Search Logs as Information Footprints: Supporting Guided Navigation for Exploratory Search

Xuanhui Wang, Bin Tan, Azadeh Shakery, ChengXiang Zhai

Department of Computer Science

University of Illinois at Urbana Champaign

{xwang20, bintan, shakery, czhai}@uiuc.edu

Abstract

While current search engines serve known-item search such as homepage finding very well, they generally cannot support exploratory search effectively. In exploratory search, users do not know their information needs precisely and also often lack the needed knowledge to formulate effective queries, thus querying alone, as supported by the current search engines, is insufficient, and browsing into related information would be very useful. In this paper, we present a formal navigation-based retrieval framework to unify querying and browsing and treat both as navigation over topic regions. To support browsing effectively, we treat search logs as "footprints" left by previous users in the information space and build a multi-resolution topic map to guide a user in navigating in the information space. To test the effectiveness of the proposed methods, we build a prototype system based on a small sample of search logs and a commercial search engine. Our experiment results show that the proposed navigation-based framework is promising and the proposed methods for guided navigation are effective.

1 Introduction

Users' search tasks vary a lot from a simple known-item search to very complex exploratory search. In known-item search, a user has a well-defined information need and can generally formulate a very effective queryand thus the current search engines often work very well. In exploratory search, however, the information need is often complex and vague, and the goal of search is mainly to gather and study information about some topic. Thus a user generally does not know well about the information to be found in exploratory search (which is the reason why the user needs to initiate the search in the first place). As a result, it is often difficult for a user to formulate effective queries in exploratory search, and the user has to reformulate queries many times in a trial-and-error manner. For example, when a user wants to buy a used car, what he/she needs is not just a single piece of information such as a list of used car dealers, but also opinions about the dealers by previous customers, advantages/disadvantages of different brands, and advice on car insurance, etc. Formulating effective queries to find all this information is quite challenging, especially for a user who does not know well about the domain. For these reasons, the current search engines generally do not perform well for exploratory search compared with known-item search [20]. Since exploratory search happens very often, it is very important to study how to help users to conduct effective exploratory search [38, 20].

Despite its importance, however, exploratory search appears to have just begun to attract serious attention in related research communities [39, 37]. Most of the current work has been summarized in a recent special issue of Communications of ACM [38]. In general, existing work on exploratory search has been mostly focused on interface design and visualization, but developing algorithms and formal retrieval frameworks for supporting exploratory search have been significantly under-addressed.

In this paper, we present a formal navigation-based retrieval framework to integrate querying and browsing, and develop related algorithms to effectively support exploratory search. Our main idea is to provide *guided navigation* so that a current user can be guided by the "footprints" in the information space left by previous users when they explore similar or related information.

Querying and browsing are two complementary ways of finding information, however the current Web search engines mainly support querying, thus a user can only browse through static hyperlinks between Web pages, which is quite restrictive because a user would unlikely be able to navigate into a topic remotely related to the current page. For exploratory search, it is quite important to help a user navigate into remotely related relevant pages to compensate for the difficulty in formulating effective queries. Our idea is thus to *break the limitation of hyperlink-based browsing and leverage the footprints of other users to allow a user to navigate into remotely related topics*.

Our emphasis on navigation can be carried further to formulate a general navigation-based retrieval framework that can unify querying and browsing. Both querying and browsing can be regarded as navigating in the information space: querying is to navigate into an arbitrary topic area specified by a query in the information space, while browsing typically navigates into a nearby topic area. Thus we can combine querying and browsing naturally in a navigation-based retrieval framework in which a user would be able to flexibly choose between browsing and querying depending on the situation. In general, when the user can formulate an effective query, he/she can choose querying, otherwise, the user could browse through related topic regions to explore the information space.

To support a user to browse more effectively in our framework, a major challenge to be solved is how to build topic regions and "semantic navigation roads" between them for users to follow. In this paper, we propose to build a multi-resolution topic map to support guided navigation. In particular, our map is built based on past queries and clickthroughs in search logs, which can be regarded as footprints left by previous users in the information space. Just as the footprints of previous visitors can help guide future visitors to famous sites in a park, the footprints in an information space left by previous users can also help future information seekers. Indeed, people often look for similar information due to various reasons (e.g., people often need to perform similar tasks, have similar occupations, or share similar cultures), thus it is possible to leverage previous users' search experiences to help a current user reach relevant information more quickly.

The idea of having multiple resolutions is to capture those frequently-visited topic regions

Figure 1: Interface snapshot of our system.

in different granularities. Topics with coarse granularities (i.e., low resolutions) subsume those with finer granularities (i.e., high resolutions). For example, "car" can subsume "car rental," "car pricing," and "car insurance". With a multi-resolution topic map, a user can easily reach topics of different granularities through *vertical navigation* (i.e., zoom in and zoom out) as well as topics with the same level of granularity through *horizontal navigation* (i.e., moving to a neighbor area), achieving flexible navigation. Typically, after a user submits a query, the user would arrive at a topic region in the information space, or a node on the map. From this node, he/she can use the map as guidance to visit many related topic regions, including those that are remotely related, to explore the information space effectively.

Based on our navigation-based retrieval framework, we develop a prototype system (see Section 4). Figure 1 shows a snapshot of our system interface. On the left pane, a fisheye view of the map is shown and on the right pane, information about a topic region (e.g., search results of a query) is shown. At any time, a user can have two ways to navigate – either moving through the topic map or formulating a new query. When the user is navigating through the map, the user can view the current region though double clicking the interesting node. Similarly, when the user submits a new query, the map in the left pane will also be updated to reflect the target region of the query. While on the surface, our interface is similar to the clustering interface of search results, there is a critical difference in the underlying information seeking model. The purpose of a clustering interface is to facilitate a

user navigating *inside* the current search results, while our map is to allow a user to navigate *outside* the current search results and potentially reach quite different topic areas.

We evaluate the potential benefit of our navigation-based retrieval framework based on a sample log data from a commercial search engine. The results show that the proposed navigation-based framework is promising and our search log based topic map is effective and can help a user reach useful pages quickly through pure navigation.

2 Navigation-Based Exploratory Search

We view exploratory search as a process of iterative navigation in an information space where navigation is broadly interpreted to include both querying and browsing. Compared with existing retrieval frameworks, our framework emphasizes more on supporting browsing which we believe is critical in exploratory search.

2.1 A Formal Navigation Framework

We first formally define a new navigation-based framework for exploratory search. Let $C = \{d_1, ..., d_{|C|}\}$ be a collection of documents to be searched. We first define the concepts *topic region* and *topic region space*.

Definition 1 (Topic Region) *A topic region $T \subset C$ is a subset of documents that are about a topic. For example, all the documents matching a phrase can form a topic region characterized by the phrase.*

Definition 2 (Topic Region Space) *The topic region space S is the set of all possible topic regions defined on C. That is, $S = 2^C$.*

The topic region space enables us to view querying as navigation in this space. More specifically, a user would end up viewing a subset of documents (i.e., search results) after submitting a query, thus we can view this process as helping the user navigate into the topic region corresponding to the search results. Note that for generality, we allow the topic region space to contain potentially non-coherent topic regions.

Since search results are generally ordered by relevance values, we further assume that at any time point t, we have a *user-specific* ranking function $r_t : C \rightarrow \Re$ which assigns a real value score to each document and thus defines an order of documents. As will be discussed later, this ranking function can be improved over time as we collect more information about the user. That is, the ranking of documents will be context-sensitive.

When a user repeatedly submits a query, the user would be essentially visiting different topic regions defined by the queries, and at any time ponit t, a user could view the documents in the current region in the order defined by the ranking function r_t.

To make browsing a "first-class citizen," we define the concept *topic map*.

Definition 3 (Topic Map) *A topic map $M = (V, E)$ is a graph with regions as vertices (i.e., $V \subset S$). An edge between two topic regions means that the user can navigate from one topic region into the other. That is, if $(v_i, v_j) \in E$, then a user would be able to navigate between v_i and v_j.*

Intuitively, the topic map provides a way to navigate in the topic region space through pure browsing. Indeed, when a user follows a path on the map, the user would be moving from one topic region to another, just like submitting reformulated queries. In general, when a user is visiting a topic region at time point t, the documents in that region would be ranked using r_t, the ranking function at time t for this user.

We now see clearly that our navigation framework naturally unifies querying and browsing in a single framework. We can now formally define these two different navigation operators:

Definition 4 (Navigation Operator) *A navigation operator is a function that maps one topic region to another. We use N as the set of all navigation operators. That is, $N = \{f : S \rightarrow S\}$.*

Definition 5 (Query Navigation Operator) *A query navigation operator $Q(q)$ is defined as $Q(q)(T) = T_q$, where q is a query and T_q is the topic region corresponding to the search results of using the query q. For any $T_1 \neq T_2$, we have $Q(q)(T_1) = Q(q)(T_2)$. Therefore, such a definition assumes that a query navigation operator returns a topic region*

regardless the current region. It is thus a "memoryless" navigator and we can use $Q(q)$ to represent T_q without incurring confusion.

Definition 6 (Browsing Navigation Operator) *A browsing navigation operator $B(v_1, v_2)$ is defined as $B(v_1, v_2)(v_1) = v_2$, where $(v_1, v_2) \in E$ is an edge on the topic map. $B(v_1, v_2)(v)$ is undefined if $v \neq v_1$. Intuitively, a browsing navigation operator $B(v_1, v_2)$ brings a user from topic region v_1 to v_2.*

Definition 7 (Compatibility) *Two navigation operators N_i and N_j are compatible if and only if one of the following three conditions holds: (1) N_j is a query navigation operator; (2) $N_i = B(v_1, v_2)$ and $N_j = B(v_2, v_3)$; (3) $N_i = Q(q)$ and $N_j = B(Q(q), v)$.*

Definition 8 (Navigation Trace) *A navigation trace is a sequence of navigation operators $N_1, N_2, ..., N_k$ such that N_i and N_{i+1} are compatible.*

With these definitions, we can describe any user's information seeking process as a navigation trace. For example, if the user submits a query q_1, navigates into a region T_1 from the search result region, navigates further from T_1 to T_2, and finally submits another query q_2, then the process can be formally described by the navigation trace $Q(q_1), B(Q(q_1), T_1), B(T_1, T_2), Q(q_2)$. The flexibility of combining multiple operators formally to describe an arbitrary information seeking process shows the expressiveness of our framework. Indeed, it provides a solid theoretical basis for studying many different ways to combine querying and browsing as well as developing systems to integrate querying and browsing.

The main task in developing a system to support the proposed navigation framework is to implement the query navigation operator and the browsing navigation operator. The query navigation operator can be implemented using any existing retrieval/ranking function. For example, we can score documents, rank them, and then take top K documents as the value of $Q(q)(T)$. The browsing navigation operator can be implemented by constructing a topic map with pre-defined regions. The map can be pre-computed to avoid query-time overhead, but it can also be constructed dynamically based on the navigation trace of the user. When we use a pre-constructed map, we may face a problem of incompatible navigation operators

when a user switches from querying to browsing as the result topic region of the query may not correspond exactly to a node/point on the map. This problem can be solved by defining a "query-extended" map in which we would add any current query result topic region to the map and connect this topic region to a few "closest" regions on the map. (We will discuss later how we define the closeness.) This would allow a user to navigate into the pre-defined regions on the map.

Viewing existing search engines in our navigation framework, we see that they mostly only support query navigation operators. A main contribution of our work is to study how to effectively support browsing navigation operators which boils down to how to construct a good topic map.

2.2 Multi-Resolution Topic Map

A topic map is to guide a user navigating in the information space just as a geographic map can guide a traveler touring a city. As a geographic map tends to include population-dense regions, our topic map also intends to include frequently-visited topic regions. Also, as a geographic map would show roads to connect different regions to enable transportation, our topic map would also have semantic connections between topic regions to enable browsing.

In any interesting application, especially an unrestricted domain such as Web, the topic map can be quite large. How to facilitate a user in navigating on this map would be itself a challenge. We solve this problem by constructing a topic map with multiple resolutions. The idea of multiple resolutions is again analogous to the idea of displaying a geographic map in multiple resolutions, and it would allow a user to get to one region from another easily on the map. Specifically, if the user wants to visit a topic region far away on the map, he/she can simply "zoom out" to a high-level general topic region (e.g., sports) and quickly navigate into a quite different (general) topic region (e.g., economy); similarly, if the user is interested in a region and wants to explore more in the region, he/she can "zoom in" and get a detailed view of the region (e.g., from "sports" to a set of regions such as "baseball," "basketball," and "football").

We now define the multiple-resolution topic map formally.

Definition 9 (K-Level Multi-Resolution Topic Map) *A k-level multi-resolution topic map consists of k ordered topic maps, $M = (M_1, ..., M_k)$, such that for any two adjacent maps $M_i = (V_i, E_i)$ and $M_{i+1} = (V_{i+1}, E_{i+1})$, we have a zooming relation $Z \subset V_i \times V_{i+1}$ which covers every topic region in both M_i and M_{i+1}.*

In a multi-resolution topic map, we can refine browsing navigation operator defined in Definition 6 into *vertical browsing operator*, in which E in Definition 6 is set as the zooming relation Z in Definition 9, and *horizontal browsing operator*, in which E in Definition 6 is set as E_i in Definition 9. The zooming relation tells us how to refocus on a map with a new resolution if the user zooms in/out on a current map. Specifically, suppose the user is currently visiting region v_i on map M_i. If the user zooms in, he/she will see a set of "children" topic regions $\{v_{i-1}|(v_{i-1}, v_i) \in Z\}$ on map M_{i-1}. Similarly, if the user zooms out, he/she will see a set of "parent" topic regions $\{v_{i+1}|(v_i, v_{i+1}) \in Z\}$ on map M_{i+1}. Thus, with the zooming relation and maps of multiple resolutions, a user can potentially navigate into remotely related topics quickly.

The remaining questions are: (1) How do we actually choose the regions to be included on our map? (2) How should we connect them? Theoretically speaking, we could include all the possible topic regions. But such a map would have many non-interesting regions and would be too complex to be useful for a user. Thus more focused maps such as one constructed based on an ontology or encyclopedia would be more useful. We may even allow a user to choose among multiple maps. In this paper we explore the idea of including only those topic regions frequently visited by previous users (thus including frequent queries and clickthroughs) as they are likely to be interesting to future users in exploratory search.

2.3 Ranking in the Navigation Framework

While not explored in this paper, ranking is another important component in our framework. It is thus worth some discussion.

Ranking is important for three reasons. First, the ranking function is critical for supporting the query navigation operator as we generally define the target topic region of a query navigation operator as the top-ranked documents using the query. Second, even when

a user reaches a region through browsing, it is still desirable to rank the documents in the region. As the user navigates from document to document within a region, the order of unseen documents can also be dynamically ordered as in the case of implicit feedback [30]. Third, when a user is landing on a region that is not exactly a region on our map, we will need to leverage the ranking function to find the closest regions on the map.

While ranking of documents has been the central research topic in information retrieval and Web search, the navigation framework raises some new interesting research questions related to ranking:

First, a user would leave a richer interaction history in the navigation framework which would include not just queries, clickthroughs, but also browsing actions such as zoom in/out operations and neibhorhood explorations. Existing work in personalized search and implicit feedback has already shown the usefulness of the existing query-based history information [30]. It would be very interesting to study how we can incorporate all the navigation information to further improve a ranking function.

Second, while traditionally, ranking is mainly to order documents, in the navigation framework, we also need to rank the topic regions of a map. How to generalize the current document ranking functions or design new ranking functions to perform region ranking is another very interesting research question.

As a first step in studying the navigation framework, in this paper, we simply reused the ranking function provided by an existing search engine, leaving all these questions for future work.

3 Search Log based Topic Map

Our idea of supporting exploratory search is through exploiting search engine logs to build a topic map. Search engine logs record the activities of Web users and generally have the following information: text queries that users submitted, the URLs that they clicked after submitting the queries, and the time when they clicked. During search processes, users may reformulate their queries several times and these form query sequences in the logs. All this

information can be regarded as footprints left by previous users.

In this paper, we only utilize text queries and clicked pages as footprints of previous users. If we think the whole web as an information space, frequently clicked pages are the items in the information space which have been frequently visited by past users and those pages which are "close" to each other in the information space form topic regions. Text queries can be thought as concise descriptions of such topic regions. In this section, we propose to build a multi-resolution topic map to characterize the topic regions having dense footprints and construct semantic roads between them.

3.1 Representing Footprints

We use both queries and clickthroughs to represent information footprints. Our method is to generate a pseudo-document for each query. We utilize the clickthrough information in search logs for this purpose. For each query in the logs, we have all the clicked URLs by all past users. However, only URL information would not give meaningful representations since URLs alone are not informative enough to capture the footprints accurately. To gather rich information, we enrich each URL with additional text content. Specifically, given any query, we can obtain its top-ranked results using the same search engine as the one from which we obtained our log data, and extract the search engine snippets of the clicked results, according to the log data. Given a query, all the snippets of its clicked URLs are used to generate a pseudo-document. Thus, each pseudo-document corresponds to a unique query and the keywords contained in the query itself can be regarded as a brief summary of the corresponding pseudo-documents. Intuitively, all these pseudo-documents and their associated queries capture the footprints in the information space and we use them to build our topic regions through clustering techniques.

3.2 Forming Topic Regions by Clustering

Given all the queries in logs $Q = \{q_1, ..., q_n\}$, we have corresponding pseudo-documents $L_0 = \{d_1, ..., d_n\}$. We use an algorithm based on graph partition: the star clustering algorithm [2] to discover the coherent topic regions. We describe the star clustering algorithm below.

Given L_0, star clustering starts with constructing a pairwise similarity graph on this collection based on the vector space model in information retrieval [27]. Then the clusters are formed by dense subgraphs that are star-shaped. These clusters form a cover of the similarity graph. Formally, for each of the n pseudo-documents $\{d_1, ..., d_n\}$ in the collection L_0, we compute a TF-IDF vector. Then, for each pair of documents d_i and d_j $(i \neq j)$, their similarity is computed as the cosine score of their corresponding vectors. A similarity graph G_σ can then be constructed using a similarity threshold parameter σ as follows. Each document d_i is a vertex of G_σ. If $sim(d_i, d_j) > \sigma$, there would be an edge connecting the corresponding two vertices. After the similarity graph G_σ is built, the star clustering algorithm clusters the documents using a greedy algorithm as follows:

1. Associate every vertex in G_σ with a flag, initialized as *unmarked*.
2. From those *unmarked* vertices, find the one which has the highest degree and let it be u.
3. Mark the flag of u as *center*.
4. Form a cluster C containing u and all its neighbors that are not marked as *center*. Mark all the selected neighbors as *satellites*.
5. Repeat from step 2 until all the vertices in G_σ are marked.

Each cluster is *star-shaped*, which consists a single *center* and several *satellites*. There is only one parameter σ in the star clustering algorithm. A big σ enforces that the connected documents have high similarities, and thus the clusters tend to be small. On the other hand, a small σ will make the clusters big and less coherent.

3.3 Building a Multi-Resolution Topic Map

For a multi-resolution topic map, we can build it in either a top-down or a bottom-up manner. In this section, we adopt a bottom-up hierarchical clustering method.

3.3.1 Generating Map Nodes and Vertical Relations

We use hierarchical star clustering to build map nodes and their zooming relations. Let L_0 be the set of individual queries. We apply our star clustering algorithm on L_0 with a high σ_1 values so that we can find small but very coherent topic regions. Each region/cluster provides a *center* query and all these center queries form a set L_1. Recursively, we can apply star clustering on L_1 with a medium threshold σ_2 to generate another set of center queries L_2. L_2 can then be used to generate L_3 with a small threshold σ_3 and etc. In our experiments, we generate a three-level topic map by setting $\sigma_1 = 0.7$, $\sigma_2 = 0.5$, and $\sigma_3 = 0.3$.

Recursive clustering gives us clusters in different granularities. Each cluster is a node in our map and all clusters in L_i form the set of nodes in i-th level of our map. A cluster in a coarse granularity subsumes several clusters in a finer granularity. Thus in our map, we have vertical or zooming relations among the corresponding nodes. Each cluster in different levels is a topic region which contains a set of pseudo-documents in L_0 and a set of queries.

3.3.2 Computing Horizontal Relations

The procedure above generates a k-level hierarchy which can support vertical navigation. Here we describe our methods to connect nodes/clusters in the same level to support horizontal navigation. In the same level, each cluster has a set of queries in Q and all these queries in the set can be used as the content of the cluster. Intuitively, semantically closely related clusters would have high similarities in their contents. Therefore, we can build a vector representation for each cluster and use cosine similarity score to measure the closeness of two clusters. In this paper, we propose a random walk based similarity measure which can be used to incorporate other useful information in logs such as query sequences in user sessions.

Specifically, given two clusters C_i and C_j, we would calculate a probability $P(C_j|C_i)$ to measure the probability of arriving cluster C_j if we start a random walk from C_i. The general random walk works as follows: From C_i, we randomly walk to a query $Q_b \in C_i$. Then we randomly walk to another query Q_a from Q_b. The last step is another random walk from Q_a

to a cluster C_j which contains Q_a. Therefore

$$P(C_j|C_i) = \sum_{Q_a,Q_b} P(C_j|Q_a)P(Q_a|Q_b)P(Q_b|C_i). \quad (1)$$

All those probabilities can be modelled flexibly. For example, $P(Q_a|Q_b)$ can be modelled as the probability of a user reformulates queries from Q_b to Q_a. Another version of random walk is to change Q_a and Q_b to two terms w_a and w_b respectively. Then we have a similar formula

$$P(C_j|C_i) = \sum_{w_a,w_b} P(C_j|w_a)P(w_a|w_b)P(w_b|C_i). \quad (2)$$

where $P(w_a|w_b)$ can be modelled as the probability of seeing w_a in a following query given its previous query containing w_b in user sessions. Without using any additional information, we can assume $P(w_a|w_b) = 1$ if $w_a = w_b$ and 0 otherwise. Then Equation (2) can be simplified as

$$P(C_j|C_i) = \sum_{w} P(C_j|w)P(w|C_i). \quad (3)$$

In our experiments, we use Equation 3 and estimation $P(w|C_i) = \frac{c(w,C_i)}{\sum_w c(w,C_i)}$ and $P(C_j|w) = \frac{c(w,C_j)}{\sum_C c(w,C)}$ where $c(w, C)$ is the count of w appearing as a content word in cluster C.

3.3.3 Labelling Map Nodes

Each cluster generated above corresponds to a node in our topic map. To provide effective guidance when end users navigate in our topic map, we need to associate a meaningful label with each node. A label should be informative enough to represent the nodes' information in the corresponding cluster. We use query words to generate labels for each node in our map since query words are more accessible from a user's viewpoint. In this paper, we use a variant of frequent pattern algorithm to generate the labels in a top-down manner. We start from the nodes in the highest level (Level 3) of our map. For each node, we take every query in the corresponding cluster as a word sequence and find the most frequent one (unigram) or two words (bigram) in the corresponding query set as its label. For example, we can get a label "car" for a node in Level 3. After generating labels for Level 3, we apply the similar procedure to Level 2, but with a constraint that a word will not be selected if it has been used by its parent node. After we get the frequent word(s) for a node in Level 2, we *append*

the label of the node's parent node in Level 3 as prefix to label the node. For example, if we get the most frequent word of a node in Level 2 as "rental" and the node's parent's label is "car", then we label the node by "car::rental". For a node in Level 1, we use the *center* queries output by the star clustering algorithm as its label.

4 Prototype System

We build a prototype system based on a small sample of search logs and a commercial search engine to integrate querying and browsing. A snapshot of our system interface is shown in Figure 1. In our system, a user can have three operators: querying, viewing a map node, and navigating in the map.

Querying. When a user submits a new query through the search box, the search results from a search engine will be shown in the right pane. At the same time, we locate the query in our map and present a ranked list of its closest nodes in Level 1 in the left pane for users to select. To rank the nodes in Level 1, we first retrieve the top m pseudo-documents using the standard Okapi method [25]. Each pseudo-document corresponds to a past query. For nodes/clusters in Level 1, we count how many of the retrieved pseudo-documents each contains and use these counts to order them.

Viewing a map node. When a user *double* clicks on a map node, we will display all the clicked URLs aggregated over all the past queries in the clicked node on the top of the right pane. At the same time, the label of clicked node will be sent to the search engine as a query to retrieve search results. The content in the right pane tells a user the most frequently visited pages for in the current node and also the search results. The user can thus visit footprints of previous users or leave his/her own footprints by examining new search results.

Navigating in the map. The left pane in our interface is to let a user navigate in the map. When a user clicks on a map node, this pane will be refreshed and a fisheye view with the clicked node as the current focus will be displayed. In a fisheye view, we show the parents, the children, and the horizontal neighbors of the current node in focus (labelled as "center" in our interface). A user can thus zoom into a child node, zoom out to a parent node, or

navigate into a horizontal neighbor node. In our current implementation, the children and neighbor nodes are ordered by Equation 3 and the parent nodes are ordered by their size (the number of children they contain).

The three different operators provide flexibility for users to either querying or browsing interchangeably. Navigating in the map provide semantic roads to help user reach related topic regions even without formulating a query by himself/herself.

5 Experiments

5.1 Data Set

Our data set is a sample of search log data from a commercial search engine. In total, this log data spans 31 days from 05/01/2006 to 05/31/2006; there are 8,144,000 queries, 3,441,000 distinct queries, and 4,649,000 distinct URLs in the raw data.

To test our system, we separate the whole data set into two parts according to the time: the first 2/3 data is used to simulate the historical data that a search engine accumulated. We treat this log data as footprints and build our topic map. The last 1/3 data is hold out to construct our test cases which will be described in details in a later section. In the history collection, we clean the logs by only keeping those frequent, well-formatted, English queries (queries which only contain characters 'a', 'b', ..., 'z', and space, and appear more than 5 times). After cleaning, we get 169,057 unique queries in our history collection in total. On average, each query has 3.5 distinct clicks. For each query, we build a "pseudo-document" based on its clicked snippets. The average length of these pseudo-documents is 68 words and the total data size of our history collection is 129MB.

5.2 Three-Level Topic Map

Based on the history collection we described above, we build a three-level topic map according to the method we described in Section 3. The first level has the finest granularity and the third level has the most coarse granularity. We show several examples of the nodes/clusters

Level 1	Level 2	Level 3	Horizontal
used+car	**car::used**	**car**	**car**
car+prices	used+car	car::used	auto 0.05
used+car+prices	car+ratings	car::blue+book	cars 0.03
new+car+prices	car+for+sale	car::rental	insurance 0.02
used+car+reviews	used+car+finder	car::pictures	acura 0.01
used+car+values	edmunds+car	car::parts	loan 0.01

Table 1: Examples of nodes in the three-level topic map.

in different levels in Table 1. In this table, the boldfaced words are the labels for the nodes/clusters. We use past queries as labels for first-level nodes, but we use query words instead of entire queries to label the nodes/clusters in the second and third levels. A major reason is that user queries are specific in most cases so that they are not always suitable to label clusters in a coarse granularity. Instead, using partial queries or query words is better. From these tables, we can see that the nodes/clusters in the first level are very coherent. All the queries in a cluster have mostly equivalent meanings to the node's label. On the other hand, the second and third level clusters represent larger scope of concepts.

Our topic map can also enable users to view topics in the same level of granularity and we show an example of horizontal neighbors of the third level in the last column of Table 1 and the closeness of neighbors is calculated by random walk based similarity in Equation 3. We can see that the top ranked neighbors are indeed related and provide flexible choices for users to navigate into related topic areas. For example, we can go from "car" to "auto," to "loan,", or to "insurance." All these neighbors provide useful guidance/choices for users to navigate into related topic regions.

5.3 Effectiveness of Guided Navigation

5.3.1 Experiment Design

To test the effectiveness of our idea of using search logs as footprints, we construct our test cases using the sessions in our hold-out test logs. The test logs consist of user sessions and

in each session, a user submitted several queries sequentially and clicked certain documents for each of the submitted queries. A typical scenario is that a user first tries an initial query and clicked certain documents. If the current results are poor or the user wants to find more relevant information, the user will reformulate the queries several times and click on more documents which are appealing to him/her. In our experiments, we use each session as a test case and use the clicked documents of in a session to *approximate* the relevant documents [15]. In the current search engines, users try to access additional relevant documents (i.e., clicked documents after the user submits the second query) through reformulating their queries. Our goal is to test the effectiveness of our guided navigation method in helping users find those additional documents only through browsing after the submission of the initial queries.

Formally, let $\{Q_1, Q_2, ..., Q_k\}$ be a sequence of queries that a user tried in a session and let R be all the clicked documents for queries $\{Q_2, ..., Q_k\}$. R is regarded as the *additional* relevant documents to the user's information need. Note that we do not include the clicked documents of Q_1 in R. Our experiments are designed to test whether our guided navigation can help a user reach the documents in R *only* through browsing.

Since our goal is to support exploratory search, we use several heuristics to filter those sessions in our test cases to better approximate exploratory search scenarios. Each session has at least 2 different queries and at least 10 clicked documents (including the clicks for Q_1). To ensure that queries in a session are about a coherent information need, we further require that two adjacent queries in a session should share at least one word. After applying the above heuristics to our test data, we obtain 76 sessions as our test cases. On average, each session has 2.22 queries and the size of R is 7.74.

To evaluate our methods, we conduct experiments to simulate a user's actions when the user uses our system. In particular, we simulate a one-step action which is to simulate that a user view 1 node in our map after the user submits the very first query Q_1. We will compare the benefit of this navigation action in our system with a query reformulation action of submitting a second query Q_2.

We compare our methods with two baselines. Our first baseline method (BL1) is to use Q_1 to retrieve a ranked list from a search engine. Our second baseline (BL2) is to use Q_2 to

retrieve documents from the same search engine. We use R to evaluate the accuracy of these two baselines. For our method, we use Q_1 as input to return a list of map nodes in Level 1 to a user. Then the user can first *examine* several nodes and finally decide to *view* a returned node. After the user views a node in our map, a ranked list of URLs of previously clicked documents in the node/cluster will be presented, as well as a list of organic search results from the search engine. For simplicity, we rank all the clicked URLs on the top of the search results and their rankings are decided by the historical click frequencies (see Figure 1). We then use R as relevant set to evaluate the returned URL lists after the user view a map node. To simulate which node a user will view in our map, we use 4 variants as follows.

Simu0Default: This variant is the most naive method which assumes that the user will view 1st ranked map node.

Simu0Best: This variant is to assume that the user will view the "best node" after examining the top 10 map nodes returned for Q_1. We will describe what is the best node soon.

Simu1Default: This variant is an extension of the first variant. In this variant, a user first clicks on the 1st ranked node and our system will display its neighbors. The user then examines both the 1st ranked node and its top 10 horizontal neighbors. The best node of these 11 nodes is finally viewed by the user.

Simu1Best: This variant is an extension of the second variant. In this variant, a user first clicks on the node in Simu0Best and our system will display the node's neighbors. The user then examine both this node and its top 10 horizontal neighbors. Finally the user decides to view the best node among all these 11 nodes.

For all the 4 variants, Simu0Best, Simu1Default, and Simu1Best assume a user would optimally choose the best node to view, where the best node is the one whose ranked list of URLs have the best P@10, evaluated based on R. These are optimal simulations which are to show the performance upper-bound of our system. However, given informative and accessible labels in our map, users can probably choose the best or nearly best node to view in reality. Simu1Default and Simu1Best are extensions of Simu0Default and Simu0Best, and are to test whether a user can get even more useful information after more exploration.

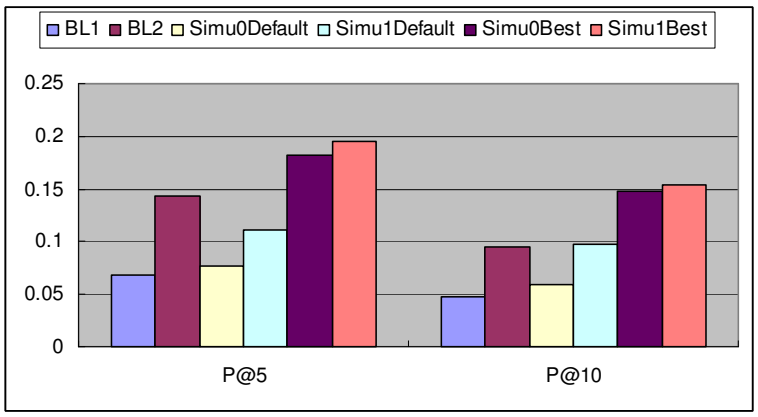

Figure 2: Comparison of different methods

Treating R as the relevant documents, we use P@5 (Precision at 5 documents) and P@10 (Precision at 10 documents) to evaluate different methods.

5.3.2 Result Comparison and Analysis

In Figure 2, we compare different methods using the two primary measures. We compare the two baseline methods BL1 and BL2 with four variants of our method. In this figure, we can see that BL1 is very poor and it means that the first query is ineffective to retrieve additional documents. Simu0Default is a naive method which assumes the user would view the first node. Since the first node is the most similar to the current query Q_1, it is not surprising that its result is also poor. BL2 uses the second query Q_2 and the result is much better. This means that reformulating queries can get more relevant documents. Our variant Simu0Best achieves much better results than BL2 and this means that selectively visiting a node in our map can reach more relevant documents accurately than reformulating a query. From this figure, we can also see that Simu1Default and Simu1Best achieve better accuracy than Simu0Default and Simu0Best respectively. This means that more documents can be reached through viewing a neighbor node. All these confirm the benefit of guided navigation and the effectiveness of our methods.

Difficult Query Analysis. We show the effectiveness of our method for difficult queries. In this experiment, we use BL1 to assess the difficulty of queries. For all the test cases, we separate them into two parts according to their P@10 in BL1. The first part (Part I)

	#Sessions	BL1	Simu0Best	Impr.
Part I (P@10 = 0)	50	0	0.114	0.114
Part II (P@10 > 0)	26	0.138	0.173	0.035

Table 2: Improvement over difficult queries with respect to average P@10. Part I corresponds to those more difficult queries.

corresponds to the cases with P@10 = 0, which means Q_1 can not retrieve any additional documents to top 10. The second part (Part II) corresponds to the cases with P@10 > 0. This means that we can retrieve at least 1 document using the original query Q_1. We compare the improvement of our Simu0Best over BL1 for these two sets of test cases using P@10. Table 2 summarizes the results. In this table, we can see that 50 test cases fall into Part I and 26 test cases fall into Part II. For Part I, we can improve P@10 by 0.114 from 0 to 0.114 on average. For Part II, the improvement is only 0.035 from 0.138 to 0.173 on average. Since the cases in Part I is more difficult than the cases in Part II, this means that navigation based on our topic map can help more for more difficult queries.

History Richness. We show the impact of history richness on our method. We use Q_2 to retrieve our history collection and use the number of returned pseudo-documents as the indicator of the history richness for a test case. According to the number of returned pseudo-documents, we separate the test cases into 4 bins. Bin 1 has 0∼40 , Bin 2 has 40∼80, Bin 3 has 80∼120, and Bin 4 has more than 120 returned pseudo-documents. Bin 1 corresponds to those cases without much history while Bin 4 corresponds to those cases having rich history. For each bin, we show the number of test cases whose P@10's are improved versus decreased, by comparing Simu0Best with BL1. The result is shown in Figure 3. From this figure, we can see that the percentage of improved test cases increases along with the increase of the history richness. For example, in Bin 4, we improve 22 and decrease 2 cases. But in Bin 2, we increase 8 and decrease 3. This confirms that the more history we have, the better we can help users for navigation. More importantly, as time goes, more and more queries will have sufficient history, so we can improve more and more exploratory searches.

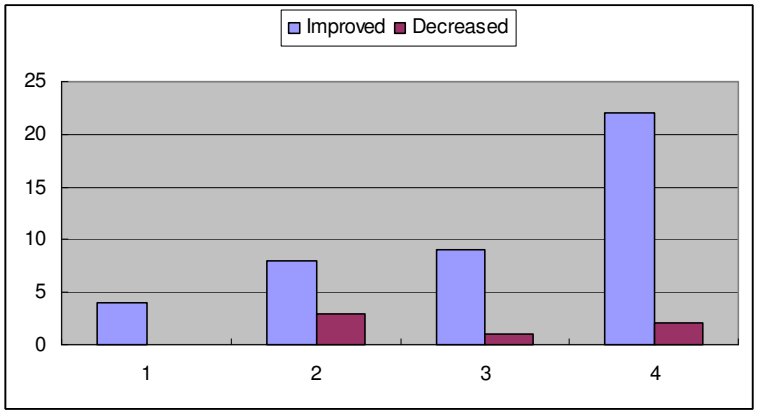

Figure 3: The impact of history richness.

Trace 1		Trace 2	
Action	Query/NodeLabel	Action	Query/NodeLabel
Querying	car accidents	Querying	health insurance
Click Node	car accidents	View Doc	www.healthinsurance.com
Zoom Out	crash::car	Click Node	health insurance
Zoom Out	crash	Horizontal	medical insurance
Horizontal	auto	Zoom Out	insurance::life
Horizontal	insurance	Zoom Out	insurance
Zoom In	insurance::car	Zoom In	insurance::traveler
Zoom In	car insurance	Zoom In	traveler insurance
Click Doc	www.geico.com	Click Doc	www.travelers.com

Table 3: System log trace examples.

5.3.3 System Log Data

We disclosed our prototype system to several users internally and logged their navigation traces. In Table 3, we show two example traces of our internal usages. In our system logs, a typical trace starting from a new query by "Querying". Then the user can navigate over our topic map or view the returned documents. On the map, the user can "Zoom In," "Zoom Out," or move "Horizontally." For example, in Table 3, a user can start from "car accidents" and finally arrive at "car insurance." Another user starts from "health insurance" and finally arrives at "traveler insurance." All these show that our map can potentially help user to navigate into remotely-related topics, which are very important for exploratory search. Apparently, all these traces provide valuable information for future research such as personalization [30].

6 Related Work

Our work is closely related to exploratory search which appears to have just begun to attract serious attention in related research communities [39, 37, 38, 20, 17, 11]. Different from the previous works in HCI community which mostly focus on interface design [38, 13, 8, 12], our emphasis is to build a topic map to guide users' navigation in the whole information space. Querying and browsing have been regarded as two information search paradigms [10, 18, 19]. Integrating querying and browsing has been studied in library science domain [3, 21, 5] where a lot of catalog or meta-data information exists. Our work is on the Web domain and relies on unstructured user history data. A recent work [29] tried to integrate browsing, searching, and visualization in a digital library and mSpace [17] project is to facilitate browsing over music space. These two works, together with [11], rely on well-defined metadata to provide multi-faceted browsing over specific domains. Their techniques can be hardly extended to Web domain and they support browsing very limitedly. Some other works such as [23, 22] do not create new ways to support browsing, but try to make the existing hyperlinks easier for users to browse. They either try to find pages which have a lot of useful hyperlinks [23] or highlight those hyperlinks which appear appealing to search queries [22]. Our work is to break the limitation of hyperlinks through a constructed topic map.

We treat search logs as footprints to guide users' navigation in the information space. In the past, search logs have been exploited for several different purposes such as query clustering [34, 4, 28], personalized search [30], latent semantic analysis [32], and learning retrieval ranking functions [24, 1]. In particular, our work is related to log based query suggestions [16, 26, 7] which is to recommend closely related queries to a current query. A main difference of our work is that we try to characterize a global structure of users's information footprints and can help users reach remotely related topic areas while query suggestions do not build a global structure and thus can only help touch local areas close to the input query. Our framework also does not restrict a map to one constructed with query log; we can use an ontology or encyclopedia to construct multiple maps for a user to choose. Recently, a special type of user logs, i.e., users' browsing trails/traces, is studied in [36]. A browsing trail is a list of pages browsed by users after a submitted query. [36] extracts the

starting queries and destination pages (i.e., the last pages in the trails) and recommend the destination pages to users when a similar query is submitted. Our work is different in that we use the general search logs and support navigation through topic maps.

Our work is also related to search result organization, which includes clustering based methods [31, 41, 40, 33, 14], categorization based methods [6, 9], and multi-facet based methods [13, 11]. All the work in search result organization is to help users navigate *inside* the current search results. A major difference of our work is that we help users navigate *outside* of the search results to explore remotely related topic regions, which is more important for exploratory search when a user's information need is not well-defined. Most of the previous work such as [14] relies on the content of search results but our work relies on search engine log data, which can generate results more accessible to users [33]. Yet our work is different from query suggestions as discussed above.

We treat search logs as "footprints" to help user search tasks. Previous work such as [35] has made a similar analogy and explored footprints to build maps, trials, and annotations to help a new user for information exploration. The novelty of our work is that we propose methods to use search logs as footprints and try to help users in Web search tasks.

7 Conclusions and Future Work

In this paper, we study how to support guided navigation for exploratory search. We define a novel navigation based exploratory search framework which can integrate querying and browsing in a navigational view. In our framework, we treat browsing as important as querying and propose to support browsing using a multi-resolution topic map. Treating search logs as information footprints, we build a topic map which can capture the frequently-visited topic regions in different granularities. Based on our topic map, we develop a prototype system which enables users to navigate from one topic region to another, and thus explore information space flexibly. We conduct experiments using a sample of search logs and the results show the promise of our framework and effectiveness of our algorithms.

Our current work can be extended in several interesting research directions. First, de-

veloping effective ranking functions to rank topic regions in our navigation framework is an interesting future topic. Second, we have built a prototype system and it provides more functionalities for users to interactively search. We can study how to collect user interaction history and use their history to evaluate our system. More importantly, these interactions give us richer navigation traces and we can study how to leverage these traces to make our map and search results more personalized. Third, we construct our topic map based on content similarity. In the future, we will study how to build a multi-view multi-resolution topic map to support users' navigation more effectively.

References

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

[2] J. A. Aslam, E. Pelekov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applicatins*, 8(1):95–129, 2004.

[3] M. J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13:407–424, 1989.

[4] D. Beeferman and A. L. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[5] N. J. Belkin. Interaction with texts: Information retrieval as information-seeking behavior. In *Information Retrieval*, pages 55–66, 1993.

[6] H. Chen and S. T. Dumais. Bringing order to the web: automatically categorizing search results. In *CHI*, pages 145–152, 2000.

[7] S. Cucerzan and R. W. White. Query suggestion based on user landing pages. In *SIGIR*, pages 875–876, 2007.

[8] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin. Fast, flexible filtering with phlat. In *CHI*, pages 261–270, 2006.

[9] S. T. Dumais, E. Cutrell, and H. Chen. Optimizing search by showing results in context. In *CHI*, pages 277–284, 2001.

[10] E. Duval and H. Olivié. Towards the integration of a query mechanism and navigation for retrieval of data on multimedia documents. *SIGIR Forum*, 26(2):8–25, 1992.

[11] J. English, M. A. Hearst, R. R. Sinha, K. Swearingen, and K.-P. Yee. Hierarchical faceted metadata in site search interfaces. In *CHI Extended Abstracts*, pages 628–639, 2002.

[12] G. W. Furnas. Effective view navigation. In *CHI*, pages 367–374, 1997.

[13] M. A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.

[14] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, pages 76–84, 1996.

[15] T. Joachims. *Evaluating Retrieval Performance Using Clickthrough Data.*, pages 79–96. Physica/Springer Verlag, 2003. in J. Franke and G. Nakhaeizadeh and I. Renz, "Text Mining".

[16] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pages 387–396, 2006.

[17] m. c. schraefel, D. A. Smith, A. Owens, A. Russell, C. Harris, and M. Wilson. The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, 2005.

[18] Y. S. Maarek. Organizing documents to support browsing in digital libraries. *SIGOIS Bull.*, 16(2):36–37, 1995.

[19] J. D. Mackinlay and P. T. Zellweger. Browsing vs. search: can we find a synergy? (panel session). In *CHI*, pages 179–180, 1995.

[20] G. Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

[21] V. L. O'Day and R. Jeffries. Orienteering in an information landscape: how information seekers get from here to there. In *INTERCHI*, pages 438–445, 1993.

[22] C. Olston and E. H. Chi. Scenttrails: Integrating browsing and searching on the web. *ACM Trans. Comput.-Hum. Interact.*, 10(3):177–197, 2003.

[23] S. Pandit and C. Olston. Navigation-aided retrieval. In *WWW*, pages 391–400, 2007.

[24] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD*, pages 239–248, 2005.

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

[26] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386, 2006.

[27] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

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

[29] R. Shen, N. S. Vemuri, W. Fan, R. da S. Torres, and E. A. Fox. Exploring digital libraries: integrating browsing, searching, and visualization. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 1–10, 2006.

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

[31] Vivisimo. http://vivisimo.com/.

[32] X. Wang, J.-T. Sun, Z. Chen, and C. Zhai. Latent semantic analysis for multiple-type interrelated data objects. In *SIGIR*, pages 236–243, 2006.

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

[34] J.-R. Wen, J.-Y. Nie, and H. Zhang. Clustering user queries of a search engine. In *WWW*, pages 162–168, 2001.

[35] A. Wexelblat and P. Maes. Footprints: history-rich tools for information foraging. In *CHI*, pages 270–277, 1999.

[36] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pages 159–166, 2007.

[37] R. W. White, S. M. Drucker, G. Marchionini, M. Hearst, and m.c. schraefel. Exploratory search and HCI. In *Proceedings of SIGCHI 2007 Workshop*, 2007.

[38] R. W. White, B. Kules, and S. M. D. m.c. schraefel. Supporting exploratory search, introduction, special issue, communications of the ACM. *Commun. ACM*, 49(4):36–39, 2006.

[39] R. W. White, G. Muresan, and G. Marchionini. Evaluating exploratory search systems. In *Proceedings of SIGIR 2006 Workshop*, 2006.

[40] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, pages 46–54, 1998.

[41] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*, pages 210–217, 2004.