Analysis of Trail Algorithms for User Search Behavior

Surabhi S. Golechha, Prof. R.R. Keole

***Abstract*— Web log data has been the basis for analyzing user query session behavior for a number of years. Web Search has become a popular tool for finding information in our daily lives. Web search engines provide keyword access to web content. In response to search queries, these engines return lists of Web pages ranked based on estimated relevance. Information retrieval (IR) researchers have worked extensively on algorithms to effectively rank documents. Web search logs have been studied mainly at session or query level where users may submit several queries within one task and handle several tasks within one session. A new concept have been introduced "task trail" to understand search behaviors. Task is defined as an atomic user information need. The goal is to obtain more precise information about the search strategy of the user. In this paper, we analyze and compare task, session and query trails for search applications as well as classified whether two queries are from same task and used clustering algorithms to combine similar queries into the same task.**

***Index Terms*—Search log analysis, Search trails, task evaluation, task trail**

I. INTRODUCTION

Logs containing the search engine interactions of many users have been mined extensively to enhance search-result ranking. Richer log data from sources such as browser toolbars offers insight into the behavior of many users beyond search engines. Search trails comprising a query and post-query page views can be mined from these logs [1]. Although trail components—origins (clicked search results) and destinations (trail end points) have been used previously to support search, the typical application of trails is to better rank Web pages [2]. Search trails are a series of web pages starting with a search query and terminating with an event such as session inactivity. Although the traversal of trails following a query is common, about how much value users derive from following the trail versus sticking with the origin (the clicked search result) or jumping to the destination page at the end of the trail [3]. In this paper we present a clustering algorithm evaluating the result of queries to users of traversing multi-page search trails.

Manuscript received April, 2015.

***Surabhi S. Golechha**, Computer Science and Information Technology, H.V.P.M COET, Amravati, India, +918983799909*

***Ranjit R. Keole**, Computer Science and Information Technology, H.V.P.M COET, Amravati, India, +919823852893*

A search trail consists of an origin page, intermediate pages, and a destination page. Origin pages are the search results that start a search trail. Query and origin pages from search engine click logs can be used to improve result set relevance.
To determine the value of trail traversal we define number of trail sources. They are as follows:

Origin: The first page in the trail after the SERP (search engine result page), visited by clicking on a search result hyperlink. This is regarded as a baseline in this study since current search engines show this source alone in search results.
Destination: The last page in the trail, visited prior to trail termination through a follow-up query or inactivity timeout. Destinations are defined similarly to the popular destinations from White et al. [3].
Sub-trail: All pages in the trail except for destination, including all post-SERP pages.
Full-trail: The complete trail, including all post-SERP pages.

Including user satisfaction analysis, user search interest prediction, website recommendation etc, previous work has shown that search logs can be used in various applications. Work had been analyzed mostly on web search logs at query or session level, where a session is defined as "a series of queries by a single user made within a small range of time". However, few of them have considered search logs at task level. Queries are often ambiguous and short therefore segmentation of sessions into tasks is non-trivial. Beyond timeout based method, several work tried to improve session boundary detection by adding features such as query reformulation patterns. Jones et al. [4] proposed to extract tasks from sessions by using features based on time stamp, query terms, etc. Lucchese et al. [5] proposed to leverage information from Wikipedia and Wiki dictionary to further improve the performance of task identification. Although previous studies have discussed the problem of session boundary detection and task identification [4], [5]. In this paper we systematically analyze and compare the effectiveness of using task trails, session trails, and query trails for search applications.

Searching activities of users in search engines is recorded by web search logs. Previous studies have shown that search logs can be used in various applications including user satisfaction analysis, page utility estimation, user search interest prediction, query suggestion, webpage re-ranking [1], website recommendation, etc. Most of previous work analyzed web search logs at session or query level, where a session is defined as "a series of queries by a single user made within a small range of time". However, few of them have considered search logs at task (atomic user information need) level.

Trails can be presented as an alternative to result lists, as instant answers above result lists, in pop-ups shown after hovering over a result, below each result along with the snippet and URL, or even on the click trail a user is following. Follow-up user studies and large-scale flights will further analyze trail appropriateness for different queries and compare trail selection algorithms and trail presentation methods.

II. LITERATURE REVIEW

In 2004, K. Sugiyama tried constructing user profiles from past brosing behavior of user.

In 2009, Attenberg used query logs to study user behavior on sponsored search results.

In 2010, R. White and J. Haung [1] presented a log-based study estimating the user value of trail following. The findings have implications for the design of search systems, including trail recommendation systems that display trails on search result pages.

In 2010, B.Xiang, D. Jiang, J.Pei, Q. He, Z. Liao [6] studied the problem of using context information in ranking documents in Web search. It conducted an empirical study on real search logs and developed four principles for context-aware ranking. Also, adopted a learning-to-rank approach and incorporated our principles to ranking models.

In 2010, A. Hassan, R. Jones, and K. Klinkner [7] addressed the problem of predicting user search goal success by modeling user behavior. It shows empirically that user behavior alone can give an accurate picture of the success of the user's web search goals, without considering the relevance of the documents displayed.

In 2010, F. Radlinski and N. Craswell [8] presented a detailed comparison between performance as measured by judgments-based information retrieval metrics and performance as measured by usage-based interleaving on five real pairs of web search ranking functions.

In 2010, A. Singla, R. White, J. Huang quantified the benefit that users currently obtain from trail following and compare different methods for finding the best trail for a given query and each top-ranked result.

In 2011, A. Hassan, Y. Song, L.-w. He [9] performed a large scale user study collected explicit judgments of user satisfaction with the entire search task. Results were analyzed using sequence models that incorporate user behavior to predict whether the user ended up being satisfied with a search or not. Metric on millions of queries collected from real Web search traffic and show empirically that user behavior models trained using explicit judgments of user satisfaction outperform several other search quality metrics.

In 2011, C. Lucchesez, S. Orlandoy, R. Peregoz, F. Silvestriz, G. Tolomeizy[4] proposed a clustering-based solution, leveraging distance measures based on query content and semantics, while query timestamps were used for a first pre-processing breaking phase.

In 2011, Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao and H. Li,[11] proposed a novel context-aware query suggestion approach. The experimental results clearly show that approach outperforms three baseline methods in both coverage and quality of suggestions.

In 2012, O. Chapelle,T. Joachims, F. Radlinski,Y. Yue[12]extended and combined the body of empirical evidence regarding interleaving, and provided a comprehensive analysis of interleaving using data from two major commercial search engines and a retrieval system for scientific literature

In 2012, Y. Song, D. Zhou, L. w. He [13] presented a novel query suggestion framework which extracted user preference data from user sessions in search engine logs. We then used the user patterns to build two suggestion models.

In 2013, H. Wang, Y. Song, M.-W. Chang, X. He, R. White, and W. Chu [14] targeted the identification of long-term, or cross-session, search tasks (transcending session boundaries) by investigating inter-query dependencies learned from users searching behaviors. A semi-supervise0d clustering model is proposed based on the latent structural SVM framework and a set of effective automatic annotation rules are proposed as weak supervision to release the

In 2013, H. Feild and J.Allan introduced a novel generalized model for generating recommendations over a search context. While only considered query text in this study, the model can handle integration over arbitrary user search behavior, such as page visits, dwell times, and query abandonment. In addition, it can be used for other types of recommendation, including personalized web search.

In 2014, Zhen Liao introduced "task trail" to understand user search behaviors. It conducted extensive analyses and comparisons to evaluate the effectiveness of task trails in several search applications: determining user satisfaction, predicting user search interests, and suggesting related queries.

III. RELATED WORK

Web logs contain a set of users, and each user has a series of successive behaviors b_1, b_2,…., b_n, where each b_i can be search behavior. A single query submitted to a search engine is search behavior. In web logs, a single query is often followed by a series of browse behaviors before the next query is submitted by the same user.

A. Query Trail

A query trail q represents a sequence of user behaviors b_1^q, b_2^q,...., b_m^q of one user u, starting from a query, followed by a sequence of browsing behaviors triggered by this query.

B. Session Trail

A session trail s is a sequence of user behaviors b_1^s, b_2^s,...., b_k^s k of one user u, where user behaviors are consecutive in search logs and any two consecutive behaviors e_i , e_{i+1} occurred within

time threshold Θ. The session strictly follows the chronological order of user behaviors in search logs. The entire search logs of one user can be segmented into a sequence of disjoint sessions along the time dimension.

C. Task Trail

A task trail *t* is a sequence of user behaviors $b_1^t, b_2^t, ..., b_r^t$ of one user *u* occurred within one session, where all user behaviors collectively define an atomic user information need.

The task segmentation actually acquires two steps since tasks are constrained in session boundary. First, segment logs into sessions according to time threshold. Second, sessions are segmented into tasks according to semantic relationships between queries. Considering that the user activities within one task may not be necessarily in series in web logs because multiple tasks can be interleaved with each other. This is one of the major differences between session and task [4], [5]. Note that we define the task trail within the session by assuming that user behavior happened beyond a timeout may have different intentions. For example, even one user searched the same query "weather" or "current time in "California" in different session; there may be variance in search results.

In this paper task is defined as an atomic user information need. Task extraction framework can be described as follows. Firstly, we learn to measure similarities between query pairs. Secondly, through clustering algorithms, queries within sessions are grouped into tasks. Using a binary classification approach, Jones and Klinkner [4] proposed to classify queries into tasks, whereas [5] proposed to cluster queries into the same task. Our method classifies whether two queries are from same task and to merge similar queries into same task, we use clustering algorithms.

As discussed and tested in [5], this method performs better than Query Flow Graph, DBScan, K-means. However, in constructing the graph and extracting connected component, the time complexity can be O ($k \cdot N^2$) where k is the dimension features and N is the number of queries.

Algorithm 1: Query Clustering (QC).

Input: Query set *Q*, cut-off threshold *ct*;
Output: A set of tasks Θ;
Initialization: Θ = *Φ*; Query to task table *L*=*Φ*;

```
for len = 1 : |Q| − 1 do
  for i = 1 : |Q| − 1 do
    // if two queries are not in the same task
    if L [Q_i] ≠ L [Q_{i+len}] then
      // compute similarity takes O (k)
      s ← sim (L [Q_i];L [Q_{i+len}]);
      if s ≥ ct then
        merge Θ (Q_i) and Θ (Q_{i+len});
        modify L;
  // break if there is only one task
  if |Θ| = 1 break;
return Θ;
```

To further reduce the time cost (especially the cost of the worst cases) we propose a bounded spread version of Query clustering, named bounded query clustering. The idea is that two queries far away from each other are not likely from the same task. By setting a length bound bl, the time complexity of bounded query clustering is reduced to O (k ·bl· N). In the end, tasks of same queries are merged into a single one.

Algorithm 2: Bounded Query Clustering (BQC).

Input: Query set *Q*, cut-off threshold *ct*; bounded length bl;
Output: A set of tasks Θ;
Initialization: Θ = *Φ*; Query to task table *L*=*Φ*; M= *Φ;*

```
// initialize same queries into one task
cid=0;
for i = 1: |Q| − 1 do
  if M [Q_i] exists then
    add Q_i into Θ (M [Qi]);
  else
    M [Q_i] = cid++
if |Θ| = 1 return Θ;
for len = 1 : bl do
  for i = 1 : |Q| − 1 do
    // if two queries are not in the same task
    if L [Q_i] ≠ L [Q_{i+len}] then
      // compute similarity takes O (k)
      s ← sim (L [Q_i];L [Q_{i+len}]);
      if s ≥ ct then
:       merge Θ (Q_i) and Θ (Q_{i+len});
        modify L;
  // break if there is only one task
  if |Θ| = 1 break;
return Θ;
```

We propose a Query Clustering approach (QC) in this paper as shown in Algorithm 1. We observe that consecutive query pairs are more likely belonging to same task than non-consecutive ones; QC prefers to first compute the similarities for consecutive query pairs by timestamps. Take an example, given a series of queries $\{q_1, q_2, q_3, q_4\}$, QC will first compute for pairs $\{q_1 \rightarrow q_2, q_2 \rightarrow q_3, q_3 \rightarrow q_4\}$, it can reduce the computational cost from O ($k \cdot N^2$) to O ($k \cdot N$) if and only if there is only one task in the session. On the basis of statistics that about 50% sessions only have one task as shown in table I (D_0 - dataset consists of user browsing logs from a widely used browser plug-in toolbar, D_1-dataset consists of web search logs from Bing), QC is efficient to identify them. For sessions with multiple tasks, QC is also faster than standard implementation.

For example, if the sequences $\{q_1, q_2, q_3, q_4\}$ are grouped into $\{q_1\}$ and $\{q_2, q_3, q_4\}$, the standard approach compute all 6 query pairs but QC only needs to calculate 5 pairs while pair $\{q_2 \rightarrow q_4\}$ is skipped. That is because it skips computing the similarity of query pairs from the same task. In addition, QC needs extra O (N) space for storing a query to task mapping table, which is affordable in current applications.

TABLE I
Basic Statistics of Search Logs

Statistics	D_0	D_1
Avg. # of Queries in Sessions	5.81	2.54
Avg. # of Queries in Tasks	2.06	1.60
Avg. # of Tasks in Sessions	2.82	1.58
% of Single-Tasks Sessions	53.29	70.72
% of Multi-Tasks Sessions	46.71	29.28
% of Interleaved Tasks Sessions	15.25	4.78
% of Single-Query Tasks	48.75	71.86
% of Multi-Query Tasks	51.24	28.13

IV. Results of Search Applications

In this section, we present results of comparison on evaluating the effectiveness of task trails in real applications. Here we present the methods, metrics and findings in search applications.

A. Estimating User Satisfaction

To know whether the user is obtaining the findings from query executed by him on the search engine, we consider features which take into account the entire pattern of user search behavior, including query, click and dwell-time as well as number of reformulations.

1. Clicks: Jung et al. [11] show that considering the last click of a session may be the most important piece of information in relating user clicks to document relevance. Clicks in a user search goal as well as the times between actions allow us to predict the user's success at that goal.

2. Dwell Time: An important feature that we consider is the dwell time. Dwell time of a click is the amount of time between the click and the next action (query, click, or end). We calculate the dwell times for all clicks during goal and use the maximum, minimum, and average dwell times as features to predict success. It is widely believed that long dwell time of clicks is an important predictor of success.

3. Goal Success: We can build two Markov models to compute the probability of user success and failure. The Markov Model includes {clicks, queries, dwell time (>30 sec)} as states {Q, SS, SS_Long}, respectively. On the basis of labelled dataset, we split two Markov models. Given a new user task trail, we can compare the probability from successful and unsuccessful models and estimate the label of user satisfaction. For more details, see [10].

By using the task success labels based on Hidden Markov Model, we can study the percentage of multitask sessions with both successful and unsuccessful tasks.

B. Prediction on User Search Interests

For improving ranking or personalization of search systems, user search interests can be captured. Previous work has taken advantage of queries and clicks preceding user current query as context to perform context-aware ranking. One of key aspect is to promote or demote the rank of URLs based on their relationship with current query's context.

Since queries submitted by users reflect user information needs, we can use queries to represent user search interests. On the other hand, queries are often short and ambiguous. Therefore, we can summarize user search interests at topic level. By taking previous co-session or co-task queries as context information to user's current query, we can construct different context models. To know which context model can predict user search interests better, we can compare topic similarities of co-session and co-task query pairs.

V. Conclusion

The result of the analysis of user search behavior trails has developed to use task trail for better understanding of user search behaviors. We observed and found result on evaluation of the effectiveness of task trails in real applications. We have taken extensive analyses and comparisons to the effectiveness of task trails in several search applications: estimating user satisfaction, prediction on user search interests. To reduce the time complexity, we have also proposed clustering algorithm to merge similar queries into same task, which performs better than other approaches. We defined query, session and task trails and found users are more likely to find useful information following the task trails.

Acknowledgment

My thanks to the guide Prof.R.R.Keole and Principal Dr.A.B.Marathe, who provided me constructive and positive feedback during the preparation of this paper.

References

[1] R. White and J. Huang, "Assessing the scenic route: measuring the value of search trails in web logs," in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, 2010, pp. 587–594.
[2] L. D. Catledge and J. E. Pitkow, "Characterizing browsing strategies in the world-wide web," Comput. Netw. ISDN Syst., vol. 27, no. 6, pp. 1065–1073, 1995.
[3] Z. Liao, Y. Song, L. -w. He, and Y. Huang, "Evaluating the effectiveness of search task trails," in Proc. 21st Int. Conf. World Wide Web, 2012, pp. 489–498.
[4] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei, "Identifying task-based sessions in search engine query logs," in Proc. 4thACMInt. Conf.WebSearch DataMining, 2011, pp. 277–286.
[5] R. White, P. Bennett, and S. Dumais, "Predicting short-term interests using activity-based search context," in Proc. 19th ACM Int. Conf. Inform. Knowl. Manage., 2010, pp. 1009–1018.
[6] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li, "Contextaware ranking in web search," in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, 2010, pp. 451–458.
[7] A. Hassan, R. Jones, and K. Klinkner, "Beyond DCG: User behavior as a predictor of a successful search," in Proc. 3rd ACM Int. Conf. Web Search Data Mining, 2010, pp. 221–230.
[8] F. Radlinski and N. Craswell, "Comparing the sensitivity of information retrieval metrics," in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, 2010, pp. 667–674.
[9] A. Hassan, Y. Song, and L.-w. He, "A task level user satisfaction metric and its application on improving relevance estimation," in Proc. 20thACMInt. Conf. Inform. Knowl. Manage., 2011, pp. 125–134.
[10] Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. He, and H. Li,"Browserank: letting web users vote for page importance," in Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, 2008, pp. 451–458.
[11] Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao, and H. Li, "Mining conceptsequences from large-scale search logs for context-aware query

suggestion," ACM Trans. Intell. Syst. Technol., vol. 3, pp. 17:1–17:40, 2011.
[12] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue, "Large-scale validation and analysis of interleaved search evaluation," ACM Trans. Inform. Syst., vol. 30, no. 1, 2012.
[13] Y. Song, D. Zhou, and L.-w. He, "Query suggestion by constructing term-transition graphs," in Proc. 5th ACM Int. Conf. Web Search Data Mining, 2012, pp. 353–362.
[14] H. Wang, Y. Song, M.-W. Chang, X. He, R. White, and W. Chu, "Learning to extract cross-session search tasks," in Proc. 22nd Int. Conf. World Wide Web, 2013, pp. 1353–1364.

Surabhi S.Golechha is currently pursuing master's degree program in Computer Science and Information Technology Engineering in HVPM's College of Engineering &Technology, Amravati, India.

.

Ranjit R. Keole is working as Assistant Professor in HVPM's College of Engineering & Technology, Amravati, India.

