Optimal Rare Query Suggestion With Implicit User Feedback

Yang Song, Li-wei He

Microsoft Research,
One Microsoft Way,
Redmond, WA 98052, USA
{yangsong, lhe}@microsoft.com

ABSTRACT

Query suggestion has been an effective approach to help users narrow down to the information they need. However, most of existing studies focused on only popular/head queries. Since rare queries possess much less information (e.g., clicks) than popular queries in the query logs, it is much more difficult to efficiently suggest relevant queries to a rare query. In this paper, we propose an optimal rare query suggestion framework by leveraging implicit feedbacks from users in the query logs. Our model resembles the principle of pseudo-relevance feedback which assumes that top-returned results by search engines are relevant. However, we argue that the clicked URLs and skipped URLs contain different levels of information and thus should be treated differently. Hence, our framework optimally combines both the click and skip information from users and uses a random walk model to optimize the query correlation. Our model specifically optimizes two parameters: (1) the restarting (jumping) rate of random walk, and (2) the combination ratio of click and skip information. Unlike the Rocchio algorithm, our learning process does not involve the content of the URLs but simply leverages the click and skip counts in the query-URL bipartite graphs. Consequently, our model is capable of scaling up to the need of commercial search engines. Experimental results on one-month query logs from a large commercial search engine with over 40 million rare queries demonstrate the superiority of our framework, with statistical significance, over the traditional random walk models and pseudo-relevance feedback models.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Text Mining

General Terms

Algorithms

Keywords

query suggestion, random walk, pseudo-relevance feedback, query log analysis

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

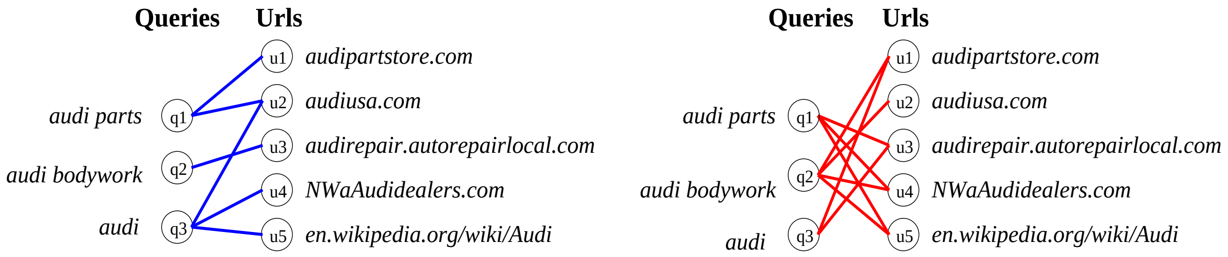
1. INTRODUCTION

Web search engines have completely changed the way people acquire information during the last ten years. By providing a comprehensive portal between the Internet users and the Web, search engines are able to take a user query and return a ranked list of web pages according to the relevance between queries and the search engine index, which consists a subset of the entire Web. Recent study indicates that search is still quite difficult, approximately 50% of times search engines fail to return relevant documents.

The reason of failure is that the length of the queries is usually quite short, so that understanding user intents correctly has been a critical yet quite difficult task for search engines. Among a variety of techniques, *query suggestion* related techniques [1, 2, 3, 8, 11, 12, 14, 18] have become an effective way to interact between users and search engines, hence to improve the relevance of search results.

Among all query suggestion techniques, one of the most important and effective techniques refers to query log analysis [2, 3, 8, 12, 18]. Specifically, query logs are server-end logs that record user activities in search engines. A typical query log entry contains *timestamp*, *query*, *clicked Urls* as well as user personal information. In order to learn a query suggestion model, a commonly used approach is to leverage graph representation which forms query and URL relationship into *bipartite graphs*. A query-URL bipartite graph usually consists of two disjoint sets of nodes, corresponding to queries and URLs respectively. An example of this bipartite representation has been shown in Figure 1(a), where the left-hand set of nodes are queries and the right-hand set are URLs. The edge between a query q and a URL u indicates user clicks of u when issuing q (for simplicity, the click numbers are omitted from the graph). The click graph possesses large amount of potential information that can be learnt for query suggestion, query clustering, query reformulation and so on. As a matter of fact, a myriad of techniques have been proposed. Among them, random walk technique is one of the most effective methods [12, 7].

However, leveraging only the click information has a serious drawback. That is, the models learnt from click graph can only benefit popular queries which possess enough user click feedbacks. While for rare queries that have only appeared a handful of times in the logs with very few clicks, click graph is unable to capture the underlying relationship between queries. For example, in Figure 1(a), q_1 and q_2 do not have commonly clicked URLs, thus a random walk



(a) Click Graph

(b) Skip Graph

Figure 1: An illustrative example of query-URL click graph (a) and skip graph (b). Query *audi parts* and *audi bodywork* are not correlated if only performs random walk on the click graph, but will be highly correlation if random walk is performed on the skip graph. More details on the text.

model which discovers query relationship according to their common clicks is unable to discover any correlation between q_1 and q_2.

While it is well known that in search engines, query frequencies follow a power-law distribution where most queries are issued very few times by users, rare queries together constitute a great amount of search traffic which potentially affects the relevance and revenue of search engines significantly. Therefore, the lack of efficient and effective proposals to deal with rare queries needs our immediate attentions.

1.1 Motivation of Our Work

Figure 1 presents a motivation of our approach. The left figure (a) shows the click graph for three queries and five URLs that returned as top SERP results. Ideally, *audi parts* should be a good suggested query for *audi bodywork* (and vice versa). However, after performing a random walk on the click graph, only the query *audi* can be suggested to *audi parts* because there is no commonly clicked URLs between *audi parts* and *audi bodywork* so that their correlation is zero. However, if we leverage the top-skipped URLs[1] for *audi parts* and *audi bodywork* as shown in Figure 1(b), it can be clearly observed that both queries skipped their top-returned two URLs: *NwaAudidealers.com* and *en.wikipedia.org/wiki/Audi*. As a result, a random walk on the skip graph will assign a high correlation score to these two queries.

Our work is inspired by the principle of *pseudo-relevance feedback* [16, 15, 10, 20, 19] which assumes that the top-k returned documents from search engines are always relevant to the queries, regardless of whether they are clicked or not. However, for rare queries, many times the top skipped URLs contain different levels of information than the clicked URLs. Because top-returned URLs are more likely to have high static rank scores which are representative of the high-level topic that the query belongs to. e.g., the URL u_5 is a general entry about *audi*, while queries *audi parts* and *audi bodywork* address different aspects of user need of the specific car model. Although users who issued these two queries clicked on more specific URLs like *audipartstore.com*, a general URL offers a potential topic link between these queries.

To further back up our argument regarding using both clicked and skipped URLs for rare query suggestion, we carefully analyzed query logs from a commercial search engine. Figure 2 shows user session statistics in one of the data sets we use in the experiment which contains 40 million unique queries. The figure compares the query frequency (x-axis) against the number of clicked and skipped URLs (y-axis). It can be observed that when the query frequency is low, more URLs are skipped than clicked during the same user session. However, with the increase of query popularity, the click patterns become more stable. Generally, users are tend to click more often on top-returned results for popular queries, while for rare queries, the clicks are more random and thus have higher entropy scores.

We further analyzed the quality of skipped URLs for rare queries. We selected 6,000 queries which have been issued less than 20 times within a week and asked human judgers to judge the relevance on a 1-5 scale (5 means the best). Figure 3 demonstrates the comparative ratings between skipped and clicked URLs. Overall, skipped URLs indicate a little bit less relevance than clicked URLs. On average, clicked URLs have a rating of 3.78 while skipped URLs have 3.65. This observation further supports our claim that skipped URLs should be leveraged for rare queries in the context of relevance measurement.

1.2 Contribution of This Paper

In this paper, we propose a novel graph combination-based rare query suggestion framework. Our proposal can be sketched into four major steps:

1. construct two query-URL bipartite graphs from query logs, where the click graph contains query-URL click information and the skip graph contains query-URL skip information,

2. perform random walk on each of the graphs, using the random walk with restart (RWR) technique [17],

3. build a correlation matrix for URLs from the category of URLs,

4. based on the URL correlation, iteratively optimize the model to estimate the best parameters of random walk and the combination rate of click and skip graphs. Finally, combine two query correlation matrices to form the optimal query correlation matrix, which is used for query suggestion.

[1] We define a URL to be skipped if it was viewed by the user without being clicked. So if a user clicked the 3^{rd}-ranked URL, then the 1^{st} and 2^{nd} URLs are said to be skipped.

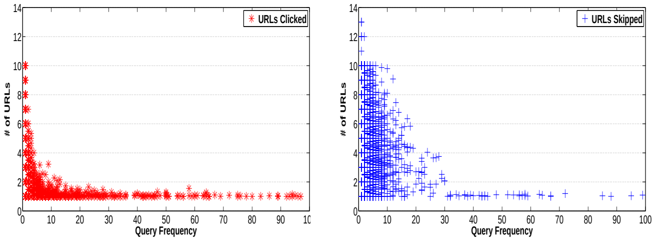


Figure 2: Number of URLs clicked vs. number of URLs skipped in the same user sessions from one week search log. There are more URLs skipped than clicked for queries with lower frequencies.

Our model specifically addresses two concerns. First, how to choose the optimal restarting rate for the random walk? Second, given two query-URL correlation matrices, how to optimally combine them? The reason is that the restarting rate directly affects the transition probability of random walk from nodes to nodes, which affects the distribution of query relevance scores that is critical for determining the most relevant neighbor nodes. On the other hand, the combination rate decides the level of contributions from click and skip graphs respectively. In pseudo-relevance feedback models, this ratio is the same for both clicked and skipped URLs, which is not optimal in practice for rare queries, as we shall see in the empirical analysis.

To the best of our knowledge, we are among the first to address the importance of the restarting rate (or jumping rate) of random walk, and optimize the parameter in a principled way. In other random walk-like models, this rate is either pre-fixed (e.g., the original PageRank paper [13] used 0.85 as the jumping rate), or empirically chosen without any support information [7].

The rest of the paper is organized as follows: Section 2 presents the literature in query suggestion, query clustering related research; Section 3 introduces our framework for optimal rare query suggestions; Section 4 provides empirical results on the performance of our model; finally, Section 5 concludes our proposal with future work.

2. RELATED WORK

A variety of research efforts have been devoted to address query suggestion related problems in literature, such as query classification, query clustering and query reformulation. Among them, most of the proposals directly or indirectly make use of query logs that contain query click information. By representing the relationship between queries and URLs into a click graph, many researchers have investigated in using random walk-related techniques for learning underlying query-document relevance. In [7], Craswell and Szummer proposed a Markov random walk model on the click graph to rank documents given a user query. The authors proposed a backward random walk comparing to the traditional forward random walk techniques such as page rank [13]. Specifically, the backward walk addresses the bias towards documents with more clicks in the forward walk models, by assuming a uniform prior on all documents. Experimental results indicate that the backward model are more effective in retrieving relevant documents for images. Essentially, the backward model can be treated as a normalization on the document clicks instead of query counts. Thus the most likely transition from a document to query will not be affect by the raw account of the queries, which eliminates the click bias.

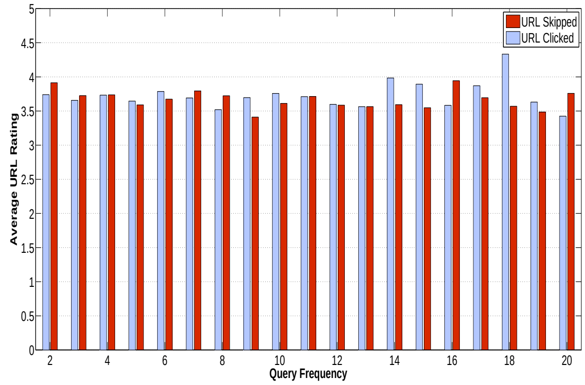


Figure 3: Human judger ratings in terms of relevance for clicked and skipped URLs in query logs. Break down accordingly to query frequency. Clicked URLs and skipped URLs have almost the same ratings for rare queries (queries with frequency less than 20).

Similarly, Deng et. al [9] proposed an entropy-bias framework to represent the edge weights between query and URLs. Comparing to the traditional raw-click frequency-based count of edge weights, the authors argued that various clicks should be treated differently based on the importance of the URLs and the queries. i.e., clicks on more specific URLs should be weigh more than clicks on general URLs. An inverse query frequency (IQF) based weighting mechanism was introduced to estimate the click quality. The authors applied the IQF model on random walk and the experimental results indicated superiority over the traditional count-frequency models.

In [12], Mei et. al introduced a parameter-free random walk model for query suggestion. This query-dependent model addressed the efficiency issue in random walk by constructing a subset of nodes in the click graph based on a depth-first search from the target node. Their model estimated the transition probabilities between two queries via an inner product-based similarity measurement. Consequently, the model is able to suggest semantically related queries to the original query by iteratively performing random walk and output the highest scored nodes.

Not until recently has the importance of rare query classification/suggestion attracted enough attention from the IR community. In [6], Broder et. al leveraged the results from search engines as an external knowledge base for building the word features for rare queries. The authors trained a classifier on a commercial taxonomy consists of 6,000 nodes for categorization. Experimental results indicated a significant boost in terms of precision than the baseline query expansion methods. Lately, Broder et. al proposed an online expansion of rare queries in [5]. Their framework started by training an offline model that is able to suggest a ranked list of related queries to the incoming rare query. The rare query is then expanded by a weighted linear combination of the original query and the related queries according to their similarity. The framework was specifically targeted for ads suggestion. The matching precision on the expanded queries demonstrated improvement over the original un-expanded version of queries.

3. OUR APPROACH

This section introduces our framework for rare query suggestion. Since our model is inspired by pseudo-relevance feedback, it is valid to assume that search engines do not generate random top results. Users click the results based on their own perception of relevance so that the a URL may be clicked by one user but skipped by others. Ideally, all returned URLs should be considered relevant. However, since we are only confident about top-ranked URLs, we will only consider the skipped URLs above the last user click. Next, we discuss how the query-URL graphs are generated.

3.1 Constructing Query-URL Graphs

We construct two bipartite graphs that correspond to explicit user feedback (clicks) and implicit user feedback (skips) from query logs respectively. The data we use is projected from the search log of a commercial search engine which serves millions of users daily. Each (simplified) record of the log contains $(q, \mathbf{u}^q, \mathbf{I}^q)$, where q corresponds to a user-issued query instance, $\mathbf{u}$ is a set of top-k URLs returned by search engine, i.e., $\mathbf{u}^q = \{u_1^q, ..., u_k^q\}$, and $\mathbf{I}^q$ is a vector that contains the corresponding binary variables indicating whether a URL has been clicked or not, i.e., $I_i^q \in \{0, 1\}$. Note that different users may issue the same query instance but click on different URLs. Thus, it is quite common that a query q have multiple query instances. To aggregate the clicks and skips from all query instances for a specific query q, we define a triplet $(q, \mathbf{u}^q, \mathbf{c}^q, \mathbf{s}^q)$, where

$$c_i^q = \sum_{j=1}^{d} \mathbb{I}(I_i^q = 1), i = 1, ..., k,$$

$$s_i^q = \sum_{j=1}^{d} \mathbb{I}(I_i^q = 0), i = 1, ..., k. \qquad (1)$$

Here $\mathbb{I}$ is an indicator function that equals 1 when the condition holds and 0 otherwise. Mathematically, c_i^q and s_i^q indicate the number of time a URL u_i has been clicked and skipped when users issued query q. For example, a query q has three query instances $\{q_1, q_2, q_3\}$, and the click pattern of the top-5 returned URLs is

$$\begin{aligned} \mathbf{I}^{q_1} &= \{1, 1, 0, 0, 0\}, \\ \mathbf{I}^{q_2} &= \{1, 0, 0, 0, 1\}, \\ \mathbf{I}^{q_3} &= \{1, 0, 1, 0, 1\}. \end{aligned} \qquad (2)$$

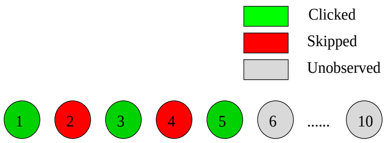


Figure 4: An example of clicked, skipped and unobserved URLs for query instance q_3 in eq.(2).

Figure 4 shows the example of q_3 that consists of clicked, skipped and unobserved URLs. Summarizing eq.(2) up by using eq.(1), we can get the aggregated clicks and skips:

$$\begin{aligned} \mathbf{c}^q &= \{3, 1, 1, 0, 2\}, \\ \mathbf{s}^q &= \{0, 2, 1, 2, 0\}. \end{aligned} \qquad (3)$$

Notice that here tw0 URLs $\mathbf{u}_2^q and \mathbf{u}_3^q$ are both clicked and skipped by different query instances.

Next, given a set of m user-issued queries $\mathbf{q}$ and the union of n returned URLs $\mathbf{u}$, we represent the relationship between queries and URLs using two squared weight matrices W^+ and W^-, defined as follows:

$$W^+ = \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{0} \end{pmatrix}, W^- = \begin{pmatrix} \mathbf{0} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{0} \end{pmatrix}$$

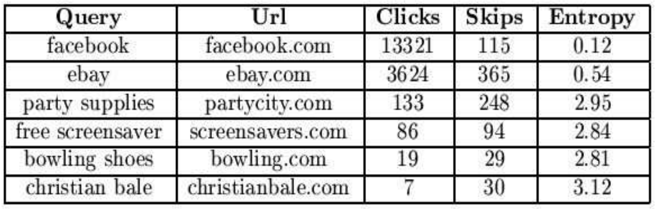
where

$$\mathbf{C} \in \mathbb{R}^{m \times n}, C(i,j) \triangleq c_{u_j}^{q_i},$$

$$\mathbf{S} \in \mathbb{R}^{m \times n}, S(i,j) \triangleq s_{u_j}^{q_i}.$$

Since the constructed matrices correspond to the bipartite graphs of a query set and a URL set, there exists no edges between queries (and URLs), as indicated by the big $\mathbf{0}$ in the diagonal of both W^+ and W^-, which should be learnt from our model.

Table 1 shows several queries and the number of times the top-returned URL being clicked / skipped. It can be observed that for popular queries like *facebook* and *ebay*, the top-ranked URLs are much more likely to be clicked, resulting a low click entropy. While for queries with lower frequencies e.g. *bowling shoes*, the top-ranked URLs, although being relevant, do not have a clear advantage over the lower-ranked URLs in terms of clicks. And in fact, these rare queries have much higher entropies that is more difficult for search engines to determine user intents.



Query	Url	Clicks	Skips	Entropy
facebook	facebook.com	13321	115	0.12
ebay	ebay.com	3624	365	0.54
party supplies	partycity.com	133	248	2.95
free screensaver	screensavers.com	86	94	2.84
bowling shoes	bowling.com	19	29	2.81
christian bale	christianbale.com	7	30	3.12

Table 1: Example of URL clicks and skips. Popular queries have more clicks than skips and thus lower entropies, while rare queries often exhibit much higher entropies.

3.2 Random Walk with Restart

Given the weight matrices of query-URL relationships, our objective is to define a score that indicates how closely two queries are. Since the original W matrices contain no such information, we propose to leverage the technique of random walk with restart (RWR) to learn the relevance score between queries [17]. RWR is a technique which specifies a starting node for a walk, then iteratively visits its neighbors with probability proportional to the edge weights. At each step, the walk has a constant probability of p to jump back to its starting node. It has been shown that after a certain number of steps, the probability of visiting a node j given the starting node i will become stable. We can thus define

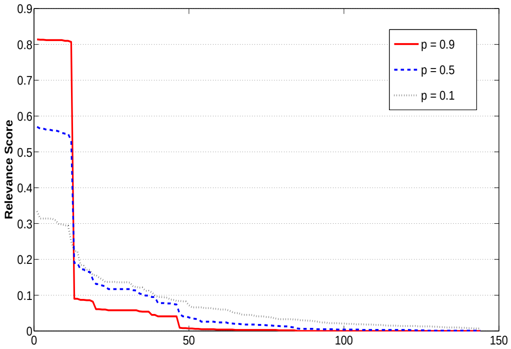


Figure 5: The neighborhood relevance score distribution w.r.t. different restarting rates p of random walks.

the ranking vector for all nodes given i as starting node:

$$\mathbf{R}_i = pW\mathbf{R}_i + (1-p)\mathbf{E}_i, \tag{4}$$

where W is the weight matrix, $\mathbf{E}_i$ defines a starting vector whose i^{th} entry is 1 and the rest 0. Each entry of R_{ij} defines the relevance score of node j to the starting node i. It can be observed that the above equation can be solved iteratively by replacing $\mathbf{R}_i$ from the previous iteration. After convergence, the finally ranking matrix $\mathbf{R} = \{\mathbf{R}_1, ..., \mathbf{R}_q\}$ for q nodes is a column-normalized matrix that contains the stable relevance scores of all nodes within the graph. For example, in Figure 1, the most relevant query for q_1 (*audi parts*) is q_3 (*audi*) according to the click graph, since $R_{13} = 0.24$ and $R_{12} = 0$. On the other hand, the skip graph suggests that q_2 (*audi bodywork*) has a higher relevance score than q_3.

The only parameter in eq.(4) is the restarting rate p which controls the shape of the probability distribution of neighborhood relevance scores. Higher p values have more local effect which assign very higher scores to its nearby nodes and generally ignoring the nodes far apart, while lower p values generate flatter probability distributions so that the start node i are more likely to reach out to distant nodes. Choosing the right restarting rate is critical for learning the query relevance scores. As it can be seen from Figure 5, the ranking vector will lose discriminative power when p is too low. On the other hand, if p is set to be very high, many nodes will end up with no correlations with the starting node i. Figure 6 further plots the density of weight matrix and shows the impact of p.

Unfortunately, optimizing the restarting rate has generally been ignored in literature. For example, in the random walk-like pagerank algorithm [13], the restarting (jumping) rate is pre-fixed to be 0.85 without optimization. In some other efforts, researchers empirically chose p in a grid-search way which is however dataset-dependent [7]. In the next two sub-sections, we propose a gradient optimization framework which leverages URL categorization information from ODP to choose the best parameters.

3.3 Determine URL Correlation

To get the ground-truth of URL correlation, we acquire the categorization labels of URLs from the Open Directory Project (ODP)[2]. ODP is a human-edited repository

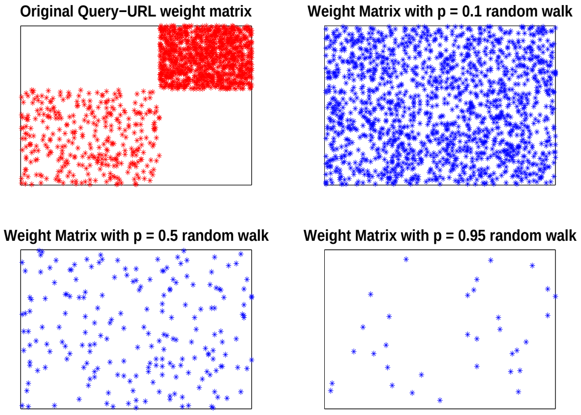


Figure 6: The density of weight matrix with different restarting rates. p=0.1 gives an over-dense matrix while p=0.95 results in a matrix that is too sparse. Note that self-correlation has been removed from the graph.

of URLs and categories that contains more than 4 million URLs with over 590,000 categories. The categories of URLs are organized in a tree-structured taxonomy where more general topics are at higher levels. In this paper, we leverage the top three levels of the ODP categorization. For example, the URL $\{research.microsoft.com\}$ belongs to $/Computers/Companies/Microsoft\_Corporation/$. An example of taxonomy can be found in Figure 7.

To efficiently generate labels for millions of URLs, we make a simplified assumption that all URLs within a domain have the same category. For URLs that do no belong to the ODP repository, we leverage a content-based hierarchical classifier to generate labels [4], which classifies a URL from top to bottom and refines the categories by propagating classification errors in a bottom-up manner. However, since in our framework we point out a general guideline of leveraging URL information for query suggestion, the generation of URL labels is beyond the scope of this paper.

In order to determine the correlation between URLs, different weights are assigned to each level of the taxonomy. Generally, lower levels receive higher weights since the categories are more specific than top levels. Algorithm 1 sketches the process to calculate correlation, where the multiplier m controls the bias towards lower levels. For example, for $m = 2$, top-three levels get weights of $\{1, 2, 4\}$ respectively. Therefore, node 1 and 2 in Figure 7 will have correlation of 3/7, node 3 and 4 correlation of 7/7, and node 1 and 3 of 1/7.

It should be noted that under our assumption that URLs having the same category in each domain, the query-URL weight matrix can therefore be compacted into query-domain matrix, which could possibly save much more computational time for random walk. However, it is critical to understand that the URL correlation is a guideline for optimizing the parameters for random walk as we shall see in the next section, but should not affect the relevance between queries. For example, users issue *buy printers* and *buy monitors* are very likely to visit both *dell.com* and *hp.com*, but apparently these two queries have different intents. Aggregating URLs into domains will possibly rank *buy monitors* to be more

[2] http://www.dmoz.org/

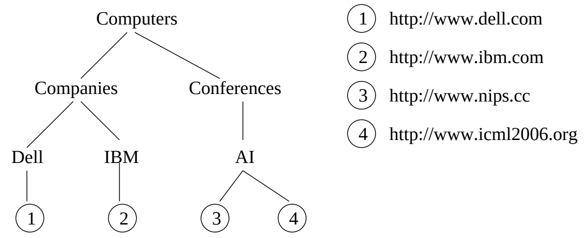


Figure 7: An example of ODP taxonomy for URLs. We leverage the top three levels of categorization in this paper.

Algorithm 1 URL Correlation Calculation

```
Input two URLs $\{u_1, u_2\}$ with hierarchical labels $\{l_1, l_2\}$, the multiplier $m$
Initialize $weight = 1$, $sim = 0$, $denominator = 0$
for each level $i$ in the tree
  if $l_1(i) = l_2(i)$
    $sim = sim + weight$
  end if
  $denominator = denominator + weight$
  $weight = weight * multiplier$
end for
$sim = \frac{sim}{denominator}$
Output $sim$
```

relevant to *buy printers* than other queries such like *printer supplies*. In practice, we also notice that the query-URL matrix performs better than query-domain matrix in terms of determining query relevance.

3.4 Parameter Optimization via Gradient Search

After performing random walk as in eq.(4), two ranking matrices $\mathbf{R}^+$ and $\mathbf{R}^-$ can be obtained from the click and skip graphs respectively, which can be further decomposed into

$$\mathbf{R}^+ = \begin{pmatrix} \tilde{\mathbf{Q}}^+ & \tilde{\mathbf{Q}}^+\tilde{\mathbf{U}}^+ \\ (\tilde{\mathbf{Q}}^+\tilde{\mathbf{U}}^+)^T & \tilde{\mathbf{U}}^+ \end{pmatrix}, \mathbf{R}^- = \begin{pmatrix} \tilde{\mathbf{Q}}^- & \tilde{\mathbf{Q}}^-\tilde{\mathbf{U}}^- \\ (\tilde{\mathbf{Q}}^-\tilde{\mathbf{U}}^-)^T & \tilde{\mathbf{U}}^- \end{pmatrix}$$

where $\tilde{\mathbf{Q}}$ is the estimated query correlation matrix, $\tilde{\mathbf{U}}$ the estimated URL correlation matrix, and $\tilde{\mathbf{Q}}\tilde{\mathbf{U}}$ the normalized estimated query-URL relationship.

Since both $\mathbf{R}^+$ and $\mathbf{R}^-$ are column-normalized weight matrices, a linear combination is proposed to estimate the ideal correlation matrix, so that the combination will preserve the property of column normalization:

$$\tilde{\mathbf{R}} = \alpha\mathbf{R}^+ + (1-\alpha)\mathbf{R}^-, \alpha \in [0,1], \qquad (5)$$

therefore, the estimated URL correlation matrix is also a linear combination

$$\tilde{\mathbf{U}} = \alpha\mathbf{U}^+ + (1-\alpha)\mathbf{U}^-, \alpha \in [0,1]. \qquad (6)$$

Along with the restarting rate p of random walk, the combination ratio α is the other parameter in our model. Since we have the ground-true URL correlations $\mathbf{U}$ from ODP data, we can optimize the parameters by minimizing the absolute loss of the estimated $\tilde{\mathbf{U}}$,

$$(p', \alpha') = \arg\min_{p', \alpha'} ||\tilde{\mathbf{U}}_{p,\alpha} - \mathbf{U}||, \qquad (7)$$

where $\tilde{\mathbf{U}}_{p,\alpha}$ is the estimated URL correlation given parameters p and α.

Algorithm 2 Parameter Optimization

```
Input two weight matrices $\{W^+, W^-\}$, URL correlation matrix $\mathbf{U}$
Initialize $p$ and $\alpha$ randomly
do
calculate $f(p,\alpha)$ according to eq. (8)
find the Jacobian and Hessian matrices numerically
    $$\partial f = \begin{pmatrix} \frac{\partial f}{\partial p}, & \frac{\partial f}{\partial \alpha} \end{pmatrix} = (df_p, df_\alpha).$$
    $$\partial^2 f = \begin{pmatrix} \frac{df_p}{\partial p} & \frac{df_p}{\partial \alpha} \\ \frac{df_\alpha}{\partial p} & \frac{df_\alpha}{\partial \alpha} \end{pmatrix}.$$
update $(p, \alpha)$ by the equation
    $$\begin{pmatrix} p' \\ \alpha' \end{pmatrix} = \begin{pmatrix} p \\ \alpha \end{pmatrix} - [\partial^2 f]^{-1}\partial f$$
Until convergence
Output optimal query ranking matrix $\mathbf{Q}'$
```

Since the above equation does not have a close-form solution, we propose to estimate the optimal value using numerical methods. To be concrete, we specify a function f of p and α,

$$f(p,\alpha) = \sum_i \sum_j \{\mathbf{U}(i,j) - \tilde{\mathbf{U}}_{p,\alpha}(i,j)\}, \qquad (8)$$

and use Newton's method to find its first and second derivatives so that during each iteration, both p and α are optimized. The new set of p' and α' is then used to perform a new round of random walk as well as the combination, in order to update the estimated URL correlation matrix $\tilde{\mathbf{U}}_{p',\alpha'}$. The process continues until the change of f becomes insignificant. Figure 8 illustrates an example of this optimization. Each sub-plot is a density plot of a matrix that shows the absolute difference between $\tilde{\mathbf{U}}_{p,\alpha}$ and $\mathbf{U}$. Each dot shows a point-level difference, where darker color indicates higher difference and white color means subtle difference (we used $1.0e^{-6}$ here). It can be seen that during the beginning with $iteration = 1$, the difference is significant since p and α are randomly initialized. With the function gradually being optimized with better parameters, the discrepancy starts to vanish. After the 20^{th} iteration, only less than 5% entries in the matrix are significantly different. Typically, the optimization finishes within 30 iterations. Finally, the optimal query correlation $\tilde{\mathbf{Q}}^{opt}$ is leveraged for query suggestion.

4. EXPERIMENTS

This section provides empirical results of how the proposed method performs on a large-scale data set, as well as the comparison between different random walk models.

4.1 Data Preparation and Evaluation Method

We collected a sample of one month query logs between March 16 and April 14 from a large commercial search engine, which contains approximately 40 million unique queries and 110 million query instances. Since our focus is on rare queries, we filtered out all popular queries and only kept queries with less than 20 appearances in the log. We also removed queries that have URL fragments and long queries (more than 10 words). Overall, we used 3,299,278 unique queries in our experiments. The data set also contains 8,139,150 user clicks and 13,577,113 skips on a total of 7,784,037 URLs.

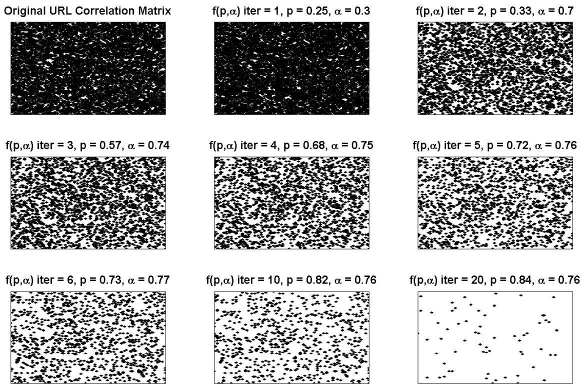


Figure 8: An illustration of the absolute loss (eq.(7)) of $||\tilde{U}_{p,\alpha} - U||$ during different iterations with p and α values. Darker color indicates higher inconsistency (more loss).

For comparison, three algorithms were tested on the same data set. The baseline approach is a random walk with restart model that only leverages click information from the query log (RWR-Base). We use the same algorithm as shown in Algorithm 2 to optimize the restarting parameter p, except that in this case the combination rate α is set to be 1. The second approach is similar to pseudo relevance feedback [15, 19], where all top URLs returned by a search engine are treated as relevant (RWR-Pseudo). We form edges between a query and top-10 returned URLs. We then perform a random walk on the click graph that includes all these edges. Finally, we also implemented the backward random walk [7] using the pseudo relevance model (RW-Back). We only considered the "101-0.9-backward" model since it performed the best among all models. Here 101 means 101-step random walk and 0.9 indicates the self-transition probability. Notice that the first and the second methods are *supervised* random walks which are optimized use ground-truth, while for the third method, the number of random walk steps and the transition probability are fixed but not optimized for the data set. Since our algorithm is a combination of positive and negative feedbacks, we denote it as RWR-Combo.

We used a similar evaluation process as in [7]. Since it is difficult to judge all the results for this data set with 3 million queries, we resort to sampling methods instead. We first uniformly sampled 500 queries from the set. The top 5 suggested queries generated from each of the four algorithms are evaluated, which results in a total of 2,500 relevance judgments. Each of the suggested queries are judged by two human judgers. Each query was judged to be either *relevant* or *irrelevant*. The survey tool was designed in the way that judgers are able to browse the search engine result page (SERP) for the given query when judging the relevance of suggested queries. The judgers are also allowed to navigate through links in SERP to help better understand the meanings of the query if necessary. The list of suggested queries from the four algorithms are randomly ordered so that the judgers are not aware of the particular algorithm behind them.

We employ two well-known metrics in information retrieval (IR) community to measure: Precision at rank N ($P@N$) and Mean Average Precision (MAP). Given a query q_i and top j suggested queries, the precision at the cut-off rank j can be defined as:

$$P(j) = \frac{\#\text{ of relevant queries}}{j}. \tag{9}$$

The top N precision is the aggregated precision from all queries in the evaluation set (K is the total number of queries):

$$P@N = \frac{\sum_{i=1}^{K} P(N)}{K}. \tag{10}$$

On the other hand, MAP averages the precision of a given query after each relevant query is retrieved, this metric focuses on both precision and recall so that the earlier relevant documents are retrieved, the higher MAP score will be. To be specific, for each query q_i,

$$AveP_i = \frac{\sum_{j=1}^{M}(P(j) \times \mathbb{I}(j))}{\#\text{ of relevant queries}}, \tag{11}$$

where $P(j)$ is the precision at position j as defined in eq.(9), $\mathbb{I}(j)$ is an indicator function that equals 1 when the j's suggested query is relevant and 0 otherwise. MAP is the average score over all K queries in evaluation.

4.2 Performance Comparison

Table 2 summarizes the overall performance of the four algorithms in the sample set. Our algorithm (RWR-Combo) successfully outperforms all other methods in both MAP and $P@5$. The random walk using pseudo-relevance (RWR-Pseudo) also has a good performance and beats the backward random walk (RW-Back). The random walk method leveraging only click information (RWR-Base) shows the worst performance. These results indicate two facts: (1) using only click information and ignoring the skipped URLs is not effective in suggesting relevant queries, and (2) treating clicks and skips with equal importance (RWR-Pseudo) is not as effective as assigning different weights to both feedbacks (RWR-Combo). We also notice that the unsupervised RW-Back method outperforms the baseline RWR-Base a little bit in both metrics.

Next, we break down the performance of these four algorithms into individual queries. Figure 9 shows $P@5$ for different queries. While performing the best on most of the queries, our RWR-Combo model also exhibits the lowest variance (0.021). The baseline RWR-Base shows the highest variance 0.036 among all. Overall, RWR-Combo outperforms RWR-Pseudo in 19 queries out of the 24 samples.

We also want to observe the quality of query suggestion at different positions. Therefore, we calculate $P@1$ as well as $P@5$ as illustrated in Figure 10. Our algorithm shows 65% of precision at position 1, an improvement of 9% over $P@5$. We notice that the baseline approach only has less than 3% of improvement at position 1, which demonstrates that with only click information, even the top-suggested queries are not quite relevant in the case of rare queries.

To illustrate the significance of the performance improvement, we conducted rigorous statistical significance test on the results of $P@5$. Usually, a p-value of less than 0.05 indicates a significant improvement. We performed paired t-test on every pair of the four algorithm. Table 3 summarizes the results. Each row shows an algorithm which compares the performance improvement over each column of algorithms. For example, the entry of the first row and first column can be interpreted as: RWR-Combo performs

Algorithm	MAP	$P@5$
RWR-Base	0.402839237	0.357734005
RWR-Pseudo	0.57829303	0.509583332
RW-Back	0.452837391	0.402468081
RWR-Combo	**0.622839822**	**0.56725468**

Table 2: Overall performance comparison of four algorithms in terms of MAP and $P@5$. Our algorithm (RWR-Combo) outperforms others in both metrics.

6% better than RWR-Pseudo on average, which is statistically significant with a p-value = 0.002. From this table, we also observe that using pseudo relevance show significantly better precision than both the backward walk and walk on only click graphs. On the other hand, although RWR-Back shows some performance improvement over the baseline as indicated in Table 2, the improvement is not significantly enough (p-value = 0.11).

To show the parameter sensitivity, we plot the value of $f(p, a)$ as in eq.(8) in Figure 11. The optimal result is achieved while f is minimized, when $\alpha = 0.765$ and $p = 0.837$. The worst result is when $p = 1$ (always jump back to starting node without hitting other nodes) and $\alpha = 0$ (without using any click information). It can also be observed from Figure 11 that for any particular value of α, the value of p around 0.6 and 0.85 gives the best results. Likewise, for any particular value of p, α between 0.65 and 0.8 performs the best. It is also observable that the loss function possesses a global minimum, so that no matter how parameters are initialized, the best result can always been achieved.

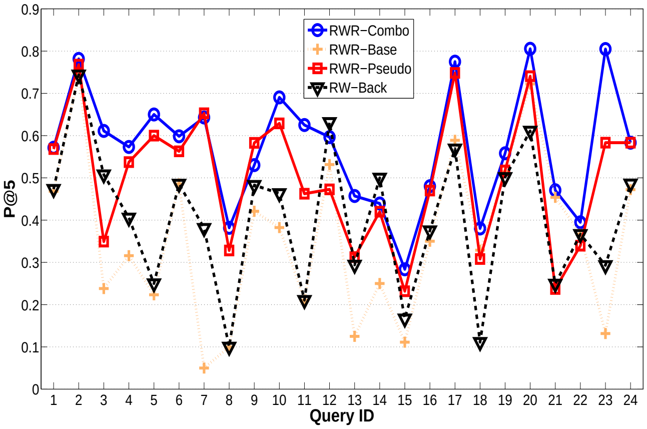


Figure 9: Performance break down of precision for each query. RWR-Combo performs best for most rare queries.

Table 4 shows some example queries and the suggestions by different algorithms, where queries with bold font are judged to be relevant by judgers. Our algorithm returns more relevant queries than any others in all four cases. The queries *valentine one* and *single ladies* are difficult queries which has different intents than their literal meanings. Our algorithm successfully discovers the underlying user intent and makes correct suggestions. On the other hand, our algorithm also exhibits its capability of diversifying suggestions to queries that have multiple intensions, e.g., *dc ups* could refer to both the power system and the postal service.

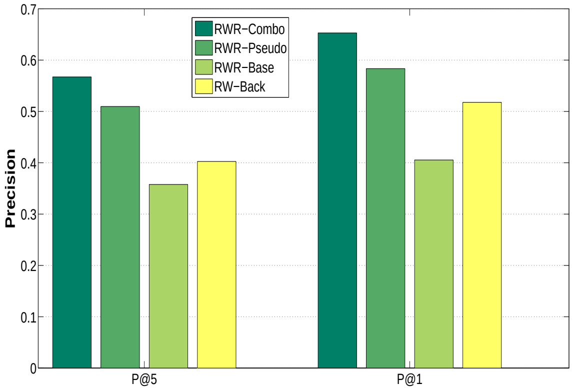


Figure 10: Precision at different rank of suggested queries. RWR-Combo achieves the highest precision of 65% among all algorithms at position 1 and 5.

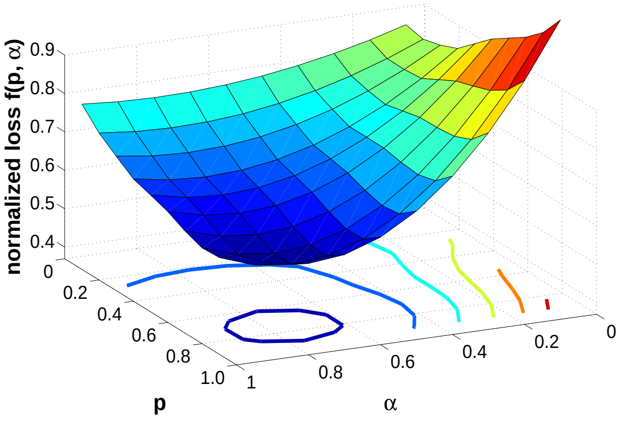


Figure 11: Normalized loss as in eq.(8), as a function of p and α. Smaller values indicate better results. Best achieved when $\alpha = 0.765$ and $p = 0.837$.

4.3 Why does our model work?

Overall, our model is quite straightforward. We combine query correlations from click and skip graphs and use URL categories as guideline to perform optimal random walk. So why is the model more successful on rare queries than others, e.g., pseudo-relevance feedback? The reason is that we somehow *smooth* the click graph by adding potential relevant edges according to the skip graph. While the pseudo-relevance feedback model treat all top-returned result as relevant, we perform it differently in a principled way. As indicated in Figure 11, the best result is achieved when $\alpha = 0.765$, suggesting that every click on a URL should be assigned with 0.765 edge weight, while every skip on a top-ranked URL should be treated with 0.235 edge weight. A rough calculation indicates that edges in click graph should be treated $0.765/0.235 \approx 3$ times more important as in the skip graph. Due to the characteristics of high entropy and low click frequency of rare queries, this model is capable of discovering potential edges between queries and skipped URLs, while not *over-smoothing* the graph like pseudo-relevance feedback which introduces lots of noises. On the other hand, the optimal restarting rate in our model is 0.837, which is surprisingly coherent with the jumping rate

	RWR-Pseudo	**RWR-Base**	**RWR-Base**
RWR-Combo (performs better than)	0.06 (p-value = 0.002)	0.22 (p-value< 0.001)	0.18 (p-value < 0.001)
RWR-Pseudo (performs better than)	/	0.16 (p-value < 0.001)	0.12 (p-value = 0.003)
RW-Back (performs better than)	/	/	0.04 (p-value = 0.11)*

*: insignificance of the comparison results.

Table 3: Statistical significance test on performance improvement of the four algorithms in comparison. Each entry contains the relative precision increase as well as the p-value. A p-value of less than 0.05 indicates a significance improvement.

Query	RWR-Base	RW-Back	RWR-Pseudo	RWR-Combo
valentine one (shopping & car)	valentines day valentine activities valentine gifts anniversary gifts free valentines crafts	valentine activities valentine gifts valentines day free valentines crafts anniversary gifts	valentine gifts **valentine one review** **radar detector** valentine one ebay valentine activities	**best radar detector** **escort radar** **radar detector** **valentine one review** valentine one ebay
single ladies (music)	dating ladies **beyonce single ladies** single women single women myspace eharmony	single women dating single ladies dating ladies dating ladies myspace single moms	**single ladies by beyonce** single ladies mp3 dating single ladies single ladies myspace **beyonce single ladies**	**beyonce single ladies** **single ladies by beyonce** single ladies lyrics single ladies mp3 **single ladies download**
nfl teams with 5 super bowl wins (sports & long)	**super bowl champions** super bowl super bowl 2009 super bowl 2008	super bowl 2009 super bowl 2008 **super bowl champs** list of superbowl	super bowl history **super bowl winners** **past nfl super bowl winner** super bowl 2009	**list super bowl winners** **super bowl winners** super bowl steelers **past nfl super bowl winner**
dc ups (ambiguous)	d-cups d cup **dc ups systems** d-cup dc control	dc power d-cups **dc ups systems** **dc power system** **universal power supply**	**dc power supply** dc ups power d-cups **dc ups systems** dc usa	dc power **dc ups power** **dc postal service** **dc power supply** **dc ups systems**

Table 4: Examples of query suggestions by four different algorithms. Bold queries are judged as relevant. Our algorithm RWR-Combo has the most number of relevant suggestions in all four cases. RWR-Combo is also capable of diversifying the suggestions to multi-intensional queries.

(0.85) as used in the pagerank algorithm. To summarize, a simplified way of leveraging our model to make suggestion for rare queries can be conducted as follows (the parameters are specified for the cut-off value of 20 for query frequency):

1. perform random walk with p around 0.85 on click and skip graphs respectively,
2. multiple the positive ranking matrix $\mathbf{R}^{+}$ by 0.75, the negative ranking matrix $\mathbf{R}^{-}$ by 0.25, respectively,
3. combine $\mathbf{R}^{+}$ and $\mathbf{R}^{-}$ linearly to get $\mathbf{R}^{opt}$ and extract the optimal query correlation matrix $\mathbf{Q}^{opt}$.

In our empirical analysis, when a query set contains more frequent queries, the skip graph becomes less important during the smoothing process. We ran three tests to obtain the optimal values of α and p on three different datasets, with query frequency cut-off values of $20, 50$ and 100 respectively. Figure 12 shows the bar plots in terms of the best parameter values. There is an obvious up trend of α values when the query frequency increases, while for the restarting value p, it stabilizes relatively around 0.85.

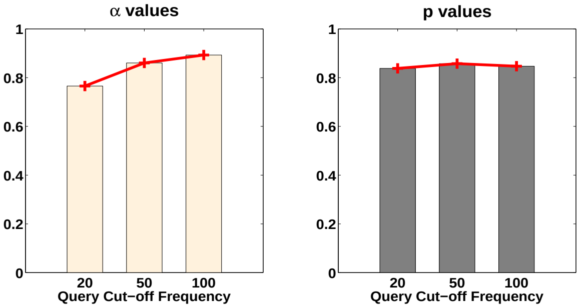


Figure 12: Optimal values of parameters w.r.t. different query frequency cut-offs. α value increases with the query frequency, while p is not correlated with the query frequency.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed an optimal solution for rare query suggestions. Rare queries are those difficult (long-tail) queries in search engines that appeared very few times. We proposed to tackle this problem by random walk on the query logs. Specifically, we leveraged both click and skip information from query log to form an optimal random walk and combination model. Our model was related to both pseudo-relevance feedback and smoothing technique used in natural language processing. Our major discovery was that user skipped URLs (observed by users but without clicks) played an important role in finding relevant information for rare queries, but they should be treated with different weight comparing to user clicked URLs. Since our model only relied on the click/skip information without the involvement of URL/query content, we were able to implement the framework on a large-scale data set which contained 40 million unique queries. The empirical result comparing with other random walk models indicates that our model can generate higher precision scores for rare query suggestion.

Since in this paper we only focused on the observed URLs from users, in the future, it would also be interesting to investigate unobserved URLs below the last user clicks to see if they can benefit our model or bring noises instead. This way we can break down eq.(5) into three different pieces (clicked, skipped and unobserved URLs) and optimize them jointly. It would also be useful to investigate other methods for finding correlations between URLs, e.g., topic models. And of course, it is worthy of further investigation how our model behaves for rare queries in different verticals, e.g., news, sports, commercial-intent and so on.

6. REFERENCES

[1] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 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 *In International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Creete*, pages 588–596. Springer, 2004.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM.

[4] P. N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA, 2009. ACM.

[5] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 511–520, New York, NY, USA, 2009. ACM.

[6] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238, New York, NY, USA, 2007. ACM.

[7] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246, New York, NY, USA, 2007. ACM.

[8] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 325–332, New York, NY, USA, 2002. ACM.

[9] H. Deng, I. King, and M. R. Lyu. Entropy-biased models for query representation on the click graph. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346, New York, NY, USA, 2009. ACM.

[10] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM.

[11] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006. ACM.

[12] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 469–478, New York, NY, USA, 2008. ACM.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, November 1999.

[14] F. Radlinski, A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search: a query substitution approach. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 403–410, New York, NY, USA, 2008. ACM.

[15] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, New York, NY, USA, 2005. ACM.

[16] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. 1971.

[17] H. Tong, C. Faloutsos, and J.-Y. Pan. Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.*, 14(3):327–346, 2008.

[18] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1):59–81, 2002.

[19] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126, New York, NY, USA, 2004. ACM.

[20] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM, 2001.