

CS598CXZ Course Project

Predicting User's Site Preferences in Web Search

Bin Tan
bintan@uiuc.edu

May 9, 2005

1 Introduction

Nowadays search engines are so powerful that they have largely taken over the functionality of bookmarks; instead of managing a lot of bookmarks for fast access to frequently visited web sites, people would send simple queries to search engines and look in the returned results for the web pages they want. Yet the typical query is very short (usually only one or two keywords) and does not carry enough context information, which makes it difficult for search engines to return the desired results. For example, since I do not keep a bookmark of the DAIS¹ group's home page, I would search in Google when I need to access it. If I am willing to use the query "DAIS UIUC", then I would get what I want at the top of the ranked results. But usually I would just type "DAIS", and have to skip one or two pages of irrelevant search results before I can find it. I will not blame Google for this search inefficiency, as the term "DAIS" without any context can mean many things. However, Google could have done better. There are plenty of clues in my past search activities; it should have associated "DAIS" with the web page <http://dais.cs.uiuc.edu/> since I always click on that and nothing else.

Generally, people trust some web sites (in the above example, dais.cs.uiuc.edu over others, and prefer to see search results of those sites. In my case, I prefer cnn.com for news, wunderground.com for weather, portal.acm.org and cite-seer.ist.psu.edu for papers. Existing search engines do not provide ranked results tailored to different user's need; their search services are not personalized. For example, Google always ranks my favorite weather web site wunderground.com below weather.com, regardless of how many times I have clicked the former site, but not the latter.

This project is motivated by the above scenarios. It aims to learn a user's site preferences from the user's search history (queries and clicked results) and promote the search result coming from a web site that is most likely to be preferred by the user. Site preferences should be topic-dependent. It would be meaningless to argue whether the user likes cnn.com better or wunderground.com better overall. It makes more sense to say the user likes cnn.com for news and wunderground.com for weather². When a user submits a query, we can determine the most appropriate topic and select the sites preferred for that topic.

¹The Database and Information Systems Laboratory at UIUC

²CNN actually also has a good weather channel: www.cnn.com/weather

The rest of this project report is organized as follows. In section 2, a few related works are discussed. The main algorithm is presented in section 3. Section 4 gives some examples of user study. Section ?? concludes this report and discusses some issues.

2 Previous Works

Personalized search has been seen as a major goal of next-generation search engines. It tries to deliver search results customized to individual users' different needs. This can be achieved by utilizing a user's search context and reorganizing the results to better match the context. There are two types of search context: long-term and short-term. Long-term context captures a user's general interests and builds a user profile from the entire search history. Short-term context only considers the immediate search activities. It can be inferred from a user's implicit feedback such as web page clickthrough and query modification. In [1], a user's profile represented as a term vector is constructed from browsed web pages and is shown to be able to improve search effectiveness. In [2], captured implicit user feedback is used to modify the query to achieve better retrieval performance. The method used in this project also tries to provide personalized search by promoting those web sites that are most likely to be preferred by a user.

My Yahoo![3] also takes a site-based personalization approach. It allows users to bookmark web sites they like and block web sites they dislike. The search engine would exclude results from the blocked sites and can be instructed to only search within the preferred(bookmarked) sites. The work in this project differs from My Yahoo! in two aspects. First, site preferences are topic dependent. No site is preferred for all topics. Second, no efforts are required from users. A user does not need to mark web sites as good/bad, nor does he/she need a special search option to limit the search within the bookmarked sites.

Another method for search personalization based on clickthrough history is described [4]. It uses Support Vector Machine to learn a customized retrieval function that minimizes the number of inversions (which refers to the case that a clicked result is ranked lower than a non-clicked result) in query logs. The retrieval function takes the form of a linear combination of features on queries and documents. However, to make it reflect users' topic-dependent site preferences, we would need features such as "query is about news and result is from cnn.com", which are too many to define beforehand. Moreover, since the method is targeted at adapting the retrieval function of a meta search engine to a group of users and it may take non-minimal training time, it may not be suitable to meet a single user's site preferences need.

3 Main Algorithm

The main idea is to learn the association between query topics and preferred sites from search history. For a new query, the topic that best matches it is selected to provide site preference information. Section 3.1 describes how the association between queries and sites is made and stored. Section ?? presents an algorithm to utilize this information for predicting preferred web sites.

3.1 Collecting Search History

A search record is appended to a log with every query. The records have the following format:

$$\langle T, Q, S \rangle$$

T : time of search

Q : query

(represented as a vector of terms and their weights $\langle t_1, w_{t_1} \rangle, \langle t_2, w_{t_2} \rangle, \dots, \langle t_n, w_{t_n} \rangle$)

S : preferred sites

(represented as a vector of sites and their weights $\langle s_1, w_{s_1} \rangle, \langle s_2, w_{s_2} \rangle, \dots, \langle s_m, w_{s_m} \rangle$)

Besides query terms, other terms are also included in Q to better capture the search context. Suppose N is the number of results, N_c is the number of clicked results, N_t is the number of results containing term t and $N_{c,t}$ is the number of clicked results containing t . With the current implementation, terms for which $N_{c,t} > 0 \wedge N_t > 0.05 \times N$ are used to expand the query. Its weight is determined by $0.1 \times N_{c,t}/N_c + 0.9 \times N_t/N$.³

Determining which sites are actually liked by a user is a particularly difficult task. In general, clicking on a search result may result from the user’s endorsement for that site, but this is not always true for two reasons. First, a clicked result is not necessarily relevant to the query. Second, even if it is, the user may not like the site where the result is from. To cope with the first problem, I set a threshold of 2 seconds such that if a user stays in a web page for less than that amount of time, the clicked result is not considered relevant. To deal with the second one, note that for easy and frequently repeated queries, the user would probably only click on the result from a preferred site and nothing else, for hard and new queries, the user may click on a lot of results before finding anything relevant. Since easy queries provide stronger evidence of site preferences, I associate each clicked site with a confidence of preference, which is inversely proportional to the number of results the user has clicked. Although the confidence values may not be accurate for a single query, when accumulated over a large number of search records, they may suggest the user’s real preferences.

To facilitate search within the search records, two indices are built. One on term, the other on site. Given an arbitrary term or site, we can look up all records containing that term or site efficiently using the two indices. Each term/site is represented by an integer id. When I run out of ids, I use the LRU algorithm to evict terms/sites that are least recently used.

3.2 Predicting Site Preferences

Predicting site preferences from search history is in essence a classification problem. Each search record $\langle T, S, Q \rangle$ can be viewed as a training example $\langle T, S \rangle \rightarrow Q$, where T and S are features and Q contains labels. Given an unlabeled example $\langle T, S \rangle$, we are to predict its labels, i.e. for each site, whether it will be preferred by a user.

Due to the large number of features and labels (each term corresponds to a feature and each site corresponds to a label), some supervised learning algorithms such as Naive Bayes are not well suited to this problem. The algorithm

³The parameters in this project are often determined heuristically/aesthetically and not best tuned

adopted in this project is *k-Nearest Neighbor*. This is a lazy learning algorithm and has the advantage that no training is needed and new examples can be used immediately. Of course, looking for neighbors might take some time, but this should not be a problem as the number of search records will not be extremely large. (Suppose a user submits 30 queries a day, which amounts to only 10,000 queries a year. We can also periodically eliminate very old queries.)

Given a new query, we need to search for its nearest (most similar) neighbors. The similarity between two examples $P_1 = \langle T_1, Q_1 \rangle$ and $P_2 = \langle T_2, Q_2 \rangle$ is calculated as follows. Since each Q is represented as a vector of terms $\langle t_1, w_{t_1} \rangle, \langle t_2, w_{t_2} \rangle, \dots, \langle t_m, w_{t_m} \rangle$, where w_t is the term frequency of t , cosine similarity can be readily computed. To incorporate document frequency, each w_t is multiplied by $\log((1+N)/f_t)$, where f_t is the number of queries in which t appears, and N is the total number of queries. To take into account of the fact that more recent queries are more important than less recent ones, the cosine similarity is multiplied by a time decay factor $\exp(-\alpha|T_1 - T_2|)$

For each nearest neighbor $P = \langle T, Q, S \rangle$, its list of preferred sites (S) is retrieved and weighted according to P 's similarity to the current query. If a site is found in multiple neighbors, then the weights are accumulated. More formally, if we denote a neighbor P_i 's similarity to the current query P_0 as $D(P_i, P_0)$ and a site s 's confidence in P_i as $w_{i,s}$, then s 's weight is

$$\frac{\sum_i D(P_i, P_0) w_{i,s}}{\log(1 + R_s)}$$

Here R_s is the rank of the highest-ranked result from site s in current query's search results. This is introduced to give advantage to a site whose result is ranked high.

I retrieve $k = 10$ nearest neighbors. At the time of the poster session, I was silly enough to believe that the larger k is, the better. The problem with an infinitely large k is that near examples may lose to faraway examples, as long as the latter greatly outnumber the former, which can cause inferior sites from those faraway examples to get a large weight. For example, when I searched for "Chengxiang Zhai", the query was expanded to include terms "retrieval". Since there were only a few past queries about "Chengxiang Zhai" but a large number of queries on SIGIR papers (which were slightly related to the current query) when I did the demo, the site "citeseer.ist.psu.edu" outweighed "www-faculty.cs.uiuc.edu", which was certainly incorrect.

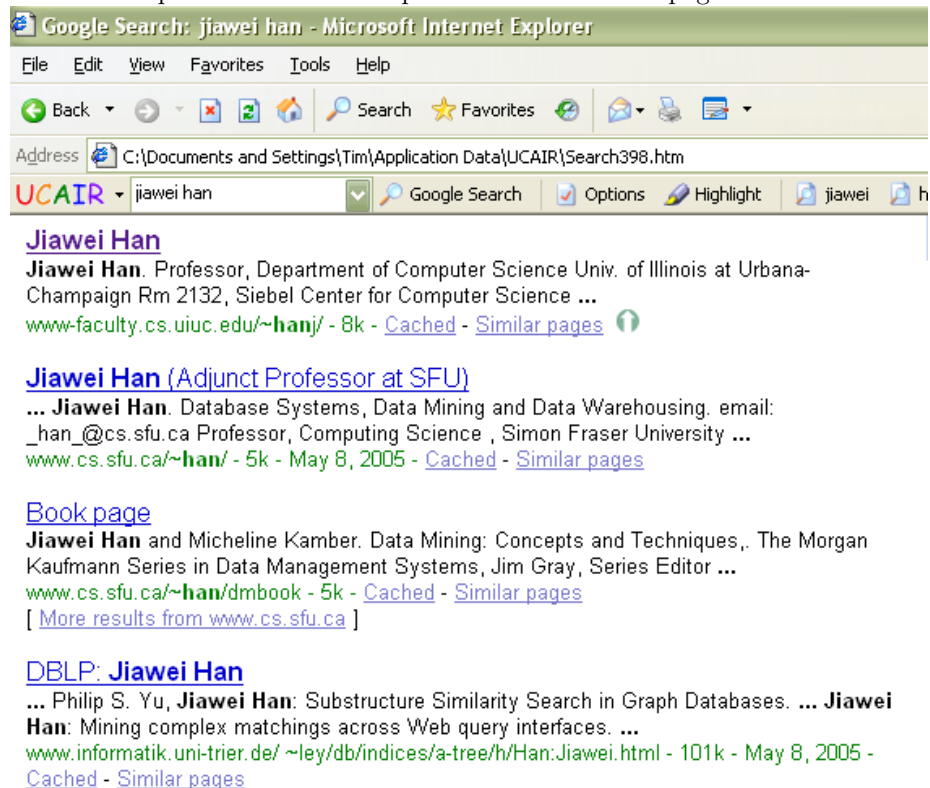
When I find the site that is most likely to be preferred by the user, the highest-ranked result that comes from that site is pulled to the top of the search result list if its weight exceeds a certain threshold. The threshold test is needed to guard against the situation where even the nearest neighbor is too far away.

4 User Study

This project is implemented on top of the UCAIR toolbar[5], which is an Internet Explorer plug-in that enables me to tweak Google's ranking. A screenshot is shown in Figure 4. The modified toolbar can be downloaded from

<http://www.ews.uiuc.edu/~bintan/IRSetup.msi>

Figure 1: Prof. Jiawei Han's homepage is originally ranked in the 4th place. The toolbar promotes it to the top after I visit his homepage.



I could ask a user to use the toolbar for a period of time and collect the statistics on how often the user clicks the promoted result from the preferred site and how often the user clicks Google's original top result. If the former exceeds the latter, then it would suggest that this site preference based search personalization can indeed help. Unfortunately, due to time constraint, I was unable to perform such a user study. Instead, I will just show a very informal example to demonstrate the usefulness of this project.

I used the toolbar to search SIGIR proceedings during the demo and regularly clicked on results from "portal.acm.org" and "citeseer.ist.psu.edu". For this informal user study, I picked a paper from each session of SIGGRAPH 2005 and searched for its title. For only 2 of the 20 papers, Google ranked "portal.acm.org" at the top. But with this project, I got "portal.acm.org" at the top for all but 2 papers. This study is a little optimistic though, as I have not accumulated enough data from other web sites to compete with "portal.acm.org".

5 Conclusions and Discussions

This project tries to capture a user's preferences of web sites in order to provide personalized search. From my usage of the software that implements the project's ideas, I feel it can pleasantly improve a user's search experience.

At the start of this project I had an idea of a probabilistic approach. Let a random variable L represent whether a user likes a site or not for a query. Then the task is to find the site s that maximizes $O(L = 1|q, s)$ for a given query q , which equals

$$\frac{P(L = 1|q, s)}{P(L = 0|q, s)} = \frac{P(q|s, L = 1)P(s|L = 1)}{P(q|s, L = 0)P(s|L = 0)}$$

$P(s|L = 1)$ is just the prior probability that a user likes site s . If we maintain a language model for each site, then $P(q|s, L = 1)$ can be computed as the probability that q is generated from s . This may be an interesting alternative approach, but one problem I fear is that if a site contains diverse topics, its competitiveness may get hurt since its language model is not focused on just one topic.

References

- [1] Kazunari Sugiyama, Kenji Hatano, Stash Yoshikawa, Adaptive Web Search Based on User Profile Constructed Without any Effort from Users. WWW 2004
- [2] Xuehua Shen, Bin Tan, Chengxiang Zhai, Context-Sensitive Information Retrieval Using Implicit Feedback. SIGIR 2005
- [3] <http://my.yahoo.com>
- [4] Thorsten Joachims, Optimizing Search Engines Using Clickthrough Data. KDD 2002
- [5] <http://sifaka.cs.uiuc.edu/ir/ucair/>