An Optimization Framework for Query Recommendation *

Aris Anagnostopoulos[1]
aris@cs.brown.edu

Luca Becchetti[1]
becchett@dis.uniroma1.it

Carlos Castillo[2]
chato@yahoo-inc.com

Aristides Gionis[2]
gionis@yahoo-inc.com

[1]Sapienza University of Rome, Italy

[2]Yahoo! Research Barcelona, Spain

ABSTRACT

Query recommendation is an integral part of modern search engines. The goal of query recommendation is to facilitate users while searching for information. Query recommendation also allows users to explore concepts related to their information needs.

In this paper, we present a formal treatment of the problem of query recommendation. In our framework we model the querying behavior of users by a probabilistic reformulation graph, or query-flow graph [Boldi et al. CIKM 2008]. A sequence of queries submitted by a user can be seen as a path on this graph. Assigning score values to queries allows us to define suitable utility functions and to consider the expected utility achieved by a reformulation path on the query-flow graph. Providing recommendations can be seen as adding shortcuts in the query-flow graph that "nudge" the reformulation paths of users, in such a way that users are more likely to follow paths with larger expected utility.

We discuss in detail the most important questions that arise in the proposed framework. In particular, we provide examples of meaningful utility functions to optimize, we discuss how to estimate the effect of recommendations on the reformulation probabilities, we address the complexity of the optimization problems that we consider, we suggest efficient algorithmic solutions, and we validate our models and algorithms with extensive experimentation. Our techniques can be applied to other scenarios where user behavior can be modeled as a Markov process.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Query formulation ; Search process*

General Terms

Algorithms

Keywords

Query reformulations; Query suggestions

1. INTRODUCTION

Query recommendations are a prominent feature of modern search engines. Query recommendations serve several purposes: correcting possible spelling mistakes, guiding users through their information-seeking tasks, allowing them to locate information more easily, and helping them explore other concepts related to what they are looking for.

The simplest form of query recommendation is *spelling correction*, a topic that we do not address in this paper. Instead we focus on more elaborate forms of query recommendations. For instance, by submitting the query "`chocolate cookie`" a user may be prompted to other queries such as "`chocolate cookie recipe`", or "`chocolate chip cookie recipe`", but also to related concepts such as "`brownies`", "`baking`", and so on.

A key technology for enabling query recommendations is query-log mining, which is used to leverage information about how people use search engines, and how they rephrase their queries when they are looking for information. Most of the proposed query-recommendation algorithms in the literature use aggregate user information mined from query logs and allowing to identify queries that are relevant to what the user is searching [2–4, 13, 14]. Current state-of-the-art methods often produce relevant query recommendations, but typically there is no clear objective to optimize and query-recommendation methods are fairly ad-hoc.

In this paper we propose a general and principled methodology for generating query recommendations. We model the query-recommendation problem as a problem of optimizing a global utility function. Our approach consists of the following ingredients:

- First, we assume that it is possible to aggregate historical information from a query log to build a query-reformulation graph [3]. The nodes of this graph are distinct queries, and an edge (q, q') is annotated with the probability that a user will submit query q' after submitting query q. We then model the querying behavior of users as *random walks* on this graph.
- Second, we assume that the queries in the query-flow graph have intrinsic score values $w(q)$, which model a

*Partially supported by EU Project N. 215270 FRONTS and by MIUR FIRB project N. RBIN047MH9: "*Tecnologia e Scienza per le reti di prossima generazione*".

WSDM'10, February 4–6, 2010, New York City, New York, USA.

desired property of the queries. For example, $w(q)$ may represent the probability that users who submit query q will be satisfied with the search-engine results. We assume that during the random walk on the query-flow graph, users collect the scores of (a subset of) the nodes that they visit. The higher the total value collected, the higher the overall utility of the system.

- Last, we assume that query recommendations can be viewed as a perturbation of the transition probabilities in the query-flow graph. The motivation is that while searching for information users tend to follow edges according to their propensity to reformulate queries, but they also move to queries that are recommended to them by the search engine.

- Given the objective to maximize the overall utility of the system, we formulate the query-recommendation problem as deciding how to perturb the transition probabilities so as to maximize the expected utility of a random walk. From the algorithmic point of view, the problem of finding k shortcut edges to add at each node order to maximize the overall utility is an NP-hard optimization problem. In this paper, we discuss the complexity of the problem and we propose approximate algorithms for obtaining solutions of high quality.

Contributions. We propose an optimization framework for query recommendation. The framework is based on the use of a random walk over the query-flow graph. An important ingredient of our framework is a model, supported by experimental evidence, of the way in which the addition of query-recommendation links affects the above mentioned random walk.

We design optimization algorithms to place query recommendation links so as to optimize the expected benefit achieved by navigating the perturbed query-flow graph. The effectiveness of the algorithms we propose is supported by theoretical analysis. For the special case of adding recommendation links to a single node, we are able to provide a characterization of optimal solutions. Our characterization allows to design a heuristic strategy that achieves near-optimal performance. For the general case of adding recommendation links from each node, we propose a heuristic based on the optimization of a modified objective function, which is a good proxy of the original objective function when the perturbation induced by adding recommendations is relatively small.

We perform experiments addressing the following issues: (i) testing the validity of our model on real query-log data; (ii) assessing the performance of our algorithms with respect to other reasonable heuristics on real data. Note that the algorithms that we propose maximize the expected benefit over the entire navigation of the user, not just the benefit over the next recommendation. To understand the effects of this choice, we also present the results of a user study intended to assess the user-perceived quality of query recommendations provided by our heuristics and the top ones returned by other systems.

Applications. While we present the problem using query recommendations as a motivation, our model and methods can be applied to other scenarios where user behavior can be modeled as a Markov process. A concrete application that also partly motivated our work is leisure-related search in entertainment sites such as Yahoo! *OMG* (celebrity/fashion/gossiping columns). Usually, users start browsing these services either from the frontpage or by performing some query. Our goal in this case is to show recommendations that take into account the browsing behavior of users in order to deliver an entertaining experience to the user, which involves a path along several of the best pages in the site.

As another example, consider exploring media sites such as YouTube or Flickr, where the system allows for browsing the collection in a guided way, suggesting related contents in order to provide an entertaining experience. In such a scenario, content should be recommended according to some "interestingness" criteria, and recommendations should depend not only on the next step of the user navigation, but also on the user future browsing path.

Roadmap. This paper is organized as follows. Section 2 discusses related work. Section 3 defines the scenario that we are interested in and formally describes the model we adopt in the rest of the paper. Section 4 formalizes providing effective recommendations as a suitably defined optimization problem. Section 5 addresses the complexity of the general problem we consider. In Section 6, we present our analysis of historical data to model the user browsing behavior and query utility. In Section 7 we build on this model to solve the query recommendation problem, and we compare our method with other natural baselines. Finally, in Section 8 we conclude and present some ideas for future work.

2. PREVIOUS WORK

2.1 Query recommendations

Devising effective strategies for query recommendation has been recognized as an important task since the early 2000's. Query recommendation tools are commonly part of the interface provided nowadays to users of search engines.

A first line of research has focused on the task of finding queries that are related to those submitted by the user. To find related queries, various strategies have been proposed, including measures of query similarity [2], query clustering [13], or association rules [8].

A different approach has been taken by Zhang and Nasraoui [14], who attempted to model the user sequential search behavior. Zhang and Nasraoui consider query graphs in which nodes represent queries and edges consecutive queries in the same user session. Edges are weighted by a damping (or forgetting) factor, providing a measure of similarity between consecutive queries, whereas the similarity for non-consecutive queries within the same session is calculated by multiplying the similarity values of arcs along the path connecting them. An overall similarity value is obtained by adding up the contributions of different sessions.

The concept of a query graph has been further expanded by Boldi et al. [3]. Here, the authors introduce the concept of a *query-flow graph*. The authors define a graph in which an edge (q, q') denotes the fact that at least one user has submitted query q' after submitting q in the same session. Edges are associated to weights, which estimate the probability that a query transition connects related queries, and are computed from aggregate information extracted from the query log. Boldi et al. consider three heuristics for query recommendations: (i) a simple one based on recommending, for a user at query q's search results page, the query

q' such that the weight of arc (q, q') is maximum; (ii) two heuristics based on random walks with restart, where restart may either occur at q or at some of the last k nodes visited by the user in the current session, according to a suitable probability distribution.

Different types of graphs can be defined over the summary information contained in query logs. In the previous paragraphs, we discuss only the approaches that are most closely related to this paper. An overview of literature, techniques, and a broader range of applications of graphs extracted from query logs is presented by Baeza-Yates [1].

One of the differences of our research with previous work, is that we use the summary information contained in the query-flow graph to define a whole optimization framework with respect to the browsing behavior of the average user. In this context, we define several optimization problems, investigating important properties of optimal solutions in significant cases. Finally, we propose and analyze heuristics that have provable performance with respect to the optimization objectives we consider.

2.2 Perturbation of Markov chains

The approach that we consider in this paper considers the scenario of perturbing a Markov chain so as to optimize some suitable function.[1] While perturbation theory is a well-studied area of matrix analysis, there is less literature on sensitivity analysis of probability distributions defined over a (possibly non-ergodic) Markov chain. In fact, most literature addresses the topic of sensitivity analysis of Pagerank [12, Chapter 6], mostly providing bounds on the change in norm in the Pagerank vector when perturbations occur. One key contribution in the area [6] provides an interesting and deep analysis to prove the intuitive fact that Pagerank is monotonic, that is, adding a link to a Web graph cannot decrease the Pagerank value of the target page.

In this paper, we consider a related, but different problem: we are interested in optimizing the change in the expected utility achieved by a random walk on a (generally non-ergodic) Markov chain when at most k outgoing links are added to the same node (or, in general, at most k links are added to each node in a subset of the nodes).

2.3 Link optimization

A scenario in which links have to be added to a weighted graph in order to optimize some function is considered in [7] and [11], which address the *hotlink assignment problem*. Here, we are a given a DAG with a distinguished root node, representing the home page of a web site. Leaf nodes have associated weights, representing the probability with which a leaf node will be visited. The goal is adding at most one link per internal page, so as to minimize the expected number of steps needed for a user to reach the desired leaf from the root. This problem is NP-hard in general and the authors prove several approximation results that depend on the probability distribution on the leaves.

The scenario we consider is similar in spirit to those considered for the hotlink assignment problem [7,11], even though there are important differences: (i) the hotlink assignment problem typically involve DAGs and trees, while here we consider general graphs; (ii) the addition of hotlinks does not change the underlying probability while in our case, as explained further below, providing recommendation links modifies the structure of the underlying Markov chain; (iii) the goal in the hotlink assignment problem is to minimize the expected distance traveled to reach the desired leaf, while we define different objective functions that depend on node weights and the visiting probabilities of a non-ergodic Markov chain.

Chakrabarti et al. [5] consider the problem of adding "quicklinks" (i.e., shortcuts) to search engine results to optimize some utility measure. In more detail, the authors assume that for some queries the most relevant page is the home page of some web site W, which becomes the source of a possible, further user navigation of the web site W. User navigation paths from W are called *trails*. A trail will possibly meet the information needs of a user. The problem considered in [5] is to recommend at most k "deep" links that are likely to meet the user information needs. The authors assume that (i) each page u in W has some probability $\alpha(u)$ to be considered a useful page in a trail meeting the user information needs; and (ii) the usefulness of pages is described by a suitable benefit function. The objective is to choose the links to add so as to maximize the sum of the benefits achieved over a given set of possible navigation trails. The authors prove that the problem is in general NP-hard and that it admits a constant approximate solution under mild assumptions on the benefit functions.

Our approach differs from the one considered in [5] in significant ways: (i) the problem studied is different, since their goal is to provide a set of quicklinks to pages belonging to a web site reported in the top position of a search engine's result list, while we consider the problem of suggesting queries that maximize the benefit over the entire navigation path of the user; (ii) their objective to optimize depends on the set of suggested quicklinks and on a measure of noticeability assigned to nodes, which is inferred from click-through data, while in our case, it depends on the Markov chain underlying the query-flow graph, which is itself perturbed by the addition of recommendation links, thus modifying the objective function itself and complicating the optimization task.

3. USER-BEHAVIOR MODEL

We now describe the modeling of user behavior as a random walk on the query graph.

3.1 Query reformulation graph

The *query-flow graph* [3] is a directed graph $G = (V, E)$. Nodes in V represent queries plus one special symbol t indicating the end of a search goal in our case. Let $|V| = n$. The edges $E \subseteq V \times V$ of the query-flow graph represent *query transitions*. When the second element of an edge is not the terminal symbol t, then the edge represents a *query reformulation*.

We associate a Markov chain to the query-flow graph. In particular, let $\mathbf{P}_{n\times n}$ be a row-stochastic matrix, arranged so that its last row represents the terminal symbol t. Accordingly, $\mathbf{P}_{n,i} = 0$ for all $i \neq n$ and $\mathbf{P}_{n,n} = 1$, so that state t is an absorption state of the Markov chain. The other transition probabilities of $\mathbf{P}$ can be estimated either (i) as observed reformulation frequencies, or (ii) by a machine-learning algorithm that uses many types of features; see [3] for details.

When a user submits a query $q \in V$, we expect that, in the absence of any query recommendation, the user will

[1] More precisely, some suitable probability distribution defined over the Markov chain under consideration.

submit the next query according to the distribution $\mathbf{P}_{q,\cdot}$, the q-th row of matrix $\mathbf{P}$. In other words, we assume that the user's querying behavior is modeled by a random walk on the matrix $\mathbf{P}$. Let $\tau_q = \mathbf{P}_{q,n}$, the probability that a user will terminate her session at query q, possibly after clicking on a search result for q.

In the remainder, we let $w : V \to \mathbb{R}$ represent node weights, so that $w(q)$ provides a quantitative indication of the intrinsic value of query q to the users (or to the search engine, as we see shortly). We assume that $w(n) = 0$. The generic weight $w(q)$ models the quality of a query q, for instance, how satisfied is a user with the search results of q. Obviously, it is impossible to know if a user is satisfied or not from the results of a query. However, we can use cheap proxies available in the query logs, such as clicks to search results, dwell time, etc. Another option for setting the weights $w(q)$ is to consider the monetization of a query q, which can be also estimated by query-log analysis, for instance by the clicks on advertisement links. We can also consider combinations of the two. Finally, other options are possible and in this paper, we are completely agnostic about the interpretation of the weights $w(q)$.

3.2 Query-recommendation model

The Markov chain $\mathbf{P}$ models the behavior of users regarding query reformulations when they are not shown any query recommendations. When users are shown recommendations, their behavior can change in complex ways. Defining sound and reasonably simple models for estimating transition probabilities between queries in the presence of recommendations is important when investigating effective query recommendation strategies. Naturally, any proposed query-recommendation model has to be validated with respect to real query-log data.

In general, providing recommendations to a query q induces a perturbation of $\mathbf{P}$. In the remainder, we assume that this perturbation only affects $\mathbf{P}_{q,\cdot}$, leaving other rows unaffected. We emphasize that this assumption and others that follow are consistent with experimental observations, as shown in Section 6. Correspondingly, the perturbation induced by adding a set of recommendations Q to q can be described by a set of values $\rho^{\mathbf{P}}_{q,q'}(Q) \in [-1, 1]$, which in general depend on $\mathbf{P}$ and Q. In the remainder, we drop $\mathbf{P}$ (and possibly Q) from $\rho^{\mathbf{P}}_{q,q'}(Q)$ when they are clear from the context. As a result, if a set of queries Q is recommended to users who submit q, the change induced for the matrix $\mathbf{P}$ is described by $\mathbf{P}'_{q,\cdot} = \mathbf{P}_{q,\cdot} + \rho^{\mathbf{P}}_{q,\cdot}(Q)$ (so actually we have that $-\mathbf{P}_{q,q'} \leq \rho^{\mathbf{P}}_{q,q'}(Q) \leq 1 - \mathbf{P}_{q,q'}$). Of course, a perturbation of $\mathbf{P}_{q,\cdot}$ affects $\mathbf{P}_{q,t} = \tau_q$. More precisely, $\tau'_q = \mathbf{P}'_{q,t} = 1 - \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'}$.

Note that this definition provides a quite general framework that can model complex interactions, such as cases of negative dependence in the transition probabilities to similar queries.

For the sake of exposition, in the rest of the paper we assume a model in which transition probabilities associated to recommendations depend on the set Q of recommendations and on $\mathbf{P}$, but not on the order in which queries are recommended. Furthermore, we restrict to scenarios in which a query appears at most once in Q (i.e., Q is not a multiset). In fact, our framework extends to these more general settings as well.

Properties of query-recommendation models. In general when the user has just submitted query q, it is not sensible to recommend q itself, so $\rho_{q,q}(Q) = 0$, if $q \in Q$. We also make other assumptions about the properties of the recommendation model: (i) $\rho_{q,q'}(Q) \geq 0$, for all Q and $q, q' \neq t, q' \in Q$, that is, we expect an increase in the transition probabilities of the queries being recommended.

We also assume that recommending a query does not affect the transition probabilities for other queries (except possibly the termination node t). Namely, we state property (ii) as follows: $\rho_{q,q'}(Q) = 0$ if $q' \notin Q$, for all Q and $q, q' \neq t$. If properties (i) and (ii) hold, we also require the following property: (iii) $\sum_{q' \in V} \rho_{q,q'}(Q) \leq \tau_q$ for all Q and $q, q' \neq t$ also holds, since the matrix $\mathbf{P'}$ has to be row stochastic. Thus, recommending queries for a query q reduces the probability of a session terminating at q.

Empirical observations. While the properties that we state in the previous paragraphs are in theory restricting, they are motivated by our findings and experiments on real query-log data. Those experiments are discussed in detail in Section 6.3. Our main finding is the verification of properties (i) to (iii): adding recommendations to a query reduces the termination probability associated to the query, without significantly affecting other transition probabilities.

For example, we observe that, in the presence of recommendations, average termination probability is reduced from $\tau_q \approx 0.90$ to $\tau'_q \approx 0.84$, while at the same time, we observe for the sum of probabilities associated to recommendation links, that $\sum_{q'} \rho^{\mathbf{P}}_{q,q'}(Q) \approx 0.06$, thus supporting assumption (ii) above. Details are given in Section 6.

4. PROBLEM STATEMENT

The general problem that we are interested in is assisting users to formulate queries by suggesting query reformulations of potential interest and biasing the query graph navigation towards queries of higher value. Given that we cannot show an arbitrarily large number of suggestions in the search engine interface, we will assume that we can provide at most k recommendations per query.

4.1 Input

We assume that we can estimate the values $\rho^{\mathbf{P}}_{q,q'}(Q)$; determining these values is an interesting but difficult statistical problem, and outside of the scope of this paper. Supported by experimental observation, we will use an estimate based on a linear function of $\mathbf{P}_{q,q'}(Q)$.

We also assume that we are given, or that we can estimate the weights $w(q)$. Estimating these quantities can be hard, for example, when they corresponds to estimating click-through rate (CTR) on ads, or even harder when they estimate user satisfaction [9]. For our experiments we used click-through-rates as a proxy, since the majority of documents that users click are relevant for their interests; however, our framework is general and other functions can be used.

4.2 Objective functions

As we mentioned in the introduction, we consider query-recommendation as an optimization problem, for which we next define plausible objective functions. As we mentioned in Section 3.1, every node in the query graph has an associated weight representing its intrinsic value. Depending on

the application of interest, we can define a *utility function* $U(\cdot)$, which associates a utility to every user session, typically a function of the weights of nodes visited during a user session on the query graph. Let $\mathbf{path}(q)$ be the (non necessarily simple) path that the user follows in the query graph, with q_t being the last query issued (just before terminating by visiting the terminal state t).

Two natural choices for $U(\cdot)$ are

- MAX-SUM: $U(\mathbf{path}(q)) = \sum_{q' \in \mathbf{path}(q)} w(q')$; and
- MAX-LAST: $U(\mathbf{path}(q)) = w(q_t)$.

The first choice captures scenarios such as maximizing total user satisfaction through the entire session (by setting, for example, $w(\cdot)$ to be the number of clicked results), or maximizing revenue by displaying ads, or minimizing user session lengths (by setting the weights to -1).

The second choice can capture user satisfaction by setting, for example, $w(\cdot)$ to 1 if the last session terminates with a user clicking to a result.

We are now ready to formally define the optimization problems that we consider in this paper.

Multi-step query recommendation. In the general problem, users start their navigation at an initial state according to some distribution π_0, whose i-th component we denote by π_{0i}. We sometimes need to consider the special case in which $\pi_0 = \mathbf{e}_q$ for some $q \in \{1, \ldots, n-1\}$, with $\mathbf{e}_q$ being the q-th canonical column vector (the vector whose q-th element is 1 and the rest are 0). Given the transition matrix $\mathbf{P}$ of the underlying query graph, perturbation functions $\rho^{\mathbf{P}}_{\cdot,\cdot}(\cdot)$, a utility function $U(\cdot)$ and a positive integer k, we seek a strategy for recommending at most k queries per node so as to maximize the expected utility for a user starting at any query in the query graph, that is, $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(i))]$, where $\mathbf{P}'$ is such that, for every q, $\mathbf{P}'_{q,\cdot} = \mathbf{P}_{q,\cdot} + \rho_{q,\cdot}(Q(q))$, with $Q(q)$ being the set of queries recommended at q, while the expectation is taken with respect to the perturbed stochastic matrix $\mathbf{P}'$.

Single-step query recommendation. A simpler version of the problem is when the search engine can only affect the user trajectory at the initial step, but not afterwards. This case can correspond to users issuing a query as a starting point for browsing.

Formally, our goal in this case is to recommend at most k queries to users visiting q, (i.e., add at most k links leaving q in the query graph), so as to maximize the expected utility after the perturbation. Note that we still want to maximize $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{P'}[U(\mathbf{path}(i))]$, but with the constraint that only the q-th row of $\mathbf{P}$ is modified, that is, $Q(q') = \emptyset$, for $q' \neq q$.

5. ALGORITHMS

Next we address the problem of solving the multi-step and the one-step query optimization problems. We first show that both versions are NP-hard and we then we look at heuristics for solving both versions of the problem.

5.1 Solving the problem

Complexity. Both of the query-recommendation problems we consider are NP-hard.

THEOREM 5.1. *Both the multi-step and the single-step recommendation problems are NP-hard.*

The proof follows from the fact that we can encode a maximum-coverage problem instance in the function ρ. The details are omitted due to space constraints and will appear in an extended version of this paper.

Optimizing multi-step query recommendations. We recall from Section 4.2 that our optimization objective is maximizing $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(i))]$, where π_0 is the initial distribution of starting queries. In fact, this can be a very difficult task and many questions remain open for the moment. For example, we show in Subsection 5.2 that the solution for the single-step case is independent of the initial distribution. It is not clear if this result carries over to the multi-step case: the optimal solution might in principle depend on the initial distribution and the set of recommendations chosen at one node might in the optimal solution depend on the choices performed at other nodes in complex ways. In our quest for a practical solution, we propose below a heuristic in which choices are performed independently for every node of the query-flow graph.

First notice that, in the special case that $\pi_0 = \mathbf{e}_q$ for some q (that is, given that the user started at query q) we can write the optimization problem as follows:

$$\max_{\mathbf{P}'} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q))] = \max_{\mathbf{P}'} \left(w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q'))] \right),$$

or

$$\max_{\mathbf{P}'} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q))] = \max_{\mathbf{P}'} \left(\tau'_q \cdot w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q'))] \right),$$

depending on which utility function we consider. The first corresponds to MAX-SUM, the case that the utility equals the sum of the weights in the path, and the second to MAX-LAST, the case that the utility equals the weight of the last node before termination. We focus on the first case as the second one is similar. Instead of maximizing

$$\max_{\mathbf{P}'} \left(w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q'))] \right),$$

we perform a one-step approximation and we maximize the expression

$$\max_{\mathbf{P}'} \left(w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))] \right) = \max_{\mathbf{P}'_{q,\cdot}} \left(w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))] \right).$$

Note that, compared to the initial formulation, the expectation is now taken with respect to $\mathbf{P}$ and not $\mathbf{P}'$, so that the last equality above follows since the terms $\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))]$ do not depend on the perturbation. We call this the *one-step approximation* because we assume that the system will perform recommendations only in the current query q and not in the subsequent user browsing session.

Now we only need to compute the expressions $V_{q'} \doteq \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))]$ and write for each query q':

$$V_{q'} = w(q') + \sum_{q'' \in V \setminus \{t\}} \mathbf{P}_{q',q''} \cdot V_{q''}.$$

This yields a set of n linear equations with n unknowns so we can compute all the values $V_{q'} = \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))]$ simultaneously. We can therefore cast the problem as

$$\max_{\mathbf{P}'_{q,\cdot}} \left(w(q) + \sum_{q' \in V \setminus \{t\}} \mathbf{P}'_{q,q'} \cdot V_{q'} \right),$$

for known values $V_{q'}$. This decomposition allows us to solve a large number of scenarios.

5.2 Single-step query recommendations

In this subsection, we consider the single-step query recommendation problem[2]. Given a query q_j, our goal is to choose a set Q of at most k recommendations to propose to users viewing q_j's search engine results, so as to optimize some utility function. In the following, we label queries as $q_1, \ldots, q_{n-1}$ and we assume that $\mathbf{P}$'s i-th row corresponds to query q_i. Our goal is to optimize $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q_i))]$, where $\mathbf{P}'$ is the perturbed matrix and π_{0i} is the probability that the user starts her navigation at q_i. For utility, we use MAX-LAST, where the utility is the weight of the last query before termination. We assume that the properties (i)-(iii) described in Section 3.2 hold, so that the perturbation only affects the row of the initial matrix $\mathbf{P}$ corresponding to q.

Note that, from the above paragraph, the objective appears to depend on the initial probability distribution π_0. In fact, the simple heuristic we propose below relies on the interesting fact that the optimal set of query recommendations to add to some given query q_j so as to maximize the expression above does not depend on π_0. This is stated by the following fact:

FACT 5.2. *Assume we add a set Q of recommendations at q_j, $|Q| \leq k$, so as to maximize $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q_i))]$, where $\mathbf{P}'$ denotes the perturbed matrix. Then, the optimal solution is independent of π_0.*

This fact implies that, in order to optimize the placement of links at some query q, we can limit ourselves to optimizing $\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q))]$ and this choice is optimal with respect to any initial distribution. We prove in the full paper that the simple, greedy algorithm given in Figure 1 performs close to optimum in the cases of practical interest.

Algorithm `SSQR-Greedy` requires the computation of the expected utility achieved by a random walk starting at q_ℓ, for $\ell = 1, \ldots, n-1$, where the expectation is taken with respect to *the unperturbed matrix* $\mathbf{P}$. These quantities can be precomputed once and stored beforehand. Denote by Q_{ALG} and Q_{OPT} the algorithm's and the optimum's choices for Q. The optimal algorithm solves the problem exactly, computing the solution maximizing the expected benefit, generally in non-polynomial time. Denote by $h(Q_{ALG})$ and $h(Q_{OPT})$ the corresponding values for the expected utility. Algorithm `SSQR-Greedy` achieves a performance that is close to optimum in cases of practical relevance, as shown by the following

[2]All proofs of this subsection are omitted for the sake of space and will appear in the extended version of the paper.

Algorithm `SSQR-Greedy`
Require: query q_j, $\rho_{j\ell}$ $(\ell = 1, \ldots, n-1)$, k
1: $U \leftarrow V - \{t, q_j\}$
2: $Q \leftarrow \emptyset$
3: **while** $|Q| < k$ **and** $\exists \ell : \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_\ell))] > w(j)$ **do**
4: $\quad i \leftarrow \arg\max_{\ell : q_\ell \in U} \{\rho_{j\ell}(\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_\ell))] - w(j))\}$
5: $\quad Q \leftarrow Q \cup \{q_i\}$
6: $\quad U \leftarrow U - \{q_i\}$

Figure 1: Greedy algorithm for single-step query recommendation.

THEOREM 5.3.

$$h(Q_{ALG}) \geq (1 - x) h(Q_{OPT}),$$

where x depends on the the unperturbed matrix and it is small for most non-pathological instances (possibly, $x \ll 1$).

The case of no incoming links. When q_j has no incoming links in the unperturbed Markov chain, it turns out that the solution provided by Algorithm `SSQR-Greedy` is optimal. In particular, in this case a random walk starting at q_ℓ has zero probability of terminating at q_j. We already observed in Section 4.2 that this case is of practical interest in the case of search done as an initial step for browsing, for example.

Remarks. If we replace $\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_\ell))]$ with $w(\ell)$ in the pseudo-code of Algorithm `SSQR-Greedy`, we obtain one of two natural heuristics, namely, maximizing the expected increase in the weight of the next query navigated by the user over the current one, provided the user follows a recommended link. In the experimental part we consider a slightly different version, that of maximizing $\rho_{j\ell} \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_\ell))]$.

6. EXPERIMENTS TO DETERMINE USER BEHAVIOR

In this section we present the dataset we used and our modeling approach. We mention again that our main goal is to design a fairly accurate but simple model to use for our optimization purposes.

6.1 Experimental framework

Dataset. We perform an experiment on the search-engine results page, for a small fraction of users. In this experiment, we remove the query recommendations (labeled "*Also try ...*" in the user interface) from the top of the page. As a control group, we sample a similar amount of normal sessions using the default interface during the same period of time.

We process the search sessions to segment them into *search goals* [10], which are sequences of queries corresponding to an atomic information need. For instance, the queries "`car battery`" and "`buy a car battery`" can be considered part of the same *goal*, but the query "`brake pads`", while related to the other queries (for the topic of cars) is on a different *goal*. Search goals contain a median of 2 queries.

Next, we aggregate information in a query-flow graph, by considering for every pair of queries, all the sessions in which those queries appear consecutively.

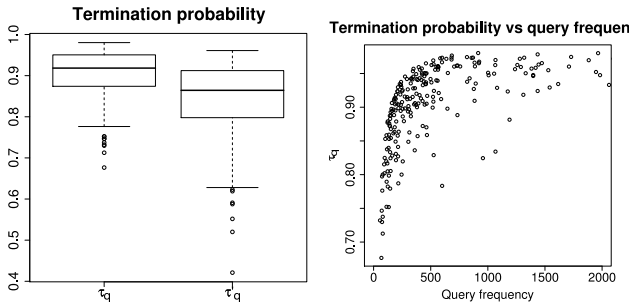
In the user-behavior characterization results below, we select queries q having frequency $\text{freq}(q) \geq 50$ and having at least 15 non-terminal transitions: $\sum_{q' \neq t} \text{freq}(q, q') \geq 15$. We also discard queries that generate spell suggestions, to avoid confusing spell correction with query recommendation.

Finally, we keep only the queries that match these conditions in both the experiment and the control group.

6.2 Impact of reformulations on the termination probability

The termination probability (the probability of stopping a search session at a query) depends on a number of factors, as it conflates both successful and unsuccessful queries. We observe that in general the more frequent a query is, the higher its termination probability, as shown in Figure 2(b).

With respect to the effect of query recommendations on this probability, it is clear that recommendations reduce the termination probability, as shown in the box-and-whisker diagram of Figure 2(a). For this query sample, without query recommendations in most cases the termination probability τ_q is between 0.8 and 1.0. When query recommendations are shown, the termination probability τ'_q is between 0.6 and 1.0, with $E[\tau_q] \approx 0.90$ and $E[\tau'_q] \approx 0.84$.



(a) Drop in termination probability (b) Frequent queries tend to have higher τ_q

Figure 2: Drop in termination probability with recommendations. τ'_q can be approximated by a linear function of τ_q

The relationship between τ_q and τ'_q can be roughly approximated by a linear function, as shown in Figure 3(a): $\tau'_q \approx 0.9\tau_q$. The linear model fits well this data, with Pearson's correlation coefficient $r = 0.82$. In this and the following plots, each point is a query and the size of its circle is proportional to its frequency in the data.

Actually, the drop in the termination probability observed in Figure 2(a) can be explained almost completely by the user clicks on the query recommendations, as $\tau'_q \approx \tau_q - 0.8\sum_{q' \in Q} \rho_{q,q'}$ ($r = 0.95$). In Figure 3(b) we can see the linear approximation.

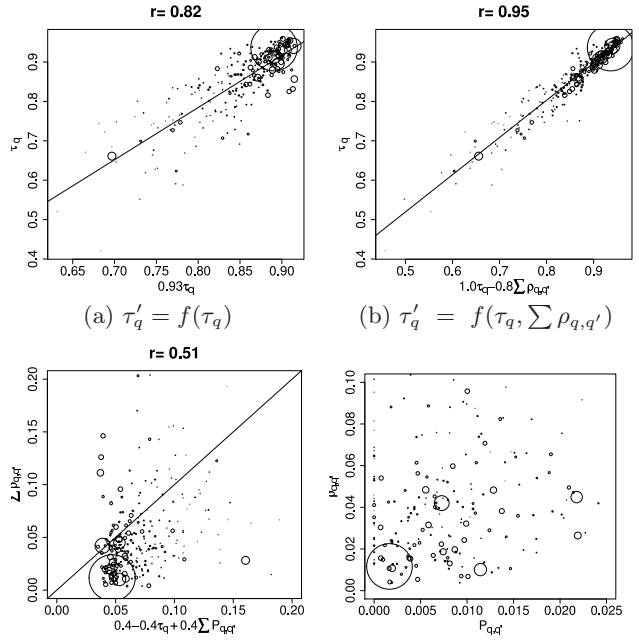
Finally, considering $\mathbf{P}_{q,q'}$ does not seem to help increase the accuracy, as the best linear we can obtain is $\tau'_q \approx \tau_q - 0.7\sum_{q' \in Q} \mathbf{P}_{q,q'}$ (r=0.82); which basically relies on the correlation between τ'_q and τ_q. The plot looks very similar to Figure 3(a) and is thus omitted.

We also gather two sets of weights for the queries. The first is a proxy of user's satisfaction: the click-through rate on the organic search results of a query. The second is a proxy for revenue for the search engine: the click-through rate of advertising in the page of search results for a query. These two sets of weights will be the weights $w(q)$ that we use in our experiments.

6.3 Impact on query transitions

We observe that 98.7% of the recommended queries at position 1 increase their transition probability with respect to the control group. On the other hand, the distribution of reformulations to queries that are not recommended ($q' \notin Q$) remains basically equal. We measured the Jensen-Shannon divergence between the probability distribution of the transitions to non-recommended queries, and it is on average 0.03 with 95% of the query pairs having a divergence of less than 0.08.

We are also interested in finding $\rho_{q,q'}$ given $\mathbf{P}_{q,q'}$. It turns out that $\rho_{q,q'}$ depends on a number of factors, and simple models based only on termination probability do not perform well. For instance, if we look at the sum of the perturbations, $\sum_{q' \in Q} \rho_{q,q'} \approx 0.4 - 0.4\tau_q + 0.4\sum_{q' \in Q} \mathbf{P}_{q,q'}$ (r=0.51), shown in Figure 3(c).



(a) $\tau'_q = f(\tau_q)$ (b) $\tau'_q = f(\tau_q, \sum \rho_{q,q'})$

(c) $\sum \rho_{q,q'} = f(\tau_q, \sum \mathbf{P}_{q,q'})$ (d) $\rho_{q,q'}$ vs $\mathbf{P}_{q,q'}$

Figure 3: The drop in the termination probability can be explained almost completely by the clicks in the recommended queries

A simple model is the following: $\rho_{q,q'} \approx 0.2 - 0.2\tau_q + 0.6\mathbf{P}_{q,q'}$ (r=0.41). Basically, the recommendation will be more clicked if the recommended query is done more frequently by users. However we have to point out that this does not hold deterministically, and for instance there are some queries with $\mathbf{P}_{q,q'} = 0$ that have $\rho_{q,q'} > 0$, as shown in Figure 3(d). For q, these queries are never written as reformulations by current users, but they are followed if shown to them as suggestions.

We computed several lexical features for each query pair, following [3], and found that if $J(q, q')$ is the Jaccard coefficient between the sets of character tri-grams of query strings q and q' (capturing basically if the queries are lexically related), then $\rho_{q,q'} \approx 0.2 - 0.2\tau_q + 0.8\mathbf{P}_{q,q'} + 0.1J(q, q')$ (r=0.50). Basically this model incorporates the fact that people will click on suggested queries that are similar (lexically) to their original query.

6.4 Correlation of $w(q')$ with $\mathbf{P}_{q,q'}$

Before attempting to solve the optimization problem, we may first ask if users naturally select queries with high weight on their own. For a fixed q, is $\mathbf{P}_{q,q'} \propto w(q')$? It does not seem to be the case. If we measure the correlation coefficient between these two values for a fixed query q, we observe an average correlation of around 0.1. More generally, even when we look at the relationship between $w(q')$ and $w(q)$ for pairs of reformulations done by users, we observe that users are not consistent in reformulating to either queries with higher or lower click-through rates than the queries they are at currently.

7. EXPERIMENTS COMPARING RECOMMENDATION ALGORITHMS

We present below the results of experiments comparing our heuristic with various other natural candidates.

7.1 Problem instances

Dataset. We take the top 420 queries by frequency and then follow all possible reformulations observed in the query-log up to distance 5 (distance=1 are direct reformulations).

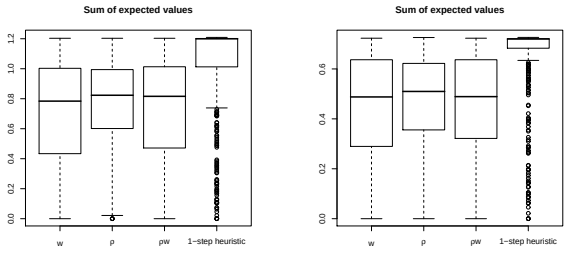
We use the two sets of weights described in Section 6.1. In terms of utility function we consider both approaches of Section 4.2. For the experiments where we use the set of weights corresponding to the click-through rate of organic search results, the utility is MAX-LAST, the click-through rate of the last page before termination – this gives an indication of the user's satisfaction during her search session. For the experiments where we use the set of weights corresponding to sponsored advertisements, the utility is MAX-SUM, the sum of the weights of the queries visited – this is a proxy of the expected number of ads clicked.

In the next two sections, we compare the performance of our approach with other natural heuristics, by using historical results in the next section and then by performing a user study. Our goal is two-fold. First, we want to examine whether simpler approaches can also optimize the expected future utility as much as our approach does. Second, since we optimize not over the next step, but over the entire session, we measure the decrease in the quality of the immediate recommendations.

Interestingly, our experimental analysis shows that our algorithms are significantly better than other heuristics with respect to the optimization goals we pursue, at the same time providing next-step recommendations whose quality is comparable to that achieved by heuristics that are explicitly designed to this purpose.

7.2 Comparison of our approach with other heuristics

First, we examine to what extent other natural heuristics optimize the objective that we are trying to optimize, the expected future utility. We consider three different heuristics. The first is to recommend the k queries that have the highest weight. The second imitates a simple recommendation system and it recommends the k queries that have the highest recommendation probability, that is, the queries q_ℓ with highest value $\rho_{j,\ell}$, assuming that the current query is query q_j. Finally, the third heuristic combines the previous two and it recommends the k queries that maximize the quantity $\rho_{j,\ell} \cdot w(\ell)$, in other words the queries that maximize the expected utility of the next query.



(a) Top-5 recommendations (b) Top-3 recommendations

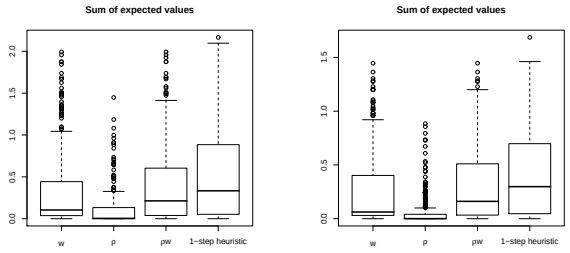
Figure 4: Average sum (over the top-k recommendations) of the expected weight of the last query before termination. Expected values are approximated using the 1-step heuristic. Weights correspond to click-through rate of the organic search results.

In Figure 4 we can see the comparison of the aforementioned three heuristics with our method when we perform k recommendations (Figure 4(a) shows the case of $k = 5$, while Figure 4(b) the case of $k = 3$). To create each plot we consider each query and we compute the top-k recommendations according to the various heuristics. For each recommendation we compute the expected future value. In fact, since this value is expensive to compute, we approximate it with the one-step heuristic of Section 5.1 (note that this is also the quantity that our approach maximizes). After computing these values we sum them for each query and then we consider the distribution over all queries. The plots in Figure 4 are the box-and-whisker diagrams of those distributions. Recall that for the organic search results, the utility of a path is MAX-LAST, the weight of the last query before termination, and in this particular experiment, the click-through rate (of organic search results) on the results page after the users performed the query.

By definition our heuristic performs better than the alternatives, so what the plots are depicting is whether more standard and "myopic" solutions suffice, or our method performs much better. From the plots we see that the three simple heuristics perform similarly, and indeed, much worse than our proposed solution. The overall improvement is about 45%.

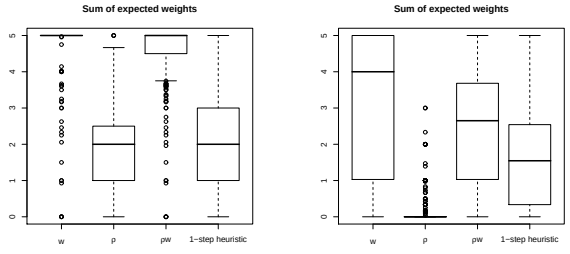
In Figure 5 we see the corresponding plots for the advertising scenario, namely when query weights correspond to the click-through rates on ads on the corresponding result pages, and when the utility is the sum of the weights of the pages that the user visited. Again we see that other heuristics cannot achieve the performance of ours, our heuristic giving values that are from 57% to 87% higher than the next best heuristic.

While in this paper we make the case that we should optimize query recommendations for the entire user session, nevertheless we would ideally like not to present recommendations to the user that appear significantly worse to the user, even if they provide better future results. In the case that the weight of a query is a measure of the user satisfaction (e.g., when the weight is the click-through rate of the corresponding pages as in our first experimental scenario)



(a) Top-5 recommendations (b) Top-3 recommendations

Figure 5: Average sum (over the top-k recommendations) of the expected sum of the weights of the queries. Expected values are approximated using the 1-step heuristic. Weights correspond to click-through rate of the sponsored advertisements.



(a) Search results (b) Sponsored ads

Figure 6: Sum of the weights of the weights of top-5 recommendations. Weights correspond to (a) click-through rate of organic search results, and (b) sponsored advertisements.

we would like the weight of the recommended queries to be large for our heuristic as well. It is not hard for someone to construct Markov chains where this can happen, and in fact, in general the one-step heuristic will perform poorly under this measure. In Figures 6(a) and 6(b) we see that this can happen in practice as well: the weights of the queries recommended are much lower than the best possible (we show results for top-5 recommendations, plots for top-3 recommendations are analogous). To explore this matter more, we perform a user study in the next section, to measure the extent to which optimizing over the entire future reduces the quality of the immediate recommendations and to examine to what extent a lower weight corresponds to a much lower user satisfaction.

7.3 User study

To address the issues raised at the end of the previous subsection, we conducted a user study to test if the recommended queries are acceptable as recommendations to users. For this, we ran our approach and the heuristics and compared the top-3 recommendations side-by-side. We then paired systems at random and showed to an assessor: (*i*) the original query, (*ii*) the top-3 recommendations generated by one system, and (*iii*) the top-3 recommendations generated by the other system. The identity of the system used for generating each recommendation was not present in the interface. For each query (original, or recommended), we asked a web search engine to retrieve the top-3 results, which were also shown to provide some context about the query and the recommendations.

We asked the panel of assessors (3 of the authors of this paper) to answer: *which system provides better recommendations?*. The options were (a) the first set is better, (b) the second set is better, (c) both sets are similar. We collected 980 such assessments.

This assessment task was highly subjective and Cohen's κ statistics of the inter-assessor agreement (on a set of 50 queries for which their assessments overlapped) shows $\kappa = 0.61$ which can be interpreted as a substantial level of agreement.

Considering the cases in which the assessors declared one set of recommendations to be better, we observed that the method based on ρw out-performed the method based on ρ in about 59% of the cases in which they were paired, which is significant at $p = 0.03$. For the other pairs of systems, we did not observe a significant ($p < 0.1$) advantage of one system over the other.

These results suggest that the recommendations generated by our method, are not perceived as being worse or better by users, while still leading them through paths that have significantly larger utility.

8. CONCLUSIONS

We have shown an approach to query recommendation that is based on casting this problem in an optimization framework, in which we perturb users' query-reformulation paths to maximize the expected value of some suitable utility function defined over search sessions. We defined two utility functions Max-Last and Max-Sum which, respectively, formalize the goals of reaching a valuable destination or traversing many valuable nodes. We have shown that this problem is in general NP-hard, but that we can provide effective and efficient approximation algorithms for it, with provable performance in significant cases. Finally, we have implemented our approximation heuristics and tested them on real test sets, also carrying out a user study that confirms that our techniques can be used to generate query recommendations that are perceived similar in quality to what users would consider more relevant to their search goals, but that at the same time bias users' browsing along reformulation paths that achieve a much higher utility than without such assistance.

Both our modeling framework and our solution approach are general and can be applied to various settings by modifying the interpretation of weights, the exact definition of utility, the transition probability matrix, and so on. While our initial motivation was the query reformulation problem, we believe that it can be applied to other settings in which users' behavior can be modeled as a Markov process.

Two key aspects of our method require further development: the way of assessing the utility of individual queries and the model for estimating the response of the user in the presence of query reformulations. The methods we have described in this paper can benefit from future improvements in these two areas. Another interesting question is the incorporation of diversity in the entire framework. On the theoretical side, providing an approximation algorithm for the general multi-step recommendation problem seems to be the hardest open problem from this research.

Acknowledgements: the authors would like to thank Linda Wang, Su-Lin Wu, and colleagues from the Search Sciences group at Yahoo! for their help in obtaining the experimental data for this paper. Finally, we would like to thank the anonymous reviewers, for their comments improved the presentation of this paper.

9. REFERENCES

[1] Baeza-Yates, R. Graphs from search engine queries. In *SOFSEM '07: Proc. of the 33rd conf. on Current Trends in Theory and Practice of Computer Science* (2007), Springer-Verlag, pp. 1–8.

[2] Baeza-yates, R., Hurtado, C., and Mendoza, M. Query recommendation using query logs in search engines. In *Proc. of int. Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Crete* (2004), Springer, pp. 588–596.

[3] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. The query-flow graph: model and applications. In *CIKM '08: Proceeding of the 17th ACM conf. on Information and knowledge mining* (New York, NY, USA, 2008), ACM, pp. 609–618.

[4] Boldi, P., Bonchi, F., Castillo, C., Donato, D., and Vigna, S. Query suggestions using query-flow graphs. In *WSCD '09: Proc. of the 2009 workshop on Web Search Click Data* (New York, NY, USA, 2009), ACM, pp. 56–63.

[5] Chakrabarti, D., Kumar, R., and Punera, K. Quicklink selection for navigational query results. In *Proc. of the 18th int. conf. on World wide web (WWW)* (New York, NY, USA, 2009), ACM, pp. 391–400.

[6] Chien, S., Dwork, C., Kumar, R., and Sivakumar, D. Link evolution: Analysis and algorithms. *Internet Mathematics 1*, 3 (2004), 277–304.

[7] Czyzowicz, J., Kranakis, E., Krizanc, D., Pelc, A., and Martin, M. V. Enhancing hyperlink structure for improving web performance. *Journal of Web Engineering 1*, 2 (2003), 93–127.

[8] Fonseca, B. M., Golgher, P. B., de Moura, E. S., and Ziviani, N. Using association rules to discover search engines related queries. In *Proc. of the 1st conf. on Latin American Web (LA-WEB)* (2003), IEEE Computer Society, p. 66.

[9] Fox, S., Karnawat, K., Mydland, M., Dumais, S., and White, T. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst. 23*, 2 (April 2005), 147–168.

[10] Jones, R., and Klinkner, K. L. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *conf. on Information and Knowledge Management (CIKM)* (October 2008), ACM Press.

[11] Kranakis. Approximate hotlink assignment. *Information Processing Letters 90*, 3 (May 2004), 121–128.

[12] Langville, A. N., and Meyer, C. D. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[13] Wen, J.-R., Nie, J.-Y., and Zhang, H.-J. Clustering user queries of a search engine. In *Proc. of the 10th int. conf. on World Wide Web (WWW)* (2001), ACM, pp. 162–168.

[14] Zhang, Z., and Nasraoui, O. Mining search engine query logs for query recommendation. In *Proc. of the 15th int. conf. on World Wide Web (WWW)* (2006), ACM, pp. 1039–1040.