-based clustering (or simply called the P-QC method). And then discuss the algorithmic complexities based on the related parameters.

Now discuss the result of experiments which compares the performances of QU, QW, QC methods. QU method is the original input of BB's algorithm, which serves as a baseline for comparison. QW method uses query-word bipartite graph, which is similar to the query concept bipartite graph in that they are both constructed. The difference is that the former contains all words (excluding stop words) from the web snippets and the latter contains the extracted concepts. QW and QC methods are necessary, since they allow us to study the benefits of concept extraction. The three methods are also employed to cluster the collected data.

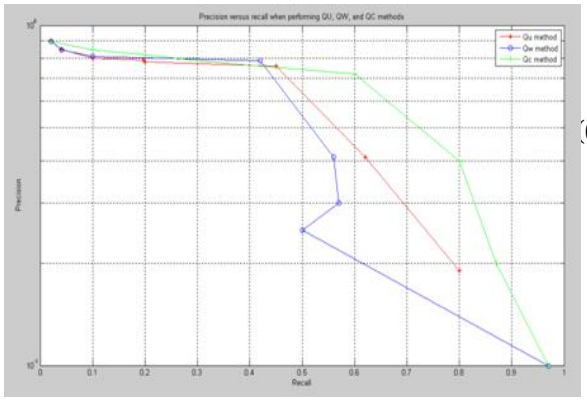

Figure 4: Precision versus recall when performing QU, QW, and QC

Table 1
Statistics of the Clickthrough Data Collected for the experiment.

Number of users	5
Number of queries assigned to each user	10
Number of test queries	50
Number of unique queries	50
Max. number of retrieved URLs for query	140
Max. no. of extracted concepts for query	279
Max. no. of extracted words for query	1203
Number of URLs retrieved	15390
Number of unique URLs retrieved	13006
Number of concepts retrieved	14490
Number of unique concepts retrieved	7098
Number of words retrieved	179843
Number of unique words retrieved	24567

The results are compared to our predefined clusters for precision and recall. Given a query q and its corresponding query cluster $\{q_1, q_2, q_3 \dots .\}$generated by a clustering algorithm, the precision and recall are computed using the following formulas:

$$precision(q) = \frac{|Q_{relevant} \cap Q\_retrieved|}{|Q\_retrieved|}$$

$$recall(q) = \frac{|Q_{relevant} \cap Q\_retrieved|}{|Q\_relevant|}$$

The precision and recall values from all queries are averaged for plotting the precision-recall figures. The performance of the three methods is compared using precision-recall figures and best F-measure values. Figure 4 shows the precision-recall figures for QU, QW, and QC methods. We observe that QC method yields better recall rate than QU method (i.e., the original BB's algorithm), while preserving high precision rates. Note that QU method can yield high precision rate because of the valuable URL overlaps between queries. However, QC method benefits both precision and recall compared to QU method, showing that the use of extracted concepts is much better for finding similar queries. We also observe that QW method performs the worst among the three methods because common nonstop words such as "discussion," "information," and "news" bring unrelated queries together, thus lowering both the precision and recall rate. The main difference between QW and QC methods is the availability of concept extraction. Intuitively, QC method outperforms QW method because the concept extraction process can successfully eliminate unrelated common words within web-snippets.

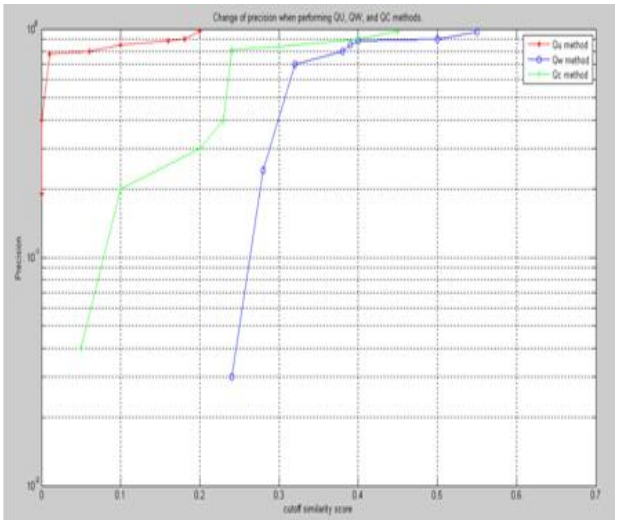

Figure 5: Change of recall when performing QU, QW and QC methods

Figures 5 and 6 show the change of precision and recall, respectively, for the three clustering methods. In Figure 5, when the cutoff similarity score is around 0.25, the precision obtained using QU method is very close to that of QC method, which is much better than the precision obtained using QW method. In Figure 6, at the same cutoff similarity score, the recall obtained using QU method is close to zero, which is much lower compared to the recalls obtained using QW and QC methods. We can easily see from Figures 5 and 6 that QC method is able to generate good recall, while achieving a precision comparable to that of QU method

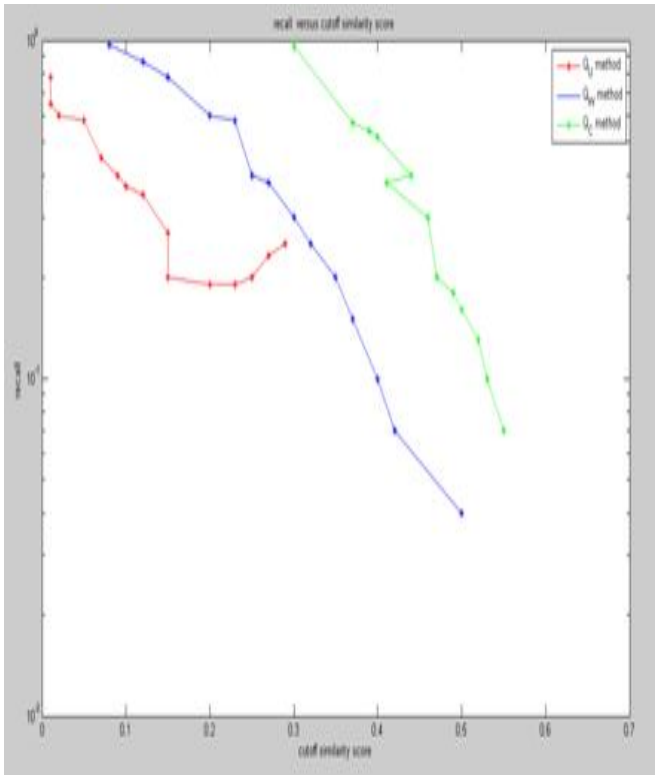

Figure 6: Change of recall when performing QU, QW and QC methods.

.

Figure 7 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 0.6. 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 Figure 7), 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.

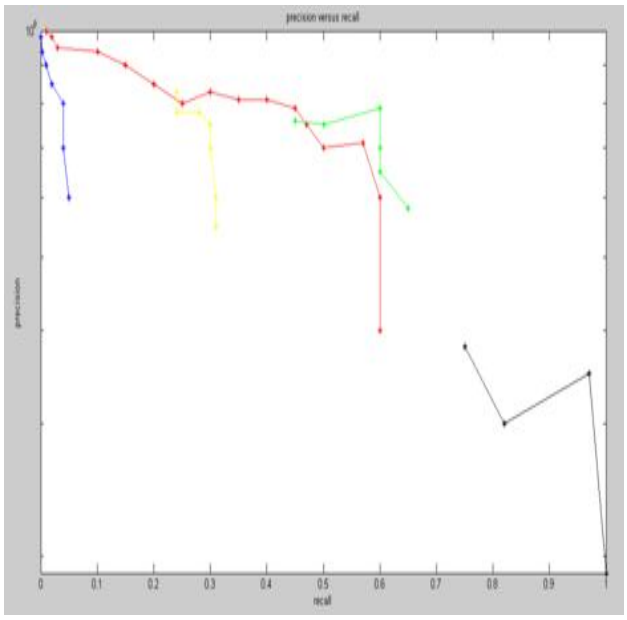

Figure 7: Precision versus recall when performing P-QC method.

When initial clustering is switched to community merging at the optimal point (see the white-circle graph in Figure 7), 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.

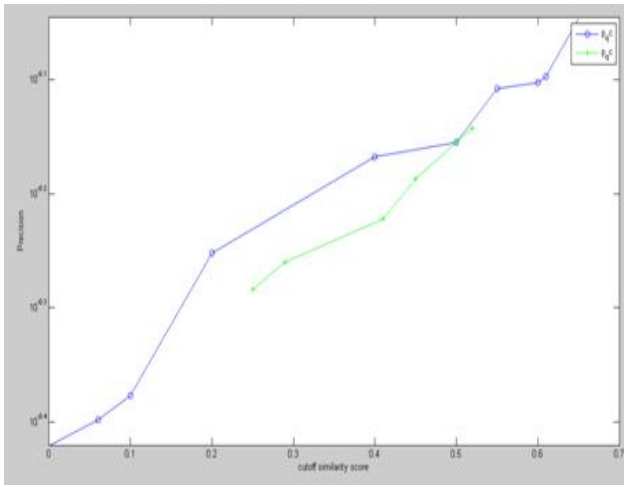

Figure 8: Change of precision when performing P-QC method.

In Figure 8, 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.

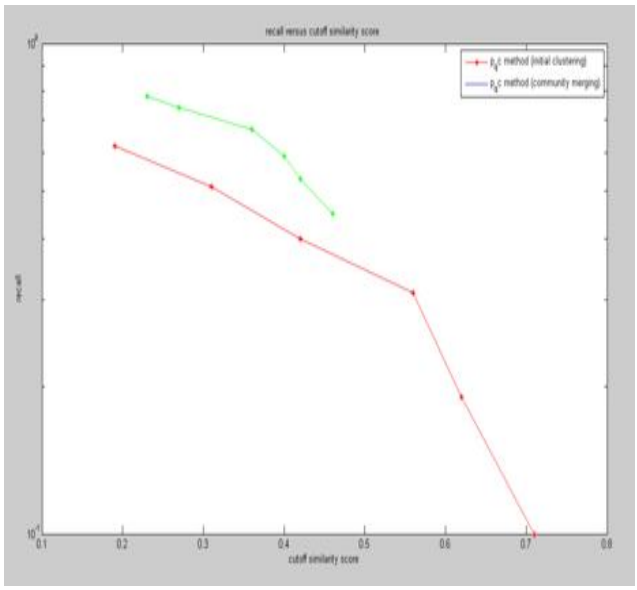

Figure 9: Change of recall when performing P-QC method.

In Figure 9, 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 Figures 8 and 9 that only a small fraction of precision is used to trade for a much better recall in community merging.

10. CONCLUSION

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. 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. A new two phase personalized agglomerative clustering algorithm and fuzzy clustering algorithm, which is able to generate personalized query clusters. The technique makes use of clickthrough data and the concept relationship graph mined from web-snippets, both of which can be captured at the back end and as such do not add extra burden to users. An adapted agglomerative clustering algorithm is employed for finding queries that are conceptually close to one another. Our experimental results confirm that our approach can successfully generate personalized query suggestions according to

individual user conceptual needs. it improves prediction accuracy and computational cost compared to BB's algorithm, which is the state-of-the-art technique of query clustering using clickthroughs for the similar objective.

REFERENCES

[1] Kenneth Wai-Ting Leung, Wilfred Ng and Dik Lun Lee, "Personalized Concept Based Clustering of Search Engine Queries." Proceeding IEEE Transactions of Knowledge and Data Engineering, vol. 20, no. 11, Nov. 2008.

[2] Kenneth Wai-Ting Leung, Wilfred Ng and Dik Lun Lee, "Deriving Concept Based Users Profiles from Search Engine Logs." Proceeding IEEE Transactions of Knowledge and Data Engineering, vol. 6, no. 1, Jan 2007.

[3] Eugene Agichtein, Eric Brill and Susan Dumais, "Improving Web Search Ranking by Incorporating User Behavior Information." Proceedings of the ACM conference on Research and Development on Information Retrieval (SIGIR) 2006.

[4] Kenneth Wai-Ting Leung, Wilfred Ng and Dik Lun Lee, "Constructing Concept Billion Network and its Application to Personalized Web Search." Proceedings of EDBT 2011, March 22-24, 2011, Uppsala, Sweden

[5] Doug Beeferman and Adam Berger, "Agglomerative Clustering of a Search Engine Query Log." Proceedings of the ACM conference on Management of Data 2000, Washington.

[6] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM SIGMOD, 1993.

[7] S.M. Beitzel, E.C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder, "Hourly Analysis of a Very Large Topically Categorized Web Query Log," Proc. 27th Ann. Int'l ACM SIGIR Conf. (SIGIR), 2004.

[8] H. Cui, J. Wen, J. Nie, and W. Ma, "Query Expansion by Mining User Logs," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 829-839, July/Aug. 2003.

[9] S. Chuang and L. Chien, "Automatic Query Taxonomy Generation for Information Retrieval Applications," Online Information Rev., vol. 27, no. 4, pp. 243-255, 2003.

[10] H. Cui, J. Wen, J. Nie, and W. Ma, "Query Expansion by Mining User Logs," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 829-839, July/Aug. 2003.

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

[12] L. Deng, W. Ng, X. Chai, and D.L. Lee, "Spying Out Accurate User Preferences for Search Engine Adaptation," Advances in Web Mining and Web Usage Analysis, LNCS 3932, pp. 87-103, 2006.

[13] S. Chuang and L. Chien, "Automatic Query Taxonomy Generation for Information Retrieval Applications," Online Information Rev., vol. 27, no. 4, pp. 243-255, 2003.

[14] L. Deng, W. Ng, X. Chai, and D.L. Lee, "Spying Out Accurate User Preferences for Search Engine Adaptation," Advances in Web Mining and Web Usage Analysis, LNCS 3932, pp. 87-103, 2006.

[15] T. Joachims and F. Radlinski, "Search Engines That Learn from Implicit Feedback," Computer, vol. 40, no. 8, pp. 34-40, 2007.

[16] B. Koester, "Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies," Proc. Sixth IEEE Int'l Conf. Data Mining (ICDM), 2006.

[17] F. Liu, C. Yu, and W. Meng, "Personalized Web Search for Improving Retrieval Effectiveness," IEEE Trans. Knowledge and Data Eng., vol. 16, no. 1, pp. 28-40, Jan. 2004.