A Structured Approach to Query Recommendation With Social Annotation Data

Jiafeng Guo[†], Xueqi Cheng[†], Gu Xu[‡], Hua-Wei Shen[†]

[†]Institute of Computing Technology, CAS
Beijing, P.R.China
guojiafeng@software.ict.ac.cn,cxq@ict.ac.cn,
shenhuawei@software.ict.ac.cn

[‡]Microsoft Research Asia
Beijing, P.R.China
guxu@microsoft.com

ABSTRACT

Query recommendation has been recognized as an important mean to help users search and also improve the usability of search engines. Existing approaches mainly focus on helping users refine their search queries and the recommendations typically stick to users' search intent, named *search interests* in this paper. However, users may also have some vague or delitescent interests which they are unaware of until they are faced with one, named *exploratory interests*. These interests may be provoked within a search session when users read a web page from search results or even follow links on the page. By considering exploratory interests in query recommendation, we attract more user clicks on recommendations. This type of query recommendation has not been explicitly addressed in previous work. In this paper, we propose to recommend queries in a structured way for better satisfying both search and exploratory interests of users. Specifically, we construct a query relation graph from query logs and social annotation data which capture two types of interests respectively. Based on the query relation graph, we employ hitting time to rank possible recommendations, leverage a modularity based approach to group top recommendations into clusters, and label each cluster with social tags. Empirical experimental results indicate that our structured approach to query recommendation with social annotation data can better satisfy users' interests and significantly enhance users' click behavior on recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation,Search Process*

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Query Recommendation, Structured Approach, Social Annotation Data, Search Interests, Exploratory Interests

1. INTRODUCTION

Nowadays query recommendation has been widely used in search engines, and considered as an important mean to help search users in their information seeking activities. In existing work, query recommendation mainly aims to provide alternative queries which typically stick to user's search intent, called *search interests* in this paper. For example, when a user issues a query "iphone", the search engine would suggest him equivalent or highly related queries such as "apple iphone", "iphone 3g" and "iphone price". In this way, query recommendation is mostly limited to help users to refine their queries and find what they need more quickly.

Search users may also have some vague or delitescent interests, named as *exploratory interests*, which users are unaware of until they are faced with one [21, 13]. Exploratory interests may be provoked within a search session, especially when users read some web pages from the search results or even further follow links. For example, when a user searches for "iphone" and notices another type of smartphones from search results, "blackberry" for example, he may become curious about "blackberry" and would like to know more about it. If we can provide recommendations from exploratory interests, e.g., "blackberry" or "nexus one" in this particular case, we can attract users to make more clicks on recommendations and thus spend more time on search engines. It will also increase the advertisement revenue for service providers. However, the importance of exploratory interests is less emphasized in previous work.

In this paper, therefore, we propose to recommend queries to satisfy both search and exploratory interests of users simultaneously. First of all, we propose a structured representation for query recommendation, where recommendations are organized into clusters and each cluster is explicitly labeled with some keywords (e.g., tags). It provides a clear view of the recommendations and can not only help users quickly refine their queries but also invoke their exploratory interests even before reading search results. Compared to the traditional query recommendations that are presented in a simple flat list, our approach has evident advantages. Grouping recommendations into clusters and further naming each cluster with keywords would greatly improve the readability of recommendations and help users digest them more

Table 1: Top 5 Frequent Next Queries Submitted after Query "iphone".

	Top 5 Frequent Next Queries
Overall	iphone unlocked, iphone price, iphone at&t, iphone apps, refurbished iphone
after clicking URL "http://en.wikipedia.org/wiki/iphone"	blackberry, itunes, ipod touch, apple, ipad
after clicking URL "http://www.apple.com/iphone"	at&t wireless, at&t, itunes, verizon wireless, ebay
after clicking URL "http://www.apple.com/iphone/iphone-3gs"	iphone at&t, 3g network, at&t phone packages, iphone problems, iphones price

easily, especially when query recommendations are diverse, e.g. containing both search and exploratory interests.

Data sources are another important fact in our recommendation problem. Previous work on query recommendation mostly focused on search logs, which merely capture the interactions between search users and search engines. Specifically, click-through data [3, 2, 25], search query sessions [20, 8] and query frequency [23, 11] are commonly employed to conduct query recommendation. In this paper, we first introduce social annotation data into query recommendation as an additional resource. Social annotation data is basically a collection of meta-data, or *social tags*, on URLs. Social tags are normally keywords created by millions of web users according to the content of the pages referred by the URLs. The nature of the "wisdom of crowds" makes social annotation data as reliable summaries of web pages. We can leverage the social tags to infer what people might think when reading the pages and more precisely predict the exploratory interests of users.

To conduct recommendation, we first construct a query relation graph from query logs and social annotation data, which captures both the search and browsing behaviors and reflects users' search and exploratory interests respectively. Based on the query relation graph, we employ hitting time to rank possible recommendations, leverage a modularity based approach [4] to group top recommendations into clusters, and label each cluster with social tags. Empirical experimental results indicate that our structured approach to query recommendation with social annotation data can better satisfy users' interests and significantly enhance users' click behavior on recommendations.

The rest of the paper is organized as follows. Section 2 introduces the structured approach to query recommendation considering both search and exploratory interests. Section 3 describes the proposed query recommendation approach in details. Section 4 shows the experimental results. Related work is discussed in Section 5 and conclusions are made in the last section.

2. STRUCTURED APPROACH TO QUERY RECOMMENDATION

2.1 Search Interests and Exploratory Interests

In this paper, we argue that search users conceive two types of different interests while using search engines: "search interests" and "exploratory interests". Search interests are explicit information needs when people do search, while exploratory interests are just vague or delitescent interests which can be provoked when people consume search results. To verify our statements, we made use of one-week search log data to study the shift of user interests during a search session. Specifically, we collected the next queries submitted after an initial query within a short time interval (i.e., less than 10 minutes), and analyzed the causality between the initial query and the consequent queries. The results

indicate that the formulation of the next queries is not only determined by the initial query but also significantly affected by the clicks on the search results.

Taking query "iphone" as an example, we present the most frequent queries from real users issued after initial query "iphone" in Table 1. On average, in the first line of Table 1, we can see that all the top queries submitted after "iphone" closely stick to users' search interests, since most next queries are just immediate refinements on the initial query. However, if users click a certain result URL (indicating users may read the webpage), the top next queries become different. For example, after clicking the wikipedia page of "iphone", users would show much boarder interests in other types of smartphones ("blackberry"), iphone related software ("itunes"), or even a new apple product ("ipad"). These interests are already beyond the original search interests, and fit our definition of exploratory interests. Moreover, we can also see that the top next queries will change when the clicked URL is different. It indicates that user interests, especially exploratory interests, will be affected visibly by what they have read.

We further conducted statistical tests to verify whether the existence of exploratory interests is common and significant. We would like to testify the following statement: given an initial query, the overall next queries and the next queries with clicks on a specific URL were generated from different underlying distributions as opposed to the same distribution. In other words, we are going to test the hypothesis: $H_1 : p_{qo} \neq p_{qu_i}$ against the null hypothesis: $H_2 : p_{qo} = p_{qu_i}$, where p_{qo} and p_{qu_i} denote the overall next queries distribution and the next query distribution after clicking a specific URL u_i given an initial query q respectively. These distributions can be estimated using Maximum Likelihood method.

We employed the likelihood ratio test proposed in [9] to calculate a confidence value for rejecting the null hypothesis, and the significance level p was set to 0.01. Users' random behaviors, e.g., clicking on irrelevant URLs or issuing arbitrary queries, may lead to a false rejection of null hypothesis. To reduce these false positives, we also removed the URLs having less clicks and infrequent queries from the one-week search log data. The results indicate that, in 80.9% of cases, clicks indeed affect the formulation of the next queries, i.e., rejecting the null hypothesis that users issue the same next queries no matter whether a result URL is clicked or not. Similarly, we also studied whether the clicks on different URLs will lead to different distributions on next queries. The results show that, in 43.1% of cases, users would issue different next queries if they clicked on different result URLs. These two tests prove that the existence of exploratory interests is common and significant, and users' interests will be affected by what they have read during a search session.

Search interests and exploratory interests can be considered as two different heading directions of query recommendation. If query recommendations emphasize search interests, it would help searchers to easily refine their queries

(a)	(b)	(c)
iphone 3g	iphone 3g	[iphone related]
apple iphone	iphone price	iphone 3g
iphone price	ipod touch	iphone price
iphone review	iphone review	iphone review
unlock iphone	moblileme	unlock iphone
iphone plans	unlock iphone	iphone apps
iphone jailbreak	blackberry	[apple product]
iphone apps	palm	ipod touch
iphone ringtones	iphone apps	mobileme
iphone verizon	nexus one	[smartphones]
		blackberry
		palm
		nexus one

Figure 1: Examples of recommendations for the query "iphone" (a) list-based recommendation with search interests, (b) list-based recommendation with search and exploratory interests, and (c) structured recommendation with search and exploratory interests.

and quickly find what they need in search. Thus it basically enhances the "search-click-leave" behavior. While if the recommendations more focus on exploratory interests, it would attract more user clicks on recommendations and further increase the staying time of users on search engines. However, the importance of exploratory interests is less emphasized in previous work. Traditional evaluation approaches, e.g. the similarity or relevance between recommendations and original queries [2, 24], can hardly be applied to measure the goodness of recommendations for exploratory interests. Increasing the number of clicks on recommendations can also be considered as an essential goal of query recommendations, and more importantly, applicable to evaluating recommendations for both search and exploratory interests. We will discuss more details in Section 4.4.

2.2 Structured Recommendation

In this paper, we propose to recommend queries in a structured way for better satisfying both search and exploratory interests of users simultaneously.

Traditionally, query recommendations are presented in a simple flat list. For example, Fig. 1(a) shows a typical list of recommendations for query "iphone" and the recommendations are mostly related to users' search interests. One may also leverage the list-based approach to recommend queries with both search and exploratory interests as shown in Fig. 1(b). However, since recommendations are quite diverse in this case, the readability of these recommendations will not be very good and searchers may not be able to consume the recommendations easily. Moreover, without any additional information on what the recommendations are about, users also may not have a strong will to make a click on them because of their limited knowledge on the recommendations. For example, users may not be likely to click the recommendation "nexus one" if they have never known it is also a smartphone like "iphone".

An effective solution for these above problems is to provide a structured query recommendation as shown in Fig. 1(c),
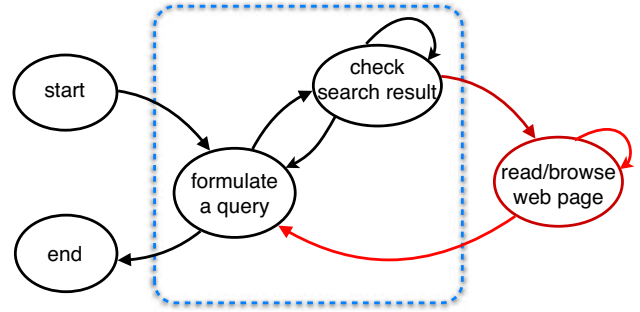

Figure 2: The Illustrated Query Formulation Model Within a Search Session.

where recommendations are organized into clusters and each cluster is explicitly labeled with some keywords (e.g., tags). In this way, it provides a clear view of the recommendation results and can not only help users quickly refine their queries but also stimulate their exploratory interests even before reading search results.

3. OUR APPROACH

The main idea of our structured approach to query recommendation considering both search and exploratory interests is as follows. We first construct a query relation graph from both query log and social annotation data, which captures the relationships between queries from the two types of interests. Based on the query relation graph, we take two steps to provide structured recommendation. Firstly, we apply a random walk on the graph and employ hitting time to rank possible recommendations with respect to the given query. We then leverage a modularity based approach to group top recommendations into clusters, and label each cluster with social tags. In this way, we can provide query recommendation in a structured way by ranking the obtained clusters according to their average hitting time.

3.1 Query Relation Graph

We construct a query relation graph which can capture both search and exploratory interests for our recommendation task. The query relation graph is a one-mode graph with the nodes representing all the unique queries and the edges capturing relationships between queries. To construct the query relation graph, we first introduce a query formulation model to help derive the relationships between queries.

3.1.1 Query Formulation Model

We consider the process how users formulate queries sequentially within a search session. Typically, the user begins with an initial query which represents his information needs and obtains some search results. After checking the search results, he may decide to refine the query to better represent his information needs, and thus issues a next query. Alternatively, he may read web pages from search results or even follow the links to browse some other web pages before he comes up with a next query. Such a query formulation model can be illustrated as Fig. 2.

Most previous work on query recommendation focused on the formulation process shown in the blue dashed box in Fig. 2, which mainly describes how users formulate their

next queries with search interests (i.e., query refinement process). Such a process can be captured by a basic query formulation model proposed in [12]. Query logs, a resource which captures the interactions between searchers and search engines, have often been leveraged for recommendation [12, 16, 25] based on the basic model.

However, the missing part of the basic model is the reading/browsing process shown as the red part in Fig. 2. In fact, this process mainly describes how users generate next queries with exploratory interests. In order to describe the proposed query formulation model, we further introduce the social annotation data as an important resource for expressing the exploratory interests in this model. Social annotation data is basically a collection of meta-data, or *social tags*, on URLs. Social tags are normally keywords created by millions of web users according to the content of the pages referred by the URLs. The nature of the "wisdom of crowds" makes social annotation data as reliable summaries of web pages. We can leverage the social tags to infer what people might think when reading the pages and more precisely predict the exploratory interests of users. Moreover, social annotation data is publicly available large scale data set which can be easily accessed.

3.1.2 Construction of Query Relation Graph

We construct a query relation graph according to the proposed query formulation model by leveraging both query logs and social annotation data. Such a graph models the relationships between queries considering both search and exploratory interests.

Typically, query logs can be represented by a Query-URL bipartite graph, as shown in the left side rectangle in blue dashed line in Fig. 3. Formally, the Query-URL graph can be denoted as $G_{qu} = (V_{qu}, E_{qu})$, where $V_{qu} = V_q \cup V_u$ denotes the node set and E_{qu} denotes the edge set. Here, $V_q = \{q_1, \ldots, q_N\}$ and $V_u = \{u_1, \ldots, u_M\}$ represent the sets consisted of unique queries and URLs respectively. E_{qu} represents the edges between queries and URLs. The weight $w_{qu}(i, j)$ on each edge $e(i, j) \in E_{qu}$ is defined by the clicks between query q_i and URL u_j.

Similarly, a URL-Tag bipartite graph can be constructed from the social annotation data. An example URL-Tag graph is shown in the right side rectangle in red dashed line in Fig. 3. The URL-Tag graph can be denoted as $G_{ut} = (V_{ut}, E_{ut})$, where $V_{ut} = V_u \cup V_t$ denotes the node set and E_{ut} denotes the edge set. Here, $V_t = \{t_1, \ldots, t_L\}$ represent the sets consisted of unique tags. E_{ut} represents the edges between URLs and tags. The weight definition is similar as that in Query-URL graph.

We obtain a Query-URL-Tag tripartite graph by connecting query logs to social annotation data through URLs as shown in Fig. 3. The proposed query formulation model can then be approximated by a transition process among query, URL and tag nodes on the tripartite graph, as illustrated by the green dot arrow lines in Fig. 3. As we can see, the forward transition from query to tags imitates the process of reading web pages related to a query and identifying some exploratory interests, while the backward transition from tags to queries imitates the process of thinking of next queries from these interests. Note that the traditional transition process over the Query-URL bipartite graph (i.e. Query→URL→Query→ · · ·) is in fact part of our transition process since each URL is always allowed to transit back to
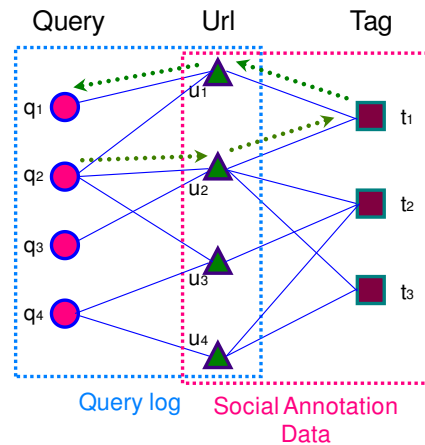

Figure 3: Example of a Query-URL-Tag Tripartite Graph.

itself through tags. In other words, the basic model where users issue queries after checking search results can also be captured by our transition process. Therefore, our transition process over the tripartite graph can well approximate the proposed query formulation model and capture both search and exploratory interests in a simple and unified way. Note that one may also design other transition processes over the tripartite graph to approximate the proposed query formulation model.

Based on the transition process on the tripartite graph, we can derive the relationships between queries and thus construct a query relation graph $G_q = (V_q, E_q)$. The query relation graph here is a weighted directed graph, with the edge weight from query node q_i to q_j defined by the transition probability under the above process as follows

$$P_{V_q|V_q}(j|i) = \sum_{u_k \in V_u} \sum_{t_l \in V_t} \sum_{u_m \in V_u} P_{V_u|V_q}(k|i) P_{V_t|V_u}(l|k) \times$$
$$P_{V_u|V_t}(m|l) P_{V_q|V_u}(j|m). (1)$$

Here the probabilities $P_{V_u|V_q}(k|i)$, $P_{V_t|V_u}(l|k)$ and $P_{V_q|V_u}(j|m)$ are defined in the form

$$P_{V_b|V_a}(j|i) = \frac{w_{ab}(i, j)}{\sum_{b_k \in V_b} w_{ab}(i, k)}.$$

It seems natural to prefer the most-clicked URL for the query, the most-annotated tag for the URL and the most-clicked query for URL, respectively. However, to use the same definition for the probability from tags to URLs (i.e., $P_{V_u|V_t}(m|l)$) may not be very appropriate. A potential disadvantage is that, for a given tag, it may prefer popularly annotated URLs (i.e., URLs annotated by a large number of people) since the absolute frequency of the tag on such URLs may be larger than other annotated URLs (i.e., $P_{V_u|V_t}$ is larger for the popularly annotated URLs). However, a popularly annotated URL does not necessarily contain more information expressed by that tag as the tag may be just a tail tag for such URL (i.e., $P_{V_t|V_u}$ is low for that tag given the popularly annotated URL). To avoid this bias, we define the probability from tags to URLs as a uniform distribution

$$P_{V_u|V_t}(m|l) = \frac{1}{d_l},$$

Algorithm 1 Ranking Queries using Hitting Time

Input: A query relation graph $G_q = (V_q, E_q)$ where V_q denotes the unique query nodes and E_q denotes the edges. The weight of the edge from q_i to q_j is defined by Eqn. (1).

1. Given a query q_s in V_q, a subgraph is constructed by using width-first search in G_q for a predefined steps.
2. For each query $q_j (j \neq s)$, iterate

$$h_{V_q|V_q}(j|i) = 1 + \sum_{k=1}^{N} P_{V_q|V_q}(k|i) h_{V_q|V_q}(j|k) \quad i \neq j$$

for all queries except q_j for a predefined number of iterations started with $h_{V_q|V_q}(i|i) = 0$.

3. Let h_s^* be the final value of $h(j|s)$ for each query q_j. Output the top k recommendations with the smallest h_s^*.

Algorithm 2 Clustering Queries using Modularity

Input: A query relation subgraph $G'_q = (V'_q, E'_q)$, where V'_q denotes the top k query nodes from Alg. 1 and E'_q denotes the edges between them. The edge weight is defined by Eqn. (1).

1. Initiate each query node q_i as a single cluster c_i.
2. **repeat**
3.   **repeat**
4.   For each query q_i, place q_i into a neighbor cluster which results in maximum positive modularity gain ΔQ.
5.   **until** There is no further improvement on modularity.
6.   Build a new graph whose nodes are clusters found in previous steps.
7. **until** There are no more changes in step $3 \sim 5$.
8. Output all the discovered clusters $C = \{c_i\}$.

---

where d_l denotes the degree of the node $l \in V_t$. An intuitive explanation for this probability is that when using tag to explore web pages, a user will equally prefer all the web pages annotated by the tag.

3.2 Ranking with Hitting Time

Based on the query relation graph, we can apply a Markov random walk on the graph and employ hitting time as a measure to rank queries. In this way, we can identify top recommendations for the given query.

The hitting time $h(j|i)$ of a random walk is the expected number of steps before node j is visited starting from node i. The hitting time from node i to node j has the property of decreasing when the number of paths from i to j increases and the lengths of the paths decrease [15]. Therefore, we can employ hitting time as a measure to rank queries with respect to the given query. Specifically, Let q_s be the given query. We compute the hitting time $h_{V_q|V_q}(j|s)$ for all the other queries q_j ($j \neq s$) based on the random walk. We then use this measure to rank queries q_j ($j \neq s$), and find the top k queries that is closest to q_s.

It can be easily verified that the hitting time satisfies the following linear system

$$\begin{cases} h_{V_q|V_q}(i|i) = 0 \\ h_{V_q|V_q}(j|i) = 1 + \sum_{k=1}^{n} P_{V_q|V_q}(k|i) h_{V_q|V_q}(j|k) \quad i \neq j \end{cases}$$

The meaning of the above recurrence formulae is quite obvious: in order to jump from node q_i to node q_j, one has to go to any adjacent node q_k of q_i and proceeds from there. Therefore, we can use the linear system to compute the hitting time iteratively. However, it would be extremely time consuming if we directly solve the linear system on the whole query graph. Since most nodes are irrelevant to the original query, we can use a width first search strategy to construct a subgraph to save the computational cost. Therefore, we propose an efficient algorithm for ranking recommendations using hitting time as shown in Alg. 1.

Note that hitting time has also been used for query recommendation by Mei et al. [25]. For a given query q_s, their approach employ hitting time from other queries q_js to q_s for ranking. In this way, they try to boost long tail queries to improve diversity in query recommendation. However, the problem with long tail queries is that they are usually unpopular queries which may not be familiar to users. Alternatively, we recommend queries considering both search and exploratory interests, and employ hitting time from the given query q_s to other queries q_js for ranking. Therefore, we can recommend diverse as well as popular queries.

3.3 Clustering with Modularity

The top k recommendations obtained above may form several groups as shown in Fig. 1(c). Therefore, we further group these recommendations into clusters and label each cluster explicitly with social tags for better understanding. Since the recommendations are nodes on the query relation graph, it is natural to apply a graph clustering approach to group them.

Clusters on a graph are basically groups of nodes in which nodes have denser connections with each other than with nodes in other clusters. There are many algorithms for graph clustering [6, 27, 10]. Recently, [26] introduced the "modularity function" Q, which measures the quality of a particular clustering of nodes in a graph. It is defined as:

$$Q = \sum_i \left[e_{ii} - a_i^2 \right],$$

where e_{ii} is the fraction of edges within cluster i, and a_i is the fraction of all *ends* of edges that are attached to nodes in cluster i. Hence, discovering the underlying clustering of a graph becomes a process to optimize the value of modularity over all possible clustering of the graph. The modularity based approaches represent the "state-of-the-art" graph clustering methods and can be easily extended to handle weighted directed graphs, e.g. the query relation graph in our problem.

Although modularity provides a quantitative method to determine the goodness of a certain clustering of a graph, brute force search of the optimal value of modularity is not always possible due to the complexity of the graph and the large number of possible divisions. Several heuristics have been proposed for optimizing modularity [17, 26]. In this paper, we will employ the fast unfolding algorithm [4] to perform clustering, as it is most efficient and performs well on graph clustering. The algorithm conducts an agglomerative clustering process on the query relation subgraph consisted of the top k recommendations as shown in Alg. 2, in which one node is merged with another to obtain the maximum gain of the modularity. The specific fast unfolding algorithm can be found in [4].

With the clusters in hand, we can label each cluster explicitly with social tags for better understanding. As we can see in Section 3.1.2, the forward transition from query to tags approximates the query's interests through search and annotation activity. Therefore, we can leverage the expected tag distribution over the URLs clicked by query q_i to

approximate q_i's interests, which is in the form

$$P_{V_t|V_q}(l|i) = \sum_{u_k \in V_u} P_{V_u|V_q}(k|i)P_{V_t|V_u}(l|k).$$

We can then approximate cluster c_j's interests by the average tag distribution over recommendations under it as follows

$$P_{V_t|C}(l|j) = \frac{1}{N_C(j)}\sum_{q_i \in c_j} P_{V_t|V_q}(l|i),$$

where $N_C(j)$ denotes the size of the j-th cluster. We choose the top ranked k' tags to label the cluster.

4. EXPERIMENTAL RESULTS

We conducted experiments to verify the effectiveness of our query recommendation approach. In this section, we first introduce the data sets and the baseline methods used in experiments. Then we present some empirical results to show the differences among these methods. Finally, we evaluate the effectiveness of our approach by comparing users' click behavior on recommendations with baseline methods.

4.1 Data Set

In our experiments, we made use of two types of data sets, i.e. query logs and social annotation data. For query logs, we used "Spring 2006 Data Asset" distributed by Microsoft Research[1]. This query log contains about 15 million records that were sampled over one month in May, 2006. The queries were processed via the following normalization steps (i) trimming of each query, (ii) converting letters into lower case, and (iii) space sequence reduction to one space character. Queries and corresponding click-through data containing adult content were filtered out. Finally, we obtained about 2.7 million unique queries and about 4.2 million unique URLs. For social annotation data, we collected a sample of Delicious[2] data during October and November, 2008. The data set consists of over 167 million taggings, which include about 0.83 million unique users, 57.8 unique URLs and 5.9 million unique tags. We disregarded user information, merged all the tags of the same URL together, and normalized each tag into lower case.

We constructed a Query-URL-Tag tripartite graph based on these two data sets by connecting queries and tags through URLs. Here we disregarded all the URLs that are not associated with either queries or tags. In this way, we obtained a tripartite graph with over 1.3 million nodes and 3.8 million edges. Table 2 gives some specific statistical numbers of the graph. Based on the tripartite graph, we finally obtained a query relation graph with $538,547$ query nodes. For experiments, we randomly sampled 300 queries for testing, and evaluated the quality of top 15 recommendations for each test query with three groups of human judges.

4.2 Baseline Methods

We implemented a query recommendation system proposed by Mei et al. [25], which is based on the basic query formulation model and only leverages query logs (i.e., Query-URL bipartite graph) to construct a one-mode query relation graph. They employed hitting time from other queries to the current query as the rank of recommendations. In later parts, we will refer their method as *BiHit*.

[1]http://research.microsoft.com/users/nickcr/wscd09/
[2]http://del.icio.us

Table 2: Statistics of the Tripartite Graph.

Type	Number
Query Nodes	538,547
URL Nodes	356,943
Tag Nodes	433,807
Query-URL Edges	682,027
URL-Tag Edges	3,174,940

Another baseline method is a list-based approach to query recommendation considering both search and exploratory interests. That is, we constructed the query relation graph based on the new formulation model, employed our proposed hitting time algorithm to rank queries, and presented the top k recommendations in a simple flat list. Since the query relation graph here is obtained based on the Query-URL-Tag tripartite graph, we will refer this method as *TriList*, and refer our structured approach as *TriStructure*.

4.3 Examples of Recommendation Results

We present the comparison of recommendations generated by our approach and baseline methods. Table 3 shows two samples from our test queries including their top 15 recommendations generated by three methods.

We can see from Table 3, the recommendations generated by BiHit method closely stick to users' search interests. For example, all the recommendations for "espn" limit to equivalent expressions (e.g., "espn go" and "espn sports") and sub-concepts of "espn" (e.g., "espn mlb" and "espn radio"). It can help users to clarify their search intent, i.e., what they want to know about ESPN.

Both the TriList and TriStructure methods can bring in some recommendations with diverse topics which are related but not close to user's search queries. For example for query "24", the recommendations include entertainment websites like "tv guide" and tv series like "grey's anatomy". Most of them would be interesting to the searchers. It confirms that we can actually infer some exploratory interests from social annotation data which are leveraged in both TriList and TriStructure methods.

From Table 3, we can also easily see that TriStructure method provides a clear view of the recommendation results and can not only help users refine their queries but also stimulate their potential exploratory interests. For example, when searching query "24", the user can easily refine their queries with the recommendations in the first cluster. Meanwhile, he can also learn about, for example "grey's anatomy", is another hot tv series, and thus probably make a click on the recommendation. The TriList method mixes all the recommendations in a simple flat list which may not be easy for users to read.

However, we also noticed that the tag labels chosen for a cluster in TriStructure method may sometimes not be so appealing as they have duplicated semantics. For example, "tv" and "television" both appeared as labels for a cluster for query "24"; "webmail", "email" and "mail" co-occurred as labels for a cluster for query "yahoo mail". This is due to that we directly leverage the expected tag distribution over the recommendations under a cluster to approximate the cluster's interests, and choose the top ranked tags to label it. We can further apply some tag selection approaches to alleviate this problem.

624

Table 3: Comparisons on Recommendation Results of Three Methods.

Query = espn			Query = 24		
BiHit	TriList	TriStructure	BiHit	TriList	TriStructure
espn magazine	espn radio	[sports espn news]	24 season 5	fox 24	[tv 24 entertainment]
espn go	espn news	espn radio	24 series	kiefer sutherland	fox 24
espn news	yahoo sports	espn news	24 on fox	tv guide	kiefer sutherland
espn sports	nba news	espn nba	24 fox	24 tv show	24 tv show
esonsports	cbs sportsline	espn mlb	fox 24	24 fox	24 fox
baseball news espn	espn nba	espn sports	24 tv show	jack bauer	jack bauer
espn mlb	sports	bill simmons	tv show 24	grey's anatomy	24 on fox
sports news	espn mlb	[sports news scores]	24 hour	24 on fox	24 spoilers
espn radio	espn sports	yahoo sports	fox television network	desperate housewives	[tv televisions entertainment]
espn 103.3	sporting news	nba news	fox broadcasting	prison break	tv guide
espn cell phone	scout	cbs sportsline	fox tv	24 spoilers	abc
espn baseball	sportsline	sports	fox sports net	abc	tv listings
sports	sports illustrated	sporting news	fox sport	tv listings	fox
mobile espn	bill simmons	scout	ktvi 2	fox	[tv television series]
espn hockey	fox sports	sportsline	fox five news	one tree hill	grey's anatomy
		sports illustrated			desperate housewives
		fox sports			prison break
					one tree hill

4.4 Evaluation of Recommendation Results

In this experiment, we compare the performances of different recommendation methods by users' click behavior. We created a label tool to simulate the real search scenario as shown in Figure 4. For each test query, the human judge is presented with the search results of the query from a commercial search engine, and the recommendations generated by one method. We ask the human judge to label for each recommendation how likely he would like to click it *given the search result list*. A 6-point scale (0, 0.2, 0.4, 0.6, 0.8, and 1) is defined to measure the click willingness, in which 0 means "totally unwilling to click it" while 1 indicates "absolutely would like to click it". Note that this evaluation method is largely different from traditional evaluation for query recommendation where judges are required to label the similarity or relevance between recommendations and the original query [24, 8]. We believe our evaluation method can better approximate the real search scenario where users check recommendations after seeing the search results. Besides, increasing the number of clicks on recommendations can also be considered as an essential goal of query recommendations.

We asked 9 judges with or without computer science background to label the recommendations. Specifically, the 300 queries were randomly divided into 3 sets with 100 each, named QuerySet1, QuerySet2, and QuerySet3. We also randomly grouped the 9 judges into 3 sets with 3 judges each, named JudgeGroup1, JudgeGroup2, JudgeGroup3. Our labeling procedure was then conducted by 3 rounds as shown in Table 4, where each judge group labeled the recommendations from QuerySet1 to QuerySet3 successively. In this way, we can reduce the bias of judges as (1) each method's recommendations on each query will be labeled by three judges; (2) each method will be labeled by all the judges (on different query sets); (3) each query will be labeled by all the judges (on different methods). Moreover, no judge will label the recommendations for a same query generated by different methods successively, thus we can reduce the bias arose by the labeling order of different methods.

4.4.1 Overall Performance

We made the evaluation at query level and used the *Clicked Recommendation Number (CRN)*, *Clicked Recommendation Score (CRS)*, and *Total Recommendation Score (TRS)* as

Table 4: Labeling Procedure.

	BiHit	TriList	TriStructure
JudgeGroup1	QuerySet1	QuerySet2	QuerySet3
JudgeGroup2	QuerySet3	QuerySet1	QuerySet2
JudgeGroup3	QuerySet2	QuerySet3	QuerySet1

evaluation measures. Given a query q, let $R = \{r_1, \ldots, r_k\}$ denote the k recommendations generated by a certain approach (note $k = 15$ in our experiments), and $L = \{l_1, \ldots, l_k\}$ denote the corresponding label scores on these recommendations. Here we define a non-zero label score on a recommendation as a *click* on it. That is, if $l_i > 0$ then the i-th recommendation for query q receives a click with the willingness of l_i. The three measures for a query q are then defined as follows

$$
\begin{aligned}
CRN_q &= |\{r_i | l_i > 0, i \in [1, k]\}|, \\
CRS_q &= \frac{\sum_{i=1}^{k} l_i}{CRN_q}, \\
TRS_q &= \frac{\sum_{i=1}^{k} l_i}{k},
\end{aligned}
$$

where $| * |$ denotes the size of a set. For each query q, we calculate the three measures for each human judge's label results and average over them to obtain its evaluation results. We then average over all the queries to obtain the final performance of each method.

Table 5 shows the final evaluation results of the three methods. The numbers in the parentheses are the relative improvements compared with BiHit method. From the results in Table 5, we can see that TriList method can outperform BiHit method in terms of all measures while TriStructure method performs best. It shows that by recommending queries with both search and exploratory interests, we can improve both the click rate and click willingness on recommendations. Moreover, by using a structured approach, the improvements become even more significant. When compared with BiHit method, the relative improvements obtained by TriStructure method are about 46.5%, 19.7% and 63.6% in terms of average CRN, average CRS and average TRS, respectively. We conducted T-Tests on the results between each pair of methods, which indicates that all these improvements are statistically significant (p-value < 0.01).
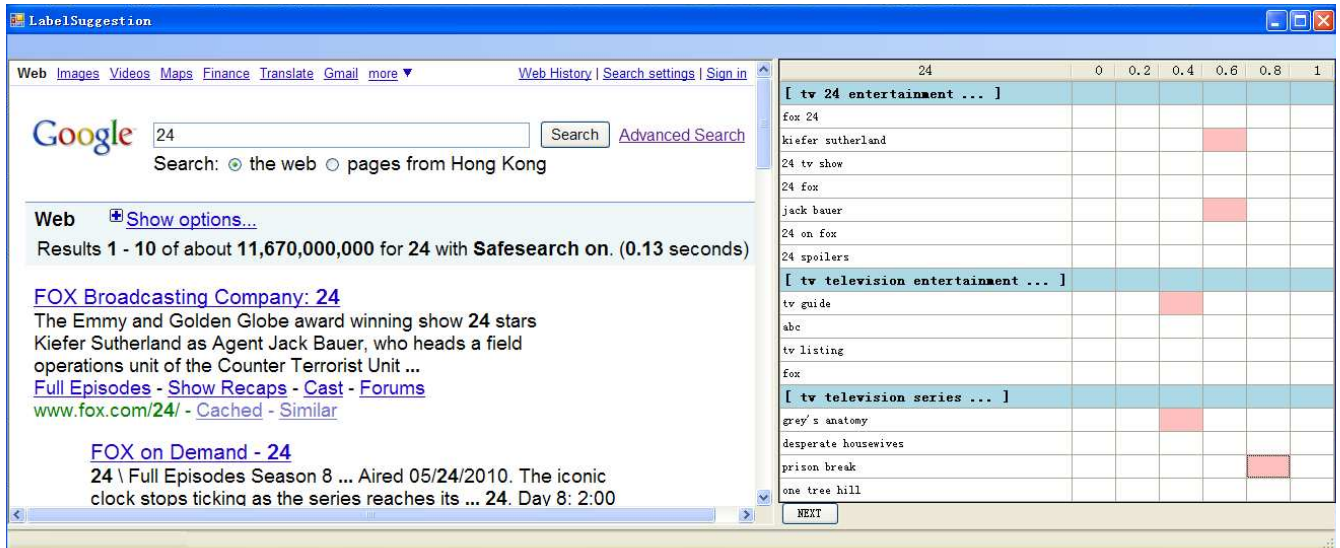
Figure 4: Our Label Tool for Query Recommendation. The left panel shows the search results of the given query from a commercial search engine (i.e. Google). The right panel contains the query recommendations generated by a certain approach (i.e. the structured approach in this case) and the 6-point scale labeling area. The grids with pink color represent the labels on the corresponding recommendations.

Table 5: Comparisons between Our Approach and Baselines on Click Performance by Query in terms of Average Clicked Recommendation Number (Ave. CRN), Average Clicked Recommendation Score (Ave. CRS), and Average Total Recommendation Score (Ave. TRS)

	BiHit	TriList	TriStructure
Ave. CRN	5.594	6.458 (+15.4%)	8.186 (+46.3%)
Ave. CRS	0.331	0.366 (+10.6%)	0.395 (+19.3%)
Ave. TRS	0.129	0.158 (+22.5%)	0.211 (+63.6%)

Table 6: Distributions of Labeled Score over Recommendations under Different Methods (%).

	0.0	0.2	0.4	0.6	0.8	1.0
BiHit	62.70	17.88	11.26	6.42	1.67	0.07
TriList	56.95	18.28	13.48	8.73	2.45	0.11
TriStructure	45.43	20.41	18.04	12.09	3.82	0.21

Table 6 further shows the detailed distributions of label scores on recommendations under different methods. From the results, we can see that both TriList and TriStructure methods receive more labels at each specific non-zero score level than BiHit method. In other words, we can attract more user clicks on recommendations at different levels by considering exploratory interests. Among the results, although there are not many recommendations which users absolutely would like to click (i.e., score level "1"), the number of such recommendations under Tristructure method reaches about 3 times of that under BiHit method. Moreover, we can see that TriStructure method obtains consistent improvements at all non-zero score levels over TriList method although they both contain the same recommendations for each query. It further demonstrates the effectiveness of structured approach for query recommendation addressing both search and exploratory interests.

4.4.2 How Structure Helps

The above experiments study the overall performance of the three methods and show the effectiveness of our approach TriStructure for query recommendation. An interesting result is that the structured method TriStructure can significantly outperform the list-based method TriList even

though they recommend a same set of queries for each input query. Here we conducted more analysis over these two methods to study how the structure affects users' click behavior on recommendations. That is, we want to figure out the changes of users' click pattern and click willingness on *each recommendation* for a given query under these two methods. In this way, we can show the benefits of structured approach for recommending queries with both search and exploratory interests.

We first analyzed how the structured approach affects users' clicks. Note here we only focus on the clicks on the recommendations (i.e. non-zero labels) but not care about the specific label scores. We compared the click entropy of these two methods, which measures how users' clicks distribute over the clusters of recommendations. The click entropy on a single query is defined as follows

$$E = -P_C(i) \log P_C(i),$$

where $P_C(i) = click_C(i)/\sum_j click_C(j)$ and $click_C(i)$ is the number of clicks in the i-th cluster of the given query's recommendations. For TriStructure method, the calculation of click entropy is quite direct. For TriList method, since its recommendations for each query are exactly the same as that of TriStructure method, we can map the clicks on TriList's recommendations into the corresponding clusters of TriStructure method and thus obtain the click entropy on each query. We show the average click entropy over queries under the two methods in Fig. 5. Here we arrange the queries
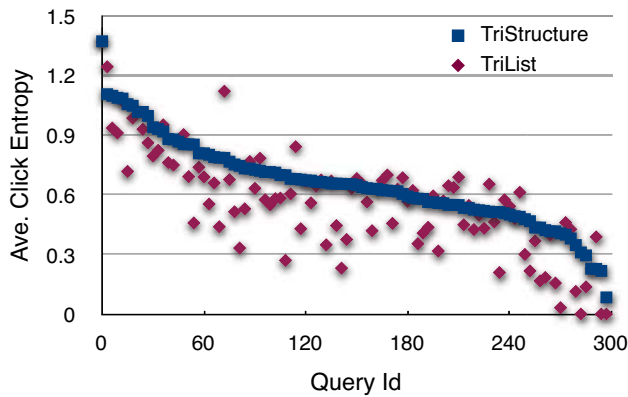
Figure 5: The Average Click Entropy over Queries under the TriList and TriStructure Methods.

by a descent order according to their average click entropy under TriStructrure method for a clear look. As we can see from Fig. 5, the average click entropy under TriStructure method is higher than TriList method on most queries, and the improvement is statistically significant under T-Test (p-value < 0.01). A higher click entropy means that users' clicks are more scattered among clusters. Therefore, it indicates that without clustering, users' clicks may not only decrease but also concentrate on certain interests. While with the structured approach where recommendations are well organized by clusters, we will be able to encourage users to click more recommendations in different clusters.

We further analyzed how the structured approach affects users' click willingness. We compared the average label scores on the same recommendation for a query under the two methods. The correlation between the two scores over all the queries is shown in Fig. 6. The x-axis and y-axis denote the average label scores of the same recommendation for a query under the TriList and TriStructure methods respectively, and the color represents the number of such recommendations. Here we focus on the score range between 0 and 0.6, since 96.58% of recommendations' average label scores are within this area. From Fig. 6 we can see that more points (around 75.2%) are close to the y-axis in the graph. It shows that for the same recommendation of a query, more of them will receive a higher label score under TriStructure method than under TriList method. That is, the willingness for users to click a same recommendation will be raised under TriStructure method. This indicates that with the structured approach, especially with the tags labeling each cluster explicitly, we can effectively stimulate user' exploratory interests even before reading any search result.

5. RELATED WORK

Query recommendation has been employed as a core utility by many industrial search engines. The general idea of query recommendation is similar to that of query expansion [29, 14], query substitution [20, 1] and query refinement [22, 18], which all focus on transforming an initial query into a "better" query to help users search. We deviate from these traditional approaches which mainly focus on users' search interests by addressing both search and exploratory interests
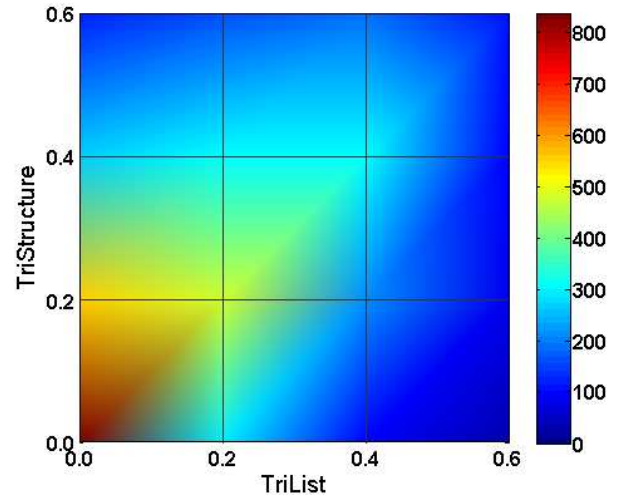

Figure 6: Correlation between the Average Label Scores on Same Recommendations for Queries under the TriList and TriStructure Methods.

simultaneously with a structured approach to query recommendation.

While most early query recommendation methods explore document information, query log data has been widely used recently. Query click-through [3, 2, 25], search query sessions [20, 8] and query frequency [23, 11] are among the most used types of information in query logs. In this paper, we further introduce the social annotation data for recommendation, a resource which mainly reflects exploratory interests on URLs with the "wisdom of crowds".

Most of the work on query recommendation is focused on measures of query similarity. Some of these studies can be categorized as cluster-based approaches. For example, Beeferman et al. [3] applied a hierarchical agglomerative method to obtain similar queries in an iterative way. Baeza-Yates et al. [2] used the k-means algorithm to derive similar queries, but the specific number of clusters is often difficult for clustering search logs. Wen et al. [28] proposed a clustering method for query recommendation that combine query content and click-through information. In [8], Cao et al. conducted context-aware query recommendation by clustering queries into concepts and mining the concept sequence patterns.

Recently, there are different query recommendation methods proposed based on random walks on a query graph. Boldi et al. [5] proposed recommending queries based on short random walks on the query-flow graph. Both PageRank [7] and personalized PageRank [19] values are used in their approach. Mei et al. [25] described a random walk on the one-mode query graph and employed hitting time to recommend queries, which is closely related to our work. However, their approach only relies on query logs and computes the hitting time from recommendations to the original query.

6. CONCLUSIONS

In this paper, we propose to recommend queries in a structured way for better satisfying both search and exploratory

627

interests of users. We introduce the social annotation data as an important resource for such recommendation. Specifically, we construct a query relation graph from query logs and social annotation data, which captures the relationships between queries considering the two types of interests. Based on the query relation graph, we employ hitting time to rank possible recommendations, leverage a modularity based approach to group top recommendations into clusters, and label each cluster with tags. Empirical experimental results indicate that our structured approach to query recommendation with social annotation data can better satisfy users interests and significantly enhance user's click behavior on recommendations.

When recommending queries in a structured way, we will face some alternative options. For example, we can show more clusters with less recommendations presented in each to bring more diversity in recommendation. Alternatively, we can show less clusters with more recommendations presented in each to concentrate on the major search and exploratory interests. There might be some trade-off between the two cases which is worth of further study.

Moreover, the social annotated data is still very sparse comparing with query logs. Many URLs clicked by long tail queries might not be annotated yet, and thus the recommendation for these queries is difficult with our approach as usual. One possible solution is to propagate the tags with the query logs or hyperlinks between URLs, so that we can enlarge the size of annotated data. This might be another interesting topic for our future work.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *WWW '01*, pages 169–178, New York, NY, USA, 2001. ACM.

[2] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology - EDBT 2004 Workshops*, pages 588–596, 2005.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00*, pages 407–416, New York, NY, USA, 2000. ACM.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, page P10008, 2008.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD '09*, pages 56–63, New York, NY, USA, 2009. ACM.

[6] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *11th Europ. Symp. Algorithms*, pages 568–579, 2003.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[8] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD '08*, pages 875–883, New York, NY, USA, 2008. ACM.

[9] M. J. Carman, M. Baillie, R. Gwadera, and F. Crestani. A statistical comparison of tag and query logs. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, New York, NY, USA, 2009. ACM.

[10] X. Q. Cheng and H. W. Shen. Uncovering the community structure associated with the diffusion dynamics on networks. *Journal of Statistical Mechanics*, page P04024, 2010.

[11] K. W. Church and B. Thiesson. The wild thing goes local. In *SIGIR '07*, pages 901–901, New York, NY, USA, 2007. ACM.

[12] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR '07*, pages 239–246, New York, NY, USA, 2007. ACM.

[13] B. Cronin. Assessing user needs. *Aslib Proceedings*, 33(2):37–47, 1981.

[14] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839, 2003.

[15] P. G. Doyle and L. J. Snell. *Random Walks and Electric Networks*. Math. Ass. of America, 1984.

[16] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *WWW '08*, pages 61–70, New York, NY, USA, 2008. ACM.

[17] R. Guimera and L. A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, February 2005.

[18] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR '08*, pages 379–386, New York, NY, USA, 2008. ACM.

[19] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03*, pages 271–279, New York, NY, USA, 2003. ACM.

[20] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, pages 387–396, New York, NY, USA, 2006. ACM.

[21] M. S. Khan and S. Khor. Enhanced web document retrieval using automatic query expansion. *J. Am. Soc. Inf. Sci. Technol.*, 55(1):29–40, 2004.

[22] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW '04*, pages 666–674, New York, NY, USA, 2004. ACM.

[23] M. Li, Y. Zhang, M. Zhu, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *ACL-44*, pages 1025–1032, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[24] H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM '08*, pages 709–718, New York, NY, USA, 2008. ACM.

[25] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08*, pages 469–478, New York, NY, USA, 2008. ACM.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[27] H. W. Shen, X. Q. Cheng, K. Cai, and M. B. Hu. Detect overlapping and hierarchical community structure in networks. *Physica A*, 388(8):1706 – 1712, 2009.

[28] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *WWW '01*, pages 162–168, New York, NY, USA, 2001. ACM.

[29] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96*, pages 4–11, New York, NY, USA, 1996. ACM.