Supporting Complex Search Tasks

Ahmed Hassan Awadallah, Ryen W. White, Patrick Pantel, Susan T. Dumais, Yi-Min Wang
Microsoft Research
Redmond, WA 98052, USA
{hassanam, ryenw, ppantel, sdumais, ymwang}@microsoft.com

ABSTRACT
We present methods to automatically identify and recommend sub-tasks to help people explore and accomplish complex search tasks. Although Web searchers often exhibit directed search behaviors such as navigating to a particular Website or locating a particular item of information, many search scenarios involve more complex tasks such as learning about a new topic or planning a vacation. These tasks often involve multiple search queries and can span multiple sessions. Current search systems do not provide adequate support for tackling these tasks. Instead, they place most of the burden on the searcher for discovering which aspects of the task they should explore. Particularly challenging is the case when a searcher lacks the task knowledge necessary to decide which step to tackle next. In this paper, we propose methods to automatically mine search logs for tasks and build an association graph connecting multiple tasks together. We then leverage the task graph to assist new searchers in exploring new search topics or tackling multi-step search tasks. We demonstrate through experiments with human participants that we can discover related and interesting tasks to assist with complex search scenarios.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*selection process*, *search process*.

Keywords
Exploratory search; Complex search tasks; Task recommendation.

1. INTRODUCTION
Search engines are the primary means by which people locate information online and complete search tasks. Queries issued to search engines have been categorized as navigational, informational or transactional [11]. This particular categorization has been useful for characterizing high-level information-search behavior and guiding the development of appropriate support (e.g., identifying definitive results for navigational queries [1]). However, as the range of tasks that are possible online grows, more complex search activities, such as exploratory search [32], and multi-step search tasks [23] have been identified. In exploratory search, people seek to learn about a topic of interest or discover new information. In multi-step search tasks, searchers attempt to fulfill a complex information need involving multiple aspects. Despite some trials [18], search engines today do not adequately support either of these scenarios. For searchers who are either unfamiliar with their problem domain, unfamiliar with the process to achieve their goal, or who lack a well-defined goal, there is a pressing need for assistance in

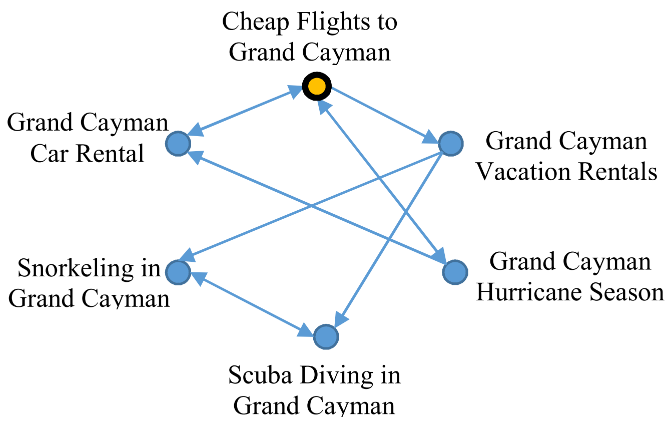
Cheap Flights to Grand Cayman
Grand Cayman Car Rental
Grand Cayman Vacation Rentals
Snorkeling in Grand Cayman
Grand Cayman Hurricane Season
Scuba Diving in Grand Cayman

Figure 1. Sub-graph generated from search-log data showing connections between the tasks recommended for the query "Cheap Flights to Grand Cayman" (highlighted),

searching. When attempting such tasks, searchers require support that extends beyond a list of search results. They need *task completion* systems that provide assistance by, among other things, outlining the necessary steps to explore or accomplish a complex task.

Some previous attempts have been made to support people engaged in complex tasks by allowing them to take notes and record results that they already examined [18], or to provide task continuation assistance, whereby the search engine can predict that a searcher is likely to resume a task and hence preemptively save and retrieve the current search state on the searcher's behalf [34]. While these are good ways to support long term tasks, they do not help searchers directly explore or identify potential next steps for their tasks. Other research efforts have focused on building tours or trails to guide the searcher through their search process [20][35]. While useful, the methods proposed to date have involved restricted domains or hypertext corpora rather than Web search [42], or have retrieved focused trails of URLs rather than lists of search results [39]. Other attempts have been made to augment browsing with serendipity, but have been limited to social media [35] or to named entities [8].

In this paper, we present and evaluate methods to automatically identify and recommend tasks that allow searchers to explore and accomplish complex (multi-aspect or multi-step) search tasks. We identify these suggestions by mining query logs from a popular commercial search engine to first identify complex tasks, and then automatically generate a graph connecting sub-tasks that are likely to have common interest from searchers. In doing so we learn from the aggregated activity of many searchers and apply this collective knowledge to assist others with similar search objectives.

Our focus is to identify that a searcher is engaged in a complex search task and to help them explore different sub-tasks related to their complex search task. To better understand this, consider the example query `[cheap flights to grand cayman]` shown in Figure 1. When the searcher enters this query, the method detects that they

CIKM'14, November 3 – 7, 2014, Shanghai, China.
Copyright 2014 ACM 1-58113-000-0/00/0010 …$15.00.
http://dx.doi.org/10.1145/2661829.2661912

are engaged in a complex search task and that they may be interested in exploring more or fulfilling more steps. The figure shows a graph where every node is a search task. Search tasks are connected if there is a high likelihood that both tasks can be steps in the same complex search task. The suggested tasks in Figure 1 appear reasonable to help searchers identify other steps for a complex task involving a trip to Grand Cayman. Note that this is a representative set of tasks and there are many similar high-quality recommendations in the task graph that we generate. Since many searchers looking to fly to Grand Cayman have also been engaged in other activities to plan their trips, the proposed method automatically identifies tasks about accommodation ("vacation rentals"), car rental ("car rental"), trip planning ("hurricane seasons"), and activities of possible interest ("snorkeling" and "scuba diving").

We make the following contributions with this research:

- Develop methods to automatically identify queries in large-scale search logs that form part of complex search tasks.
- Learn models to identify query intent and different aspects of the search tasks represented by each set of queries.
- Construct a graph connecting tasks that are likely to be of interest to searchers. This graph could be useful for a range of applications beyond task recommendation (e.g., advertising).
- Apply the graph for the recommendation of a set of interesting and diverse tasks to support searchers during exploration.
- Devise and apply metrics to measure aspects of task recommendations, including novelty, diversity, and interestingness.

The remainder of the paper is structured as follows. In Section 2, we describe related work in exploratory search, search trails, serendipity, and query suggestions. Section 3 describes how we identify exploratory search intent from search activity. We describe the task extraction and graph construction in Section 4. Section 5 presents our recommendation approach. Our experiments and findings are summarized in Section 6 and we conclude in Section 7.

2. RELATED WORK

There are several areas of related work: (i) exploratory search, (ii) serendipity in Web search, (iii) creating trails and guided tours through information spaces, and (iv) query suggestions and related searches. We now describe previous work in each area in more detail and discuss how our method and study extend this prior work.

Exploratory Search: Many Web queries are directed searches where the searcher seeks to navigate to a particular Web resource or to locate a specific information item. Another class of search activity is *exploratory search* [32][45]. Searchers engaged in exploratory search activities purposely try to learn about a topic and discover new information. This may be associated with information goals such as seeking different opinions on a topic, exploring or discovering aspects of a topic, or obtaining an overview of a topic.

Research on exploratory search has focused on characterizing the exploratory search process and the different types of support that are required to help people perform exploratory searches [32][45]. Examples of this support include developing new search interfaces to helping media studies researchers refine their research questions and explore diverse topics [12], and creating interfaces to support complex search tasks [40]. Other research has focused on the prolonged nature of exploratory search tasks that can cause these tasks to span multiple sessions, and proposed solutions to preserve and restore the search state across these sessions [34]. Others examined personalizing the search experience for multi-session search tasks, showing the impact of task stage and task type [30].

To support exploration, previous work determines if a searcher is engaged in a complex search task and supports them within the current session using tours or trails [23]. Other research has focused on identifying sessions where searchers are exploring and studying the impact of exploration on predicting search success [24]. Recent research has also examined the intrinsically-diverse nature of some information-seeking tasks. Such tasks typically require multiple queries on different aspects of the same information need [38]. The authors proposed an approach that could alter the result rankings and also provide them information on aspects of the task which they are likely to search for in the future. This relates to the aspectual retrieval task of the Text Retrieval Conference (TREC) Interactive Track [41], whereby searchers were expected to identify the different aspects or instances of a given topic.

Serendipity in Web Search: Serendipity is the act of encountering information unexpectedly. It has long been identified as valuable, both as a pleasure in itself and as part of task-focused problem solving [2]. Several research efforts have sought to characterize, understand, and support serendipity in Web search. For example, André et al. [2] studied whether search-result personalization could reduce the potential for serendipitous information discoveries (e.g., by creating a filter bubble in which searchers are only shown limited information). They found that personalization does not harm serendipity and may in fact be useful for supporting serendipity.

Serendipity has also been considered in the context of collaborative filtering where interesting content is identified by matching individuals with similar interests. Previous work on collaborative filtering has considered promoting novelty and serendipity by helping searchers to uncover less popular and more diverse items [46].

Another line of related research is that associated with entity search and recommendation. Lin et al. [29] proposed "active objects" where entities are paired with actions and given a query about some entity, all possible actions are recommended to searchers (e.g., actions on a book may be purchasing it, reading reviews, etc.). Previous work has also considered recommending queries based on an individual's browsing behavior. In [9], entities are extracted from the page that a searcher is visiting and similar entities and queries are suggested. Finally, Bordino et al. [8] proposed a method to support serendipity in entity recommendation. They constructed an entity graph based on Wikipedia and Yahoo! Answers and devised an algorithm to recommend new entities given a particular entity of interest to the searcher. They also examined the emotions attached to different entities and its impact on searcher interests.

Tours and Trails: Another related research direction concerns the construction of tours and trails to guide searchers' resource selection decisions during the search process [5][44]. Chalmers et al. [15] suggested that human recommenders build and share their Web navigation paths to support future searchers. Wexelblat and Maes [42] introduced annotations in Web browsers called *Footprints*, assembled by the Website's designer, that reveal trails that searchers take through a Website. They found that searchers required fewer steps to find information using the *Footprints* system.

Trails have also been proposed as a way to guide users through the specific steps required to accomplish search tasks. Singla et al. [39] proposed *trailfinding* methods to support Web search by identifying query-relevant trails from logs that could be shown to complement or replace traditional search result lists. An alternative to presenting the full trail or guided tour are step-at-a-time recommendation. *ScentTrails* [36] combined browsing and searching into one interface by highlighting potentially valuable hyperlinks. *WebWatcher* [26] accompanied people as they explored the Web. The system highlighted hyperlinks, and learned from implicit feedback collected during earlier tours that it believed was of interest.

Query Suggestions: There has been a significant amount of research on the problem of finding and recommending query suggestions [3][16][28][33]. Most query suggestions techniques use similarity measures between queries using query terms, clicked documents, or sequences of queries in sessions. Although this work is related to ours, unlike query suggestion, *our goal is not to help the user refine their current query*. Rather, our objective is to help them identify and explore aspects related to their current complex task.

Baeza-Yates et al. [3] built models of query expansion based on a vector representation of queries. The query-click graph was used by Craswell and Szummer [16] to find related documents and queries via random walks. Mei et al. [33] also used a bipartite graph connecting queries and clicks to find query suggestions via hitting time. Jones et al. [28] presented a method to generate query suggestions by substituting the whole query or its sub phrases by new phrases collected from other searchers' querying behavior. Other approaches address the challenge of generating query suggestions by modeling query flow in user search sessions. Boldi et al. [6] presented the concept of the query-flow graph which represents chains of related queries in search logs. They use this model for finding logical session boundaries and generating query recommendations.

Prior work also studied the problem of predicting the next search action based on the current actions, either by predicting the next result click [13] or by predicting searchers' short-term interests at a more general level of abstraction (e.g., topical categories [43]).

Contributions of Our Study: We extend previous research in a number of ways. First, we devise and evaluate methods to automatically identify queries, intents and different aspects of complex search tasks. Second, in contrast to prior work on query suggestions, we do not try to help searchers refine their current query, rather we focus on recommending future queries that they should consider beyond current query refinements. Third, our work is not limited to entities, rather we cover a large span of aspects and tasks. Finally, we focus on supporting exploration in Web search rather than post-query navigation, or navigation though particular Websites or restricted collections such as Wikipedia, all considered in prior research.

3. COMPLEX SEARCH TASKS

3.1 Data

Our data consists of a sample of hundreds of thousands of search sessions from the usage logs of the Microsoft Bing Web search engine during all of July 2013. Every log entry contained an anonymized user identifier, a timestamp, a query, and all search result clicks and their associated dwell times. Automated bot traffic, intranet, and secure URLs (https) were removed at the source prior to analysis. Only queries tagged as English and from the United States locale were retained to remove geographic or linguistic variations.

The log entries are segmented by time into sessions. In this study, *session* refers to a sequence of search activities terminated by a prolonged period of inactivity. We used 30 minutes of idle time to demarcate sessions, as has often been performed in related studies, e.g., [43] . Since a session can contain multiple related and unrelated searches, we further segment sessions into tasks and complex tasks [27]. We adopt the following definitions from prior work:

Definition: A *topically-coherent session* is a set of related information needs that belong to the same session.

Definition: *A search task* is an atomic information need resulting in one or more queries.

Definition: A *complex search task* is a multi-aspect or a multi-step information need consisting of a set of related tasks.

Note that the terms "tasks" and "goals" and the terms "topically-coherent session" and "missions" have been used to describe similar concepts in the literature [24][27]. Note also that since we only use topically-coherent sessions in our analyses, we use the terms "sessions" and "topically-coherent sessions" interchangeably to denote a related set of information needs within a fixed timeframe.

3.2 Sessions with Complex Tasks

We adopt the definition of complex tasks used in previous work [32][45]. Since our objective is to support searchers tackling complex tasks, we want to focus on search sessions where searchers are:

1. Engaged in learning and discovery (e.g., learning all aspects of a particular topic, comparing products, etc.), or
2. Browsing information on a topic or a person of interest (e.g., a celebrity, a sports team, etc.), or
3. Tackling a multi-step search task (e.g., planning a trip).

Note that we use the terms exploratory tasks and complex tasks interchangeably to denote one of the task types explained above. Since not all tasks meet these criteria, we need to appropriately handle the cases that do not. The two most frequent cases are navigational searches (searcher is trying to reach a known site) or struggling searches (searcher is experiencing difficulty in finding the required information) [24].

Navigational Searches: We removed the 500 most frequent queries that also have low click entropy (≤ 0.15), representing the variance in result clicks for these queries [19]. These are typically navigational queries (e.g., facebook, nytimes) where the searcher is seeking to navigate to a particular website. Given their nature, these queries are unlikely to be part of exploratory tasks.

Struggling Sessions: Struggling sessions are cases where searchers are experiencing difficulty in locating required information and hence issue multiple related queries. Previous work [24] has shown that search sessions with multiple queries (three or more queries) can comprise exploration or struggle; the latter describing a situation where the searcher is having trouble locating required information. To identify struggling sessions, we adopt the following list of session features shown to be useful for this task [24]:

- **Query features:** Number of queries; time between queries; average query length.
- **Query-transition features:** Average cosine similarity between queries, number of added, deleted, and substituted terms.
- **Click features:** Number of clicks per query; average dwell time; percentage of unique URL and domain clicks.
- **Topical features:** Open Directory Project (ODP, dmoz.org) categories of all visited URLs; count and entropy of all topics.

We constructed a classifier to distinguish between struggling sessions and other sessions using the features listed above and a Multiple Additive Regression Trees (MART) classifier [20]. The classifier was trained and tested using 10-fold cross validation. For this purpose, we use the labeled data described in [24]. The data contained 3000 labeled sessions and over 13,000 queries. Every session in these data was labeled as either a struggling session or not by a large number of crowd-sourced judges. The classifier could identify struggling sessions with an accuracy of 78.5 and an AUC of 83.4. We applied this classifier to our data to remove all struggling sessions in addition to the navigational queries that we have removed earlier. We used the remaining sessions for our study.

4. TASK GRAPH EXTRACTION

In this section, we describe how we use the exploring sessions identified per the process in the previous section to identify and extract different tasks. We then show how we can build a graph to connect these tasks. This graph allows us to provide task recommendations to searchers as will be described in Section 5. In doing so, we can use the aggregated behavior of many searchers to support others who are also engaged in related search exploration.

4.1 Identifying Tasks from Queries

Using all unique queries from the dataset described in the previous section, our objective is to identify tasks. As described earlier, we adopt the definition of the task as an atomic information need. As such, a task can be a step in multi-step task (e.g., locating an eatery while planning a night out) or an aspect in a multi-aspect task (e.g., finding images of a celebrity while learning about their latest news). A task is represented by a group of queries with the same intent. For example, recalling Figure 1, finding a flight to *Grand Cayman* is a task that forms part of the broader complex task of planning a trip to Grand Cayman. Notice that a task can be represented by multiple distinct query statements (e.g., `[flights to grand cayman]`, `[flying to grand cayman]`, `[grand cayman flights]`, etc.). As part of grouping queries, we pre-processed each query by lowercasing the text, stripping punctuation, replacing all runs of whitespace with a single space, and trimming any leading or trailing spaces.

We now describe the different lexical constituents that we seek to identify in every query and how they are used to find tasks.

4.1.1 Entities

We begin by tagging text spans in our queries that refer to an entity. For our purposes, we consider a text span to be an entity if it is represented in a knowledge base such as Freebase or Wikipedia. In our experiments, we pool all of the entities present in Wikipedia and Freebase, including people, places, companies, as well as events, concepts, and famous dates.

We construct a lexicon by extracting each English lexical name associated with an entity in our knowledge base and represent it using a perfect hash data structure [17]. For each query, we lookup each possible n-gram in the perfect hash. Nested matches are resolved by greedy admission using a left longest match heuristic.

Many of the knowledge sources used by our approach, such as Freebase, represent ontological items such as */time/event* and */business/employment_tenure*, as well as reified relations such as */film/performance* and */education/education*. In order to filter these out, following [37], we identified all the lexical names in our query data and manually annotated the 300 most frequently matched types according to whether they represented non-entity types (as above) or entity types, e.g., */music/record_label*, */aviation/airport*, and */military/conflict* (accounting for over 90% of the query traffic matching a lexical name). We then exclude any entity that is typed with a non-entity and none of the entity-annotated types.

It is well known that lexical names are highly ambiguous. Since our goal is to tag queries with the presence of an entity, i.e., not to resolve the entity to a particular entry in the knowledge base, we are only concerned with names that are ambiguous with a non-entity sense. For example, the name "something" is problematic since it can refer to the Beatles song "Something" as well as the very common non-entity pronoun. We filtered out such highly ambiguous names from our lexicon by building a binary ambiguity classifier trained on 500 manually annotated names. A name is ambiguous if it holds a non-entity sense, such as the name "something". For our learning algorithm, we use boosted decision trees [20]. We tune our hyper-parameters (i.e., number of iterations, learning rate, minimum instances in leaf nodes, and the maximum number of leaves) using ten-fold cross-validation. The resulting binary classifier is then applied as a filter to all names in the lexicon. This operation generated a lexicon comprising around 11 million names that we use in building the task graph described later in this section.

4.1.2 Collocations

Definition: A *collocation*, also known as a *multi-term keyword,* is a sequence of words or terms that co-occur more often than would be expected by chance [31].

Consider the query `[cheap hotels in new york city]` as an example. A bag of word representation would treat the query as a set of six words: {"cheap", "hotels", etc.} in no particular order. If we try to understand the intent behind the query, we will determine that the user is searching for "cheap hotels" in "new york city" and that breaking these multi-term keywords into their constituent terms results in a loss of semantic meaning.

Identifying term collocations is strongly related to the problem of query segmentation. Query segmentation involves dividing search query into individual phrases or semantic units [4]. Many solutions have been proposed to tackle the query-segmentation problem. Some employ a supervised learning framework [4], while others adopt an unsupervised framework [21]. We use unsupervised techniques since they are simpler to implement, only require access to raw Web n-gram frequencies, and can achieve comparable performance to state-of-the-art supervised methods [10].

To segment queries into collocations (keywords), we adopt a widely-used approach [21]. A segmentation is obtained by computing the point-wise mutual information score for each pair of consecutive terms. More formally, for a query $q = \{q_1, q_2, \dots, q_n\}$:

$$association(q_i, q_{i+1}) = log \frac{p(q_i, q_{i+1})}{p(q_i)p(q_{i+1})}$$

where $p(q_i, q_{i+1})$ is the joint probability of occurrence of the bigram (q_i, q_{i+1}) and $p(q_i)$ is the occurrence probability q_i.

A collocation break is introduced whenever the association between two words falls below a certain threshold T. Following [22], we used $\mathrm{T} = 1.91$. Note that this is larger than thresholds used in the literature (e.g., 0.895 [28]) since the objective of this work is to find collocations, which tend to require a higher degree of association, rather than phrases as is usually the case in previous work.

4.1.3 Terms and Prepositions

In addition to entities and collocations, we use two additional tags to describe the remaining tokens in a query: "Prep" and "Term". *Prep* describes prepositions such as (for, in, of, etc.). Any term not labeled as an entity, a collocation, or a preposition is tagged as *term.*

4.1.4 Patterns

Our objective is to identify queries that can be parsed into a task. We focus on queries that can be characterized using one of the following three patterns:

- Refiner Prep Pivot (e.g., cheap_hotels in new_york)
- Pivot Refiner (e.g., apple_iphone reviews)
- Pivot (e.g., tom cruise)

Definition: *A pivot* is the central point of the query and can be any concept that is well defined and has been labeled as an entity or a collocation (e.g., "new york city").

Definition: *A refiner* is a query constituent intended to characterize a precise distinction or subtlety in a query (e.g., "hotels").

Table 1. Examples of task queries and corresponding lexical tags. Multi-word entities and collocations are separated by an underscore.

Lexical Tags	Example Query
Term Prep Entity	reviews of iphone
Term Prep Entity	attire for fall_wedding
Term Prep Entity Entity	reviews of apple iphone
Collocation Prep Entity Entity	user_reviews for iphone
Collocation Prep Collocation	center_pieces for fall_wedding
Collocation Prep Entity Entity	user_reviews for apple iphone
Entity Term	iphone reviews
Collocation Term	fall_wedding dresses
Entity Object Term	apple ipad prices
Entity Collocation	tom_cruise latest_movies
Collocation Collocation	fathers_day gift cards
Entity Entity Collocation	apple iphone protection_plan

Restricting the queries that we use to those that follow these patterns allows us to identify the pivot and refiner in every query using a simple set of dependency parsing rules. For phrases of the form "NNX NNX", where NNX is a singular, plural, or proper noun, the pivot is the first noun. For phrases of the form "NNX IN NNX", where IN is a preposition, the second noun is the pivot. These simple rules lead to higher precision analyses when compared to full dependency parsers, which tend to perform poorly on queries since they are typically short and often not grammatically well-formed.

To find pivots and refiners in queries using the entity, collocation, term and prep tags, we start by resolving nested entity and collocation matches. Nested matches are resolved by greedy admission using a left longest match heuristic. For example, in the search query `[reviews for apple iphone]`, the terms "apple" and "iphone" are identified as entities, and "apple iphone" is identified as a collocation. In this case the match is resolved by treating "apple iphone" as a single concept. We retain the information about the subsumed entities with the concept and we treat the concept as an entity.

Figure 2 shows how a pivot and a refiner can be identified using the tags (entity, collocation, etc.) assigned to each token or multi-token keyword. As shown in Figure 2, a pivot can be either an entity or a collocation. Allowing collocations to serve as pivots significantly increases the coverage of our method and allows us to cover concepts that are not typically labeled as entities (e.g., "fall wedding", "resume writing", etc.) as well as consecutive entities that are typically treated as a single entity (e.g., "apple iphone"). Conversely, a refiner could be a term or a collocation that it is intended to define a specific aspect of some entity (e.g., "cheap hotels"). Examples of queries and their corresponding lexical tags appear in Table 1.

Since queries often lack syntactic structure, we also allow some patterns such as "Refiner Pivot", e.g., "pictures tom_cruise", only when the pivot is an entity and it is the only entity in the query. We also allow this pattern when the refiner is a question word, e.g., "what is adaptive radiation". Restricting queries to the three patterns described above allows us to understand query intent and group multiple queries with the same intent as we show later. In total, 72.9% of the queries in the exploration sessions identified in Section 3 followed these patterns. Therefore, these patterns allow

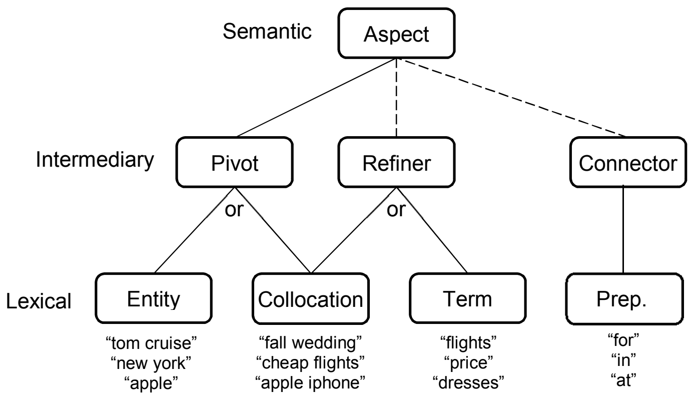

Figure 2. The basic lexical elements are entities, collocations, prepositions and terms. A pivot can be an entity or a collocation. A refiner can be a collocation or a term. An aspect is either a pivot, or a pivot and refiner optionally connected by a connector. Dashed lines indicate that a constituent is optional.

us to capture a significant fraction of online exploratory searching. We now show how these queries are grouped to find tasks.

4.2 Grouping Queries into Tasks

Using the methodology described above, we extract 10.27 million queries from the search log dataset described in Section 3. Recall that a task is either a step in a multi-step complex task or an aspect in a multi-aspect task as defined earlier. Since the same task can be represented by multiple queries, we need to group all queries pertaining to the same task together. One way to address this problem is to define a query similarity function and apply it to all pairs of queries and then cluster the resulting graph. However, this approach is very expensive since the number of pairs is quadratic in the millions of queries in our dataset.

Alternatively, we can use the metadata we have about the queries (the refiner and pivot tags, the entity identifier, etc.) to significantly reduce the computational cost of the grouping process. To group queries with similar intent, we perform the following three steps:

1. Every entity in the knowledge base is assigned a unique identifier. We assign entities present in queries identifiers by looking them up in the knowledge base and applying the identifier used for the entity therein. This allows us to match different surface forms of the same entity (e.g., "new york city" and "nyc").Surface forms with more than one identifier are ignored for this step. We then replace all entities with its entity identifier.
2. We normalize the syntactic structure of all queries by transforming all patterns of the form "Refiner Prep Pivot" to "Pivot Refiner" (e.g., "hotels in new york city" → "new york city hotels").
3. For queries with the same pivot, we match two refiners if they: (1) have the same lemma (lemmatization is the process of reducing an inflected spelling to its lexical root or lemma form); or (2) have a normalized edit distance of less than 0.2. This allows us to capture spelling mistakes and spelling variations.

Applying these steps allows us to group queries such as: `[hotels in new york city]`, `[hotels in nyc]`, `[nyc hotel]`, `[nyc hotls]`, etc. into a single query group representing a single aspect.

Every group of queries represents one aspect and is represented by the most frequent query in the group. This representation is used as the recommended surface form as we will explain in Section 5. We apply this method to the set of 10.27 million queries and identify 7.69 million query groups (tasks) that we use to construct a graph.

4.3 Task Graph

Recall that our main objective is to help searchers tackle complex tasks by recommending related and interesting tasks with respect to their current query. A good list of recommendations will contain different tasks that are related to the current one. A natural way to do that would be to construct a graph where related tasks are connected. This graph can be used to suggest future queries given the current one. Since individuals may lack the necessary knowledge to explore all related tasks, we aggregate the search behavior of all searchers and encode inter-task relationships in a graph structure.

We construct a graph $G = (T, E, w)$ where T is the set of all tasks (query groups) described in the previous section. $E = T \times T$ is the set of possible associated tasks. $w : E \rightarrow [0..1]$ is a function that assigns to every pair of tasks (i, j) a weight $w(i, j)$ representing the strength of their association.

To measure the association between pairs of tasks we use the Normalized Pointwise Mutual Information [10]. The Pointwise Mutual Information of any two discrete events x and y quantifies their degree of association by the discrepancy between the probability of their coincidence given their joint distribution and the probability of their coincidence given only their individual distributions, assuming independence. The PMI value is 0 if the two variables are independent. Positive values of PMI indicate positive association while negative values indicate negative association. Since PMI can take arbitrary positive or negative values, we normalize it into NPMI, which has a value between $[-1,+1]$, as follows:

$$npmi(x, y) = -\,log\frac{p(x, y)}{p(x)p(y)}/log\, p(x, y)$$

To compute the association value we need to determine when two tasks have co-occurred. We define co-occurrence between two tasks when the same searcher issues both queries within a 48-hour period. To reduce noise in this calculation, we discard all pairs that co-occurred less than 10 times unless they share the same pivot.

5. TASK RECOMMENDATION

Using the graph connecting the different tasks, our objective is recommend other tasks related to the current task that help the searcher explore related and novel aspects. Given the nature of the task graph, we operate in the space of search queries and recommend queries for a given query. To be valuable in this context, the set of recommended queries needs to be diverse, and the set should also be interesting and cover as many related tasks as possible.

Given these requirements, we identify candidate recommendations using a random walk approach. Random walk based methods (e.g., personalized page rank [25]) have been widely used in the literature for a broad range of recommendation tasks [6][7][16]. Inspired by these methods, we define the random walk method as follows.

Imagine a random surfer walking along the task graph G. Starting from one node i (i.e., a searcher query that matches one of the query groups), it either stays at i with probability β or moves to another adjacent node with probability $1 - \beta$. When it moves to an adjacent node, it selects a node j with probability P_{ij} that is proportional to the weight of the edge connecting i and j.

We define the transition probabilities $P_{t+1|t}(j|i)$ from i to j by normalizing the weights of the edges in the task:

$$P_{t+1|t}(j|i) = \frac{W_{ij}}{\sum_k W_{ik}}$$

Where k represents all nodes in the neighborhood of i. $P_{t2|t1}(j|i)$ denotes the transition probability from node i at step t_1 to node j at step 2. We note that neither the weights W_{ij}, nor the transition probabilities are symmetric.

We introduced the self-transition loops to reinforce the importance of the starting node and to slow the diffusion of the random walk to other nodes. Previous work has set the value of the self-loop probability to $\beta = 0.9$ [6][16]. We investigated different values of probability β and demonstrate their impact effect on performance in the evaluation section. We terminate the walk after a maximum of 30 iterations or when the norm of the difference between two successive iterations is less than 10^{-6}. We rank the recommended tasks based on the stationary distribution of the random walk.

Before applying the random walk model, we removed any edge if its weight is less than 0.2 and we also removed nodes that no connections to any other nodes. We also noticed that some nodes (tasks) have appeared in the graph with a very large degree (number of edges). These nodes are typically navigational queries that were not excluded using the navigational query filter described earlier in the paper. These nodes will result in connecting many unrelated tasks since they are very central to the graph. To alleviate this problem, we remove any node that has more than M edges in the graph. M was set heuristically to 300 and that resulted in removing of less than 1% of the nodes from the task graph.

6. EVALUATION

To evaluate our approach we build a task graph and use the methods outlined in Section 5, along with other baselines that will be described below, to generate lists of exploratory query suggestions. For our data, we use four weeks of logs from the Microsoft Bing Web search engine as described in Section 3.1. The graph contains over one million nodes (tasks) and over 35 million edges (connections between tasks). The number of nodes is less than the number of tasks identified in Section 4.1 since we performed a lot of pruning (e.g., removing unconnected nodes, nodes with very high degree, etc.), as described in the preceding sections.

We tested the performance of our system using a set of 300 test queries, of varying frequency. We split the nodes in the graph into three equally-sized groups according to their frequency, and sampled 100 queries from each group resulting in a total of 300 queries. Examples of these queries are shown in Table 2. Each query is associated with a task (i.e., a node in the task graph).

6.1 Research Questions

To assess the quality of our task recommendation method for supporting complex tasks, we performed crowd-sourced assessments to answer the following research questions:

RQ1: Relatedness: Are the recommended tasks related to the original query? Relatedness is important since searchers are unlikely to be interested in unrelated suggestions.

RQ2: Interestingness: Will searchers be interested in exploring the recommended tasks given their original query? Interestingness is important since we are not trying to propose rewrites or refinements of the current query. Hence, a searcher is likely to be interested in the suggestions if they are both related and novel.

RQ3: Diversity: Are the recommendations intrinsically diverse? Diversity is important since we are providing multiple suggestions to searchers, and it is preferable to avoid redundancy.

RQ4: Completeness: Do the recommendations address most of the sub-tasks (aspects or steps) associated with the task needed to accomplish the complex task? Given our objective of supporting exploration, it is desirable to assess the coverage of the suggestions.

Table 2. Examples of queries in the evaluation set.

Example Queries
vacation packages to cancun
baltimore inner harbor attractions
adam levine engaged
celebrities born on halloween
george zimmerman verdict
nbc canceled shows
what is adaptive radiation
side effects of amitriptyline
bland diet menu
gettysburg deaths
mother's day gift ideas
fall wedding wedding decorations

RQ5: Task recommendations vs. related searches: How are the task recommendations different from traditional related searches?

6.2 Study Methodology

Suggestions were labeled by judges who were recruited from the crowdsourcing service Clickworker.com, which provided access to crowd workers under contract. Judges resided in the United States and were fluent in English. As is necessary with a study on a remote crowdsourcing platform, we took several precautions to maintain data integrity. We restricted annotators to those based in the US because our logs came from searchers based in the US. We also used hidden quality control questions to filter out poor-quality judges. We had three judges work on every instance. In total we employed 69 judges and collected 6300 judgments for the first experiment (Section 6.2.1) and 900 judgments for the second experiment (Section 6.2.2).

6.2.1 Evaluating Task Recommendations

The objective of the first experiment is to evaluate the quality of task recommendations and answer RQ1 – RQ4 described above. As comparator methods, we generate and compare suggestions using the following techniques, which includes variations of the parameters in the proposed methods (i.e., Random Walk and Random Walk + Div) and some other methods used as baselines in the study:

- **Random Walk (two variants):** As explained in Section 5, we use a random walk based model over the aspect graph to generate recommendations. Recommendations are ranked based on the stationary distribution of the random walk. We experiment with two variants of this method by setting the self-probability β to 0.7 and 0.9. This allows us to study the effect of allowing the random walk to diffuse more from the original query. We noticed a clear reduction in quality as β decreases, so we did not experiment with any other values of β during the human judgment process.
- **Random Walk + Diversity (two variants):** To measure the need for an extra diversity step, we use a Maximal Marginal Relevance (MMR) [14] like function that tries to promote "relevant novelty" instead of just relevance. To measure relevant novelty, we measure relevance and novelty independently and then rank recommendations based on a linear combination of both. Formally we seek to maximize the following function:

$$Score(s_i) = \lambda\, Relev(s_i, Q) - (1 - \lambda) \max_{j<i} Sim(s_i, s_j)$$

where Q is the original query, $S = \{s_i, \dots, s_n\}$ is the list of suggestions, $Relev(s_i, Q)$ is the stationary distribution score described above normalized to be $\epsilon[0,1]$, $Sim(s_i, s_j)$ is function to measure the similarity between different aspects. In this study, we define $Sim(x, y)$ as the cosine similarity between word text frequency representations of x and y. Finally, $\lambda \epsilon [0,1]$ is a parameter to control the tradeoff between aspect relevance and aspect diversity. We set λ to 0.5 in all our experiments. We tried to change λ to 0.3 and 0.7, but we did not notice a significant change in the results. We applied these criteria to re-rank the top 20 results as defined by the value of the random walk stationary distribution. For queries with fewer than 20 aspects recommended, we re-rank all available suggestions.

- **Second Order Similarity:** This baseline identifies candidate recommendations using a second order co-occurrence model over our graph G. Offline, we compute the similarity matrix between all pairs of tasks as follows. For each task t, we construct a multi-dimensional vector $\vec{t}$ consisting of $|T|$ coordinates, where $t_i = w(t, i)$, i.e., the weight of the edge between tasks t and i in G. The similarity between two tasks t_1 and t_2 is then computed as the cosine of the angle between $\overrightarrow{t_1}$ and $\overrightarrow{t_2}$. This baseline identifies recommendations for a task by selecting other tasks with the highest pairwise similarity to the current task.
- **Neighbors Ranked:** For every query, we obtain the corresponding node (task) and retrieve all neighbors of that node. Neighbors are ordered by the weight of the edges connecting them to the original node and the neighbors with the highest such weight are returned.
- **Neighbors Random:** This baseline is similar to the *Neighbors Ranked* baseline except that we do not order the neighbors according to their weights. Rather nodes are selected uniformly at random without replacement from the set of neighbors.

For every method, we show a maximum of eight recommendations to judges. Judges were provided with an original query (one of the 300 used in the experiment) and a list of suggestions, and asked to judge the following dimensions on a three-point scale:

Relatedness: Suggestions are: (1) *Related*: all suggestions are related to the original query; (2) *Somewhat Related*: many suggestions are related to the original query; or (3) *Not Related*: most or all of the suggestions are not related to the original query.

Interestingness: Suggestions are: (1) *Interesting*: all suggestions are interesting given the original query; (2) *Somewhat Interesting*: many suggestions are interesting; or (3) *Not Interestingness*: most or all suggestions are uninteresting.

Diversity: Suggestions are (1) *Diverse*: suggestions are completely distinct from one another; (2) *Somewhat Diverse*: many suggestions are distinct but some are redundant; or (3) *Not Diverse*: most or all suggestions are redundant.

Completeness: Suggestions are (1) *Complete*: I cannot think of any missing aspects or steps related to the task; (2) *Somewhat Complete*: I can think of a few missing aspects or steps; or (3) *Not Complete*: I can think of many missing aspects or steps.

Since most judges label largely disjoint sets of aspects, we do not report the standard Cohen's kappa for inter-annotator agreement. Instead, we report label agreement, which was 87.4%, 81.4%, 87.0% and 70.8% for relatedness, interestingness, diversity, and completeness respectively. This level of agreement demonstrates that judgment variance is quite small, and increases our confidence in the reliability of the judgments for evaluating our methods.

Table 3. Performance in terms of Relatedness, Interestingness, Diversity and Completenes for the Task recommendations methods and baselines. All the differences between the random walk methods are statistically significant at $p \leq 0.001$ using a χ^2 test. Signficance of the random walk methods to the second order, neighbors ranked and neighbors random is denoted as *, †, ^ respectively. $p \leq 0.05$ is denoted as *, $p \leq 0.01$ is denoted as $*^2$ and $p \leq 0.001$ is denoted as $*^3$.

	Random Walk (β=0.7) + Div.	**Random Walk (β=0.7)**	**Random Walk (β=0.9) + Div.**	**Random Walk (β=0.9)**	**Second Order Similarity**	**Neighbors Ranked**	**Neighbors Random**
				Relatedness			
Related	67.56%$^{*2,\dagger 3,\wedge 3}$	**70.80%**$^{*3,\dagger 3,\wedge 3}$	62.22%$^{*2,\dagger 3,\wedge 3}$	67.70%$^{\dagger 3,\wedge 3}$	68.85%	61.94%	55.33%
Somewhat Related	23.67%	21.52%	29.78%	25.63%	21.58%	26.64%	29.67%
Not Related	8.77%	7.68%	8.00%	**6.67%**	9.56%	11.42%	15.00%
				Interestingness			
Interesting	**66.67%**$^{*3,\dagger 3,\wedge 3}$	58.19%$^{*1,\dagger 3,\wedge 3}$	43.44%$^{*3,\dagger 3,\wedge 3}$	58.44%$^{*3,\dagger 3,\wedge 3}$	63.39%	51.21%	44.33%
Somewhat Interesting	25.11%	32.01%	47.11%	29.72%	26.05%	31.83%	35.00%
Not Interesting	**8.22%**	9.80%	9.44%	11.84%	10.56%	16.96%	20.67%
				Diversity			
Diverse	**63.44%**$^{*3,\dagger 3,\wedge 3}$	62.43%$^{*3,\dagger 3,\wedge 3}$	60.89%$^{*1,\dagger 1,\wedge 3}$	55.75%$^{*3,\dagger 3,\wedge 3}$	**72.13%**	58.82%	53.33%
Somewhat Diverse	32.33%	33.22%	37.78%	37.21%	20.22%	30.45%	32.33%
Not Diverse	4.22%	4.35%	**1.33%**	7.04%	7.65%	10.73%	14.33%
				Completeness			
Complete	**48.67%**$^{*3,\dagger 3,\wedge 3}$	47.60%$^{*1,\dagger 3,\wedge 3}$	44.22%$^{\dagger 3,\wedge 2}$	40.45%$^{*1,\dagger 2,\wedge 3}$	43.44%	43.94%	35.00%
Somewhat Complete	38.33%	39.58%	44.67%	39.66%	34.15%	31.83%	38.33%
Not Complete	13.00%	12.82%	**11.11%**	19.89%	22.40%	24.22%	26.67%

6.2.2 Task Recommendations vs. Related Searches

The objective of the second experiment is to answer RQ5, which tries to characterize the differences between our task recommendations and traditional related searches. This is valuable in quantifying the degree of difference between the two approaches. We generated recommendations using the task graph and the random walk method (Random Walk (β=0.7)) and suggestions using the related searches from a state-of-the-art related search system that used query similarity (e.g., [28]), query-log based learning (e.g., [6][33]) and diversification (e.g., [46]).

To understand the relation between the two lists of suggestions, we conducted a comparative experiment where we showed judges the original query and the two lists side-by-side. The ordering in which the result sets were assigned to sides (left or right) was randomized. We ask judges to select the best list according to the rating dimensions described in the previous subsection (i.e., relatedness, interestingness, diversity. and completeness). Judges reported their preferences using a five-point scale where the points ranged from a strong preference for the left side to strong preference for the right side. The five response options are: *Left is better*, *Left is slightly better*, *About the same*, *Right is slightly better*, and *Right is better.*

To compare the lists A and B, we define a WinLoss measure as the number of times A is preferred over B minus the number of times B is preferred over A, divided by the total number of instances. This ranges between –1 (B always wins) and +1 (A always wins).

6.3 Findings

6.3.1 Task Recommendation

Table 3 shows the percentage of each response for the proposed methods and baselines in terms of relatedness, interestingness, diversity and completeness. A χ^2 test shows that the difference between proportions is significant at the $p \leq 0.001$ level for all variants of the random walk methods. The significance of the difference between the random walk methods and the other baselines is shown in the table where appropriate.

Relatedness: The table shows that the random walk based methods perform best when it comes to relatedness with the best performing method being the random walk method with a self-loop probability $\beta = 0.9$ and with no re-ranking for diversity applied. Relatedness slightly decreases whenever the re-ranking for diversity is used for both random walk method irrespective of β. The second order similarity baseline performs well too. Surprisingly, the recommendation based on the edge strength performs worse than the random walk methods even for relatedness. One explanation is that the way that relatedness is measured in a random walk takes both direct connections and indirect connections (through other nodes) into consideration, resulting in recommendations that are more relevant. As expected, when we select neighbors at random, we get the worst relatedness performance.

Interestingness: Interestingness is a more important measure than relatedness given that the focus of this paper is in devising techniques to support exploration. Like relatedness, we notice that the random walk methods outperform all other baselines. Unlike relatedness, the best performing method is the random walk method with a self-loop probability $\beta = 0.7$. Re-ranking for diversity has a slight effect on the performance when applied. This shows that smaller self-loop probability values are likely to allow more diffusion from the original node resulting in less related but obvious recommendations and more related and interesting recommendations. We also notice that the re-ranking for diversity also has a minor positive effect on the interestingness of the suggestions showing that the results are interesting to the judges even without the application of the re-ranking for diversity step.

Diversity: Our measure here is intended to measure the redundancy of the suggestions when compared to one another. The table shows that all methods perform very well in terms of diversity with most

Table 4. Win-Loss for the task recommendations vs. related searches. Positive = task recommendations preferred.

	Relatedness	Interestingness	Diversity	Completeness
Win-Loss	–0.60	0.44	0.54	0.65

suggestions labeled as "*Diverse*" or *'Somewhat Diverse"* with the exception of the neighbor recommendation methods which does not intentionally promote diversity. We also notice that the re-ranking for diversity has a higher effect on the random walk method when the self-loop probability is set to 0.9 compared to when it is set to 0.7. This shows that when we add more diffusion in the random walk model (smaller self-loop probabilities), we already obtain diverse results leaving little or no room for the re-ranking for diversity to achieve any gains in performance.

Completeness: We observe that all proposed methods and baselines achieve lower performance on completeness as compared to other metrics (relatedness, interestingness, etc.) with all of them achieving similar performance. These results should be interpreted with some caution since the degree of completeness is directly related to the number of suggestions that we provide. As part of our experiment, we tried to increase the number of task recommendations from 8 to 12 and noticed that the completeness improves but other metrics such as relatedness and interestingness are degraded. We should also consider that increasing the number of suggestions may have a negative effect on how the suggestions are perceived, since many of the suggestions may be confusing or distracting.

Overall, the findings of this analysis show that the task recommendations generated using our methods yield significant gains over the baselines in a number of important measures of recommendation value. However, the baselines in this part of the study are all based on the task graph. As part of our analysis, we also wanted to compare the output from our methods with that from other sources. As such, we evaluated against a strong comparator: related searches from a commercial search engine, which generates suggestions via many signals including many searchers' within-session refinements. As mentioned earlier, related searches address a different problem than task recommendations. Related searches focus on helping searchers *refine* queries, whereas task recommendations purposely direct people to different aspects of the search task.

6.3.2 Task Recommendations vs. Related Searches

Given that we are suggesting queries (although for a different purpose—exploration rather than refinement), it seems reasonable to compare the performance of our method with a state-of-the-art related search system. The results of the side-by-side judgments described in Section 6.2.2 are shown in Table 4 where we report the win-loss measure for both lists of suggestions. A positive value indicates that the proposed method wins (+1 means it always wins) and a negative value indicates that related searches win (–1 meaning that it always wins). The table shows that related searches are perceived to be much better in terms of relatedness to the original query by judges. They are also perceived to be much worse in terms of all other measures (interestingness, diversity and completeness). The differences are significant at the $p < 0.001$ level using a Z-test of proportions.

These results should be interpreted in light of the different objectives of related searches and exploratory suggestions. Related searches are intended to help searchers refine their query while the focus of the methods proposed in this paper is to identify queries that are typically involved in exploratory and complex tasks, and assist searchers in exploring more aspects of those tasks. The task recommendations are not intended to replace related searches and in fact they should also be used in different types of task (e.g., when searchers are in an exploratory mode [24]) or in different types of information seeking (e.g., when people are browsing the Web).

7. DISCUSSION AND CONCLUSIONS

There are many different types of search tasks that people pursue on search engines. Some searches are directed with the objective of navigating to a website or looking up a simple piece of information. Other searches are more exploratory in nature where the user is seeking to learn about different aspects of a topic or perform a complex multi-step task. Current search systems provide little, if any, support for the latter type of search. In this paper, we proposed a method that learns from the behavior of many searchers to support new searchers engaged in complex tasks. We do that by identifying different tasks commonly explored by searchers and connecting them using a task graph constructed to relate similar search tasks. We employ this task graph to generate task recommendations for searchers to assist them in exploring different aspects of a topic or identifying necessary steps to attain task completion. We demonstrate through experimentation with humans that our task recommendations are related and interesting with respect to the original query. Importantly, given our objectives in supporting search exploration, we also show that the lists of suggestions that our method generates are diverse and cover most of the sub-tasks related to the search tasks studied. Importantly, we also show that our methods generate recommendations more suited to supporting exploratory search than related searches in a competitive Web search engine.

Despite our promising findings, we should acknowledge some limitations of the method and this study. The approach used to identify tasks is currently limited in the queries that we can handle to those with a particular syntax (although it did cover almost three quarters of the queries in the exploration sessions identified through our automated analysis). More work is required to generalize the method to all query types. In addition, although we demonstrated significant promise in our methods along dimensions important for supporting exploration (diversity, interestingness, etc.), we need to evaluate the *utility* of the task recommendations in helping searchers complete exploratory and complex tasks. To this end, laboratory studies of searchers using the task recommendations are a necessary next step, coupled with other studies in more naturalistic settings.

The task recommendations can help searchers discover interesting tasks to assist them in exploration. At present, related searches are typically shown on the result page of the original query. This may not be an ideal place to show the task recommendations since they may distract the searcher from fulfilling the information needs associated with the current query. Search engines could present the recommendations on result pages or in a browser toolbar at a time where they estimate that the searcher has met their current needs and does not seek to refine their queries (e.g., after pressing the back button following a long dwell click, etc.). Personalized task recommendations, tailored to searchers' long-term interests (and what aspects of the complex task they have attempted thus far), could also be shown on the search engine homepage. Several Web browsers now offer page recommendations for particular websites (e.g., the *Flip Ahead* and *Suggested Sites* features in Internet Explorer suggest pages of potential interest to users given their current browsing context). This could offer on-demand or proactive assistance to those in complex search tasks. Future work involves further analysis, including measuring the value of the task recommendations for task completion. Other opportunities include model refinements such as richer query intent to handle other query types and extending the edges to cover templates (e.g., "cheap_flights to CITY", "hotels in CITY") rather than specific search queries.

REFERENCES

[1] Agichtein, E. and Zheng, Z. (2006). Identifying “best bet” web search results by mining past user behavior. *Proc. KDD*, 902–908.

[2] André, P., Teevan, J., and Dumais, S. (2009). From x-rays to silly putty via Uranus: Serendipity and its role in Web search. *Proc. SIGCHI*, 2033–2036.

[3] Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. *Proc. EDBT*, 588–596.

[4] Bergsma, S. and Wang, I. (2007). Learning noun phrase query segmentation. *Proc. EMNLP*, 816–826.

[5] Bilenko, M. and White, R.W. (2008). Mining the search trails of surfing crowds: Identifying relevant websites from user activity. *Proc. WWW*, 51–60.

[6] Boldi, P., Bonchi, F., Castillo, C., Donato, D., and Vigna, S. (2009). Query suggestions using query-flow graphs. *Proc. WSCD*, 56–63.

[7] Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., and Venturini, R. (2012). Efficient query recommendations in the long tail via center-piece subgraphs. *Proc. SIGIR*, 345–354.

[8] Bordino, I., Mejova, Y., and Lalmas, M. (2013). Penguins in sweaters, or serendipitous entity search on user-generated content. *Proc.* CIKM, 109–118.

[9] Bordino, I., Francisci Morales, G. D., Weber, I., and Bonchi, F. (2013). From machu_picchu to “rafting the urubamba river”: Anticipating information needs via the entity-query graph. *Proc. WSDM*, 275–284.

[10] Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proc. GSCL*, 31–40.

[11] Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, 36(2), 3–10.

[12] Bron, M., Gorp, J., Vishneuski, A., Nack, F., Leeuw, S., and De Rijke, M. (2012). A subjunctive exploratory search interface to support media studies researchers. *Proc. SIGIR*, 425−434.

[13] Cao, H., Jiang, D., Pei, I., Chen, E., and Li, H. (2009) Towards context-aware search by learning a very large variable length hidden Markov model from search logs. *Proc. WWW,* 191−200.

[14] Carbonell, J. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *Proc. SIGIR*, 335−336.

[15] Chalmers, M., Rodden, K., and Brodbeck, D. (1998). The order of things: Activity-centered information access. *Proc. WWW*, 359−367.

[16] Craswell, N. and Szummer. M. (2007). Random walks on the click graph. *Proc. SIGIR*, 239−246.

[17] Czech, Z. J., Havas, G., and Majewski, B. S. (1992). An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5): 257–264.

[18] Donato, D., Bonchi, F., Chi, T., and Maarek, Y. (2010). Do you want to take notes? Identifying research missions in Yahoo! Search Pad. *Proc. WWW*, 321–330.

[19] Dou, Z., Song, R., and Wen, J.R. (2007). A large-scale evaluation and analysis of personalized search strategies. *Proc. WWW*, 581–590.

[20] Friedman, J.H., Hastie, T., and Tibshirani, R. (1998). *Additive Logistic Regression: A Statistical View of Boosting.* Technical Report, Stanford University.

[21] Hagen, M., Potthast, M., Stein, B., and Brautigam, C. (2010). Query segmentation revisited. *Proc. WWW*, 97–106.

[22] Hassan, A. (2013). Identifying web search query reformulation using concept based matching. *Proc. EMNLP*, 1000–1010.

[23] Hassan, A. and White, R.W. (2012). Task tours: Helping users tackle complex search tasks. *Proc. CIKM*, 1885–1889.

[24] Hassan, A., White, R.W., Dumais, S., and Wang, Y. (2014). Exploring or struggling? Disambiguating long search sessions. *Proc. WSDM*, 53–62.

[25] Jeh, G. and Widom, J. (2003). Scaling personalized web search. *Proc. WWW,* 269–273.

[26] Joachims, T., Freitag, D., and Mitchell, T. (1997). WebWatcher: A tour guide for the World Wide Web. *Proc. IJCAI*, 770–775.

[27] Jones, R. and Klinkner, K.L. (2008). Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. *Proc. CIKM*, 699–708.

[28] Jones, R., Rey, B., Madani, O., and Greiner, W. (200). Generating query substitutions. *Proc. WWW*, 387–396.

[29] Lin, T., Pantel, P., Gamon, M., Kannan, A., and Fuxman, A. (2012). Active objects: Actions for entity-centric search. *Proc. WWW*, 589–598.

[30] Liu, J. and Belkin, N.J. (2010). Personalizing information retrieval for multi-session search tasks. *Proc. SIGIR*, 26–33.

[31] Manning, C.D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

[32] Marchionini, G. (2006). Exploratory search: From finding to understanding. *CACM*, 49(4): 41−46.

[33] Mei, Q., Zhou, D., and Church, K. (2008). Query suggestion using hitting time. *Proc. CIKM*, 469−478

[34] Morris, D., Morris, M.R., and Venolia, G. (2008). SearchBar: A search-centric web history for task resumption and information refinding. *Proc. SIGCHI*, 1207−1216.

[35] O’Connor, B., Krieger, M., and Ahn, D. (2010). TweetMotif: Exploratory search and topic summarization for twitter. *Proc. ICWSM*, 384–385.

[36] Olston, C. and Chi, E. (2003). ScentTrails: Integrating browsing and searching on the web. *TOCHI*, 10(3): 1–21.

[37] Pantel, P., Lin, T., and Gamon, M. (2012). Mining entity types from query logs via user intent. *Proc. ACL*, 563–571.

[38] Raman, K., Bennett, P.N., and Collins-Thompson, K. (2013). Toward whole session relevance: Exploring intrinsic diversity in web search. *Proc. SIGIR*, 463–472.

[39] Singla, A., White, R.W., and Huang, J. (2010). Studying trailfinding algorithms for enhanced web search. *Proc. SIGIR*, 443–450.

[40] Villa, R., Cantador, I., Joho, H., and Jose, J. (2009). An aspectual interface for supporting complex search tasks. *Proc. SIGIR*, 379-386.

[41] Voorhees, E.M. and Harman, D.K. eds. (2000). *TREC-9. The ninth Text REtrieval Conference*. Washington, D.C.: GPO.

[42] Wexelblat, A. and Maes, P. (1999). Footprints: history-rich tools for information foraging. *Proc. SIGCHI*, 270–277.

[43] White, R.W., Bennett, P., and Dumais, S. (2010). Predicting short-term interests using activity-based search context. *Proc. CIKM*, 1009–1018.

[44] White, R.W. and Huang, J. (2010). Assessing the scenic route: Measuring the value of search trails in web logs. *Proc. of SIGIR*, 587–594.

[45] White, R.W. and Roth, R.A. (2009). *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan Claypool.

[46] Ziegler, C., McNee, S.M., Konstan, J.A., and Lausen, G. (2005). Improving recommendation lists through topic diversification, *Proc. WWW*, 22–32.