Task Trail: An Effective Segmentation of User Search Behavior

Zhen Liao, Yang Song, Yalou Huang, Li-wei He, Qi He

Abstract—In this paper, we introduce "task trail" to understand user search behaviors. We define a task to be an atomic user information need, whereas a task trail represents all user activities within that particular task, such as query reformulations, URL clicks. Previously, 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. Although previous studies have addressed the problem of task identification, little is known about the advantage of using task over session or query for search applications. In this paper, we conduct 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. Experiments on large scale datasets of a commercial search engine show that: (1) Task trail performs better than session and query trails in determining user satisfaction; (2) Task trail increases web page utilities of end users comparing to session and query trails; (3) Task trails are comparable to query trails but more sensitive than session trails in measuring different ranking functions; (4) Query terms from the same task are more topically consistent to each other than query terms from different tasks; (5) Query suggestion based on task trail is a good complement of query suggestions based on session trail and click-through bipartite. The findings in this paper verify the need of extracting task trails from web search logs and enhance applications in search and recommendation systems.

Index Terms—Search log mining, task trail, task evaluation, log analysis

1 Introduction

Web search logs record the searching activities of users in search engines. Previous studies have shown that search logs can be used in various applications including user satisfaction analysis [1], page utility estimation [2], user search interest prediction [3], query suggestion [4], web page re-ranking [5], web site recommendation [6], 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" [6], [7]. However, few of them have considered search logs at task (atomic user information need) level.

Consider the example shown in Table 1, which is a real user search session from Bing (http://www.bing.com). This session contains 4 different search tasks: Facebook, Amazon Kindle Books, Gmail, and lyrics of a song. The "Gmail" task is interleaved with the "Amazon Kindle Books" task. The reasons causing the interleave phenomenon are: (1) web search logs are ordered chronologically; (2) users often open several tabs or browsers and conduct multiple tasks at the same time.

Table 1 indicates that the granularity of session is too coarse to manifest details of user behaviors by missing multiple tasks within a session. The query level analysis is the finest grained, but fails to capture the interleave relationships between tasks and the generalization/specification/refinement relationships between adjacent queries within the same task. Our analytics based on search logs of 159, 668, 543 users in 3 months find that following a widely used session definition (30 minutes timeout [8], [9], about 30% of the sessions contain multiple tasks and about 5% of the sessions contain interleaved tasks. Thus, task-level log analysis strikes a good balance between details of user behaviors and relationships between queries.

Previous work attempted to enhance session-level analysis by distilling semantic features such as query reformulation patterns [10], [11] into the session boundary detection. However, the tasks have not been explicitly identified. Jones et al. [12] first extracted tasks from sessions based on time and query word features. Lucchese et al. [13] further improved task identification by leveraging external encyclopedias like Wikipedia. However, all these work only focuses on task extraction. To the best of our knowledge, this paper is the first work that systematically analyzes the utilities of task-level search log analysis and compare it with session-level and query-level search log analyses in real applications.

In this paper, we name the task-level search log analysis as task trail because task trail vividly reveals the image of chronologically ordered tasks buried in massive search logs. Accordingly, we compare task, session and query trails in the following search applications: (1) Determining user satisfaction, where implicit feedback such as clicks, dwell time [1] and success scores of Markov models [14] are mined to measure user satisfaction and page utility. (2) Predicting user search interests, where ODP (http://www.dmoz.org/) category information is adapted to measure topic similarity for predicting user search

This paper was supported by the National Natural Science Foundation of China under Grant NO. 61105049 and NO. 61300166.

TABLE 1
A sample session from web search logs.

Time	Event	Value	Task
09:03:26	Query	facebook	1
09:03:39	Click	www.facebook.com	1
09:06:34	Query	amazon	2
09:07:48	Query	faecbook	1
09:08:02	Click	facebook.com/login.php	1
09:10:23	Query	amazon kindle	2
09:10:31	Click	kindle.amazon.com	2
09:13:13	Query	gmail log in	3
09:13:19	Click	mail.google.com/mail	3
09:15:39	Query	amazon kindle books	2
09:15:47	Click	amazon.com/Kindle-eBooks...	2
09:15:59	Click	astore.amazon.com/Amazon..	2
09:17:51	Query	i'm picking up stones	4
09:18:54	Query	i'm picking up stones lyrics	4
09:19:28	Query	pickin' up stones lyrics	4

interests. (3) Suggesting related queries, where co-occurrence-based query suggestion methods based on task trail are compared to the same methods based on session trail [15], [16] and the method based on click-through bipartite graph [17]. (4) Measuring ranking functions, where user satisfaction rates are mined to measure the difference of two ranking functions in session, task and query levels. All these experimental results collectively verify the advantages of task trails.

From our extensive analytics on 3-months search logs of 0.16 billion users on Bing, the interesting findings include: (1) Task Trail performs better than session and query trails in determining user satisfaction; (2) Task trail increases web page utilities of end users comparing to session and query trails; (3) Task trails are comparable to query trails but more sensitive than session trails in measuring different ranking functions; (4) Query terms from the same task are more topically consistent to each other than query terms from different tasks; (5) Query suggestion based on task trail is a good complement of query suggestions based on session trail and click-through bipartite graph. Our findings pave the way for practical applications of task trail in Web search and recommender systems.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we introduce the definition and extraction methods of task trails. The experiment results of evaluating the effectiveness of task trails are reported in Section 4. We draw conclusions in Section 5.

2 Related Work

Web search logs record user activities on search engines, such as queries and clicks. Search trails record the footprints left by users in their search processes. In the literature of studying search trails, much of previous work have applied them in the applications of user satisfaction analysis, ranking function evaluation, query suggestion, etc. Here we classify related works into four categories: (1) user behavior segmentation, (2) user satisfaction and interests analysis, (3) enhance ranking based on web logs, and (4) query suggestion.

2.1 User Behavior Segmentation

Session and task are two primary types of user search behavior segmentations. The term *session* was proposed in [7], [8]. Catledge et al. [8] analyzed user browsing logs captured from client-side user events. They found that 25.5 minutes timeout is good for separating consecutive user activities into different sessions. Silverstein et al. [7] defined "session" as "a series of queries by a single user made within a small range of time" in their study of AltaVista search logs, where they used 5 minutes as the timeout threshold. He et al. [10] proposed to detect session boundary based on time and query reformulation patterns and found that 10 to 12 minutes timeouts are good. Jansen et al. [11] clarified the session as "a series of interactions by the user towards addressing a single information need" and found that about 30 minutes timeout is better than others. As a result, later studies [4]–[6], [11], [18] often used 30 minutes timeout for session segmentation.

Considering the multitasking behaviors within a session, Jones and Klinkler [12] proposed to classify query pairs into a same task via features based on time, word, web search results, etc. Their approach achieved about 90% accuracy in task boundary detection and same task identification. Boldi et al. [19] applied the *query flow graph* in finding logical session and query recommendation. They formulated the problem of mining logical sessions as an Asymmetric Traveling Salesman Problem. Lucchese et al. [13] proposed to identify task-based sessions by combining content (query word, edit distance) and semantic (Wikipedia) features. Donato et al. [20] proposed to identify those complex tasks as research missions which need users to explore multiple pages. Kotov et al. [21] proposed to model and analyze cross-session search tasks, and they applied classification approach to predict the revisiting likelihood of tasks.

In this paper, we adapt methods described in [12], [13] to extract tasks from sessions. We learn the query similarity function via machine learning and cluster queries within a session into tasks.

2.2 User Satisfaction and Interest Analysis

Implicit feedback existing in the user search process can be used to measure user satisfaction and interests. Fox et al. [1] studied the relationship between implicit feedback signals and explicit user satisfaction ratings. According to "gene analysis" on patterns of user behaviors, they found that the dwell time on search result pages is a good indicator for user satisfaction. Hassan et al. [14] proposed to formulate the search process by Markov models. The experiment results on 2,712 labeling goals showed that their approach could model the search process well and have a better prediction of user goal success than discounted cumulative gain (DCG). White et al. [2], [6] conducted extensive studies on search and browser logs. They found that following the query trails, users can find more useful information. White et al. [22] found that short-term user interests could be well captured by the previous submitted queries and visited web pages within the same session. Olston and Chi [23] proposed ScentTrail to combine searching and browsing activities into a single interface, and they found it can help users in finding information faster than by only searching or browsing alone.

2.3 Enhance Ranking Based On Web Logs

Utilizing search and browser logs to enhance ranking is a promising and important direction. There are several methods to enhance ranking by web logs: (1) improving the page importance estimation of documents [18], (2) improving the relevance of query-document pairs [24], [25], (3) improving the evaluation of ranking functions using log-based measures [26].

Liu et al. [18] modeled the user browsing behavior as a continuous Markov model and propose a BrowseRank algorithm to estimate the web page importance. Craswell and Szummer [24] proposed a backward random walk on click graph and validated its effectiveness in image retrieval. To address the sparseness problem of the click-through bipartite graph, Gao et al. [25] proposed two smoothing techniques for estimating the relevance of query document pairs. Consider context information is useful for ranking, Shen et al. [27] proposed context sensitive information retrieval framework based on language model and tested their approach on TREC datasets. Xiang et al. [5] proposed several context-aware ranking principles by promoting or demoting web pages based on relationship of adjacent query pairs within a session.

Implicit metrics extracted from web logs can be used for measuring ranking functions, which can save the cost of traditional ranking metrics (e.g., NDCG) based on human annotation. Recent work [26], [28] showed that results of interleaving experiments can be used to measure ranking functions. They showed that interleaving experiments are reliable, sensitive, and also have high correlation with standard measures. Hassan et al. [29] showed that a task level metric can be sensitive to tell different ranking functions apart.

2.4 Query Suggestion

The related queries mined from sessions and click through bipartite graphs can be used for query suggestion. Beeferman et al. [30] proposed to group queries and URLs in the click graph for query suggestion. Huang et al. [15] proposed to use co-occurred query pairs from sessions as suggestions. Jones et al. [16] generated query substitution for sponsored search and applied log likelihood ratio (LLR) to measure the correlation of query pairs. Boldi et al. [19] proposed *query flow graph* and applied query flow graph in query recommendation. Mei et al. [17] proposed to use hitting time of query pairs on a click graph for query suggestion. Cao et al. [4] combined both click-through and session logs to mine concept sequences for context-aware query suggestion. After grouping similar queries into concepts via their efficient algorithm, suggestions can be generated at concept level. Song et al. [31] mined the term transition graph from consecutive query pairs and applied term transition graph into tail query suggestion. Jain et al. [32] proposed to generate high utility query suggestions based on session logs, click-through logs, and web corpus.

2.5 Our Goal

The study described in this paper differs from previous work in that we focus on the comparison of task, session, and query trails in the applications of determining user satisfaction, predicting user search interests, suggesting related queries and measuring ranking functions, rather than identifying session boundary [10], [11], extracting tasks from session [12], [13], or estimating web page relevance [5], [25], [27]. As a result, task trails can be considered as a novel way to segment search logs and an additional information source to classic session and query trails.

3 Task Definition and Extraction

We introduce the definition of task as well as the process of identifying and extracting tasks from web logs in this section.

3.1 Definition

Web logs contains a set of users, and each user has a sequence of consecutive behaviors $e_1, e_2, ..., e_n$, where each e_i can be a search behavior or a browse behavior. A search behavior is a single query submitted to a search engine. A browse behavior belongs to one of the following activities: 1) user starts to surf from the homepage of the browser; 2) user types a URL address in the browser; 3) user pastes the URL address from other place into browser; 4) user clicks a bookmark or favorite page in the browser; 5) user clicks the "back" or "forward" button in the browser; 6) user clicks an anchor link or a search result.

In web logs, a single query is often followed by a sequence of browse behaviors before the next query is submitted by the same user. Thus, the simplest search log segmentation is to treat one query plus its followers as an independent query trail, as defined below.

Definition 3.1 (Query Trail): A query trail q represents a sequence of user behaviors $e^q_1, e^q_2, ..., e^q_m$ of one user u, starting from a query, followed by a sequence of browsing behaviors triggered by this query.

For example, in Table 1, URLs "http://www.amazon.com/Kindle-eBooks/b?ie=UTF8\&node=1286228011" and "http://astore.amazon.com/Amazon.Kindle.Books.Store.-20/" belong to the query trail starting from query "amazon kindle books". For simplicity, we use the starting query q to represent the whole query trail as well as information need of the trail.

The disadvantage of query trails is obvious: the semantic associations between adjacent query trails are lost. So, related work [2], [4], [18] used time threshold (i.e., 30-minute) to group query trails into sessions.

Definition 3.2 (Session Trail): A session trail s is a sequence of user behaviors $e^s_1, e^s_2, ...e^s_k$ 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.

As Table 1 illustrates, it is common that a session contains multiple atomic information needs ("Facebook", "Amazon Kindle Books" etc.) which are semantically irrelevant to each other, even under the well-accepted 30-minute time threshold setting. In fact, we have examined various time threshold settings in experiments for session segmentation, none of which is able to satisfy our predefined goals (ref. Section 4).

As a result, in this paper we use the "task" as the log segmentation unit to represent individual atomic information need, as defined below.

Definition 3.3 (Task Trail): A task trail t is a sequence of user behaviors $e^t_1, e^t_2, ..., e^t_r$ of one user u occurred within one session, where all user behaviors collectively define an atomic user information need.

Tasks are constrained in session boundary. Therefore, the task segmentation actually incurs two steps: 1) segment logs into sessions according to time threshold; 2) segment a session into tasks according to semantic relationships between queries. Note that the user activities within one task may not be necessarily consecutive in web logs because multiple tasks can be interleaved with each other (ref. Table 1). This is one of the major differences between session and task [12], [13]. 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 New York" in different session, the search result may vary a lot.

3.2 Task Extraction

A task can be seen as a set of semantically relevant query trails within one session. Since we use the starting search query of each query trail to represent the entire query trail, a task can be simply represented by all such starting search queries from query trails. The problem left is to merge similar queries together. However, how to unambiguously define the semantic relevance between queries remains challenging. Some straightforward rules can be applied to group two queries into the same task if: (1) they are identical; (2) one is a part of the other (e.g., "Disney" and "Disney movies"); (3) two partially agree to each other (e.g., "Seahawks result" and "Seahawks score"); (4) one is a typo of the other (e.g., "machnie learning" and "machine learning"). We leverage these rules in the annotation process and propose an efficient clustering framework to automatically group queries into tasks.

The basic ideas of our clustering framework are described as follows. First, since tasks are extracted out from each session, we follow the time threshold method to segment logs into sessions by choosing a time threshold θ. In experiments, to alleviate the bias of a time threshold, different settings of θ ranging from 1 minute, 2 minutes, 5 minutes, 10 minutes, 20 minutes, 30 minutes, 60 minutes to a day are compared. Second, we quantitatively compute the similarity between any two queries. Last, queries similar to each other are clustered into the same task. This approach is motivated by [12], [13], where Jones and Klinkner [12] proposed to classify queries into tasks using a binary classification approach and Lucchese et al. [13] proposed to cluster queries into tasks based on empirically designed query distance functions.

Briefly speaking, we trained a SVM classifier [33] to learn the weights of various features of query similarity function and proposed two efficient algorithms to group similar queries into tasks. We present the details about the query similarity function and clustering algorithms with empirically evaluation in following sections.

3.2.1 Query Similarity

To learn a good query similarity function, we constructed a labeled dataset for task classification. 10,368 sessions were randomly sampled from search logs and 17,924 query pairs in those sessions are extracted. 26 annotators were organized as judge pools and each query pair is shown to at least three of them. Each time, one annotator compared two queries with their top ten search results to judge whether they are submitted for the same task according to the above rules. Labels include *same task*, *different task* or *unknown*. The final label of each query pair was

TABLE 2
Examples of labeled query pairs.

Label	Query A	Query B
Same Task	gmail.com florida statutes	login gmail florida evidence code
Diff. Task	facebook.com definitions of tarsorraphy	fallout 3 books attwireless
Unknown	sunday night football snowmobiling in Minnesota	nbc sports nfl snowmobile parts

TABLE 3
Features of query pair.

Feature Description	Weight
temporal features	
timediff_1: time difference in seconds	-0.1121
timediff_2: category for 1/5/10/30 mins	-0.0623
word features	
lv_1: Levenshtein distance of two queries	0.0106
lv_2: lv_1 after removing stop-words.	-0.1951
prec_1: average rate of common terms.	-0.2870
prec_2: prec_1 after removing stop words.	1.2058
prec_3: prec_1 (If term A contains B, A=B)	0.5292
rate_s: rate of common characters from left	1.6318
rate_e: rate of common characters from right	0.4014
rate_l: rate of longest common substring	0.4941
b_1: 1 if one query contains another, else 0	0.6361

obtained by voting. After labeling, we totally obtained 5,668 positive pairs, 9,370 negative pairs, and 1,334 unknown pairs. We ignored the unknown pairs since they are hard to understand even by human annotators. To better understand labeling results, we show some labeled query pairs in Table 2. Although previous work [12], [13] conducted their studies by manually labeling whole session into tasks, we chose to label query pairs since: (1) it can ease the effort of human annotators, and (2) the goal of our first step is to obtain a query similarity function between query pairs, so a labeled dataset with query pairs can provide adequate information.

We constructed 11 features to measure similarity between queries. These features can be classified into two categories: 1) time based (temporal); 2) query word based. We present details of features in Table 3, where 215 frequent searched but meaningless words are selected as stop words. The column *weight* in the table lists the weight of each feature for similarity function. We obtained the weights by training a linear SVM model [33] on labeled data. We chose linear-SVM as classifier because of its good performance in many applications and theoretical soundness, and recent study [34] also shows that it is a state-of-art method in computing query similarity. The whole labeled dataset was split into 5 folds for cross validation. Each time 3 folds were used for training, 1 fold was used for tuning parameter (C in SVM), and the rest 1 fold was used for testing.

Studies on these features showed that using temporal features can only achieve about 70% accuracy, using word features can achieve 91% accuracy, and

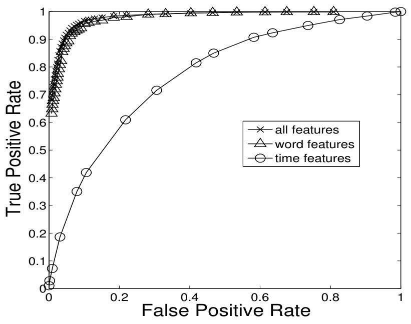


Fig. 1. ROC of features for task classification.

combining them can achieve 93% accuracy in our label dataset. The performance of temporal features is in accordance with [12], [13] no matter how long or short the timeout threshold is, it cannot achieve good precision in identifying tasks. The ROC curves of features are shown in Figure 1. As a result, the average precision on testing folds is about 93% by using all features. Therefore, given two queries, we can compute their similarity from features described in Table 3.

3.2.2 Clustering Queries into Tasks

Upon the query similarity function, we build an undirected graph for queries within a session. The vertices of the graph are queries and the edges are similarity scores between queries. After removing the suspicious edges with scores below a threshold, any connected component of the remain graph is identified as a task. This approach is named as QC-WCC (Query Clustering using Weighted Connected Component of a Graph) and outperformed other popular clustering algorithms like Query Flow Graph, K-means, and DB-Scan in task clustering, as indicated in [13]. One major shortcoming of QC-WCC is its high time complexity of constructing the graph and extracting connected component, which is $O(k \cdot N^2)$ where N is the average number of queries of a session and k is the dimension of features. Considering the massive volume of search logs and some sessions can be very long (depends on the time threshold of session segmentation), the overall time complexity for the entire search logs is intolerable.

To address the problem, Lucchese et al. [13] proposed another head-tail component query clustering approach (QC-HTC) to reduce the time complexity. The QC-HTC approach utilized the heuristic that queries are submitted sequentially by users, so that only the similarity between the head and tail parts of a query sequence should be computed. QC-HTC is an approximation of QC-WCC.

Only computing the similarity between the head and tail parts of a query sequence actually violates the task interleaving observation found by us in search

Algorithm 1: Spread Query Task Clustering (QC-SP).

```
Input: Query set $Q$, cut-off threshold $b$;
Output: A set of tasks $\Theta$ ;
Initialization: $\Theta$ = $\emptyset$; Query to task table $L$=$\emptyset$;
 1: for $len = 1 : |Q| - 1$ do
 2:   for $i = 1 : |Q| - len$ do
 3:     // if two queries are not in the same task
 4:     if $L[Q_i] \neq L[Q_{i+len}]$ then
 5:       // compute similarity takes $O(k)$
 6:       $s \leftarrow$ sim($L[Q_i], L[Q_{i+len}]$);
 7:       if $s \geq b$ then
 8:         merge $\Theta(Q_i)$ and $\Theta(Q_{i+len})$;
 9:         modify $L$;
10:   // break if there is only one task
11:   if $|\Theta| = 1$ break;
12: return $\Theta$;
```

Algorithm 2: Bounded Spread Query Task Clustering (QC-BSP).

```
Input: Query set $Q$, cut-off threshold $b$, bounded length $bl$;
Output: A set of tasks $\Theta$ ;
Initialization: $\Theta$ = $\emptyset$; Query to task table $L$=$\emptyset$, $M$=$\emptyset$;
 1: // initialize same queries into one task
 2: $cid$=0;
 3: for $i = 1 : |Q| - len$ do
 4:   if $M[Q_i]$ exists then
 5:     add $Q_i$ into $\Theta(M[Q_i])$;
 6:   else
 7:     $M[Q_i]$=$cid$++;
 8: if $|\Theta| = 1$ return $\Theta$;
 9: for $len = 1 : bl$ do
10:   for $i = 1 : |Q| - len$ do
11:     // if two queries are not in the same task
12:     if $L[Q_i] \neq L[Q_{i+len}]$ then
13:       // compute similarity takes $O(k)$
14:       $s \leftarrow$ sim($L[Q_i], L[Q_{i+len}]$);
15:       if $s \geq b$ then
16:         merge $\Theta(Q_i)$ and $\Theta(Q_{i+len})$;
17:         modify $L$;
18:   // break if there is only one task
19:   if $|\Theta| = 1$ break;
20: return $\Theta$;
```

logs. From the intuition that consecutive queries more likely belong to the same task than non-consecutive ones, a better approximation is to compute the pair-wise similarity for all consecutive query pairs. In this paper, we thus propose two new clustering algorithms, named QC-SP (Query Clustering using SPread method) shown in Algorithm 1 and QC-BSP (Query Clustering using Bounded SPread method) shown in Algorithm 2, for task extraction.

Let us use a toy example to explain our algorithms. Given a sequence of 4 queries $\{q_1, q_2, q_3, q_4\}$, QC-WCC needs 6 times of pair-wise relevance computations. For QC-SP, if q_1 is similar to q_2 and q_2 is similar to q_3, there is no need to compute the relevance between q_1 and q_3 any more. If q_1 is similar to q_2 but q_2 is *not* similar to q_3, QC-SP still has to compute the relevance between q_1 and q_3 to avoid the task interleaving. So, for those sessions only containing one task ($\sim$50% of logs), QC-SP reduces the time cost from $O(k \cdot N^2)$ to $O(k \cdot N)$. For sessions having multiple tasks, if some tasks have more than 2 consecutive queries, the time cost can still be reduced for the same reason. In the worst case that all tasks are short and interleaved with each other, QC-SP has the same time complexity as QC-WCC. In addition, QC-SP needs extra $O(N)$ space for storing all queries, which is affordable in practice. QC-SP is actually equivalent to QC-WCC (not an approximation).

To further reduce the time cost (especially the cost of the worst cases), we propose a bounded spread version of QC-SP, named QC-BSP. 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 QC-BSP is reduced to $O(k \cdot bl \cdot N)$. Considering the case where users repeat queries after a while [35] that possibly exceeds our length bound, same queries within a session are identified first; then, QC-BSP will only examine their own neighbors within the length bound separately. In the end, tasks of same queries are merged into a single one.

Now we compare our proposed QC-SP and QC-BSP methods with two baselines: QC-WCC and QC-HTC used [13] in two aspects: efficiency and effectiveness. To measure the efficiency, the runtime of different clustering methods are used. To measure the effectiveness, we chose to use the results of QC-WCC as ground-truth since QC-WCC was reported to have best performance in previous work [13]. Since it is expensive to annotate long sessions into tasks in a relatively large scale, we also measure the topic similarity of the clustering results using ODP (http://www.dmoz.org/) categories.

First, all clustering methods are comparing with QC-WCC as ground-truth first. The similarity of clustering results are computed by two metrics: Rand Index and Jaccard Index, where $RI = RandIndex = \frac{n_{00}+n_{11}}{n_{00}+n_{11}+n_{01}+n_{10}}$ and $JI = JaccardIndex = \frac{n_{11}}{n_{11}+n_{01}+n_{10}}$. For different clustering results C_1 and C_2, n_{11} denotes the number of object pairs belonging to same cluster in both C_1 and C_2, n_{10} denotes the number of object pairs belonging to same cluster in C_1 but not in C_2, n_{01} denotes the number of object pairs belonging to same cluster in C_2 but not in C_1, and n_{00} denotes the number of object pairs belonging to different clusters in both C_1 and C_2.

Second, queries are mapped into ODP categories using their search results. For each query, we first scratched its top ten search results from Bing(http://www.bing.com). Then we crawled the content information of each URL in the results. Topics of URLs are obtained by a URL to ODP mapping table and a content-based ODP classifier. The content-based ODP classifier was built similar to [36] based on a combination of uni-gram, bi-gram and trigram lan-

TABLE 4
Efficiency and effectiveness of different clustering methods. (RI=Rand Index, JI=Jaccard Index, HI=Histogram Index)

Method	Time (sec.)	RI	JI	HI
QC-WCC	3,093	1.000	1.000	0.528
QC-HTC	1,001	0.949	0.899	0.519
QC-SP	1,902	1.000	1.000	0.528
QC-BSP-3	242	0.939	0.855	0.533
QC-BSP-5	418	0.966	0.919	0.531
QC-BSP-10	807	0.988	0.972	0.529

guage models. The second level ODP categories with 385 sub topics are used, where topics in "/world" and "/regional" are excluded. Afterwards, topics of queries are obtained from topics of their top search results. Finally, we normalize topic distribution to let $\sum_t P(t|Q) = 1$ (t denotes a topic) for each query Q. Then we compute the topic similarity between queries Q_x and Q_y by Histogram Intersection [37] $HI = Sim(Q_x, Q_y) = \sum_t \min(P(t|Q_x), P(t|Q_y))$.

To measure the runtime performance in long sessions, we use top 1% longest sessions from a random sampled search log dataset. The average length of the sessions is 24.32 and the total number of sessions is 88,792. We set the bounded length of QC-BSP as 3, 5, and 10. Since we take QC-WCC as ground-truth, its own Rand Index and Jaccard Index are both 1.

The results are shown in Table 4. From the table, we have the following observations: (1) The runtime of QC-WCC is highest as expected. QC-SP is an acceleration of QC-WCC without any approximation, its time cost is about 63% of QC-WCC. (2) QC-BSP-3 has the lowest time cost and its result is least similar with QC-WCC since it only allows query pairs to be grouped if their order differs less than 3 (e.g., there are less than three queries between them). (3) The time cost of QC-HTC is about 32% of QC-WCC. QC-HTC performs only better than QC-BSP-3 in effectiveness, but its efficiency is worse than QC-BSP-3, QC-BSP-5, and QC-BSP-10. (4) By using a bounded length as 10, QC-BSP-10 performs similar to QC-WCC and its time cost is only about 26% of QC-WCC. That is, QC-BSP is more efficient and effective than QC-HTC. Although the improvement on the runtime seems to be marginal, when we apply the algorithm on large scale dataset where each process node needs to process lots of session and responds in limited time, we should guarantee that the worst runtime is bounded. Therefore the application of QC-BSP is non-trivial. (5) All methods have similar topic similarity in terms of ODP category without significantly difference, which means all of them can preserve the queries' topic well. Since QC-HTC may involve dissimilar queries into one task, then its HI is the lowest. Similarly, QC-BSP-10 and QC-BSP-5 are more likely group dissimilar queries than QC-BSP-3, and QC-BSP-3 is slightly better than other methods.

4 Experimental Results of Search Applications

In this section, we present detailed experimental observations and results on evaluating the effectiveness of task trails in real applications. We first show the statistics on the datasets used in the experiments, and we present the methods, metrics and findings in three search applications (determining user satisfaction, predicting user search interests, and suggesting related queries). We also present the application of measuring ranking functions in the section of analyzing user satisfaction.

4.1 Session and Task Statistics

We extracted two log datasets for the experiments. The first dataset D_0 consists of user browsing logs from a widely used browser plug-in (e.g., toolbar). It contains URL visits by anonymized users who opted in to provide data. The second dataset D_1 consists of web search logs from Bing. The information in these datasets contains: (1) user anonymized unique identifier (machine ID). (2) A unique browser identifier. (3) User clicked/visited URLs as well as queries related to user clicks. (4) A referrer URL where current URL comes from. (5) Time stamps of user events. For preserving user privacy, intra-net and secure URLs (such as URLs beginning with https:) are not recorded. Both datasets are from May to June 2011 in United States search market where main language is English. To further clean the data, we preprocessed datasets as follows: (1) filtering sessions which have no search event (such as checking emails) or too many search events (which are likely generated by robots); (2) filtering entries with non-English language settings, e.g., users searched in other languages; (3) keeping those sessions with search events in "Web" search vertical and filtering verticals like "Image", "Video", etc. (4) keeping sessions with search events from Google, Bing and Yahoo! since these are main search engines in U.S. market. The dataset D_0 contains $2,673,335$ unique users, and dataset D_1 contains $159,668,543$ unique users.

We tried different thresholds θ to extract session ranging from 1 minute, 2 minutes, 5 minutes, 10 minutes, 20 minutes, 30 minutes, 60 minutes to a day. The purpose of using different thresholds is to illustrate the phenomenon of multi-task and interleave-task behavior in user search behavior. We report the multi-task and interleave-task rate of extracted sessions in Table 5. From the table, we have the following observations: (1) The multi-task behavior always exists in the search process, even if we set the timeout of session as one minute. (2) The interleave-tasking behavior is not obviously while the threshold θ is less than 5 minutes.

Then, we mainly focused our studies on session with θ as 30 minutes. Based on this setting, dataset D_0

TABLE 5
Multi-task (MT) and interleave-task (IT) rate in sessions extracted with different timeout θ at different dataset D_0 and D_1.

	D_0		D_1	
	MT(%)	IT (%)	MT (%)	IT (%)
θ =1 minute	12.88	1.57	5.78	0.34
θ =2 minutes	19.12	3.07	9.58	0.69
θ =5 minutes	27.52	5.81	16.11	1.54
θ =10 minutes	33.66	8.29	21.31	2.51
θ =20 minutes	39.42	10.97	26.21	3.68
θ =30 minutes	42.65	12.65	28.85	4.41
θ =60 minutes	46.51	18.43	32.75	5.69
session=day	60.27	39.04	42.18	10.26

TABLE 6
Basic statistics of browse and 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 Multi-Task Sessions	42.65	28.85
% of Interleaved Task Sessions	12.65	4.41
% of Single-Query Tasks	48.75	71.86
% of Multi-Query Tasks	51.24	28.13

contains $30,071,190$ sessions and dataset D_1 contains $488,648,153$ sessions. Although the magnitude of D_0 is smaller than D_1, it contains more information than D_1, for example, (1) queries and clicks from different search engines; (2) post-query clicks, which are URL visits after search result clicks. We used these additional information in experiments of determining user satisfaction, deriving page utility, and predicting user search interests.

We used Algorithm 1 to group queries from obtained sessions (θ =30 minutes) into tasks since QC-SP is faster than QC-WCC and it outputs the same clustering results as QC-WCC. Note that the results of QC-BSP-3/5/10 are not reported in the following since the experimental results of QC-BSP-3/5/10 are almost the same to QC-SP because most of the lengths of sessions are less than 10 and the QC-BSP algorithm is proposed to saving the runtime of QC-SP. When we have to deal with long sessions, we argue that QC-BSP is one of the best choices. Afterwards, dataset D_0 contains $67,464,863$ tasks and dataset D_1 contains $770,759,594$ tasks. We report several basic statistics with tasks on D_0 and D_1. Table 6 shows the detailed results. From the table, we can see that the average numbers of queries in sessions and tasks of D_0 are bigger than those of D_1. The reason is that sessions in D_0 are usually longer than sessions in D_1 since users often browse before and after searching. Meanwhile, percentages of multi-task sessions in D_0 and D_1 are about 43% and 29%, respectively, which indicate that a large part of sessions consists of multiple tasks. Furthermore, about 13% and 5% of tasks are interleaved in D_0 and D_1, respectively, which indicate that users sometimes perform several tasks at the same time. Besides, about 43% tasks in D_0 and 29% tasks in D_1 contain multiple queries, which means that users often issue multiple queries for the same search information need.

Next, Figure 2 shows the distribution regarding the relationship between tasks and session length (in terms of number of queries). From the figure, we observe that: (1) As the length of sessions increases, the percentage of sessions decreases, which is in

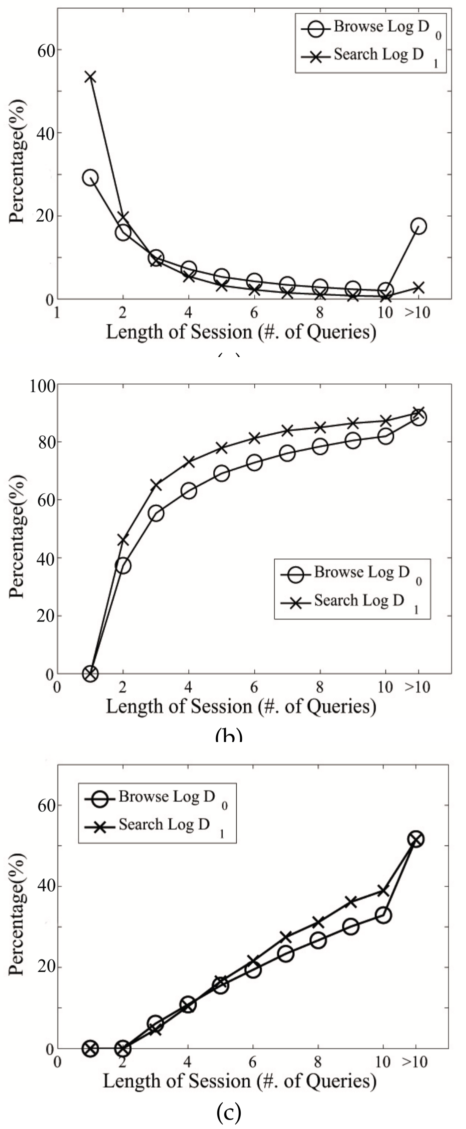


Fig. 2. Search and task distribution in browse and search Logs: (a) distribution of session length, (b) distribution of multi-task, (c) distribution of interleave-task. Note that the percentage of sessions is computed in total but the percentages of multi-task and interleaved-task sessions are computed at each length.

accordance with previous studies [4], [13], [38]. (2) As the number of queries in sessions increases, the percentage of multi-task and interleave-task sessions also increases. While the number of queries in sessions is greater than three, more than half sessions contain multiple tasks.

4.2 Determining User Satisfaction

To understand whether a user was satisfied or not after the search process, we adapted several implicit feedback signals as measures.

- *Clicks*. Previous work [25] showed that clicking on search results often indicates the relevance between queries and clicked pages.
- *Dwell Time*. Fox et al. [1] showed that dwell time on clicked results is a good indicator of user satisfaction. It is because users are more likely to stay on useful pages. White et al. [2] conducted their study on estimating page utilities by using 30 seconds as indicator of deriving page utility. We therefore took 30 seconds dwell time on a click result as a user satisfied signal.
- *Markov Model Success Score*. As shown in [29], user's search activities can be modeled as a sequential process using a Markov Model. The Markov model takes {queries, clicks, dwell time (>30 seconds)} as states {Q, SR, SR_Long}, respectively. We can build two Markov models to compute the likelihood of user satisfaction and dissatisfaction. Given a new user's search activities, we can compute the score of Markov Models to determine the label of user satisfaction. [1]. See [29] for more details.

Based on the above implicit feedback signals, we can compute ClickRate,30sClickRate,MMSuccessRate at session, task and query levels.

For a given user u, the ClickRate at query level is computed as:

$$ClickRate_q(u) = \frac{1}{Q}\sum_q I(q), \tag{1}$$

where Q is the total number of queries issued by user u, and I(q) is an indicator function which equals to 1 if q has clicks and 0 otherwise.

The ClickRate at task level is computed as:

$$ClickRate_t(u) = \frac{1}{T}\sum_t \frac{\sum_{q\in t} I(q)}{\sum_{q\in t} 1}, \tag{2}$$

where T is the total number of tasks of user u.

The ClickRate at session level is computed as:

$$ClickRate_s(u) = \frac{1}{S}\sum_s \frac{\sum_{q\in s} I(q)}{\sum_{q\in s} 1}, \tag{3}$$

where S is the total number of sessions of user u.

1. We leverage the same model as used in [29]

TABLE 7
User satisfaction rate using different signals at different levels in both D_0 and D_1.

	On D_0			On D_1		
Measure	Session	Task	Query	Session	Task	Query
CR	0.489	**0.494**	0.482	0.600	**0.614**	0.595
30s-CR	0.219	**0.224**	0.219	0.383	**0.411**	0.374
MM-SR	0.504	**0.508**	0.499	0.518	**0.541**	0.511

TABLE 8
Utilities of different sources based on sessions and tasks (N=182,533).

Source	Avg. Utility(%)
Sequence-Full	55.50
Sequence-Origin of Sessions	49.41
Sequence-Follow of Sessions	57.38
Sequence-Origin of Tasks	53.58
Sequence-Follow of Tasks	**58.47**

After getting the ClickRate of each user, we can compute the average ClickRate among all users. The average 30sClickRate and MMSuccessRate were computed in a same manner. Note that the computational steps are different from that in [9]. Since in the previous version we took a session as clicked if it contains at least one clicked query, which is not suitable since a session can contains many queries and the success of one query does not indicate the success of the whole session. To address the problem, we computed the average user satisfaction at different levels in this article.

In the experiments, we computed the average ClickRate, 30sClickRate, and MMSuccessRate at query, task, and session levels. The results on both browse and search logs are reported in Table 7. From the table, we can observe that: the task-level user satisfaction rates are higher than those at session and query levels (all p-values < 0.01, t-test). Before illustrating the reasons of the results, we show a simple example as follows. Suppose a user issued 4 queries $\{q_1, q_2, q_3, q_4\}$ while in the first query he went to a daily used website (e.g., Facebook) and come back to search something but gave up after three trials. Therefore, the first query $\{q_1\}$ belongs to a task T_A and the rest queries $\{q_2, q_3, q_4\}$ belong to another task T_B. At both session and query levels, the average success rate is 0.25 (one query successes and three fails). However, at task level, the average success rate is 0.5 (one task T_A successes and the other T_B fails). Based on previous studies [29], failed tasks tend to contain more failed queries since the user may try more times. If our task extraction method can group user's queries as atomic information needs, then the satisfaction score of a failed query is added to its own task. Then a failed query is likely belonging to a failed task which is longer than other tasks. Then we can infer that shorter tasks tend to have higher satisfaction scores than longer tasks. When we calculate the average user satisfaction rate at task level, the result can be higher than that of query or session level since several failed queries can be grouped by only one failed task. If the experiment results verify this conjecture, then our task extraction method is likely to be correct. (Note that although the average user satisfaction rate is higher at our extracted task level, it may not lead to the correct task extraction.) From another side, it also shows that task-level user satisfaction rates are more precise than session and query levels.

As we can observe from Table 7, task level user satisfaction rates are higher than session and query levels among all implicit measures. The results indicate that extracting tasks from sessions is non-trivial since we can more accurately capture users success or fail search experience in task level, which is to use logs for determining user satisfaction.

4.2.1 Utility Comparison

Next, we computed the web page utility of a query trail as the percentage of pages where users stayed longer than 30 seconds [1], [2]. Since tasks and sessions can be considered as a sequence of query trails, we compute average utilities on different parts of the sessions and tasks as: (1) Sequence-Full (all query trails of tasks or sessions), (2) Sequence-Origin (first query trails of tasks or sessions), and (3) Sequence-Follow (other query trails except the first one in tasks or sessions). The average utilities from different sources can be compared to know which source provides highest web page utilities for users in finding useful information.

To conduct this experiment, we used dataset D_0 since those post-search click events are recorded. We filtered D_0 by selecting sessions with at least two query trails where each query trail contains at least one click. After that, we obtained 182,533 sessions which are in high quality for utility calculation and adequate for significant test.

The results of utility comparison are presented in Table 8. On average there are more than half pages deriving utilities to users (Sequence-Full), which is in accordance with previous work [2]. If we only focus on the first queries, the average utilities will decrease. That is because users are more likely to fail at the first attempt in search. After issuing first queries, utilities of sessions and tasks both increase and the utility of tasks is even higher. Significant test shows that the source of tasks excluding first query (Sequence-Follow of tasks) outperforms all other sources in deriving utilities (all p-values $< 10^{-5}$, t-test). Besides, the Sequence-Origin of task is higher than session since it is more likely to fail at the beginning of a session than that of a task. The Sequence-Follow of task is higher than session because the user tends to be more successful to continue a task than to continue a session in which they may start with a new task. Therefore, following task trails, users can find more useful information.

4.2.2 Measuring Ranking Functions

In above we have evaluated the effectiveness of task in user satisfaction analysis, now we examine one interesting hypothesis: the tasks are more sensitive to the goodness of a ranking function. That is to say, if a ranking function A is better than another ranking function B, the user satisfaction at task-level is more significantly different than session and query levels. The impact of verifying this hypothesis is that we can use task-level query segmentation to implicitly measure a ranking function, which is one potential application of task segmentation.

To design this experiment, we used a ranking function from a commercial search engine as the good ranking function, called RankerA. We manually created another worse ranking function named RankerB by intentionally degrading RankerA. Here RankerB randomly picks up one of the top three search results at RankerA, and removes it from the ranking list. Based on the offline testing via NDCG on a labeled dataset, RankerB is significantly worse than RankerA. Then, nearly equal number of users are sampled (77,555 for RankerA and 77,997 for RankerB) to use the rankers in the commercial search engine for a while. We expect the users who used RankerA have better user satisfaction rate (measured by ClickRate, 30sClickRate, or MMSuccessRate) than the users who used RankerB. Such user satisfaction gap can be measured at query, task, and session levels. Our hypothesis is that the gap at task-level is more clear.

We used p-value (t-test) to measure the sensitiveness (gap) at different levels. It was computed as follows. First, we collected the average user satisfaction rates in both rankers at different levels. Then, for each level, we performed the independent two sample t-test and get the probability (p-value) that the average user satisfaction rates in RankerA and RankerB are the same. Note that the smaller the p-value is, the more significantly different two satisfaction rates are. We expect that the p-value of task-level is smaller than that of session and query levels.

Based on above settings, we conducted the experiments. Totally 232,235 sessions were collected, in which 139,080 (60%) sessions contain multiple tasks and 48,244 (20%) sessions contain interleaved tasks. We report the results of p-values (t-test) at different levels in Table 9. From Table 9, we can observe that: (1) Among the different implicit signals, MMSuccessRate has the smallest p-value. This suggests that MMSuccessRate is the more sensitive than other two measures. This result is accordance with previous study [29]. (2) By using ClickRate as the implicit measure, the ranking functions are not significantly (p-value > 0.05) different at query, task, and session levels. This indicates that use ClickRate

TABLE 9
Sensitivity results (p-value, t-test) of measuring ranking functions at different level.

Measure	Session	Task	Query
Click Rate	0.0796	0.0614	**0.0562**
30s Click Rate	0.0273	**0.0188**	0.0193
MM Success Rate	0.0030	**0.0018**	0.0021

as an implicit measure may not be good enough. (3) By using 30sClickRate and MMSuccessRate, the measures at task level are more sensitive than session and query levels. The results suggest that task is comparable to query and more sensitive than session in telling different ranking functions apart. Since the query and task level measures are comparable, we conclude that by using implicit signals to measure ranking functions, it is better to compute on both task and query levels.

4.3 Suggesting Related Queries

To compare the difference among tasks, sessions and click-through logs in suggesting related queries, we built several suggestion models. Since sessions and tasks both contain several queries, we can leverage the co-occurrence information for suggestions. Besides, queries sharing lots of co-clicked URLs can be used as suggestions. The details of suggestion models are described as follows.

- *Co-occurrence* [15]. One of its shortages is that it may suggest popular but useless queries. We build co-occurrence model of both sessions and tasks.
- *Log Likelihood Ratio (LLR)* [16]. Given two queries q_1 and q_2, LLR makes null hypothesis H_1 : $\Pr(q_2|q_1) = \Pr(q_2|\neg q_1)$ and the alternative hypothesis H_2 : $\Pr(q_2|q_1) \neq \Pr(q_2|\neg q_1)$. The LLR is calculated as $-2\ln\lambda$, where $\lambda = \max_p L(H_1)/\max_{p_1,p_2} L(H_2)$. A higher LLR indicates a higher correlation between query pairs, where LLR=3.84 indicates 95% confidence for rejecting H_1. To further increase the quality, we only mine query pairs having LLR larger than 100 as suggestions.
- *Random Walk* [17], [24]. The query to URL transition probability is computed as $P(u|q) = \frac{\#(q,u)}{\#(q)}$, and URL to query transition probability is computed as $P(q|u) = norm(\frac{\#(q,u)}{\#(u)} \cdot iqf(u))$. Here $\#(\cdot)$ is short for count and $norm(\cdot)$ is short for normalization. We introduce iqf [39] to decrease weights of popular URLs connected to many queries. The transition weights between query pairs $P(q_j|q_i) = \sum_u P(q_j|u) \cdot P(u|q_i)$. After constructing matrix P, random walk model [39] can propagate as:

$$r^{i+1}(q) = \alpha \cdot P \cdot r^i(q) + (1-\alpha) \cdot r^0(q) \quad (4)$$

Here $r^i(q)$ is a transition vector at i-th iteration for query q. α is set as 0.7 according to previous heuristic [40]. The propagation can continue until converge (L1-difference between $r^i(q)$ and $r^{i+1}(q)$ less than 10^{-6}) or stop after maximum steps (set as 1,000) of iterations. Finally we can provide suggestions for query q via $r^n(q)$.

TABLE 10
Coverage of all methods in terms of number of queries indexed by the method.

Methods	Distinct Query	Query Frequency
Session-based	2,115,815	650,550,492
Task-based	1,517,291	609,960,534
Random walk	6,104,997	720,678,502
Total	298,563,791	1,239,776,369

TABLE 11
Quality of query suggestions.

Source	High	Middle	Low	Total
Session Co-occur	54%	62%	67%	61%
Session LLR	71%	73%	72%	72%
Task Co-occur	84%	83%	72%	80%
Task LLR	**86%**	**85%**	73%	**82%**
RandWalk(0)	69%	71%	85%	75%
RandWalk(n)	74%	77%	**88%**	80%

As a result, we compared several query suggestion methods as follows: (1) session co-occurrence based; (2) session LLR based; (3) task co-occurrence based; (4) task LLR based; (5) co-click based (random walk at 0-th iteration); (6) random walk based (at n-th iterations).

We chose dataset D_1 from search logs for this experiment, since (1) D_1 is about ten times larger than D_0, and (2) D_1 can provide sufficient information for all query suggestion models. To make D_1 less noisy, we conducted some preprocessing as in [4], [38]. In session-based co-occurrence and LLR methods, we pruned query pairs with co-occurrence less than 5 times. In task-based co-occurrence and LLR methods, we pruned query pairs with task co-occurrence less than 5 times. In random walk approach, we pruned edges of query-URL pairs having less than 5 clicks. Note that co-occurrence and LLR methods are used on sessions and tasks with more than 2 queries, while random walk approach can leverage information from sessions having only one query with clicks.

Firstly, we compared the coverage of all methods in query suggestions. Instead of counting the percentage of testing cases each model is able to provide suggestions, we compared the number of indexed queries in session-based, task-based, and random walk approaches. The results are shown in Table 10. According to the distinct number of queries, session-based, task-based, and random walk methods preserve only 0.7%, 0.5% and 2% queries from D_1, respectively. However, according to query frequency, session-based, task-based, and random walk methods

TABLE 12
Suggestions of session-based models.

	Methods	
Test Query	Session Co-occur	Session LLR
amazon	facebook ebay google youtube yahoo	ebay walmart target best buy barnes and nobel
cell phone	facebook verizon wireless sprint verizon google	cheap cell phones phone all cell phone companies verizon cell phones sprint

keep 52%, 49% and 58% of whole dataset, respectively. These results indicate that after pruning, all methods can generate suggestions for high and middle frequent queries but cannot offer suggestions to most tail queries. Since random walk approach can make use of information from single-query sessions, it can generate suggestions on more testing queries than session-based and task-based methods.

Secondly, we compared the quality of query suggestions on a hold-out testing set. The testing queries were sampled from high, middle and low frequent parts of another search logs dataset D_2 in July 2011. High frequent queries have more than 100,000 submissions; middle frequent queries have 100 to 100,000 submissions; and rest queries are are of low frequency. We picked out 100 testing queries from each part and force every test query to have at least one suggestion by all models. To avoid providing duplicated suggestions, we conducted a post-processing step. Specifically, each suggestion is labeled as duplicated if it is too similar with another suggestion ranked higher than it. This simple method can remove duplicate suggestion "amazonkindle" when there is another suggestion "amazon kindle" ranked higher.

All models were allow to provide at most 5 suggestions. We invited 3 annotators to label whether these suggestions are meaningful or meaningless. The Fleiss kappa agreement [41] among the three judges is 0.68 which indicates that there is a substantial agreement. Each time, annotators were shown with query words and search results of a testing query and a suggested query. Those suggested queries which are either too similar or irrelevant to testing queries are judged as meaningless. The final judge was obtained by voting and the quality of each model was computed via the number of meaningful suggestions divided by the total number of suggestions. We show the results in Table 11. From the table, we can observe that: (1) Session co-occurrence based method performs worst. This is reasonable since session co-occurrence model generates many popular but meaningless suggestions. Using LLR can ease the problem and increase suggestion quality. To better understand the difference between co-occurrence and LLR methods, we show an example in Table 12. (2) Random walk approach performs best on low frequent queries, which indicates that random walk approach has a good property for easing the sparsity of query logs. (3) Task-based methods perform best on high and middle frequent queries, while task co-occurrence and LLR methods do not have much difference, since irrelevant queries are usually not grouped into same task. According to t-test, the performance improvements of task-based methods over the baselines are statistically significant on high and middle frequent queries (all p-value$<$0.05).

Finally, we studied several queries from high, middle, and low frequent parts with their suggestions. These examples are shown in Table 13. From the table, we can find that: (1) Session-based models often generate related queries in a broad range such as provide "verizon" as a suggestion to "att". (2) Random walk approach may generate suggestions which are too similar to test queries, such as providing "at & t" as a suggestion to "att". That is why random walk approach does not perform best on high and middle frequent queries (see Table 11). (3) For low frequent queries, task-based and session-based methods generate nearly same suggestions. This is because low frequent queries exist in less sessions and tasks and they do not have many alternative searches from sessions. (4) Task-based methods often generate more specific queries for further narrowing down user's information need, which are different from session-based and random walk approaches. As a result, suggestions provided by task-based methods can be treated as a complementary source to the results provided by session-based and random walk approaches.

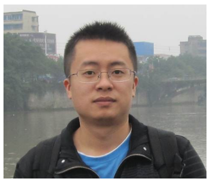
5 Conclusions

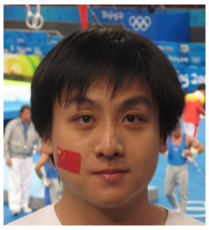
In this paper we proposed to use task trail as a useful segmentation of user search behaviors. Users often perform multiple tasks during their search processes. Statistical results on 0.5 billion sessions from web search logs showed that: (1) about 30% of sessions contain multiple tasks, and (2) about 5% of sessions contain interleaved tasks. To evaluate the effectiveness of task trails, we compared task, session and query trails in determining user satisfaction, predicting user search interests, and suggesting related queries. First, comparing to session and query trails, task trail is more precise to determine user satisfaction. Second, users are more likely to find useful information following the task trails. Third, we found that measuring ranking functions at task level is comparable to query level and more sensitive than session level. Forth, since tasks represent atomic user information needs, they can well preserve topic similarity between query pairs. Last but not least, we found that task-based query suggestion can provide complementary results

TABLE 13
Example of query suggestions by different methods. h, m, l denotes high, middle, and low frequency.

	Methods		
Test Query	Session LLR.	Task LLR	Random Walk(N)
amazon^h	ebay walmart target best buy barnes and nobel	amazon books amazon kindle amazon electronics amazon music amazon dvd movies	amazon music amazon books amazon book search amazon music cds amazon mp3
att^h	at&t my account verizon sprint tmobile att wireless	at&t my account att wireless at&t email at&t bill pay at&t customer service	at&t.com at & t att wireless att.net mail att uverse
exchange^m	military exchange exchange rate easyfreexbox360 tennis aafes	military exchange exchange rates navy exchange microsoft exchange base exchange	currency converter microsoft exchange aafes currency exchange exchange rates
harry truman^m	winston churchill robert byrd nelson mandela neil armstrong teddy roosevelt	harry truman quotes bess truman harry truman facts harry s truman	harry s truman truman truman state university truman college truman library
"popular irish baby names"^{l}	top irish baby names unique irish baby names irish baby boy names irish baby names "traditional irish baby names"	"unique irish baby names" irish baby names irish baby boy names top irish baby names top 100 baby names	"most popular irish boy names" popular irish names irish boys names irish baby names irish names

to other models. These findings verify the need to extract tasks from web search logs and suggest potential applications of using task trails in search and recommendation systems.


Zhen Liao received his BS degree in software engineering and PhD degree in computer science at Nankai University, Tianjin China, in 2008 and 2013, respectively. His research interests include data mining and information retrieval. Particularly, he has focused on the research problems about search log mining, context-aware search, and task-based search.


Yang Song received the BS degree in computer science from Zhejiang University, China, in 2003 and the PhD degree in computer science from The Pennsylvania State University in 2009. His research interests include machine learning, data mining, information retrieval, and search engines. Since 2009, he has been working for the Internet Services Research Center (ISRC) at Microsoft Research, Redmond.

Yalou Huang was born in 1964. He received the MS degree in computer science and the PhD degree in control engineering from Nankai University, Tianjin, China, in 1990 and 1993, respectively. He has been a professor and Ph.D. supervisor of Nankai University since 1996 and 1998 respectively. His primary research covers data mining, information retrieval and intelligent robotics.

Li-wei He is currently the Principal Development Manager in charge of the Bing online experimentation platform. His previous roles include: Architect in the Bing Ranking and Intent group working on spell correction algorithm; Lead of Search Quality Intelligence Group in the Internet Services Research Center (ISRC) of Microsoft Research, which built tools for gathering and diagnosing Bing search quality problems.

Qi He is a senior researcher at LinkedIn. After receiving the Ph.D. degree from Nanyang Technological University in 2008, he conducted 2-year postdoctoral research at Pennsylvania State University for CiteSeerX, followed by a research staff member position at IBM Almaden Research Center till May 2013. His research interests cover Information Retrieval/Extraction, Data/Text Mining, Social Network Analysis, Web Search and Mining.

REFERENCES

[1] Fox, S., Karnawat, K., Mydland, M., Dumais, S. and White, T., "Evaluating implicit measures to improve web search," *ACM Trans. Inf. Syst.*, vol. 23, pp. 147–168, 2005.

[2] White, R. and Huang, J., "Assessing the scenic route: measuring the value of search trails in web logs," ser. SIGIR '10. ACM, 2010, pp. 587–594.

[3] White, R., Bennett, P. and Dumais, S., "Predicting short-term interests using activity-based search context," ser. CIKM '10, 2010, pp. 1009–1018.

[4] Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E. and Li, H., "Context-aware query suggestion by mining click-through and session data," in *KDD '08*, 2008, pp. 875–883.

[5] Xiang, B., Jiang, D., Pei, J., Sun, X., Chen, E. and Li, H., "Context-aware ranking in web search," ser. SIGIR '10. ACM, 2010, pp. 451–458.

[6] White, R., Bilenko, M. and Cucerzan, S., "Studying the use of popular destinations to enhance web search interaction," ser. SIGIR '07, 2007, pp. 159–166.

[7] Silverstein, C., Henzinger, M.R., Marais, H. and Moricz, M., "Analysis of a very large web search engine query log," *SIGIR Forum*, vol. 33, pp. 6–12, 1999.

[8] Catledge, L.D. and Pitkow, J.E., "Characterizing browsing strategies in the world-wide web," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 1065–1073, 1995.

[9] Liao, Z., Song, Y., He, L.-w. and Huang, Y., "Evaluating the effectiveness of search task trails," ser. WWW '12, 2012, pp. 489–498.

[10] He, D., Göker, A. and Harper, D.J., "Combining evidence for automatic web session identification," *Inf. Process. Manage.*, vol. 38, no. 5, pp. 727–742, 2002.

[11] Jansen, B., Spink, A. and Kathuria, V., "How to define searching sessions on web search engines," ser. WebKDD '06, 2007, pp. 92–109.

[12] Jones, R. and Klinkner, K.L., "Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs," ser. CIKM '08, 2008, pp. 699–708.

[13] Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei, G., "Identifying task-based sessions in search engine query logs," ser. WSDM '11, 2011, pp. 277–286.

[14] Hassan, A., Jones, R., and Klinkner, K., "Beyond dcg: user behavior as a predictor of a successful search," ser. WSDM '10, 2010, pp. 221–230.

[15] Huang, C.K., Chien, L.F. and Oyang, Y.J., "Relevant term suggestion in interactive web search based on contextual information in query session logs," *Journal of the American Society for Information Science and Technology*, 2003.

[16] Jones, R., Rey, B., Madani, O. and Greiner, W., "Generating query substitutions," ser. WWW '06. ACM, 2006, pp. 387–396.

[17] Mei, Q. and Zhou, D. and Church, K., "Query suggestion using hitting time," ser. CIKM '08. ACM, 2008, pp. 469–478.

[18] Liu, Y., Gao, B., Liu, T.-Y., Zhang, Y., Ma, Z., He, S. and Li, H., "Browserank: letting web users vote for page importance," ser. SIGIR '08, 2008, pp. 451–458.

[19] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A. and Vigna, S., "The query-flow graph: model and applications," ser. CIKM '08, 2008, pp. 609–618.

[20] Donato, D., Bonchi, F., Chi, T. and Maarek, Y., "Do you want to take notes? identifying research missions in yahoo! search pad," ser. WWW '10, 2010, pp. 321–330.

[21] Kotov, A., Bennett, P., White, R., Dumais, S. and Teevan, J., "Modeling and analysis of cross-session search tasks," ser. SIGIR '11, 2011, pp. 5–14.

[22] White, R., Bailey, P. and Chen, L., "Predicting user interests from contextual information," ser. SIGIR '09, 2009, pp. 363–370.

[23] Olston, C. and Chi, E.H., "Scenttrails: Integrating browsing and searching on the web," *ACM Trans. Comput.-Hum. Interact.*, vol. 10, pp. 177–197, September 2003.

[24] Craswell, N. and Szummer, M., "Random walks on the click graph," ser. SIGIR '07, 2007, pp. 239–246.

[25] Gao, J., Yuan, W., Li, X., Deng, K. and Nie, J.-Y., "Smoothing clickthrough data for web search ranking," ser. SIGIR '09. ACM, 2009, pp. 355–362.

[26] Radlinski, F. and Craswell, N., "Comparing the sensitivity of information retrieval metrics," ser. SIGIR '10, 2010, pp. 667–674.

[27] Shen, X., Tan, B. and Zhai, ChengXiang, "Context-sensitive information retrieval using implicit feedback," ser. SIGIR '05, 2005, pp. 43–50.

[28] Chapelle O., Joachims T., Radlinski F. and Yisong Yue, "Large-scale validation and analysis of interleaved search evaluation," *ACM Trans. Inf. Syst.*, vol. 30, no. 1, p. 6, 2012.

[29] Hassan, A., Song, Y. and He, L.-w., "A task level user satisfaction metric and its application on improving relevance estimation," ser. CIKM '11, 2011.

[30] Beeferman, D. and Berger, A., "Agglomerative clustering of a search engine query log," ser. KDD '00, New York, NY, USA, 2000, pp. 407–416.

[31] Song, Y., Zhou, D., and He, L.-w., "Query suggestion by constructing term-transition graphs," ser. WSDM '12, 2012, pp. 353–362.

[32] Jain, A., Ozertem, U. and Velipasaoglu, E., "Synthesizing high utility suggestions for rare web search queries," ser. SIGIR '11, 2011, pp. 805–814.

[33] Vapnik V., *Statistical learning theory*. Wiley, 1998.

[34] Wang, H., Song, Y., Chang, M.-W., He, X., White, R. and Chu W., "Learning to extract cross-session search tasks," in *WWW*, 2013, pp. 1353–1364.

[35] Teevan, J., Adar, E.. Jones, R. and Potts M., "Information re-retrieval: repeat queries in yahoo's logs," ser. SIGIR '07, New York, NY, USA, 2007, pp. 151–158.

[36] Shen, D., Pan, R., Sun, J.-T, Pan, J., Wu, K., Yin, J. and Yang, Q., "Q2c@ust: our winning solution to query classification in kddcup 2005," *SIGKDD Explor. Newsl.*, vol. 7, pp. 100–110, 2005.

[37] Swain, M. and Ballard, D., "Color indexing," *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[38] Liao, Z., Jiang, D., Chen, E., Pei, J., Cao, H. and Li, H., "Mining concept sequences from large-scale search logs for context-aware query suggestion," *ACM Trans. Intell. Syst. Technol.*, vol. 3, pp. 17:1–17:40, 2011.

[39] Deng, H. and Irwin, K. and Michael L., "Entropy-biased models for query representation on the click graph," ser. SIGIR '09, 2009, pp. 339–346.

[40] Haveliwala T., Kamvar, S., and Jeh G., "An analytical comparison of approaches to personalizing pagerank," in *Technical Report in Stanford*, 2003.

[41] Fleiss, J. L., "Measuring nominal scale agreement among many raters," *psychological bulletin*, vol. 76, 1971.