Mining Long-lasting Exploratory User Interests from Search History

Bin Tan, Yuanhua Lv and ChengXiang Zhai
Department of Computer Science, University of Illinois at Urbana-Champaign
bintanuiuc@gmail.com, ylv2@uiuc.edu, czhai@cs.uiuc.edu

ABSTRACT

A user's web search history contains many valuable search patterns. In this paper, we study search patterns that represent a user's long-lasting and exploratory search interests. By focusing on long-lastingness and exploratoriness, we are able to discover search patterns that are most useful for recommending new and relevant information to the user. Our approach is based on language modeling and clustering, and specifically designed to handle web search logs. We run our algorithm on a real web search log collection, and evaluate its performance using a novel simulated study on the same search log dataset. Experiment results support our hypothesis that long-lastingness and exploratoriness are necessary for generating successful recommendation. Our algorithm is shown to effectively discover such search interest patterns, and thus directly useful for making recommendation based on personal search history.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithms

Keywords

User modeling, search log mining, recommendation system

1. INTRODUCTION

People have become increasingly reliant on web search engines to find answers about virtually every aspect of their lives. As a result, a user's personal search history contains rich clues about the user's information needs, thus is an invaluable source for learning about the user's preferences and interests. Discovering user preferences and interests from search history is a major task in search log mining, as it not only directly facilitates understanding of a user's search behavior, but can also be leveraged to improve user experience in search and other information tasks. A lot of existing works on search long mining such as [1, 4, 14, 18] use collaborative filtering techniques to mine query and click patterns that are learned from mass-scale search log data aggregating millions of users. The discovered search patterns are intended to help any user according to the collective behavior of similarly minded users. However, since aggregation inevitably leads to lost of individuality, the collective search patterns cannot precisely capture an individual user's unique information needs. Works on user-specific search log mining, on the other hand, have not explicitly targeted at mining particular patterns, but mainly exploit user search history as query context [15, 16, 17] for personalized search.

In this paper, we define a new search log mining problem, in which we aim at discovering user-specific search interest patterns from an individual user's search history, to support proactive recommendation of new information to the user. As noted in existing information filtering works [11, 20, 7], for a recommendation to be effective, it needs to satisfy two criteria: *relevancy* and *novelty*. The goal is to recommend things that match a user's interests but not already known to the user.

In order to satisfy the relevance criterion, we specifically look for *long-lasting* search interest patterns. A long-lasting interest pattern is a series of searches with closely related information need that continually occur in a user's search history over a long period. It reflects a strong and stable search interest, which can be a need to repeatedly visit specific web pages, or a hobby that motivates the user to frequently browse relevant web pages. Compared to a *short-lived* information need, where the user only shows passing interest on a subject, we hypothesize that a search pattern representing long-lasting interest is more likely to generate recommendation that would stay relevant to the user.

In order to satisfy the novelty criterion, we are particularly interested in mining *exploratory* search interest patterns. An exploratory search interest cannot be satisfied by a single information seeking activity and involves free roaming of an unknown information space to expand knowledge, such as learning in depth about a topic, or keeping track of an evolving event. Compared to a *focused* information need, where the user only looks for certain information with little exploration, we hypothesize that a search pattern representing exploratory interest is more likely to generate recommendation that would provide new information value to the user.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

Therefore, to satisfy both relevance and novelty criteria, we focus on ming *Long-lasting and Exploratory Interest Patterns* (LEIP).

An example LEIP would be a gamer continually searching for information on console games. Presumably, the user can benefit from proactive recommendation of relevant new games. Both long-lastingness and exploratoriness are required to make this work: long-lastingness guarantees the user would keep being interested in games, while exploratoriness ensures the recommendation can be generalized to games that are similar but not identical to those found in search history.

To the best of our knowledge, the problem of mining long-lasting exploratory search patterns for recommendation has not been studied previously. One related work is [6], in which, given a multi-query task, the authors predict whether the user will return to this task in the future. While their goal is prediction, our goal is to discover search patterns that are directly useful for personalized recommendation. Also, most SDI and information filtering tasks are focused on similarity between a stream of new documents and a user's long-lasting stable information need [11], but how to identify a user's long-lasting exploratory information need from web search logs has not been addressed before.

2. PROBLEM FORMULATION

We now formally define the problem of discovering Long-lasting Exploratory Interest Patterns from a user's search history.

A user's search history H consists of a series of searches $S_1, S_2, \cdots, S_n$ ordered by search time. Each search $S_i = (t_i, q_i, R_i, C_i, SLM_i)$ consists of a timestamp t_i, a query q_i, a set of top search results $R_i = \{r_{i1}, r_{i2}, \cdots, r_{im}\}$ returned from a search engine, clickthrough information $C_i : R_i \rightarrow \{0, 1\}$, and an estimated search language model SLM_i that represents search need. For each result r_{ij}, there is a snippet s_{ij} (concatenation of result title and summary) and a url u_{ij}. The searches are grouped to sessions. We put S_i and S_j in the same session if $|t_i - t_j| < 30\text{min}$ and $\cos\text{sim}(SLM_i, SLM_j) > 0.1$.

Given this search history, we would like to discover search patterns that represent user interests. More specifically, a pattern P_k is represented as a set of relevant searches $S_{k1}, S_{k2}, \cdots$ that together capture the user's interest in a particular aspect such as games or computer science. Additionally, we associate with it a cluster language model CLM_k that models word usage in searches inside the pattern. Thus we write $P_k = (\{S_{ki}\}, CLM_k)$.

3. DISCOVERING LONG-LASTING EXPLORATORY INTEREST PATTERNS

In this section, we discuss how to discover LEIPs from a user's search history. Our approach is based on language modeling and clustering.

3.1 Language model representation of search

We capture the user's information need behind a given search S using a unigram language model [9] SLM, where term w has the probability $P(w|SLM)$. Compared to search result texts, the search language model, being just a multinomial word distribution, is a very concise representation of the search need. This makes it possible to store a user's entire search history and build a recommendation app at client side to maximize user privacy. Another advantage of search language models is that they can be combined to generate a cluster language model, as we will see later.

In web search, query alone is often not accurate enough to capture the user's information need, due to text sparseness in short queries. We overcome this using pseudo feedback [3, 13] and implicit feedback [5, 15], both of which can be gathered without user effort. Pseudo feedback considers all top search results as relevant, while implicit feedback considers clicked search results as relevant, as clickthrough is a strong relevance indicator. Terms are extracted from the feedback documents to generate a smoother model of user intent than the one constructed from query only. Existing works [4, 15, 10, 16] show implicit feedback, especially clickthrough, can significantly improve retrieval performance over a baseline of no feedback or just pseudo feedback. We use pseudo feedback to complement implicit feedback because result snippets are short, and often there are no clicks at all.

To estimate the search need SLM, we first create a pseudo document by concatenating query with search results in the feedback set:

$$c(w) = c(w, q) + \alpha \sum_{r \in R, C(r)=0} c(w, r) + \beta \sum_{r \in R, C(r)=1} c(w, r)$$

where $c(w, t)$ denotes word count of w in text t. Terms from unclicked results are introduced by pseudo feedback with weight α, while terms from clicked results are brought in by implicit feedback with weight β. Since implicit feedback is more useful than pseudo feedback, we generally set $\beta >> \alpha$.

We use the mixture model [19] to estimate the search language model SLM. The pseudo document is assumed to be generated by a mixture of search model SLM and background model B, so we try to find an SLM that maximizes log likelihood:

$$\sum_{w} c(w) \log \big((1 - \mu) \cdot P(w|SLM) + \mu P(w|B) \big)$$

where $P(w|B)$ is term w's probability in the background collection, and μ is the mixture coefficient of the background model. We set $\alpha = 1$, $\beta = 20$ and $\mu = 0.9$ based on settings from our previous works.

3.2 Clustering algorithm

We implement the star clustering algorithm [2] to generate search clusters as candidate interest patterns. The algorithm computes star-shaped clusters to cover the document set, and has guaranteed lower bound for intra-cluster similarity. It is particularly suitable for our search log mining task for the following reasons. First, the cluster number is not fixed, thus it can adapt to different input data and accommodate search outliers. Second, a search may be shared by more than one clusters and contribute to all their language models, yielding better accuracy. Third, the algorithm is efficient, running linearly to size of the similarity graph. Finally, the algorithm has an efficient online version that does not require re-computation from scratch. As new searches arrive, the LEIPs can keep updating themselves to match evolving user interests.

Since we represent each search with a language model, we calculate the similarity between two searches using cosine similarity, by treating the unigram language models as vectors. We use 0.1 as cosine similarity threshold.

3.3 Filtering and ranking interest patterns

At the end of clustering, we have a lot of search clusters, many of which consist of either recurring series of navigational queries or outliers of ad-hoc queries. Even among the truly exploratory search patterns, we need to rank them by how frequently/strongly they represent user interests. Therefore, we define long-lastingness and exploratoriness measures to rank and filter search clusters to only keep long-lasting and exploratory interest patterns.

For long-lastingness, we can count number of queries, clicks or sessions. Sessions are more accurate because they are not affected by short-time bursts of queries/clicks. For exploratoriness, we can take queries or clicked urls as multinomial distribution and calculate the entropy, or compute the ratio of new queries/clicks. We can further combine the long-lastingness and exploratoriness measures. For example, we treat the measures as features in a machine learning setting to minimize classification error. We find "number of sessions containing new queries" to be the single best measure for predicting recommendation performance. We omit the details here due to space limit.

4. USING SEARCH PATTERNS FOR RECOMMENDATION

Given a set of interest patterns extracted from user's search history, we can use them to recommend items for the user. We generate a cluster language model CLM to concisely represent the user interest as reflected by all searches in pattern P. CLM can then be used to score the relevance of any candidate recommendation item with regard to P.

In Section 3.1, we have computed language models SLM to represent the information need behind an individual search S. It is convenient to estimate the cluster language model directly from the component search language models with a mixture model:

$$P(w|CLM) = \sum_{S_k \in P} \nu_k P(w|SLM_k)$$

where ν_k is the mixture coefficient satisfying $\sum \nu_k = 1$.

This can be interpreted as follows: when we generate a word from the cluster model CLM, we first choose a search S_k with probability ν_k, then choose a word from SLM_k according to its distribution.

Different weighting schemes are possible for setting ν_k. The simplest is to assign equal weight to each search in the cluster, but this has several drawbacks.

First, a query can occur many times in the interest pattern, so we want to avoid assigning too much weight to a repeated query. Inspired by the well-known TF formula in BM25 [12, 13], we propose a sublinear transformation to normalize repeated queries:

$$\nu_i = 1/(C(q_i, P) + \alpha)$$

where $C(q_i, P)$ is query q_i's frequency in cluster P.

Second, some sessions may contain a lot of searches, which can cause redundancy when combining the search models, so we use a similar sublinear transformation scheme:

$$\nu_i = 1/(|session(S_i)| + \beta)$$

where $|session(S_i)|$ is search S_i's session size.

To incorporate both query frequency and session size, we use the following empirical formula:

$$\nu_i = 1/(\max(C(q_i, P) + \alpha, |session(S_i)| + \beta))$$

This way, a session with more searches or a query with more occurrences still takes a larger share in the cluster model, but the marginal share increase decreases with the size.

It is also possible that searches that are more recent in time or have more clicks are more important. We'd like to explore weighting schemes for such cases in future work.

5. EXPERIMENT RESULTS

5.1 Data collection

We use a web search log from a commercial search engine [8] that spanned three months. Each query is associated with a user id, so we can recover the complete search history of a particular user. We randomly sample 100 users who have made at least 100 unique clicks during this three-month period. For an average such user, there are 334 queries, out of which 267 are unique. Each session has 1.8 queries on average, and each query has 1.3 clicks.

Because the search log does not come with result text, we crawled the same search engine for up to 20 result snippets per query. Many results have changed their ranking or content between initial search and the backfill crawl, but we only use result text during pseudo feedback and implicit feedback to compute search language models. Being concise search summaries, the language models would not be sensitive to minor result changes. We also pool all the terms from the crawled search results, which we use to compute the background language model in Section 3.1.

5.2 Simulated evaluation methodology

Due to the novelty of our problem of recommendation from search log, evaluation is a challenging task. An ideal way would be to develop a real recommendation system backed by a real search engine, so as to accumulate search logs and directly engage users. With such a system, we can run content recommendation using the proposed algorithms to serve live user traffic, and evaluate their performance based on online user feedback/rating. However, it needs substantial engineering efforts and resources to build and deploy a real-world search engine and recommendation system that are useful enough to attract any reasonable amount of user commitment.

The alternative is to use existing search logs that are already available. One way is to do offline evaluation by recruiting a panel of judges, who are presented with recommendations made from users' search logs, and decide whether they are relevant or not. The results from different judges can be cross-checked for agreement. Because the judges are not the original users, they have to guess a user's expected response to a recommendation. They can do this by examining a user's search log to understand user background and infer user interests, but this third-party interpretation is difficult, subjective and unreliable. Moreover, manual inspection of user search log, which can contain sensitive personally identifiable information, is a potential infringement on user privacy.

Given an existing search log collection, if we use the discovered LEIPs to recommend items such as news, products or ads, we don't have the original user to give feedback, so we face the problems of having to seek relevance decisions from recruited judges. However, if what we recommend are search results instead of genre like news, then we have abundant implicit feedback available from the search users. Consider a clicked search result that has not been seen by the user prior to the search. The search action alone reveals the user's interest in the query subject, and the click action is a strong implicit feedback that indicates the result's relevancy. And because the search result is previously unknown to the user, it provides new information value. Therefore, such search result satisfies the relevancy and novelty criteria that are needed for a recommendation to be successful. Indeed, if we can somehow recognize new search results that are likely to be clicked by the user, it would make very effective recommendation.

This motivates us to design a simulated method for evaluating recommendation based on personal search log. We divide a user's search log into training data and test data. The training part is used to learn user interests, while the test part is used for simulated evaluation. We consider all newly clicked search results in the test part as items having recommendation value, so the aim of the recommendation algorithm is to recover as many such items as possible. The evaluation is simulated because we use search results, which are usually not candidates for recommendation, and the judgments come from implicit feedback rather than real user input.

We propose to evaluate recommendation performance using recall:

$$recall = \frac{\#(\text{new clicked results recommended by algorithm})}{\#(\text{total new clicked results})}$$

A new click is defined as the first time in search history that the user clicks on a given url. Later clicks do not count.

We do not use precision because clicks as implicit feedback is not very useful for indicating negative relevance: a user often stops at the first result that satisfies the search need, but the remaining unclicked ones could still be interesting to the user.

5.3 Evaluating the LEIP hypothesis

We hypothesized that LEIPs are good for recommendation because they satisfy both relevancy and novelty criteria. We reason that by capturing long-lastingness we ensure the recommended items would be something that the user has stable interest in, and by focusing on exploratoriness we ensure the recommended items would be on subjects that the user is willing to explore for new information. In this section we show LEIPs indeed yield better recommendation performance than ordinary interest patterns.

Following the simulated evaluation setup, we divide each user's search log equally into a training half and a test half. We compute search language model for every search, then cluster searches from the training half to get candidate interest patterns. We obtain an average of 28 interest patterns per user, each containing 6 queries and 8 clicks on average. For each interest pattern, we estimate the cluster language model, then compute its recall on the test half. We consider a clicked search result to be recommended by the interest pattern if the cosine similarity between the interest pattern and clicked search is greater than a threshold (0.1).

We measure the long-lastingness of an interest pattern using *number of clicks*, and measure its exploratoriness using *click url entropy*. For each user, we sort the interest patterns in increasing order of long-lastingness measure and partition them into four strata. The first stratum contains bottom 25% patterns in terms of long-lastingness, the second stratum contains 25-50%, the third stratum contains 50-75%, and the fourth one contains top 25% patterns in terms of long-lastingness. Within each stratum, we further sort the interest patterns in increasing order of exploratoriness measure, and divide each stratum into 4 sub-strata that are differentiated by their exploratoriness. This way each sub-stratum contains a roughly even number of patterns (minor difference due to rounding). We compute the average recall of patterns for each long-lastingness and exploratoriness combination, across all 100 users, as shown in Table 1. An average recall of 0.10 would mean the average pattern captures 10% of all clicked search results in the test half of the search log that have not been previously seen by the user.

Table 1: Average recall at different long-lastingness & exploratoriness strata

Strata L1-4 are partitioned by increasing long-lastingness (number of clicks). Strata E1-4 are partitioned by increasing exploratoriness (click url entropy).

	E1	E2	E3	E4
L1	0.029	0.023	0.025	0.022
L2	0.030	0.039	0.035	0.037
L3	0.039	0.048	0.061	0.071
L4	0.068	0.094	0.098	0.129

We make the following observations from Table 1:

First, average recall is consistently higher for interest patterns that are in a more long-lasting stratum. For example, between L4:E4 and L3:E4, which have the same exploratoriness sub-stratum, average recall increases from 0.071 to 0.129. This verifies our hypothesis that long-lasting interest patterns are more likely to generate recommendations that are still of interest to the user.

Second, for interest patterns in the higher long-lastingness strata (E3-4), average recall is consistently higher for interest patterns that are in a more exploratory sub-stratum. For example, between L3:E3 and L3:E4, which have the same long-lastingness stratum, average recall increases from 0.061 to 0.071. This verifies our hypothesis that exploratory search interest patterns are more likely to generate recommendations that provide new information to the user.

Finally, for interest patterns in the lower long-lastingness strata (E1-2), there is no obvious trend when we move between the exploratoriness sub-strata. Our explanation is that, for search patterns that are not long-lasting, it may not help much to focus on exploratoriness, because the patterns might reflect a passing information need that does not persist into the future anyway. With the smaller number of searches in a non-long-lasting pattern, having higher exploratoriness value might mean more difficulty in estimating a coherent cluster language model, due to the higher diversity of searches in the cluster.

The recall numbers may look low, because we test individual search pattern's ability to retrieve *all* new clicks in the test set. Naturally a single search cluster cannot cover all topics in the entire test half. But if we combine the search patterns (e.g. top 5 LEIPs), their collective recall is much higher. This is shown in the next section.

We use number of clicks and click url entropy here, but using other long-lastingness/exploratoriness measures (e.g. number of queries and query entropy) yields similar trends.

5.4 Evaluating different weighting schemes for cluster language model computation

Each LEIP is associated with a cluster language model CLM that captures the user's information need for the search interest pattern. The quality of the language model directly affects LEIP's ability to match items that are suitable for recommendation. In Section 4, we discussed several weighting schemes for computing the cluster language model. We now compare their recommendation performance.

For each user, we select top 5 clusters ranked by "number of sessions with new queries", which is a measure that combines long-lastingness and exploratoriness. These LEIPs are evaluated on the test search log to see how many newly clicked search results they can collectively recover, and the recall is averaged over all users. Because the LEIPs stay the same and only the cluster language models change, we are able to compare the different weighting schemes by looking at their recommendation performance.

Also included here is a baseline approach that does not involve clustering. Basically we create a degenerative single cluster that holds all searches in the training part, and compute its language model by averaging all search language models. We'd like to see how the multiple-LEIP approach compares to this single-model approach.

Table 2: Performance of different cluster models

	average recall of new clicks
single cluster	0.232
top 5 clusters	0.281
query α=0	0.287
session β=0	0.294
query $\alpha = 0$, session β=0	0.297

We make the following observations from Table 2.

- If we only use a single degenerative cluster to capture user interest, the average recall is relatively low at 0.232. This is because the single language model needs to cater to different topics in search log, so it does not track well multiple aspects of user interests.
- We get a 21.1% improvement if we use the top 5 LEIPs. The multiple search clusters have capacity to capture the many facets of user interests. We also observe that a relatively large number of future new clicks (28.1%) are related to some searches in the past.
- If we introduce α to reduce the weight on frequent queries, we are able to improve recommendation performance. This shows the necessity to prevent frequent queries from dominating the cluster language model.
- Similarly, if we introduce β to reduce the weight on large sessions, we see recommendation performance improves.
- We get the best performance by combining both penalty on frequent queries and large sessions. We tried other values of α and β, but were not able to see improvement.

6. CONCLUSIONS

In this paper, we define a new search log mining problem, in which we aim at discovering user-specific long-lasting exploratory interest patterns (LEIP) from a user's personal search history. We hypothesize that long-lastingness and exploratoriness are necessary for a search interest pattern to generate successful recommendation. We use an approach that is based on language modeling and clustering. Due to the size of search log data and it being highly personal, we propose a fully automatic approach for evaluating recommendation performance to replace human judgment. The simulated evaluation validates our LEIP hypothesis, and shows we can discover interest patterns from search logs that would be suitable for personalized recommendation.

7. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grant CNS-1027965, by U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265, and by a Sloan Research Fellowship.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26, 2006.

[2] J. A. Aslam, E. Pelekhov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *J. Graph Algorithms Appl.*, 8:95–129, 2004.

[3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART. In *TREC*, 1994.

[4] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[5] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[6] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. SIGIR '11, pages 5–14, 2011.

[7] Y. Lv, T. Moon, P. Kolari, Z. Zheng, X. Wang, and Y. Chang. Learning to model relatedness for news recommendation. WWW '11, pages 57–66, 2011.

[8] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *The First International Conference on Scalable Information Systems*, 2006.

[9] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. SIGIR '98, pages 275–281, 1998.

[10] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. KDD '05, pages 239–248, 2005.

[11] S. Robertson. Introduction to the special issue: Overview of the trec routing and filtering tasks. *Inf. Retr.*, 5(2-3):127–137, 2002.

[12] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. SIGIR '94, pages 232–241, 1994.

[13] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *TREC*, page 0, 1994.

[14] D. Shen, M. Qin, W. Chen, Q. Yang, and Z. Chen. Mining web query hierarchies from clickthrough data. In *AAAI*, pages 341–346, 2007.

[15] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05*, pages 43–50, 2005.

[16] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. KDD '06, pages 718–723, 2006.

[17] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. SIGIR '05, pages 449–456, 2005.

[18] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, pages 87–94, 2007.

[19] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01*, pages 403–410, 2001.

[20] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. SIGIR '02, pages 81–88, 2002.