Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 956468, 14 pages
http://dx.doi.org/10.1155/2015/956468

Research Article

Recommending High Utility Queries via Query-Reformulation Graph

JianGuo Wang,[1,2] Joshua Zhexue Huang,[1,3] and Dingming Wu[3]

[1]*Shenzhen Key Laboratory of High Performance Data Mining, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China*
[2]*Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen 518055, China*
[3]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China*

Correspondence should be addressed to Dingming Wu; dmwu@cs.hku.hk

Received 11 December 2014; Revised 29 March 2015; Accepted 3 April 2015

Academic Editor: Chaudry Masood Khalique

Query recommendation is an essential part of modern search engine which aims at helping users find useful information. Existing query recommendation methods all focus on recommending similar queries to the users. However, the main problem of these similarity-based approaches is that even some very similar queries may return few or even no useful search results, while other less similar queries may return more useful search results, especially when the initial query does not reflect user's search intent correctly. Therefore, we propose recommending high utility queries, that is, useful queries with more relevant documents, rather than similar ones. In this paper, we first construct a query-reformulation graph that consists of query nodes, satisfactory document nodes, and interruption node. Then, we apply an absorbing random walk on the query-reformulation graph and model the document utility with the transition probability from initial query to the satisfactory document. At last, we propagate the document utilities back to queries and rank candidate queries with their utilities for recommendation. Extensive experiments were conducted on real query logs, and the experimental results have shown that our method significantly outperformed the state-of-the-art methods in recommending high utility queries.

1. Introduction

Search engines such as Google, Yahoo!, and Bing have become ubiquitous tools for people to find information from the Web. However, since it is difficult for the users to formulate proper queries from which they can find useful search results, they always reformulate current queries several times for better search results until feeling satisfied. Therefore, to help the users find useful search results quickly, given an initial query q, query recommendation technology recommends some related queries for it and displays them as hyperlinks at the bottom (for Google and Yahoo!) or left side (for Bing) in q's search results page.

Most existing query recommendation approaches are similarity-based. Given an initial query q, similarity-based methods rank its candidate queries $\{q_1, q_2, \ldots, q_m\}$ by similarity function $S(q, q_i)$ $(1 \le i \le m)$ and recommend the top $k(< m)$ most similar queries to the user. $S(q, q_i)$ measures the similarity between initial query q and its candidate query q_i, where S is computed from the log data of queries q and q_i. Different log data are used to compute the similarity function S, such as common clicked URLs [1–3] and adjacent queries in the same search session [4–6]. The basic assumption of similarity-based approaches is that the similar recommended queries can return useful search results.

However, the main problem of these similarity-based approaches is that even some very similar queries may return few or even no useful search results, while other less similar queries may return more useful search results, especially when the initial query does not reflect user's search intent correctly. For example, given an initial query "iphone available time market" which tends to find "what's the time of iphone to sell on the market," the candidate recommendations may include "iphone market sale time,"

"iphone start selling market," and "iphone release date." Obviously, the three queries are all similar to the user's initial query, especially the former two queries which are all similar ones in terms of textual similarity, but the search results show that the last one can find more useful information that satisfies users' information needs. This example shows that the similar queries are perhaps not useful ones, and therefore we should recommend useful queries, that is, high utility queries with useful search results rather than similar queries.

Therefore, in this paper, we propose a novel query recommendation approach that aims at recommending high utility queries via a *query-reformulation graph*. We first build a query-reformulation graph that consists of query nodes, satisfactory document nodes, and *an interruption* node, by mining the historical query log data. The query-reformulation graph can capture users' behaviors of reformulating queries, clicking search results and feeling satisfied with them, and interrupting their search tasks without any clicks, at the same time. Then we employ *an absorbing* random walk starting from the initial query on the query-reformulation graph. In this phase, we define query nodes as transit states, satisfactory document, and interruption node as absorbing states, and model the document utility with the transition probability from initial query node to the satisfactory document node. At last, we propagate the document utility back to the queries and rank the candidate queries with their utilities for recommendation. In this way, the users can find more useful search results from the recommendations generated by our approach via query-reformulation graph.

Extensive experiments have been conducted on a real query log data to evaluate our approach. We compared our approach with other six baselines in recommending high utility queries. The experimental results have shown that the recommendations generated by our approach could return more relevant documents than those generated by the state-of-the-art methods in comparison. We also evaluated the document utility and its experimental results have demonstrated that the documents generated by our approach were more relevant to the initial query than those generated by other baselines.

The remainder of this paper is organized as follows. Section 2 gives a brief review on related work. Section 3 introduces the query-reformulation graph. Section 4 describes the proposed query recommendation approach in detail. Section 5 shows the experimental results. Conclusions are made in Section 6.

2. Related Work

In this section, we review the research topic which is related to our work: query recommendation. According to the principles of different approaches, we will divide them into two categories: similarity-based query recommendation and utility-based query recommendation.

2.1. Similarity-Based Query Recommendation. Most existing query recommendation approaches are similarity-based. Given an initial query q, they commonly rank the candidate queries set $\{q_1, q_2, \ldots, q_m\}$ of initial query q by a similarity function $S(q, q_i)$, where S is computed from the query log data of q and q_i. Different log data have been used to compute S.

The clicked URLs in query log data were first used to compute the similarity between two queries. A query-URL bipartite graph was created from the URLs in query log data and then used to compute the similarities between queries. Reference [1] used an agglomerative clustering algorithm to cluster queries and find similar queries for recommendation. Reference [7] applied two types of random walk process to propagate the query similarity along the query-URL bipartite graph and obtained better similarity scores between queries. Reference [2] folded the query-URL bipartite graph to an affine graph and applied the HAC-based ranking method to recommend similar queries. Reference [8] transformed the undirected query-URL bipartite graph into a directed one and applied a random walk to find queries similar to the initial query. Instead of random walk, [3] used heat diffusion to model similar information propagation on the directed query-URL bipartite graph for query recommendation.

The log data of search sessions were also used to compute the similarity between two queries. A search session is the sequence of queries issued by the same user in a given time period. Reference [9] represented search sessions as transactions of queries and applied association rule mining algorithms to find associated queries for recommendation. Reference [10] represented each query as a vector of search sessions and each search session index recorded the frequency of the query that occurred in that search session. The similarity between two queries was computed from the two query vectors. Reference [11] proposed using a Mixture Variable Memory Markov model built from search sessions to predict the next query to be selected by the user given the current search session.

From the adjacent queries in search sessions, [4, 5] built a query-flow graph and applied a random walk method, starting from the initial query, to find similarities between queries. Reference [12] proposed a method to project a large query-flow graph to a low dimension space to reduce the similarity computation between queries.

The key words in queries were also used to compute similarity between two queries. Reference [13] used the LDA algorithm [14] to build topic models from training log data and computed the similarity between queries on topic features. Sparsity and diversity of key words in queries present a big challenge to the LDA method. Integration of query features from multiple data sources was investigated to solve the sparsity problem. Reference [15] combined the URL features and user IDs extracted from search sessions to construct a directed weighted query-URL bipartite graph and then used heat diffusion to model the similar information propagation. Reference [16] investigated a method to compose a recommended query in a natural language from the features of search sessions and query terms. Reference [17] proposed a context-aware approach exploiting the features of URL and search sessions.

2.2. Utility-Based Query Recommendation. Recently, several research works [6, 18–20] proposed recommending high utility queries to users. Both studies [6, 18] defined a

global utility function over the recommended queries set, which emphasize either the diversity [18] or the expected click-through rate [6] of the recommendations. But, they did not define and learn the query utility toward users' postclick experience as ours. Reference [19] employed a query utility mining (QUM) model to mine query utility from users' search behaviors. This approach differs from ours mainly in two aspects: (1) QUM models the search sessions with a dynamic Bayesian model and considers the search session that ends with a query with clicks to be satisfactory one. If a query occurs in a satisfactory search session, its utility is high. In contrast, in our work, we estimate satisfaction for each query; (2) QUM does not use the information of specific clicked documents and infer query utility directly. To avoid this disadvantage, query-reformulation graph models the specific clicked documents to infer document utility and then calculates query utility from document utility indirectly.

The paper that is most related to our work is [20]. Reference [20] constructed a session-flow graph and leveraged an absorbing random walk to infer query utility. However, our query-reformulation graph is different from session-flow graph as follows.

2.2.1. Nodes Are Different. Our query-reformulation graph consists of query nodes, satisfactory documents nodes, and an interruption node while session-flow graph consists of query nodes, documents nodes, and failure nodes.

Satisfactory Document Node versus Document Node. We evaluate the satisfaction status for each query in query log with features of reformulation and clicks. If a query with clicks has no reformulation, we consider that the user feels satisfied with it and define its clicked documents as *satisfactory documents*. The rest clicked documents are defined as dissatisfactory documents. We consider that satisfactory documents which can make users feel satisfied are of high utility; therefore, only satisfactory documents are used in building query-reformulation graph. In contrast, [20] did not evaluate satisfaction status and all the clicked documents were used in building session-flow graph. Since the walkers in the process of random walk would be absorbed by dissatisfactory documents, session-flow graph cannot infer the document utility accurately. This is the key reason that our approach can perform better than [20].

Interruption Node versus Failure Nodes. We introduce an interruption node into query-reformulation graph to capture users' interruption behaviors. If a query has no reformulation and no clicks, we consider that the user has interrupted his/her search task at that query, and we link corresponding query node to the interruption node. Since a user can interrupt his/her search task at every query, all query nodes may be linked to the interruption node. In contrast, Zhu et al. used failure nodes in session-flow graph to capture the users' failure behaviors and considered the end of search session as the failure of search task. Thus, only the nodes of queries that were formulated at the end of search session can be linked to failure nodes. However, a user could fail his/her search task anywhere. For example, a user may fail his search task inside a search session but continue it by reformulating next query. Therefore, the failure node used session-flow graph cannot capture the users' failure behaviors accurately.

2.2.2. The Edges from Query Nodes to Query Nodes Are Different. In query-reformulation graph, query 1 node is linked to query 2 node if query 2 is reformulated by any user for a same search intent after formulating query 1, while in session-flow graph, query 1 node is linked to query 2 node if query 1 is followed by query 2 in any search session. Since query 1 node and query 2 node in the session-flow graph may originate from different search intents, the "walker" in the process of random walk will go into another topic.

2.2.3. The Probabilities of Transition Are Different. We use the features of reformulation and click to estimate the user behaviors at each query. Specifically, according to different combinations of reformulation and click, we classify the user behaviors at distinct query q into three types of those who feel satisfied and stop the search, feel dissatisfied and reformulate next query, feel dissatisfied and interrupt the search. The three types of user behavior correspond to three types of transition in query-reformulation graph, respectively: the transition from query node q to satisfactory document nodes, to other query nodes, and to the interruption node. Then, the transition probabilities from query q node to other nodes, that is, reformulation query nodes, satisfactory document nodes, and the interruption node, are calculated from the ratios of specific combination of reformulation and click at query q. In contrast, in session-flow graph, [20] set the transition probabilities from a query q node to other nodes as constant values. Therefore, the query-reformulation graph can capture user behaviors much better than session-flow graph and thus performs better than it in inferring query utility.

3. Query-Reformulation Graph

In this section, we provide the basic idea behind the query-reformulation graph. We first give the definitions of query reformulation and query utility.

Definition 1. Query reformulation is the act of submitting a next query q' to modify the previous query q in hope of retrieving better search results.

Definition 2. Query utility is defined as the useful information that the user can obtain from the clicked search results.

To capture query utility which cannot be computed directly, we first predict the query satisfaction. The basic idea is that if users have obtained some useful information from query q's search results, then they are satisfied; otherwise they are dissatisfied.

The features of click have been widely used in predicting search task satisfaction. The satisfied users more often click search results than dissatisfied users. Fox et al. [21] learned a Bayesian model to predict user's satisfaction and find a strong relationship between user's satisfaction and the

Table 1: The satisfaction status, user behavior, and document type according to reformulation and clicks.

Reformulation	Clicks	Satisfaction	User behavior	Document type
No	Yes	Satisfied	Stop	Satisfactory documents
Yes	Yes	Dissatisfied	Reformulate	Dissatisfactory documents
Yes	No			
No	No		Interrupt	

Figure 1: The reformulated query is evidence of dissatisfaction even though the search results of initial query received some clicks.

number of clicked search results in search task. Hassan et al. [22] trained two *Markov models*, describing user behavior in case of satisfied and dissatisfied search task, respectively. The transition probability from the action of "query" to the action of "click search results" in satisfied model is twice as large as that in unsatisfied model. In other words, users are nearly twice as likely to click search results after issuing a query in the satisfied search task compared to unsatisfied one. Ageev et al. [23] asked 200 users to find answers to questions using search engine and made a comparison on the user behavioral features for different user success levels. The comparison results showed that the users with high success level, who submitted correct answers to more than 80% questions, clicked more search results in the search tasks than the users with low success level who submitted correct answers to fewer than 50% of questions.

However, *click feature alone is not enough* to indicate that a query is satisfied. The reformulation is also important. For example in Figure 1, a user submitted the query "greenfield, mn accident" and was looking for information about an accident that took place in Greenfield, Minnesota. Even though the user has clicked *a search result from 2010*, he reformulated a second query "woman dies in a fatal accident in greenfield, minnesota" and clicked another search result from 2012. A likely interpretation of this is that the user was looking for the 2012 accident and failed to find it on the first query. If we only consider the first query and its click, we might have made a mistake that the user was satisfied. Besides, extensive experiments in previous work [24] have also demonstrated the importance of reformulation in predicting query satisfaction. The experimental results in [24] have shown that a classifier using both feature of reformulation and click preformed much better than that using feature of click only.

Therefore, as [24], we use the query reformulation and click as predictors of query satisfaction. Specifically, if a user clicks some search results and does not reformulate another query after submitting a query q, then he/she is satisfied with query q. According to different combination of reformulation and click, we can summarize the satisfaction status and user behavior in Table 1. As in previous literatures about satisfaction [22–25], for simplicity, we also consider that satisfaction status is binary: satisfied or dissatisfied. From Table 1, we can see that there are three types of user behavior: those who feel satisfied with the clicked search results and stop the search task, those who feel dissatisfied and reformulate the next query to continue the search task, and those who feel dissatisfied and interrupt the search task without any click.

We consider that the clicked documents that make the users feel satisfied with the queries contain useful information and we define them as *satisfactory documents*.

Definition 3. Satisfactory documents are defined as the documents clicked by the query that has no reformulation.

Accordingly, the documents clicked by the query that has reformulation are defined as *dissatisfactory documents*. The relationships between satisfactory document and other information, such as user behaviors, are shown in Table 1.

To model users' stop behaviors, reformulation behaviors, and interruption behaviors simultaneously, the query-reformulation graph is proposed, which is the central contribution in our paper. An example of query-reformulation graph is plotted in Figure 2. As shown in Figure 2, query-reformulation graph consists of three types of nodes, that is, query nodes, satisfactory document nodes, and interruption node. The query nodes represent the queries that were reformulated in search tasks, the satisfactory document nodes represent the corresponding clicked documents of satisfied queries, and the interruption node represents the situation that the users interrupt their search tasks without any click and reformulation. The query-reformulation graph is defined formally in Definition 4.

Definition 4. The query-reformulation graph is a directed graph as follows:

$$G_{qr} = \left\{Q, S, i, E_{qr}, E_{qs}, E_{qi}\right\}, \tag{1}$$

where

(i) $Q = \{q_1, q_2, \ldots, q_n\}$ is the set of distinct queries in the query log;

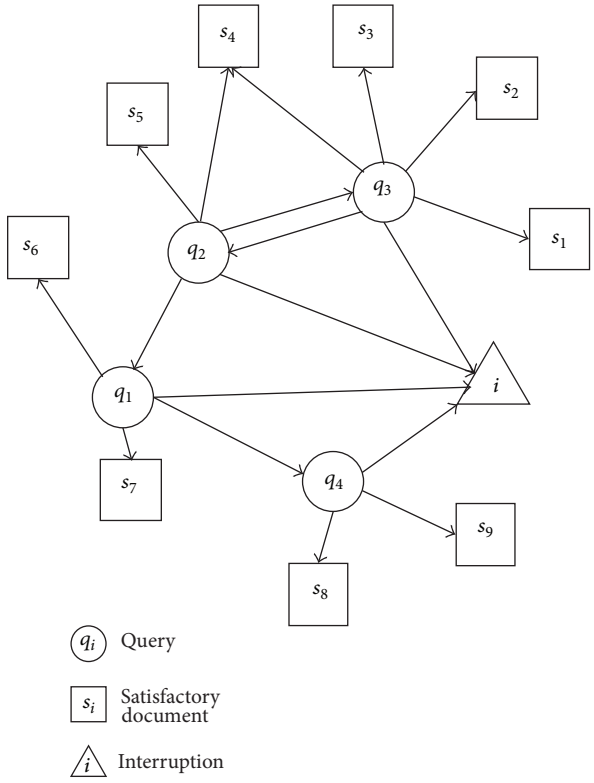

Figure 2: An example of the query-reformulation graph.

(ii) $S = \{s_1, s_2, \ldots, s_m\}$ is the set of distinct satisfactory documents;

(iii) i is the interruption node;

(iv) $E_{qr} = \{(q, q') \mid q \in Q, q' \in R(q)\}$ is the set of edges from queries to their reformulated queries, where $R(q)$ denotes the set of reformulated queries of the query q;

(v) $E_{qs} = \{(q, s) \mid q \in Q, s \in S(q)\}$ is the set of edges from queries to their satisfactory documents, where $S(q)$ is the subset of S clicked by q;

(vi) $E_{qi} = \{(q, i) \mid q \in Q\}$ is the set of edges from query nodes to the interruption node.

The weighting schemes are described in the following section.

4. Our Approach

With the definition of query-reformulation graph, we can introduce our query recommendation approach in detail. In Section 4.1, we introduce the algorithm to predict query reformulation and construct a query-reformulation graph from the query logs; in Section 4.2, we first define the transition probabilities in query-reformulation graph and then leverage absorbing random walk on it to capture the document utility. In Section 4.3, we compute the distribution of the Markov model. At last, in Section 4.4, we propagate the document utility back to the queries and rank them for query recommendation.

4.1. Predicting Query Reformulation. Query reformulation is the act of submitting a next query q' to modify the previous query q in hope of retrieving better search results. Hence, query reformulation is considered an indication of dissatisfaction with the previous query. If q' is considered to be the reformulation of q, they must be intended to satisfy the same information need. In this subsection, we propose an unsupervised rule-based method for automatically predicting whether the current query is a reformulation of the previous query.

4.1.1. Query Normalization. We preform standard normalization by (1) replacing all letters with their corresponding lower case representation; (2) replacing all runs of whitespace characters with a single space and removing any leading or trailing spaces; (3) removing all characters except for letters, Arabic numerals, and whitespace characters.

4.1.2. Matching Query. Lexical similarity between queries has been often used to identify related queries. However, the problem with it is that it brings many false negatives (e.g., synonyms and acronyms). To handle this problem, we define five levels of matching queries ranked *from the largest to the smallest similarity* as follows:

(1) Reorder match: the words in current query and previous query are the same, but their orders are different (e.g., Jordan NBA → NBA Jordan).

(2) Spelling correction match: to capture spelling variants and misspelling, we allow two queries to match if the Levenshtein edit distance between them is less than 2 (e.g., reformualtion → reformulation).

(3) Acronym match: the current query is the acronym of previous one (e.g., personal computer → pc) or vice versa.

(4) Semantic match: the semantic similarity between two terms (queries or words) can be measured based on WordNet [26]. WordNet is a large lexical database of English. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept [26]. Synsets are interlinked by means of conceptual-semantic and lexical relations. We match two terms t_i and t_j if their similarity according to the WordNet Wu and Palmer measure (wup) is greater than 0.5. The Wu and Palmer measure [27] $\text{wup}(t_i, t_j)$ is defined as

$$\text{wup}\left(t_i, t_j\right) = \frac{2 * \text{depth (LCS)}}{\text{depth}\left(t_i\right) + \text{depth}\left(t_j\right)}, \tag{2}$$

where LCS denotes the Least Common Subsumer and $\text{depth}(t_i)$ denotes the depth of synset in WordNet that contains t_i. The depth of any synset in WordNet is the length of the path that connects it to the root node plus one.

(5) Words vector match: treating query as a whole unit in computing similarity cannot cover all reformulation

situation, such as stemming. For example, "running over bridges" → "run over bridge." Therefore, we further define four levels of matching words in queries ranked from the most to the least similar as follows:

(a) Exact match: the two words are the same.

(b) Stemming match: one word is the stem of another word (e.g., run and running) and we use Porter's stemming algorithm [28] to stem every word in both queries.

(c) Substring match: one word is the substring of another word (e.g., Nevada police rec → Nevada police records 2008).

(d) Semantic match: given two words w_i and w_j, if $\text{wup}(w_i, w_j) > 0.5$, then we consider that the two words are semantic match (e.g., hand and finger).

Given two queries q and q', we try to match every pair of words (w, w') from rules (a) to (d), where $w \in q$ and $w' \in q'$, respectively. Then, we use Jaccard similarity to measure the similarity between q and q'. Consider

$$\text{wvs}\left(q, q'\right) = \frac{\text{match}\left(q, q'\right)}{\text{length}\left(q\right) + \text{length}\left(q'\right) - \text{match}\left(q, q'\right)}, \quad (3)$$

where $\text{match}(q, q')$ denotes the number of matched words in q and q' and $\text{length}(q)$ denotes the number of words in query q. If $\text{wvs}(q, q') > 0.5$, we consider that q and q' are matched.

Given current query q' and previous query q, if they are matched in any rule of (1)~(5), then we say q' is the reformulation of q.

The advantages of this rule-based query-reformulation prediction approach are that (1) it is easy to reimplement and does not need any training data and can be easily used in future work and that (2) the results of this rule-based method is better than that of state-of-the-art classifier [24] using expensive features such as search results content. We test our method on 100 pairs of labeled queries; its precision, recall, and $F1$ measure are all 86% while $F1$ measure in [24] is recorded as 81.06%.

4.2. Capturing Document Utility by Absorbing Random Walk. With the query-reformulation prediction method mentioned above, we can construct a query-reformulation graph from a real query log. We model the process that the utility score transits from initial query to satisfactory documents by applying an absorbing random walk on it and measure the document utility with the transition probability from the initial query to satisfactory document.

The random walk at a certain query node can be reasonably interpreted as follows. If the users feel satisfied with their clicked documents, they will stop their search tasks and the walkers are "absorbed" into the corresponding satisfactory documents nodes and never come out; if the users feel dissatisfied and want better search results, they will reformulate next query and the walkers walk to reformulation query nodes. (If the user reformulates a next query, either after clicking on documents or not, the user behavior is considered as reformulation); if the users feel dissatisfied and impatient, they will interrupt their search tasks without any click and reformulation, and the walkers walk into the interruption node. *These three walk schemas correspond* to the three user behaviors in Table 1: stop, reformulate, and interrupt, and absorbing random walk can describe them at the same time.

We set query nodes as transiting states and satisfactory document nodes and interruption node as absorbing states. The reasons for them can be interpreted as follows. We set the query nodes as transiting states, *because the users can reformulate* another query for better search results, click some satisfactory documents for satisfaction, or interrupt their search tasks directly, after issuing a query; we set the satisfactory document nodes as absorbing states, because when the users feel satisfied with their clicked documents, they will stop their search tasks and the utility scores are all absorbed into the clicked documents; we also set the interruption node as absorbing state, *because when the users* interrupt their search tasks, they will do nothing. In this way, the absorbing random walk can model user search process very well and make the utility score absorbed by satisfactory documents.

Based on the analysis mentioned above, we present the transition probabilities of query-reformulation graph. Let $f(q)$ be the total frequency that query q occurs in the query log; let $f_r(q)$ be the frequency that users reformulate other queries after issuing query q; let $f_s(q)$ be the frequency that users feel satisfied with their clicked documents and **s**top their search tasks after issuing query q; let $f_i(q)$ be the frequency that users interrupt their search tasks after issuing query q. Obviously, $f_r(q) + f_s(q) + f_i(q) = f(q)$.

When the user chooses to reformulate another query for better search results, the transition probability from query q to its reformulation q' is defined as the fraction of the frequency that q is reformulated by q' over the total frequency that query q occurs in the query log, formally. Consider

$$P\left(q' \mid q\right) = \frac{f_r\left(q, q'\right)}{f\left(q\right)}, \quad (4)$$

where $f_r(q, q')$ denotes the frequency that q is reformulated by q'.

When the user clicks the satisfactory document s and feels satisfied with the query q, the corresponding transition probability is formally defined as

$$P\left(s \mid q\right) = \frac{f_s\left(q\right)}{f\left(q\right)} \frac{f_s\left(q, s\right)}{\sum_{s' \in S(q)} f_s\left(q, s'\right)}, \quad (5)$$

where $S(q)$ denotes the set of distinct satisfactory documents from query q and $f_s(q, s)$ denotes the frequency that users click satisfactory document s when the users feel satisfied with query q. Specifically, $f_s(q)/f(q)$ denotes the probability that users feel satisfied with query q, and $f_s(q, s)/\sum_{s' \in S(q)} f_s(q, s')$ denotes the probability that users

click satisfactory document s when they feel satisfied with query q.

From formula (5), only satisfactory documents can be linked to query node q and absorb the utility score from query q. It is the key reason why we can find high utility documents via query-reformulation graph.

When the user interrupts his/her search task after issuing query q, the transition probability from query q to interruption node i is defined as the fraction of the frequency that users interrupt their search tasks after issuing query q over the total frequency that query q occurs in the query log, formally. Consider

$$P(i \mid q) = \frac{f_i(q)}{f(q)}. \tag{6}$$

In the existing absorbing random walk literatures [20, 29], there is a parameter to control the transition probabilities from query node to other nodes. However, in our work, there is no parameter and we use $f_r(q)/f(q)$, $f_s(q)/f(q)$, and $f_i(q)/f(q)$ that mined from query log to control the transition probabilities from query node to reformulation nodes, satisfactory document nodes, and interruption node, respectively. Given a query q and its transition probabilities, we have

$$\begin{aligned}
&\sum_{q' \in R(q)} P\left(q' \mid q\right) + \sum_{s \in S(q)} P(s \mid q) + P(i \mid q) \\
&= \sum_{q' \in R(q)} \frac{f_r\left(q, q'\right)}{f(q)} + \sum_{s \in S(q)} \frac{f_s(q)}{f(q)} \frac{f_s(q, s)}{\sum_{s' \in S(q)} f_s(q, s')} \\
&\quad + \frac{f_i(q)}{f(q)} = \frac{f_r(q)}{f(q)} + \frac{f_s(q)}{f(q)} + \frac{f_i(q)}{f(q)} \\
&= \frac{f_r(q) + f_s(q) + f_i(q)}{f(q)} = 1.
\end{aligned} \tag{7}$$

Since both satisfactory document node s and interruption node i are absorbing states, the walkers can never transit to other nodes if they reach these nodes; thus $P(s \mid s) = 1$ and $P(i \mid i) = 1$. In this way, the utility score can be absorbed by the corresponding satisfactory document nodes and interruption node. Since we model the document utility with the transition probability from the initial query to document, we introduce how to compute the distribution in the next subsection.

4.3. Computing the Distribution. Let n be the number of distinct queries in query-reformulation graph G_{qr} and let m be the number of distinct satisfactory documents in G_{qr}; thus in the stage of absorbing random walk, the corresponding matrix of transition probabilities can be represented as

$$P = \begin{bmatrix} P_Q & P_S & P_I \\ 0 & I_S & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

where P_Q is a $n \times n$ matrix of transition probabilities on query nodes, P_S is a $n \times m$ matrix of transition probabilities from query nodes to satisfactory document nodes, P_I is a $n \times 1$ matrix of transition probabilities from query nodes to interruption node, and I_S is a $m \times m$ identity matrix of transition probabilities from satisfactory document nodes to themselves.

Since the above matrix of transition probabilities is reducible, there is no stationary distribution. Therefore, we use an iterative way to compute the absorbing distribution:

$$P^t = \begin{bmatrix} P_Q^t & \sum_{k=0}^{t-1} P_Q^k P_S & \sum_{k=0}^{t-1} P_Q^k P_I \\ 0 & I_S & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

where $P^t[i, j]$ represents the probability from node i to node j after t step walk. Here we only need to compute the probability from query to satisfactory document, that is, computing the upper middle matrix $\sum_{k=0}^{t-1} P_Q^k P_S$ of P^t, and the computing complex is $O(tn^3 + n^2 m)$. Moreover, in the query recommendation scenario, we only need to compute the probability from the initial query to the satisfactory documents, that is, computing the distribution of the matrix row corresponding to the initial query. Let v (a $1 \times m$ row vector) denote the corresponding row of the initial query, thus we will compute vP_Q^{k-1} instead of P_Q^k, and the computing complex is $O(tn^2 + nm)$.

Besides, in reality, the dimension of transition probability matrix P is very high. (The query-reformulation graph constructed by us from the real query log "Microsoft 2006 RFP dataset" contains about 4.5 million distinct queries and 1.9 million distinct documents); therefore, *given an initial query* q, we first apply a wide first traverse starting from q on the whole query-reformulation graph G_{qr}. Since the number of query recommendations is relatively less and most of queries far from q in G_{qr} are not related to it, thus, when the wide first traverse has collected k ($k = 500$) query nodes, we stop the traverse progress and extract the subgraph $G_{qr}(q)$ that has been traversed. Then we compute the absorbing distribution of $G_{qr}(q)$ for q rather than that of G_{qr}. In this way, we can save lots of resources and time.

4.4. Recommending High Utility Queries. With the document utilities we have learned, we can propagate them from satisfactory document nodes to query nodes. Since the utility of a query is represented by its documents which users will click after issuing the query, an intuitive way to infer query utility is to aggregate the utilities of its satisfactory documents. Although this method is simple, the experimental results have shown that it was effective and stable. We can also use other ways to calculate query utility, for example, running a random walk on a query-documents bipartite graph [7].

With the utilities of candidate queries that computed from document utilities, we rank them in descending order of query utility, which represents the useful information they can provide to the users for satisfying their information

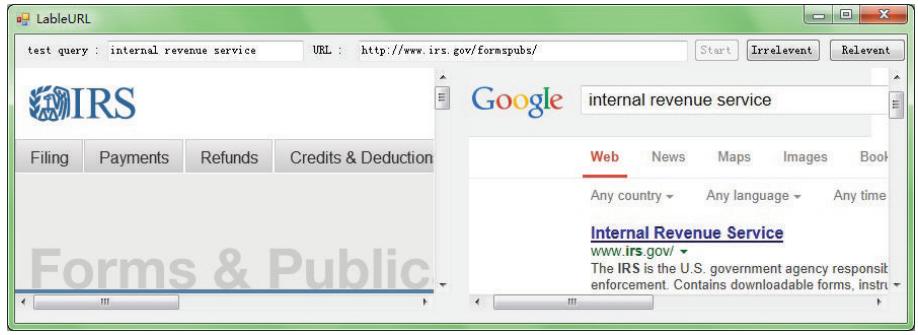

FIGURE 3: A tool for labeling URLs. The left panel shows the web document navigated by the URL. The right panel contains the search results of the test query "internal revenue service" from a commercial search engine. A human expert labels the URL by manually clicking the buttons "Relevant" or "Irrelevant."

needs. Then, we recommend top 10 queries with highest utilities to the users.

5. Experiments Results

To verify the effectiveness of our query recommendation method in recommending high utility queries, we conducted extensive experiments on a real query log and compared our method with six baselines. Furthermore, we also evaluated the learned document utility and made comparison with other two baselines.

5.1. Data Set. In our experiments, we used a real query log "Spring 2006 Data Asset" distributed by Microsoft Research (http://research.microsoft.com/users/nickcr/wscd09/). 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) remove symbols such as "?," "#," and "+," (ii) convert letters into lower case, and (iii) replace all runs of whitespace characters with a single space and remove any leading or trailing spaces. Then, we swept out the queries that occurred less than 2 times and their clicked documents. After that, we obtained about 4.5 million unique queries and about 3.3 million unique URLs. A query-reformulation graph was constructed based on the processed query log data. Finally, we got a query-reformulation graph with about 4.5 million query nodes and 1.9 million satisfactory document nodes.

5.2. Evaluation of Query Utility

5.2.1. Metrics. Query utility is defined as the useful information that the users can obtain from its clicked search results. To evaluate a query's utility, we consider the relevance of its search results. We labelled the clicked documents of recommendations as relevant or irrelevant and used the metrics of Query Relevant Ratio (QRR) and Mean Relevant Document (MRD) proposed in [19] to measure the performance of recommendations.

Given a test query q, the metric QRR is formally defined as

$$\mathrm{QRR}(r,q) = \frac{\mathrm{RQ}(r,q)}{N(r)}, \tag{10}$$

where r denotes the recommendation generated by a kind of method for the test query q; $\mathrm{RQ}(r,q)$ denotes the total frequency of recommendation r with relevant clicked search results in terms of test query q; $N(r)$ denotes the total frequency of recommendation r that occurs in query log. The metric of $\mathrm{QRR}(r,q)$ measures the probability that users can find relevant search results from recommendation r when they have reformulated q in their search tasks. The higher the QRR is, the more the users can likely find useful results from r.

Besides, given a test query q, the metric MRD is formally defined as

$$\mathrm{MRD}(r,q) = \frac{\mathrm{RD}(r,q)}{N(r)}, \tag{11}$$

where r denotes the recommendation generated by a kind of method for the test query q; $\mathrm{RD}(r,q)$ denotes the total frequency of relevant clicked search results from recommendation r in terms of test query q; $N(r)$ denotes the total frequency of recommendation r that occurs in query log. The metric of $\mathrm{MRD}(r,q)$ measures the average number of relevant search results that the users can find from recommendation r. The higher MRD is, the more useful results the users can find from r.

The metrics of QRR and MRD need the clicked search results of recommendations which are labelled as relevant or irrelevant. We created a label tool for human judges as shown in Figure 3. For each URL, the human judge is presented with the search results of the corresponding test query from a search engine and the web document navigated by the URL. We asked the human judge first to check the search results list and understand the intent of test query as far as possible and then check the web document and label it as relevant or irrelevant. We invited three human judges to label the URLs and each URL was labelled by three human judges according to the search intent of corresponding test query. For each URL, the relevance labeled more than two times was selected as the final label result. Finally, we labelled 84,744 URLs as relevant or irrelevant for 1150 test queries.

For a test query q, we evaluate the average performance of top 10 recommendations generated by a kind of method for it. One has

$$\text{average@10}\,(q) = \frac{\sum_{k=1}^{10} \text{metric}\,(r_k, q)}{10}, \tag{12}$$

where r_k denotes the kth recommendation generated by a kind of method for the test query q and metric(r_k, q) can be QRR(r_k, q) or MRD(r_k, q).

Besides, for a test query q, to consider the rank of recommendation, we evaluate the performance of top 10 recommendations with DCG [30] measure:

$$\text{DCG@10}\,(q) = \sum_{k=1}^{10} \frac{2^{\text{metric}(r_k,q)} - 1}{\log\,(k+1)}, \tag{13}$$

where r_k denotes the kth recommendation generated by a kind of method for the test query q and metric(r_k, q) can be QRR(r_k, q) or MRD(r_k, q). From the metric of DCG@10, we can see that the higher the rank of a recommendation is, the more important it is.

Since we have two metrics of QRR and MRD for single recommendation and two metrics of average@10 and DCG@10 for top 10 recommendations generated for a test query, we get 4 metrics of average@10 QRR, average@10 MRD, DCG@10 QRR, and DCG@10 MRD for top 10 recommendations generated for a test query. The higher these metrics are, the better a query recommendation method performs on a test query.

5.2.2. Baselines. To evaluate the performance of our method, we compare it with six baseline query recommendation methods. They are divided into two categories: similarity-based query recommendation and utility-based query recommendation. Details are given below.

Similarity-Based Query Recommendation

(i) Adjacency (ADJ): given a test query q, the top k frequent queries in the same search session adjacent to q are recommended to the user [31].

(ii) Cooccurrence (COO): given a test query q, the top k frequent queries cooccurred in the same search session with q are recommended to the user [10].

(iii) Query-flow graph (QFG): this was a recent query recommendation method proposed by [4]. In this method, a query-flow graph is first constructed based on the queries appearing in succession in the same search session and a random walk is performed on this graph for query recommendation.

(iv) Hitting time (HT): this method is proposed by [32] and emphasizes long tail recommendations for diversity. In this method, a query relation graph is first constructed from a query-URL bipartite graph by mining query logs. Then, a random walk is conducted on the query relation graph and the hitting time is leveraged as a measure to select queries for recommendation.

Utility-Based Query Recommendation

(i) Query Utility Model (QUM): this is a utility-based query recommendation approach proposed in [19]. QUM models the search sessions with a dynamic Bayesian model. It estimates the satisfaction of search session rather than that of query. It considers the search session that ends with a query with clicks to be satisfactory one. If a query occurs in a satisfactory search session, its utility is high. In addition, QUM does not use the information of specific clicked documents and infer query utility directly.

(ii) Session-Flow Graph (SFG): this is another utility-based method proposed in [20]. In this method, a session-flow graph, which consists of query nodes, document nodes, and failure nodes, is first constructed. Then, *an absorbing random* walk is leveraged to capture the query utility. The differences between SFG and our QRG are presented in the section of related work.

5.2.3. Evaluation Results on 100 Fully Labeled Test Queries. We first randomly selected 100 test queries and ran all seven query recommendation approaches on the whole data set to generate recommendations for them. We fully labelled the clicked search results of these recommendations as relevant or irrelevant according to their web page contents. For each test query, we calculated the four measures of average@10 MRD, average@10 QRR, DCG@10 MRD, and DCG@10 QRR for its top 10 recommendations generated by each method. We then averaged over 100 test queries to obtain the final performance of each method on one of four metrics. Finally, we got four groups of comparison and showed them in Figures 4(a), 4(b), 4(c), and 4(d), respectively.

From Figure 4, we can see that HT method performed worst under the four metrics. The reason is that HT method emphasizes the long tail recommendations to improve their diversity so that it often generates some irrelevant recommendations. The two frequency-based methods ADJ and COO also performed poorly in finding relevant search results. It shows that by simply considering the most frequently adjacent or cooccurring queries in the same search session with the initial query, (which are usually highly similar to the initial query) we cannot guarantee recommending useful queries with relevant documents. The other similarity-based method QFG showed better performance than frequency-based methods. It indicates that by modelling the users' reformulation behaviors, we are able to find better query recommendations. However, since QFG also aims at recommending similar query rather than useful query, it performed worse than the three utility-based methods QUM, SFG, and QRG.

The utility-based method QUM ranks and recommends candidate queries by their perceived utility and posterior utility. The perceived utility is defined as the probability that the query's search results are clicked by the users, while posterior utility is defined as the useful information that the users can obtain from the clicked search results. According to these definitions, QUM can recommend more useful

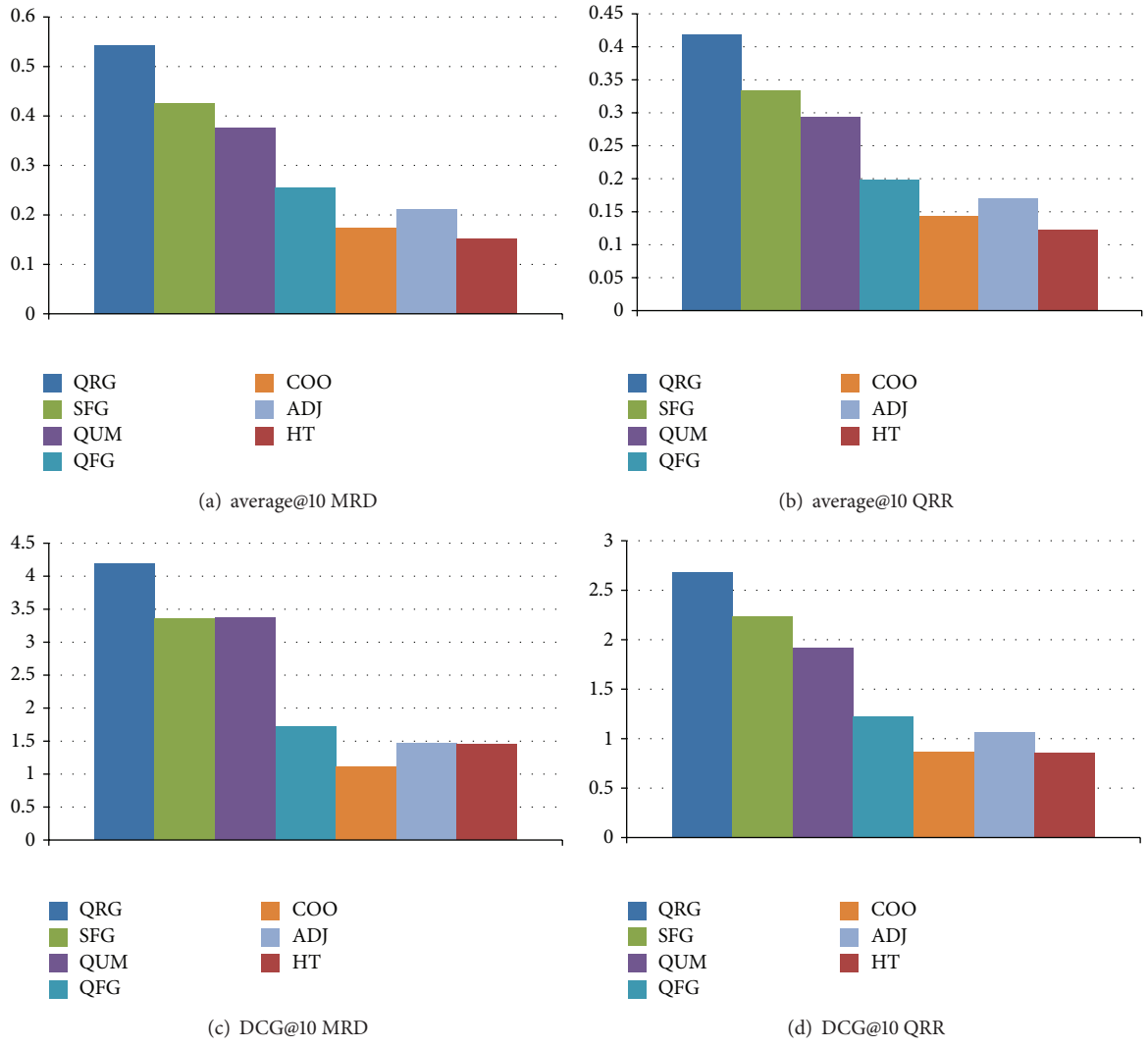

Figure 4: Evaluation results of seven query recommendation methods on 100 fully labeled test queries.

queries than those similarity-based methods. By further modelling the utility of clicked documents, another utility-based method SFG performed better than QUM.

Finally, as we can see in Figure 4, our QRG method performed better than all other query recommendation baselines. We also conducted t-test (p value ≤ 0.05) over the results and found that the performance improvements are significant as compared with all the baseline methods. The reasons that QRG performed better than QUM can be summarized as follows. (1) QUM models the search sessions with a dynamic Bayesian model and treats the search session that ends with a query with clicks as satisfactory one. If a query occurs in a satisfactory search session, its utility is high. However, a dissatisfactory query can also occur in a satisfactory search session. In addition, the method of satisfaction estimation used by QUM is too naive and unrealistic. Therefore, the quality of the recommendations generated by QUM is poor. (2) QUM does not use the information of specific clicked documents and infer query utility directly. To avoid the two disadvantages of QUM, QRG estimates satisfaction for each query and models the specific clicked documents to infer document utility and then calculates query utility from document utility. The reasons that QRG performed better than SFG can be summarized as follows. (1) QRG classifies the clicked documents into satisfactory documents and dissatisfactory documents by estimating satisfaction for each query and only the satisfactory documents are used in QRG. In contrast, all the clicked documents are used in SFG. Since the satisfactory documents contain more information that was considered useful, the utility of satisfactory document is high. Therefore, the document utility inferred by QRG is higher than that inferred by SFG. (2) QRG can capture users' behaviors more accurately than SFG. First, QRG employs an interruption node and can capture the users' interruption behaviors while SFG cannot capture them. Second, our QRG explicitly classifies the users' behaviors at a distinct query as stop, reformulation, and interruption while SFG does not. In addition, QRG calculates the transition probabilities that correspond to the users' behaviors from the statistical information in query log while

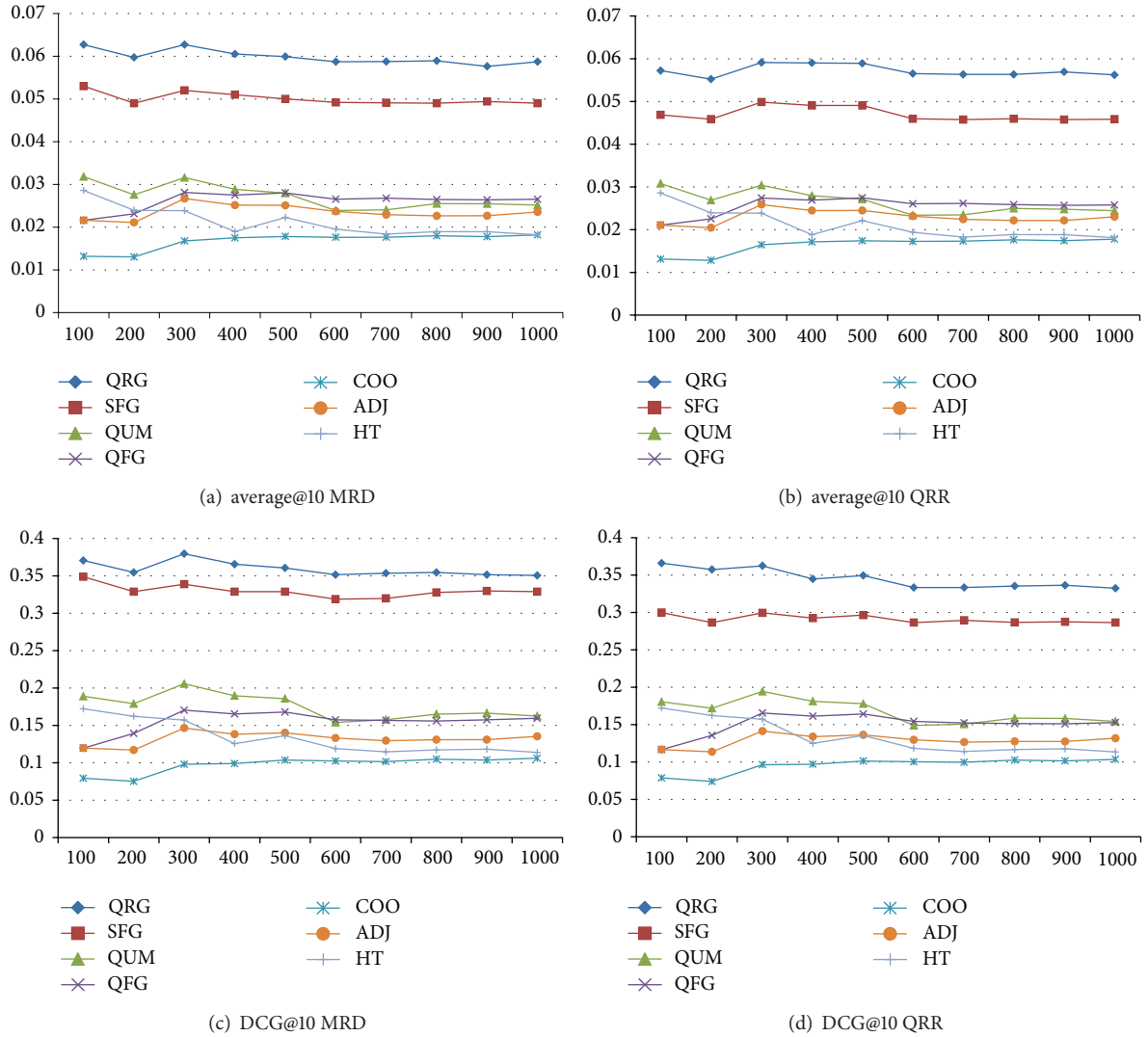

Figure 5: Evaluation results of seven query recommendation methods on 1000 partially labeled test queries.

SFG sets the transition probabilities from query node to other nodes as constant values.

5.2.4. Evaluation Results of Incremental Experiment on 1000 Partially Labeled Test Queries. Besides, we also conducted an incremental experiment to investigate the stability of our QRG approach in recommending high utility queries. We first randomly selected 100 test queries and ran all seven query recommendation approaches on the whole data set to generate recommendations for them. We collected the clicked search results of these recommendations together, randomly selected 10% from the collection, and labelled them as "relevant" or "irrelevant" according to their web page contents. The rest 90% search results were labelled as "irrelevant." For each test query, we calculated the four measures of average@10 MRD, average@10 QRR, DCG@10 MRD, and DCG@10 QRR for its top 10 recommendations generated by each method. We then averaged over 100 test queries to obtain the final performance of each method on one of four metrics. After evaluations of the first 100 test queries, we randomly selected 100 new test queries and merged them with the previously selected queries to form a new set of 200 test queries. In the same way, we evaluated the performance of seven methods on the 200 test queries with the four metrics. We repeated the same process 8 more times until 1000 test queries were evaluated. 100 more new test queries were added each time.

Figure 5 shows evaluation results of the seven methods under four metrics in the incremental experiment. The horizontal axis is the number of test queries and the vertical axis is the performance by each metric. We can see that the utility-based method QRG performed the best and the SFG performed second best. Other five baseline methods were far below the top lines generated by QRG and SFG. The similarity-based recommendation methods HT, COO, and ADJ took the bottom places, whereas QUM and QFG were in the middle. When the number of test queries was in a few hundreds, the performance of these query recommendation

methods converged. Their performance became stable after the number of test queries approached 500. At each given number of test queries, the performance of the seven methods in Figure 5 is consistent with the results in Figure 4. It indicates that the performance of QRG is stable and it stably performs better than all other baseline query recommendation methods.

5.3. Evaluation of Document Utility. Since the performance of our QRG method relies on the inferred document utility, in this subsection, we will evaluate the learned utilities of documents generated by our method.

5.3.1. Metric. Since the utility of document is the useful information *it can provide to satisfy* user's information need, the traditional document labelling strategy, that is, labelling as "relevant" or "irrelevant," can be employed. We first introduce precision at position k as the first metric on document utility. Consider

$$P@k = \sum_{j=1}^{k} r(j), \tag{14}$$

where k denotes the number of documents to be measured and $r(j)$ denotes the relevance of jth document:

$$r(j) = \begin{cases} 1, & \text{if } j\text{th document is relevant} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

$P@k$ measures the number of relevant documents in top k results.

However, $P@k$ does not consider the position of relevant document; therefore, we also introduce DCG@k to measure document utility considering the position information, formally. One has

$$\text{DCG@}k = \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log(j+1)}, \tag{16}$$

where k denotes the number of documents to be measured and $r(j)$ denotes the relevance of jth document as mentioned above. From the metric of DCG@k, we can see that the higher the rank of a recommendation is, the more important it is.

5.3.2. Baselines. We compare our QRG against two baseline approaches.

(i) Document Frequency-Based Method (DF): it is based on the click frequency of a document when the users browse the search results of the initial query. This method assumes that the click frequency of a document reflects users' preference to that document, and the higher the click frequency is, the more useful the document is. Thus, it can be used to measure the relevance of the document to the initial query.

(ii) Session Document Frequency-Based Method (SDF): since the document clicks of a query are biased on the position of search result, the relevant document ranked at low position may have no clicks. To alleviate this problem, another baseline method referred to as SDF is used. SDF assumes that the documents of queries in the same search session convey the similar search intent, and thus their click frequencies can be used to measure their relevance to the initial query.

Table 2: Comparison of document relevance of three methods. (The percentages in the parentheses are the improvement of our QRG method over the corresponding methods).

Method	SDF	DF	QRG
$P@5$	0.614 (10.4%)	0.634 (6.8%)	**0.678**
$P@10$	0.562 (11.8%)	0.578 (8.8%)	**0.629**
$P@15$	0.490 (21.7%)	0.513 (16.2%)	**0.596**
$P@20$	0.440 (30.4%)	0.453 (26.7%)	**0.573**
DCG@5	2.693 (9.4%)	2.780 (5.9%)	**2.945**
DCG@10	3.870 (10.6%)	3.985 (7.4%)	**4.279**
DCG@15	4.529 (16.8%)	4.716 (12.2%)	**5.290**
DCG@20	5.022 (22.5%)	5.180 (18.7%)	**6.150**

5.3.3. Evaluation Results. We first randomly selected 50 test queries and ran all three approaches DF, SDF, and QRG on the whole data set to rank the documents according to the measure used. For each test query, we calculated the two measures of $P@k$ and DCG@k for its top k ($k = 5, 10, 15, 20$) documents generated by each method. We then averaged over 50 test queries to obtain the final performance of each method on $P@k$ and DCG@k. Table 2 presents the performances of three approaches on learning document utility.

From Table 2, we can see that, among three methods, SDF performed worst. This is not surprising that SDF suffered from the problem that a query in the same search session with *the initial query may be irrelevant* to the initial query. DF performed better than SDF since the returned documents of initial query are always relevant to it. Our QRG method performed better than the other two baselines, especially when the number of measured documents was large. It indicates that QRG can help the users find more relevant documents from recommended queries of initial query than its search results. The reason is that QRG aims at finding the useful documents that make the users feel satisfied according to the initial query by simultaneously modelling the users' behaviors of stop, reformulation, and interruption.

6. Conclusion

In this paper, we have investigated the problem of how to recommend high utility queries to the users. We first proposed the concept of query-reformulation graph which can model the users' behaviors of stop, reformulation, and interruption at the same time. Then a novel two phrase models based on absorbing random walk on the query-reformulation graph was proposed. Experimental results on a real query log have shown that our proposed approach

achieved significant improvements over the baselines in recommending high utility queries.

Given query log data, we can construct a query-reformulation graph and recommend high utility queries to the users. However, there still exist some interesting problems needed to be addressed in the future work as follows. (1) Since QRG *method is dedicated to mining* utility of single query, a recommendation set generated by QRG may suffer from its redundant utility; for example, two recommendations may return the same relevant documents to the user. Therefore, one important future work is to mine the utility of whole set of recommendations and reduce the redundant utility in the recommendation set; (2) QRG is based on query-level satisfaction and assumes that if a query is satisfied, any clicked document in *its search results is satisfactory*. However, since some dissatisfied documents may be clicked in the search results of a satisfied query, query-level satisfaction may reduce the quality of recommendations. Therefore, in our future work, *we will focus on predicting* the click-level satisfaction and use it to capture better query utility.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

Joshua Zhexue Huang was supported by The National Natural Science Foundation of China under Grant no. 61473194.

References

[1] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '00)*, pp. 407–416, August 2000.

[2] L. Li, Z. Yang, L. Liu, and M. Kitsuregawa, "Query-URL bipartite based approach to personalized query recommendation," in *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*, vol. 2, pp. 1189–1194, 2008.

[3] H. Ma, I. King, and M. R. Lyu, "Mining web graphs for recommendations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 1051–1064, 2012.

[4] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: model and applications," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*, pp. 609–617, October 2008.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna, "Query suggestions using query-flow graphs," in *Proceedings of the Workshop on Web Search Click Data (WSCD '09)*, pp. 56–63, February 2009.

[6] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis, "An optimization framework for query recommendation," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM '10)*, pp. 161–170, February 2010.

[7] N. Craswell and M. Szummer, "Random walks on the click graph," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pp. 239–246, July 2007.

[8] L. Zhiyuan and S. Maosong, "Asymmetrical query recommendation method based on bipartite network resource allocation," in *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pp. 1049–1050, April 2008.

[9] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani, "Using association rules to discover search engines related queries," in *Proceedings of the 1st Conference on Latin American Web Congress (LA-WEB '03)*, pp. 66–71, 2003.

[10] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang, "Relevant term suggestion in interactive web search based on contextual information in query session logs," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 7, pp. 638–649, 2003.

[11] Q. He, D. Jiang, Z. Liao et al., "Web query recommendation via sequential query prediction," in *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE '09)*, pp. 1443–1454, April 2009.

[12] I. Bordino, C. Castillo, D. Donato, and A. Gionis, "Query similarity by projecting the query-flow graph," in *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pp. 515–522, July 2010.

[13] L. Li, G. Xu, Z. Yang, P. Dolog, Y. Zhang, and M. Kitsuregawa, "An efficient approach to suggesting topically related web queries using hidden topic model," *World Wide Web*, vol. 16, no. 3, pp. 273–297, 2013.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[15] H. Ma, H. Yang, I. King, and M. R. Lyu, "Learning latent semantic relations from clickthrough data for query suggestion," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*, pp. 709–718, October 2008.

[16] M. Strohmaier, M. Kröll, and C. Körner, "Intentional query suggestion: user goals more explicit during search," in *Proceedings of the Workshop on Web Search Click Data (WSCD '09)*, pp. 68–74, February 2009.

[17] H. Cao, D. Jiang, J. Pei et al., "Context-aware query suggestion by mining click-through and session data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)*, pp. 875–883, ACM, August 2008.

[18] A. Jain, U. Ozertem, and E. Velipasaoglu, "Synthesizing high utility suggestions for rare web search queries," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*, pp. 805–814, July 2011.

[19] X. Zhu, J. Guo, X. Cheng, and Y. Lan, "More than relevance: high utility query recommendation by mining users' search behaviors," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*, pp. 1814–1818, November 2012.

[20] X. Zhu, J. Guo, X. Cheng, Y. Lan, and W. Nejdl, "Recommending high utility query via session-flow graph," in *Advances in Information Retrieval: Proceedings of the 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013*, Lecture Notes in Computer Science, pp. 642–655, Springer, Berlin, Germany, 2013.

[21] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White, "Evaluating implicit measures to improve Web search," *ACM Transactions on Information Systems*, vol. 23, no. 2, pp. 147–168, 2005.

[22] A. Hassan, R. Jones, and K. L. Klinkner, "Beyond DCG: user behavior as a predictor of a successful search," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM '10)*, pp. 221–230, February 2010.

[23] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein, "Find it if you can: a game for modeling different types of web search success using interaction data," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information (SIGIR '11)*, pp. 345–354, July 2011.

[24] A. Hassan, X. Shi, N. Craswell, and B. Ramsey, "Beyond clicks: query reformulation as a predictor of search satisfaction," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM '13)*, pp. 2019–2028, November 2013.

[25] Y. Kim, A. Hassan, R. W. White, and I. Zitouni, "Modeling dwell time to predict click-level satisfaction," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*, pp. 193–202, ACM, New York, NY, USA, February 2014.

[26] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[27] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL '94)*, pp. 133–138, Las Cruces, New Mexico, June 1994.

[28] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[29] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy, "Clustering query refinements by user intent," in *Proceedings of the 19th International World Wide Web Conference (WWW '10)*, pp. 841–850, April 2010.

[30] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[31] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proceedings of the 15th International Conference on World Wide Web*, pp. 387–396, May 2006.

[32] Q. Mei, D. Zhou, and K. Church, "Query suggestion using hitting time," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*, pp. 469–477, October 2008.