INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Concept Based Personalized Search Engine

Jagadeeshwar Reddy.V
B.Tech in Computer science
Pursuing M.Tech in Computer science Engineering
Aurora's Technological And Research Institute, JNTU, Hyderabad, India

S.Siva Sankar Rao
Assoc Professor, Department of Computer Science and Engineering
Aurora's Technological And Research Institute, JNTU, Hyderabad, India

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. The separation provides a clear threshold for an agglomerative clustering algorithm to terminate and improve the overall quality of the resulting query clusters.

Keywords: *Negative Preferences, Personalization, Personalized query clustering, Search engine, User profiling.*

Introduction

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 preference.

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
- Existing clickthrough-based
- Most existing user profiling strategies

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 f 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 [1] 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 [2] 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 a 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 [1], 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 [1]. 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.

Related Work

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 click through 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.

Document-Based Methods

On Web search engines, click through data are important implicit feedback mechanism from users. Table 1 is an example of click through 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 d1, d5, and d8 are the documents that have been clicked by the user.

Joachims' method assumes that a user would scan the search result list from top to bottom. If a user has skipped a document di at rank i before clicking on document dj at rank j, it is assumed that he/she must have scan the document di and decided to skip it. Thus, we can conclude that the user prefers document dj more than document di (i.e., dj

<r0 di, where r0 is the user's preference order of the documents in the search result list). Using Joachims' proposition and the example click through data in Table 1.

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. [4] proposed a user profiling method based on users' search history and the Open Directory Project (ODP) [5]. The user profile is represented as a set of categories, and for

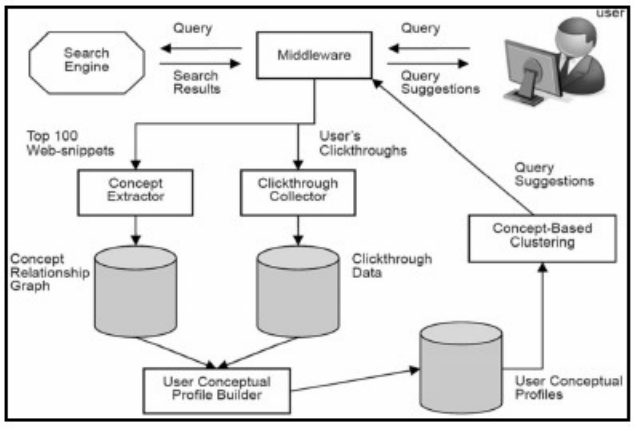

Figure 1: The general process of Query based clustering.

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. [6] 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 [7]. A classifier is employed to classify user-browsed documents into concepts in the reference ontology. Xu et al. [8] proposed a scalable method which automatically builds user profiles based on users' personal documents (e.g., browsing histories and e-mails). The user profiles summarize users' interests into hierarchical structures.

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

Table 1

The method assumes that terms that 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 reference ontology (e.g., ODP) to develop the hierarchical user profiles, while Xu et al. automatically extract possible topics from users' browsed documents and organize 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 reference ontology such as ODP is not always up-to-date. Thus, all of our proposed users profiling strategies rely on a concept extraction method as described in Section 3.1.1, which extracts concepts from Web-snippets2 to create accurate and up-to-date user profiles.**

Personalized Concept-Based Query Clustering

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

Extracting Concepts From Web-Snippets

After a query is submitted to a search engine, a list of Websnippets is 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

coexists 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 [3], to measure the interestingness of a particular keyword/phrase ci extracted from the Web-snippets arising from q:

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

Where sf (ci) is the snippet frequency of the keyword/ phrase ci (i.e., the number of Web-snippets containing ci), n is the number of Web-snippets returned, and |ci| is the number of terms in the keyword/phrase ci. If the support of a keyword/phrase ci is greater than the threshold s (s = 0:03 in our experiments), we treat ci as a concept for the query q. shows an example set of concepts extracted for the query "apple." Before concepts are extracted, stop words, such as "the," "of," "we," etc., are first removed from the snippets.

Mining Concept Relations

We assume that two concepts from a query q are similar if they coexist 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 [3] to establish the similarity between terms t1 and t2:

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

Where n is the number of documents in the corpus, df(t) is the document frequency of the term t, and df(t1 U t2) is the joint document frequency of t1 and t2. The similarity sim(t1; t2) obtained using the above formula always lies between [0, 1].

Query Clustering Algorithm

Our personalized concept-based clustering algorithm [1] 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 (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 are 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:

$$sim(x,y) = \frac{N_x \cdot N_y}{\| N_x \| \| N_y \|}, \qquad \text{-------} \quad (3)$$

where Nx is a weight vector for the set of neighbor nodes of node x in the bipartite graph G, the weight of a neighbor node nx in the weight vector Nx is the weight of the link connecting x and nx in G, Ny is a weight vector for the set of neighbor nodes of node y in G, and the weight of a neighbor node ny in Ny is the weight of the link connecting y and ny 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 Equation (3).

2: Merge the pair of most similar query nodes (qi,qj) that does not contain the same query from different users. Assume that a concept node c is connected to both query nodes qi and qj with weight wi and wj, a new link is Created between c and (qi; qj) with weight w = wi + wj.

3: Obtain the similarity scores in G for all possible pairs of concept nodes using Equation (3).

4: Merge the pair of concept nodes (ci,cj) having highest similarity score. Assume that a query node q is connected to both concept nodes ci and cj with weight wi and wj, a new link is created between q and(ci; cj) with weight

W = wi + wj.

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 (3).

7. Merge the pair of most similar query nodes (qi,qj) that contains the same query from different users. Assume that a concept node c is connected to both query nodes qi and qj with weight wi and wj, a new link is created between c and (qi; qj) with weight w = wi + wj.

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.

User Profiling Strategies

We propose user profiling strategy which are concept-based and utilize users positive and negative preferences. They are PJoachims_C, In addition, we use PClick, which was proposed in [1], as the baseline in the experiments. PClick is concept-based but cannot handle negative preferences.

JOACHIMS-C METHOD (Pjoachims_C)

Joachims [10] assumed that a user would scan the search results from top to bottom. If a user skipped a document di before clicking on document dj (where rank of dj > rank of di), he/she must have scanned di 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 dj <r' di.

Joachims' original method was based on users' document preferences. If a user has skipped a document di at rank i before clicking on document dj at rank j, he/she must have scanned the document di and decided to skip it. Thus, we can conclude that the user prefers document dj more than document di (i.e., dj <r0 di, where r0 is the user's preference order of the documents in the search result list).

We extended Joachims' method, which is a documentbased method, to a concept-based method (Joachims-C). Instead of obtaining the document preferences dj <r0 di, Joachims-C assumes that the user prefers the concepts C(dj) associated with document dj to the concepts CðdiÞ associated with document di, 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 dj at rank j, all the concepts C(di) in the unclicked documents di above rank j are considered as less relevant than the concepts C(dj) in the document dj, i.e., (C(dj) <r0 C(di), where r0 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 d2, d3, and d4, but clicked on d5. Thus, according to Proposition 1, we can conclude that the concepts Cðd5Þ is more relevant to the user than the concepts in the other three unclicked documents (i.e., Cðd2Þ, Cðd3Þ, and Cðd4Þ). The concept preference pairs extracted using Joachims-C method.

After the concept preference pairs are identified using Proposition 1, a ranking SVM algorithm [2] 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 w! and a feature vector of query concept mapping φ(q, c), which describes how well a concept c matches the user's interest for a query q.

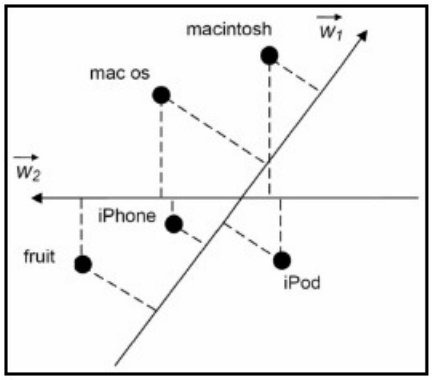

Figure 2: Ordering of concepts "macintosh," "mac os," "iPod," "iPhone," and "fruit" using weight vectors w1 and w2.

Fig. 2 is an example showing how the weight vector 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 w1 is better than w2, because w1 correctly ranks the concepts as("macintosh" <w1 "mac os" <w1 "iPod" <w1 "iPhone" <w1"fruit"), while w2 ranks the concepts as ("fruit"<w2 "macos" <w2 "iPhone" <w2 "macintosh" <w2 "iPod").

The feature vector $\Phi(q, c) = [Feature\_c1, Feature\_c2, \ldots, Feature\_cn]$ **for the ranking SVM training is composed of all the extracted concepts for a query q. For each concept ci, we create a feature vector** $\varphi(q, ci) = [Feature\_c1, Feature\_c2, \ldots, Feature\_cn]$ **which is defined as follows:**

$$Feature\_c_k = \begin{cases} 1, & \text{if } k = i, \\ sim_R(c_i, c_j), & \text{if } sim_R(c_i, c_k) > 0, \\ 0, & \text{otherwise.} \end{cases} \quad \text{---- 4}$$

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 w such that the maximum number of the following inequalities holds:

$$\forall (c_i, c_j) \in r'_k, (1 \le k \le n): \quad \vec{w} \cdot \phi(q_k, c_i) > \vec{w} \cdot \phi(q_k, c_j), \quad \text{-----5}$$

where (ci; cj) € r'k is a concept pair corresponding to the concept preference pair (ci <r0k cj) of the query qk, which means that ci should rank higher than cj in the target concept ordering of r'k.

The weight vector $W = (w\ Feature\_c1, w\ Feature\_c2, \ldots, w\ Feature\_cn)$ **determines the user preferences on the extracted concepts.**

For all the concepts c1, c2; . . . ; ci extracted for the query q, the user preferences are stored in the corresponding weight values w Feature c1 , w Feature c2 , . . . , w Feature cn, creating a concept preference profile PJ*oachims--C* $= (w\ Feature\_c1, w\ Feature\_c2, \ldots, w\ Feature\_cn)$ **for the query q. Table 1 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.**

Experimental Results

We first describe the setup for clickthrough collection. The collected clickthrough data are used by the proposed user profiling strategies to create user profiles.

Experimental Setup

To evaluate the performance of our user profiling strategies, we developed a middleware for Google3 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).

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

Table 2: **Statistics of the Collected Clickthrough Data**

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. The 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 is 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., PClick, PJoachims_C, PmJoachims_C, PSpyNB_C, PClickþJoachims_C, PClickþmJoachims_C, and PClickþSpyNB_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 2 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 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:

$$precision(q) = \frac{|Q_{relevant} \bigcap Q_{retrieved}|}{|Q_{retrieved}|}, \quad \text{------ 6}$$

$$recall(q) = \frac{|Q_{relevant} \bigcap Q_{retrieved}|}{|Q_{relevant}|}, \quad \text{------- 7}$$

where q is the input query, Qrelevant is the set of queries that exists in the predefined cluster for q, and Qretrieved 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.

Termination Points For Individual Clustering To Community Merging

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 ΔSimilarity, 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, 4Similarity is defined as

$$\triangle Similarity(i) = sim_i(P_{q_m}, P_{q_n}) - sim_{i+1}(P_{q_o}, P_{q_p}), \quad \text{---- 8}$$

where qm and qn are the two most similar queries in the ith

step of the clustering process, P(qm) and P(qn) are the concept-based profiles for qm and qn, qo and qp are the two most similar queries in the ði þ 1Þth step of the clustering process, P(qo) and P(qp) are the concept-based profiles for qm and qn, and simðÞ is the cosine similarity. Note that a positive 4Similarity means that step i + 1 is producing worse clusters than that of step i. In our previous work [1], it is not easy to determine where to cut

the clustering tree in PClick, because the similarity values decrease uniformly during the clustering process.change in similarity values when performing initial clustering and community merging of the personalized clustering algorithm using PClick, PClickþJoachims_C, PClickþmJoachims_C, and PClickþSpyNB_C.

Conclusions

In these we can maintain the Relationships among the users can be mined from the concept-based user profiles to perform collaborative filtering. Because of this concept the users can be easily access the information from the other users profiles information. Here the time consumption is less and the process is done easily. The different profile concepts can be mined differently. 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.

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 also 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.**References**

1. 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.

2. T. Joachims, "Optimizing Search Engines Using Clickthrough Data," Proc. ACM SIGKDD, 2002.

3. 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, Lawrence Erlbaum, 1991.

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

5. Open Directory Project, http://www.dmoz.org/, 2009

6. S. Gauch, J. Chaffee, and A. Pretschner, "Ontology-Based Personalized Search and Browsing," ACM Web Intelligence and Agent System, vol. 1, nos. 3/4, pp. 219-234, 2003.

7. Magellan, http://magellan.mckinley.com/, 2008

8. Y. Xu, K. Wang, B. Zhang, and Z. Chen, "Privacy-Enhancing Personalized Web Search," Proc. World Wide Web (WWW) Conf., 2007.