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 Labs Barcelona, Spain

ABSTRACT

Query recommendations are an integral part of modern search engines. Their goal is to facilitate users' search tasks, as well as help them discover and 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 user-querying behavior by a probabilistic reformulation graph, or query-flow graph [Boldi et al. CIKM 2008], so that the 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 performing a random walk on the query-flow graph. Furthermore, 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 come up 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 we consider, and we suggest efficient algorithmic solutions. We validate our models and algorithms with extensive experimentation.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

1. INTRODUCTION

A prominent feature of modern search engines is the presence of query recommendations in response to user queries.

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

Submitted for confidential review.
August 2009.

Query recommendations serve several purposes: correcting possible spelling errors, 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 a query recommendation is *spell 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`", "`chocolate chip cookie recipe`", but also to related queries such as "`brownies`", "`baking`", and so on.

A key enabling technology for query-recommendation 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 found on query logs to find potentially successful queries that are relevant to what the user is searching [2–4, 15, 16]. Current state-of-the-art methods often produce relevant query recommendations, but often there is no clear objective to optimize and the query-recommendation algorithms 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 methodology makes the following assumptions, which are also the main ingredients of our approach:

- First, we assume that it is possible to aggregate historical information from the query logs to build a query-reformulation graph G [3]. The nodes of the 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 *weighted random walks* on this graph.
- Second, we assume that the queries in the query-flow graph have intrinsic score values $w(q)$ that are increasing on a desired property of the query q, for example, the probability that users that issue 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 are visiting. The higher the total value collected the higher is the overall utility of the system.

- Last, we assume that query recommendations can be viewed as *shortcuts* that perturb the transition probabilities in the query-flow graph. The motivation is that during the random walk, 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. Thus, given the desiderata to maximize the overall utility of the system, we formulate the query-recommendation problem as the problem of deciding where to add shortcuts in a graph so that the expected utility collected during a random walk is maximized.

From the algorithmic point of view, the problem of finding the best k shortcut edges to add at each node in order to improve 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.

Assumptions. We discuss below the most important assumptions we make in the framework we propose.

First, we discuss notions of score values $w(q)$ that can be used to define suitable utility functions to optimize. One possible choice is to use values $w(q)$ that model the quality of the query q and the satisfaction of the user after submitting the query q and inspecting the results. Measuring user satisfaction is a challenging task on its own, and we resort to user clicks on search engine results for submitted queries as a surrogate for user satisfaction. Another choice for the values $w(q)$, resulting in a different utility function, is to use expected click-through rate to ads, or expected revenue. Notice that even when attempting to explicitly optimize the revenue for the search engine, one has to provide valuable and interesting recommendations to the users: recommendations that do not keep users interested and engaged with the search engine are not going to be good solutions for the revenue-related utility objective.

Second, we discuss how to estimate the effect of query recommendations on transition probabilities. Even if a query q' has high value $w(q')$ but q is not related to q', the transition probability of a recommendation/shortcut (q, q') will probably be small, since users who type q are not likely to be interested in clicking q'. This is a challenging problem in its own right; based on empirical observations, we propose a simple model in which the transition probability for a recommendation (q, q') is related to the transition probability on the edge (q, q') in the query-flow graph. Learning these transition probabilities is an interesting statistical problem involving explore-exploit ideas and it is outside of the scope of this paper.

We note here that our framework is very general and works with any other model used to estimate the utility of a query or the recommendation transition probabilities. In particular, improving the models used to estimate transition probabilities will improve the performance of our algorithms.

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 users' future browsing behavior in order, for example, to deliver an entertaining experience to the user, involving 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, contents should be recommended according to some "interestingness" criteria, and they should depend not only on the next step in the user navigation, but also on her future browsing path.

In general, our work can be applied to recommendation systems where we take into account the entire future browsing behavior of the user and where the ideal recommendation to propose might depend on various factors.

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, while Section 4 formalizes the task of providing effective recommendations as a suitably defined optimization problem. Section 5 addresses the complexity of the general problem we consider. In this section, we further present effective algorithms for the general problem and for an important, special case. For the latter, we provide a thorough analysis of the structural properties of optimal solutions and we present an easy, greedy heuristic with provably good performance on instances of practical interest. In Section 6, we present our analysis of historical data to model the user browsing behavior and query utility, while in Section 7 we build on this model to solve the query recommendation problem using the methodology presented in Section 5 and we compare it with other natural baselines. Finally, in Section 8 we conclude and present some ideas for future work.

2. PREVIOUS WORK

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

A first line of research has focused on the task of finding queries that are related to those submitted by the user. To this purpose, strategies based on measures of query similarity [2], on query clustering [15] or association rules [8] have been proposed.

A different approach has been taken by Zhang and Nasraoui [16], who attempted to model the users' sequential search behavior. To this purpose, they considered query graphs in which nodes are the queries and arcs link consecutive queries in the same user session. In [16], arcs 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 in [3]. Here, the authors introduce the concept of a *query-flow graph*. The authors define a graph in which nodes are the queries appearing in the query log and arc (q, q') is present if at least one user submitted query q' after submitting q in the same session. Arcs come with weights that, generally speaking, estimate the probability that a query transition connects related queries, and are computed from aggregate information extracted from the query log. In [3], the authors considered 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. We considered above the approaches that are most closely related to the contents of this paper. An overview of literature, techniques and a broader range of applications of graphs extracted from query logs is presented in [1].

Our work uses the query-flow–graph framework. However, while in previous work the query-flow graph is used to recommend queries that are related to the search goals of the user, the novelty of our approach 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.

Perturbation of Markov chains. The applications that we consider in this paper revolve around the problem of perturbing a Markov chain so as to optimize some suitable function of it[1]. While perturbation theory is a well assessed area of matrix analysis, there is no large literature on sensitivity analysis of probability distributions defined over a (possibly non-ergodic) Markov chain. In fact, most literature about it concerns 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 apparently 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).

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 ([11] considers the special case of a tree) 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 paper [5] considers the problem of adding "quicklinks" (i.e., shortcuts) to search engine results to optimize some utility measure. More in detail, the authors assume that, upon submitting a query, a user is returned the most relevant page for the query, which in some cases will be the home page of some Web site W. This in turn becomes the source of a possible, further user navigation of Web site W, which they call *trail.* A trail will possibly meet the user's information needs. The problem considered in [5] is recommending at most k further links that are likely to meet the user's information needs. To this purpose, 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's information needs and (ii) provided this happens, the page gives a benefit described by a suitable benefit function. The goal is choosing 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.

The scenario we consider is similar in spirit to cthose considered in [7] and [11], though with some remarkable differences: they consider DAGs and trees in particular, while we consider general graphs. In [7, 11], probabilities are assigned to leaves of a tree suitably defined over the graph and rooted at the home page, whereas in our case each node is assigned its visiting probability in the underlying random walk. In [7,11], 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. Furthermore, in [7,11] the goal is minimizing the expected distance travelled 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. The approach in [5] differs from ours in significant ways: i) the problem studied is different, since the goal in [5] is providing 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 will maximize the benefit over the entire navigation path of the user, i.e., the recommendation is for other result lists; ii) the objective to optimize in [5] depends on the set of suggested quicklinks and on a measure of noticeability assigned to nodes, which is inferred from click-through data. In our case, it depends on the Markov chain underlying the query-flow graph, with the complication that this is itself perturbed by the addition of recommendation links, thus modifying the objective function itself and complicating the optimization task; iii) the approach of [5] requires knowledge about user trails which is collected from toolbar data, while our approach relies on query-log data.

3. USER-BEHAVIOR MODEL

3.1 Query reformulation graph

Users interact with a search engine in sessions. During a *session*, a user submits a sequence of queries within a rea-

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

sonably short time interval. It is clear that, defined in this way, a session may consist of a sequence of topically-related queries, or it may contain unrelated queries, corresponding to unrelated tasks performed by the same user during that period. More precise and refined definitions of user sessions have been proposed in the literature [10, 14]. In our experiments we will use *search goals* as defined in [10], which are sequences of queries with the same atomic intent.

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 reformulations done by users.

We also 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 session end t. Accordingly, $\mathbf{P}_{n,i} = 0\ \forall i \neq n, \mathbf{P}_{n,n} = 1$, so that the only possible transition from the terminal state is a self-loop. 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 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 users' 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 $w : N \to \mathbb{R}$ 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 later). We will assume $w(t) = 0$, given that t is not an actual query. The generic weight $w(q)$ models the quality of a query q, for instance, how satisfied a user is with the result list obtained upon submitting query 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 to 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 further 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 clear from 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 $-\mathbf{P}_{q,q'} \leq \rho^{\mathbf{P}}_{q,q'}(Q) \leq 1 - \mathbf{P}_{q,q'}$). Note that this definition provides a pretty general framework that can model complex interactions, such as cases of negative dependence in the transition probabilities to similar queries. Of course, a perturbation of $\mathbf{P}_{q,\cdot}$ affects $\mathbf{P}_{q,t} = \tau_q$. More precisely, $\tau'_q = \mathbf{P}'_{q,t} = 1 - \sum_{q' \in V\setminus\{t\}} \mathbf{P}'_{q,q'}$.

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 $\mathbf{P}$, 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 results extend to these more general settings as well.

Properties of query-recommendation models. Obviously when the user is at query q, it does not pay off 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, then we also have 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. The properties we state in the previous paragraphs 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), that is, the fact that 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, termination probabilities are reduced from $\tau_q \approx 0.90$ to $\tau'_q \approx 0.84$, while at the same time, we observe that $\sum_{q'} \rho^{\mathbf{P}}_{q,q'}(Q) \approx 0.06$, thus supporting assumption ii) above. Details are given in Section 6. In addition our experimental results suggest an even simpler model, in which the values $\rho^{\mathbf{P}}_{q,q'}(Q)$ depend linearly on the values $\mathbf{P}_{q,q'}(Q)$.

4. PROBLEM STATEMENT

The general problem 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. Henceforth, we assume that we can provide at most k recommendations per query. The restriction to k recommendations per query represents screen real-state constraints.

4.1 Problem input

We assume we can estimate $\rho^{\mathbf{P}}_{q,q'}(Q)$; determining these values itself 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 the $\mathbf{P}_{q,q'}(Q)$'s.

We also assume we can estimate $w(q)$. Estimating this quantity can be hard, e.g., when it corresponds to estimating CTR on ads, or even harder when it estimates user satisfaction [9]. For our experiments we used click-through-rates as a proxy, as we have observed consistently that the majority of clicked documents are relevant for user's 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 one (before terminating by visiting t).

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

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

The first choice can capture scenarios such as maximizing total user satisfaction through the entire session (by setting, for example, $w(\cdot)$ to be the number of clicked results), 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}. In the remainder, we sometimes need to consider the special case in which $\pi_0 = \mathbf{e}_q$ for some $q \in \{1, \dots, n-1\}$, with $\mathbf{e}_q$ the q-th canonical column vector. Given the transition matrix P of the underlying query graph, perturbation functions $\rho^P_{\cdot,\cdot}(\cdot)$, 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, $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{P'}[U(\mathbf{path}(i))]$, where $\mathbf{P'}$ is such that, for every q, $\mathbf{P}'_{q,\cdot} = \mathbf{P}_{q,\cdot} + \rho_{q,\cdot}(Q(q))$, with $Q(q)$ 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.

More in detail, we focus on a particular query q. 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, i.e., $Q(q') = \emptyset$, for $q' \neq q$.

5. ALGORITHMS

We next 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 look at heuristics for solving both versions of the problem.

5.1 Solving the problem

Complexity. The query recommendation problem is NP-hard in general. In fact, it is possible to prove that even the single-step query recommendation problem is NP-hard. This is stated in the following.

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 just 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 $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{P'}[U(\mathbf{path}(i))]$, π_0 being 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 application realistic solutions, we propose below a heuristic in which choices are performed independently for every node of the query-flow graph. To this purpose, first notice that, in the special case $\pi_0 = \mathbf{e}_q$ for some q, we can write the optimization problem for random walks starting at node q 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

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

according to which of the utility functions defined in Section 4.2 we consider. The first corresponds to the case that the utility equals the sum of the weights in the path, while the second to 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

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

where the equality now follows since the $\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(q'))]$'s 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's browsing session. Now we only need to compute the expressions $V_{q'} \doteq \mathbf{E}^{P}[U(\mathbf{path}(q'))]$ and write for each query q':

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

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

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

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

5.2 Single-step query recommendations

In this section, we consider the single-step query recommendation problem[2]. Given a query q, our goal is to choose a set Q of at most k recommendations to propose to users viewing q's search engine results, so as to optimize some utility function.

In the remainder of the paper, we label queries $q_1, \ldots, q_{n-1}$ and we assume $\mathbf{P}$'s i-th row corresponds to q_i. Our goal is to optimize $\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q))]$, where $\mathbf{P}'$ is the perturbed matrix, and for utility we use the second definition of Section 4.2, where the utility is the weight of the last query before termination. We assume 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 $\tilde{\mathbf{P}}$ has the following structure:

$$\mathbf{P} = \left(\begin{array}{c|c} \tilde{\mathbf{P}} & \mathbf{P}_n \\ \hline \underline{\mathbf{0}}_{1 \times n-1} & 1 \end{array} \right),$$

where $\tilde{\mathbf{P}}$ is sub-stochastic (elements are nonnegative and the sum of the rows is at most 1) and $\mathbf{P}_n$ is an $(n-1) \times 1$ column vector whose i-th component is the termination probability at q_i. We denote by $\pi_0 \in [0,1]^{n-1}$ the initial distribution, that is, the i-th component of π_0 gives the probability that the random walk starts at q_i (we assume that the probability to start at the terminating state q_n is 0 and we consider the probability distribution over the rest of the states). We further define π as the $(n-1)$-dimensional column vector whose i-th component gives the probability that q_i is the last query visited by the user in her random walk over the query-flow graph. Note that we are interested in real queries, so we omit the component of this vector corresponding to the absorbing state of the Markov chain. Let $\mathbf{e}_i$ be the i-th canonical column vector. We denote by π^l the above defined probability vector when $\pi_0 = \mathbf{e}_l$.

Assume that we add recommendations to the j-th node q_j of the query graph. We let $\hat{\mathbf{P}}$ denote the perturbed version of $\tilde{\mathbf{P}}$ and $\hat{\mathbf{P}}_n$ be the perturbation of $\mathbf{P}_n$. This means that $\hat{\mathbf{P}}_{in} = \mathbf{P}_{in}$ if $i \neq j$, while $\hat{\mathbf{P}}_{jn} = \mathbf{P}_{jn} - \sum_{q_s \in Q} \rho_{js}$. First we prove the following fact, which describes an important property of optimal solutions.

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

The proof of Fact 5.2 shows that optimizing $\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(\pi_0))]$ amounts to maximizing $\sum_{s=1}^{n-1} [(\mathbf{I} - \hat{\mathbf{P}})^{-1} \hat{\mathbf{P}}_n]_{js} w(s)$. On the other hand, given $\tilde{\mathbf{P}}$, q_j, the ρ_{jl}'s and the weights, this is solely a function $h(Q)$ of Q, the set of recommendations:

$$h(Q) = \sum_{s=1}^{n-1} [(\mathbf{I} - \hat{\mathbf{P}})^{-1} \hat{\mathbf{P}}_n]_{js} w(s).$$

We provide a simple greedy algorithm to maximize $h(Q)$ and we prove that it performs close to optimum in all cases of practical interest.

```
Algorithm SSQR-Greedy
Require: query $q_j$, $\rho_{jl}$ $(l = 1, \ldots, n-1)$, integer k
 1: U = V - {$t$, $q_j$}
 2: Q = $\emptyset$
 3: while |Q| < k AND ($\exists$ l: $\mathbf{E}^{\mathbf{P}}[U(\mathtt{path}(\mathbf{e}_l))] > w(j)$)
    do
 4:    i = $\arg\max_{l: q_l \in U} \{\rho_{jl}(\mathbf{E}^{\mathbf{P}}[U(\mathtt{path}(\mathbf{e}_l))] - w(j))\}$
 5:    Q = Q $\cup$ {$q_i$}
 6:    U = U - {$q_i$}
```

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

The algorithm is given in Figure 8 and it requires the computation of the expected utility achieved by a random walk starting at q_l, for $l = 1, \ldots, n-1$, computed with respect to *the unperturbed matrix* $\mathbf{P}$. These quantities can be pre-computed once for all and require the computation of $(\mathbf{I} - \tilde{\mathbf{P}})^{-1} \mathbf{P}_n$. Denote by Q_{ALG} and Q_{OPT} the algorithm's and the optimum's choices for Q. Algorithm `SSQR-Greedy` achieves a performance that is close to optimum in cases of practical relevance, as shown by the following

THEOREM 5.3. *Let $x = \max_Q \sum_{q_l \in Q} \frac{\rho_{jl}}{\mathbf{P}_{jn}} \pi_j^l$. Then:*

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

Note that, in practice, x should be small for most non-pathological instances, (possibly, $x << 1$). In fact, it is

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

possible to prove that the generic $x(Q) = \frac{1}{\mathbf{P}_{jn}} \sum_{q_l \in Q} \rho_{jl} \pi_j^l$ is, up to the factor $1/\mathbf{P}_{jn}$, the probability that a random walk starting at q_j follows one of the recommendations as a first step and then proceeds for an arbitrary number of steps along a possibly non simple path, terminating at q_j itself. On the other hand, $\mathbf{P}_{jn}$ can be large in practice, as shown in Figure 2(a).

The case of no incoming links. Note that, when q_j has no incoming links in the unperturbed Markov chain, the solution provided by Algorithm `SSQR-Greedy` is optimal. This trivially follows by observing that, no matter which is the set Q of recommendations chosen by the algorithm, we have $x(Q) = 0$ in this case. This in turn is a consequence of the considerations in the last paragraph above. 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.

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. In mid-2009, we carried out an experiment on the search engine results page, for a small fraction of users. In this experiment, we removed the query recommendations (labeled "*Also try ...*" in the user interface) from the top of the page. As a control group, we sampled a similar amount of normal sessions from the same period of time.

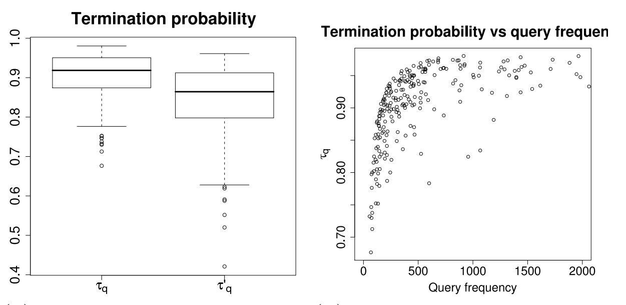
We processed the search sessions to segment them into search goals [10], which are sequences of queries corresponding to an atomic information need, containing a median of 2 queries. Next, we aggregated information in a query-flow graph, by considering for every pair of queries, all the sessions in which those queries appear consecutively.

In the user-behavior characterization results below, we selected 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 discarded queries that generated spell suggestions, to avoid confusing spell correction with query recommendation. Finally, we kept only the queries that matched 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 Figure 2(a). For this query sample, without query recommendations in most cases the termination probability is between 0.8 and 1.0. When query recommendations are shown, the termination probability 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 clicks on the query recommendations, as $\tau'_q \approx \tau_q - 0.8 \sum_{q' \in Q} \rho_{q,q'}$ (r=0.95).

Finally, if we were to predict the termination probability based on the behavior of users in the absence of recommendations, we could state the model $\tau'_q \approx \tau_q - 0.7 \sum_{q' \in Q} \mathbf{P}_{q,q'}$ (r=0.82); which basically arises from the correlation with the termination probability. The plot looks very similar to Figure 3(a) and is thus omitted.

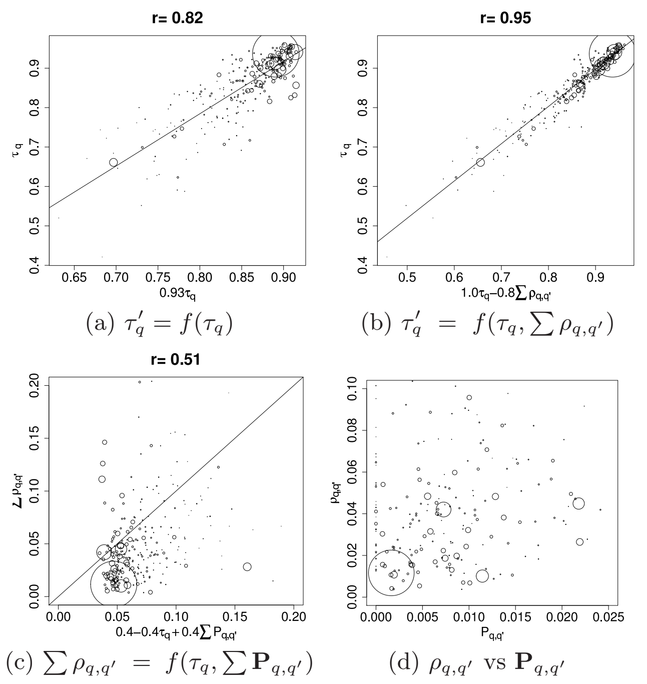
We also gathered two sets of weights for the queries. The first is a proxy of user's satisfaction and is the click-through rate of organic search results in the page returned by the search engine when the user performs the query. The second is a proxy for revenue for the search engine and it is the click-through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

6.3 Impact of recommendations on query transitions

With recommendations, 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 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



(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

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).

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. Formally, for a fixed q, is $P(q, q') \propto w(q')$? 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 took the top 420 queries by frequency and then followed all possible reformulations observed in the query-log up to distance 5 (distance=1 are direct reformulations).

We used 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 equals 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 of the user's walk on the graph equals the sum of the weights of the queries visited – this is a proxy of the total revenue gathered by the search engine. For a more detailed description of these objective functions refer to Section 4.2.

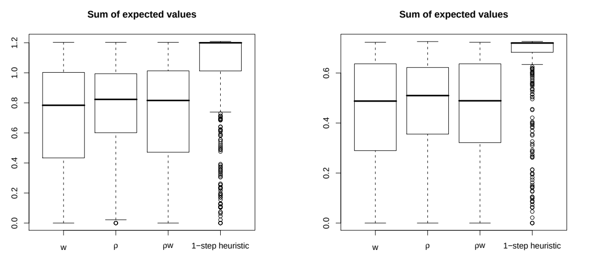
In the next two sections, we compare the performance of our approach with other natural heuristics, by using historical results in the next section and then by performing a user study. Our goal is two-fold. First we want to examine whether simpler approaches can also optimize the expected future utility as much as our approach optimizes. Second, since we optimize not only over the next step but over the entire session, we measure if we decrease the quality of the immediate recommendations. It is clear that, in general, our algorithms might not be the best possible with respect to the quality of the next recommendation suggested, for which other heuristics, explicitly designed to this purpose, might prove more effective.

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

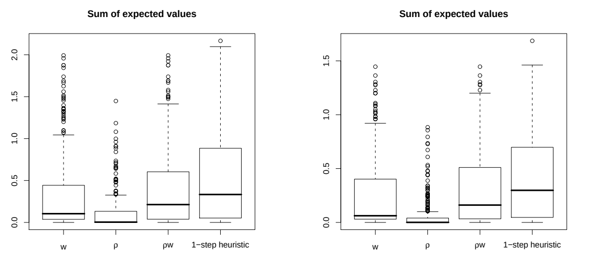
First we examine to what extent other natural heuristics optimize the objective that we are trying to optimize, the xpected future utility. We consider three different heuristics. The first is to recommend the k queries that have the highest weight. The second imitates a simple recommendation systems 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.

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



(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 approximated using the 1-step heuristic. Weights correspond to click-through rate of the organic search results.



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

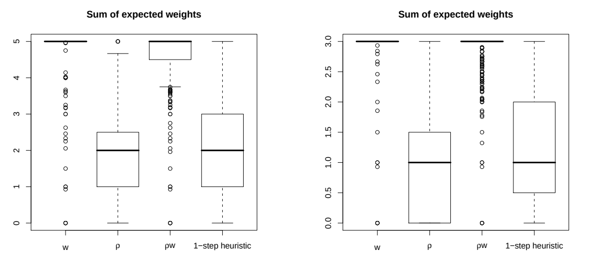
Figure 5: Average sum (over the top-k recommendations) of the expected sum of the weights of the queries. Expected values approximated using the 1-step heuristic. Weights correspond to click-through rate of the sponsored advertisements.

in Figure 4 are the box-and-whisker diagrams of those distributions. Recall that for the organic results utility of a path is the weight of the last query before termination, and in this particular experiment, the click-through rate (of organic search results) on the resulted 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%.

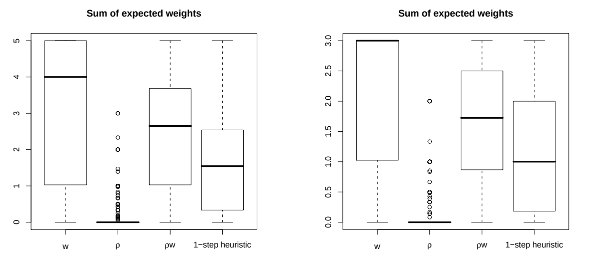
In Figure 5 we see the corresponding plots for the advertising scenario, namely when query weights correspond to the click-through rates on ads on the corresponding result pages, and when the utility is the sum of the weights of the pages that the user visited. Again we see that other heuristics cannot 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 not present recommendations to the user that appear significantly worse to the user, even



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

Figure 6: Sum of the weights of the top-k recommendations. Weights correspond to click-through rate of the organic search results.



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

Figure 7: Sum of the weights of the top-k recommendations. Weights correspond to click-through rate of the sponsored advertisements.

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) 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 and 7 we see that this can happen in practice as well: the weights of the queries recommended are much lower than the best possible. 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 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. And 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.

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), Creete* (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] Meyer, C. D., Ed. *Matrix analysis and applied linear algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[14] Radlinski, F., and Joachims, T. Query chains: learning to rank from implicit feedback. In *Proceeding of the 11th int. conf. on Knowledge discovery in data mining (KDD)* (New York, NY, USA, 2005), ACM Press, pp. 239–248.

[15] 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.

[16] 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.

APPENDIX

A. NP-COMPLETENESS

Theorem 5.1.

Proof. The proof follows from the fact that we can encode a problem instance just in the function ρ. For completeness we provide the details, by reducing the maximum coverage problem to it. (In the maximum coverage problem the input is a set of n elements, m sets where each set covers a subset of the elements, and an integer $k < m$. The objective is to return k sets that cover the maximum number of elements.) Given an instance of the maximum coverage problem, we create a graph with $m+1$ queries $\{q_0, \ldots, q_m\}$ and a terminal state t. For all i we set $P_{ij} = 1$ if $j = t$ and 0 otherwise. Also we set $w(q_0) = 0$ and $w(q_i) = 1$, for $i \geq 1$. We assume that the user is initially at query q_0 and we can add k recommendations to each query.

To encode the maximum coverage problem to the ρ function we set $\rho_{q_i,q_j}(Q) = 0$ for all $i \geq 1$ and all j, Q. For the initial query q_0 we set

$$\rho_{q_0,q_j}(Q) = \frac{\text{number of elements covered by the sets in } Q}{nm}.$$

If the recommendations from q_0 is the set Q, then the total expected utility from node q_0 is

$$\sum_{q_j \in Q} \rho_{q_0,q_j} w(j)$$

since from any other query the set of recommendations does not affect the expected value. This equals to

$$k \frac{\text{number of elements covered by the sets in } Q}{nm}.$$

Subject to the conditions that we make no more than k recommendations, this quantity is maximized when we chose the k queries corresponding to the sets that cover the most elements. $\square$

B. SINGLE-STEP QUERY RECOMMENDATIONS

In this section, we consider the single-step query recommendation problem. Given a query q, our goal is to choose a set Q of at most k recommendations to propose to users viewing q's screenshot, so as to optimize some utility function. We have already seen in Subsection 5.1 that the single-step recommendation problem is NP-hard.

In the sequel, we label queries $q_1, \ldots, q_{n-1}$ and we assume $\mathbf{P}$'s i-th row corresponds to q_i. Our goal is to optimize $\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(q))]$, where $\mathbf{P}'$ is the perturbed matrix, and for utility we use the second definition of Section 4.2, where the utility is the weight of the last query before termination. We assume 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. From the definitions given in Section 3.1, we also have that:

$$\mathbf{P} = \left(\begin{array}{c|c} \tilde{\mathbf{P}} & \mathbf{P}_n \\ \hline \underline{\mathbf{0}}_{1 \times n-1} & 1 \end{array} \right),$$

where $\tilde{\mathbf{P}}$ is sub-stochastic (elements are nonnegative and the sum of the rows is at most 1) and $\mathbf{P}_n$ is an $(n-1) \times 1$ column vector whose i-th component is the termination probability at q_i.

In the sequel, we denote by $\pi_0 \in [0,1]^{n-1}$ the initial distribution, that is, the i-th component of π_0 gives the probability that the random walk starts at q_i (we assume that the probability to start at the terminating state q_n is 0 and we consider the probability distribution over the rest of the states). We define $\pi(t)$ as the $(n-1)$-dimensional column vector whose i-th component gives the probability that q_i is the last query visited by the user *and* this event occurs at step t. Note that we are interested in real queries, so we omit the component of this vector corresponding to the absorbing state of the Markov chain. Let $\mathbf{e}_i$ be the i-th canonical column vector. We denote by $\pi^l(t)$ the above defined probability vector when $\pi_0 = \mathbf{e}_l$. We further denote by π the overall distribution of the termination probability, so that $\pi_i = \sum_{t=0}^{\infty} \pi_i(t)$.[3]

In the sequel, assume that we add recommendations to the j-th node q_j of the query graph. We let $\hat{\mathbf{P}}$ denote the perturbed version of $\tilde{\mathbf{P}}$ and $\hat{\mathbf{P}}_n$ be the perturbation of $\mathbf{P}_n$. This means that $\hat{\mathbf{P}}_{in} = \mathbf{P}_{in}$ if $i \neq j$, while $\hat{\mathbf{P}}_{jn} = \mathbf{P}_{jn} - \sum_{q_s \in Q} \rho_{js}$. First we prove the following fact.

Fact B.1. *Assume we add a set Q of recommendations at q_j, $|Q| \leq k$, so as to maximize $\sum_{i=1}^{n-1} \pi_{0i} \mathbf{E}^{\mathbf{P}'}[U(\boldsymbol{path}(q_i))]$, where*

$$\mathbf{P}' = \left(\begin{array}{c|c} \hat{\mathbf{P}} & \hat{\mathbf{P}}_n \\ \hline \underline{\mathbf{0}}_{1 \times n-1} & 1 \end{array} \right),$$

Then, the optimal solution is independent of π_0.

Proof. Our first task is to derive an explixit expression of the perturbed termination probability vector $\hat{\pi}(t)$ ($\hat{\pi}(t)$ is defined similarly to $\pi(t)$ with the difference that it depends on matrix $\mathbf{P}'$ instead of $\mathbf{P}$). To this aim, we note that, for every i, $\hat{\pi}_i(t)$ is the sum of the probabilities of all walks on the Markov chain that (i) bring from any of the possible initial nodes to q_i in t steps and (ii) have a final transition from q_i to the absorbing state q_n. Note that by this definition q_n is only traversed once, precisely in the final transition. As a result, we have $\hat{\pi}(t)^T = \pi_0^T \hat{\mathbf{P}}^t \hat{\mathbf{P}}_n$. Furthermore, we can express $\hat{\pi}(t)$ as $\hat{\pi}(t) = \overline{\pi}(t) + \tilde{\pi}(t)$. $\overline{\pi}(t)$ and $\tilde{\pi}(t)$ are such that $\overline{\pi}_i(t)$ gives the probability that the random walk terminates at q_i without ever traversing q_j, while $\tilde{\pi}_i(t)$ gives the probability that the random walk terminates at q_i after traversing q_j at least once. We naturally set $\overline{\pi} = \sum_{t=0}^{\infty} \overline{\pi}(t)$ and we define $\tilde{\pi}$ analogously.

It is straightforward to see that $\overline{\pi}$ does not depend on $\hat{\mathbf{P}}_{j\cdot}$, hence the effect of the perturbation is on $\tilde{\pi}$ alone. Now, $\tilde{\pi}(t) = \sum_{l=0}^{t} \tilde{\pi}(t,l)$ where, for every $i = 1, \ldots, n-1$, $\tilde{\pi}_i(t,l)$ denotes the probability that (i) the session terminates at q_i and (ii) it traverses q_j for the first time after l steps. Let $\bar{\mathbf{I}}$ denote the matrix obtained from the identity matric $\mathbf{I}$ by setting to 0 the entry in position (j,j) and let

[3]The probability of terminating the session at q_i is the probability of terminating it at any of the possible infinite steps. Note that the corresponding events are disjoint.

$\tilde{\mathbf{I}} = \mathbf{I} - \bar{\mathbf{I}}$. We then have: $\tilde{\pi}(t,l) = \pi_0^T \bar{\mathbf{I}}(\hat{\mathbf{P}}\bar{\mathbf{I}})^{l-1}\hat{\mathbf{P}}\tilde{\mathbf{I}}\hat{\mathbf{P}}^{t-l}\hat{\mathbf{P}}_n = \pi_0^T(\bar{\mathbf{I}}\hat{\mathbf{P}})^l\tilde{\mathbf{I}}\hat{\mathbf{P}}^{t-l}\hat{\mathbf{P}}_n$. Here, the first equality follows by considering that the generic contribution to $\tilde{\pi}(t,l)$ comes from a path that starts at any node but q_j, first traverses q_j after l steps and then for $t-l$ steps follows a random walk starting at q_j. As a result:

$$\tilde{\pi} = \sum_{t=0}^{\infty}\tilde{\pi}(t) = \sum_{t=0}^{\infty}\sum_{l=0}^{t}\tilde{\pi}(t,l) = \sum_{t=0}^{\infty}\sum_{l=0}^{t}\pi_0^T(\bar{\mathbf{I}}\hat{\mathbf{P}})^l\tilde{\mathbf{I}}\hat{\mathbf{P}}^{t-l}\hat{\mathbf{P}}_n$$
$$= \sum_{l=0}^{\infty}\pi_0^T(\bar{\mathbf{I}}\hat{\mathbf{P}})^l\tilde{\mathbf{I}}\sum_{t=l}^{\infty}\hat{\mathbf{P}}^{t-l}\hat{\mathbf{P}}_n = \sum_{l=0}^{\infty}\pi_0^T(\bar{\mathbf{I}}\hat{\mathbf{P}})^l\tilde{\mathbf{I}}\sum_{t=0}^{\infty}\hat{\mathbf{P}}^{t}\hat{\mathbf{P}}_n.$$

Now, note that $\bar{\mathbf{I}}\hat{\mathbf{P}}$ is the matrix obtained from $\hat{\mathbf{P}}$ by setting to $\mathbf{0}$ its j-th row. Also, recall that $\hat{\mathbf{P}}_{i\cdot} = \tilde{\mathbf{P}}_{i\cdot}$, whenever $i \neq j$. As a result, $\sum_{l=0}^{\infty}\pi_0^T(\bar{\mathbf{I}}\hat{\mathbf{P}})^l\tilde{\mathbf{I}} = c\mathbf{e}_j^T$, where c does not depend on the perturbation (i.e., on $\hat{\mathbf{P}}_{j\cdot}$). Hence, for the expected utility we have:

$$\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(\pi_0))] = \sum_{s=1}^{n-1}\overline{\pi}_s w(s) + c\sum_{s=1}^{n-1}\left[\sum_{t=0}^{\infty}\hat{\mathbf{P}}^t\hat{\mathbf{P}}_n\right]_{js} w(s)$$
$$= \sum_{s=1}^{n-1}\overline{\pi}_s w(s) + c\sum_{s=1}^{n-1}[(\mathbf{I}-\hat{\mathbf{P}})^{-1}\hat{\mathbf{P}}_n]_{js}w(s),$$

where the first equality follows from the observations above, while the second follows since $\hat{\mathbf{P}}$ is sub-stochastic and its Neumann series converges to $(\mathbf{I}-\hat{\mathbf{P}})^{-1}$.[4] At this point, note that $\sum_{s=1}^{n-1}[(\mathbf{I}-\hat{\mathbf{P}})^{-1}\hat{\mathbf{P}}_n]_{js}w(s)$ is the expected utility for random walks starting at q_j. Hence, the optimization can be performed considering only these and the result does not depend on π_0. $\square$

The fact above shows that optimizing $\mathbf{E}^{\mathbf{P}'}[U(\mathbf{path}(\pi_0))]$ amounts to maximizing $\sum_{s=1}^{n-1}[(\mathbf{I}-\hat{\mathbf{P}})^{-1}\hat{\mathbf{P}}_n]_{js}w(s)$. Note that, given $\tilde{\mathbf{P}}$, q_j, the ρ_{jl}'s and the weights, this is solely a function $h(Q)$ of Q, the set of recommendations:

$$h(Q) = \sum_{s=1}^{n-1}[(\mathbf{I}-\hat{\mathbf{P}})^{-1}\hat{\mathbf{P}}_n]_{js}w(s).$$

We provide below a simple greedy algorithm to maximize $h(Q)$ and we prove that it performs close to optimum in all cases of practical interest.

```
Algorithm SSQR-Greedy
Require: query $q_j$, $\rho_{jl}$ $(l = 1,\dots,n-1)$, integer k
 1: U = V - {$t$, $q_j$}
 2: Q = $\emptyset$
 3: while |Q| < k AND ($\exists$ l:  $\mathbf{E}^{\mathbf{P}}[U(\mathtt{path}(\mathbf{e}_l))] > w(j)$)
    do
 4:    i = $\arg\max_{l:q_l\in U}\{\rho_{jl}(\mathbf{E}^{\mathbf{P}}[U(\mathtt{path}(\mathbf{e}_l))] - w(j))\}$
 5:    Q = Q $\cup$ {$q_i$}
 6:    U = U - {$q_i$}
```

Figure 8: Greedy algorithm for single-step query recommendation.

The algorithm is given in Figure 8 and it requires the computation of the expected utility achieved by a random walk starting at q_l, for $l = 1,\dots,n-1$, computed with respect to *the unperturbed matrix* $\mathbf{P}$. These quantities can be pre-computed once for all and require the computation of $(\mathbf{I}-\tilde{\mathbf{P}})^{-1}\mathbf{P}_n$. Algorithm `SSQR-Greedy` achieves a performance that is close to optimum in cases of practical relevance. In particular:

[4]More precisely, the Neumann series of $\tilde{\mathbf{P}}$, $\hat{\mathbf{P}}$ and $\bar{\mathbf{I}}\hat{\mathbf{P}}$ converge, since the spectral radius of these matrices is strictly less than 1. To see this, it is enough to remember that any matrix norm is an upper bound to the spectral radius and then consider the norm $\|\cdot\|_\infty$.

THEOREM B.2. *Let* $x = \max_Q \sum_{q_l\in Q}\frac{\rho_{jl}}{\mathbf{P}_{jn}}\pi_j^l$ *and* $\alpha = \frac{1}{1-x}$. *Denote by* Q_{ALG} *and* Q_{OPT} *the algorithm's and the optimum's choices for* Q. *Then:*

$$h(Q_{ALG}) \geq \frac{1}{\alpha}h(Q_{OPT}).$$

PROOF. To prove this result, we first note that $\hat{\mathbf{P}} = \tilde{\mathbf{P}} + \mathbf{e}_j\mathbf{d}^T$, where $\mathbf{d}_i = \rho_{ji}$ if $q_i \in Q$, $\mathbf{d}_i = 0$ otherwise. Applying Sherman-Morrison formula [13, Section 3.8, page 124] we have:

$$(\mathbf{I}-\hat{\mathbf{P}})^{-1} = (\mathbf{I}-\tilde{\mathbf{P}})^{-1} + \frac{(\mathbf{I}-\tilde{\mathbf{P}})^{-1}\mathbf{e}_j\mathbf{d}^T(\mathbf{I}-\tilde{\mathbf{P}})^{-1}}{1-\mathbf{d}^T(\mathbf{I}-\tilde{\mathbf{P}})^{-1}\mathbf{e}_j}$$
$$= (\mathbf{I}-\tilde{\mathbf{P}})^{-1} + \frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{\cdot j}\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{l\cdot}}{1-\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}$$

As a consequence:

$$[(\mathbf{I}-\hat{\mathbf{P}})^{-1}\hat{\mathbf{P}}_n]_{j\cdot}$$
$$= [(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{j\cdot}\mathbf{P}_n + \frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{l\cdot}}{1-\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}\mathbf{P}_n$$
$$-[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}\cdot\mathbf{e}_j^T$$
$$-\frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}{1-\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}\sum_{q_l\in Q}\rho_{jl}\cdot\mathbf{e}_j^T,$$

where the equality follows by recalling that $\tilde{\mathbf{P}}_{in} = \mathbf{P}_{in}$ for $i \neq j$ and $\hat{\mathbf{P}}_{jn} = \mathbf{P}_{jn} - \sum_{q_l\in Q}\rho_{jl}$ and by rearranging terms. We thus have:

$$h(Q) = \sum_{s=1}^{n-1}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{js}\mathbf{P}_{sn}w(s)$$
$$+\frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{ls}}{1-\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}\mathbf{P}_{sn}w(s)$$
$$-[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}\cdot w(j)$$
$$-\frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}{1-\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}}\sum_{q_l\in Q}\rho_{jl}\cdot w(j)$$
$$= \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_j))] + \frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_l))]}{1-\sum_{q_l\in Q}\frac{\rho_{jl}}{\mathbf{P}_{jn}}\pi_j^l}$$
$$-[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\rho_{jl}\cdot w(j)$$
$$-\frac{[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{jj}\sum_{q_l\in Q}\frac{\rho_{jl}}{\mathbf{P}_{jn}}\pi_j^l}{1-\sum_{q_l\in Q}\frac{\rho_{jl}}{\mathbf{P}_{jn}}\pi_j^l}\sum_{q_l\in Q}\rho_{jl}\cdot w(j).$$

Here, the first equality follows from simple algebraic manipulations. As to the second equality, the first term on the right-hand side follows since, for every s, $[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{js}\mathbf{P}_{sn}$ is the probability of termination at query q_s for a random walk starting at q_j. Analogous considerations hold for $[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{ls}\mathbf{P}_{sn}$ (the probability of termination at q_s for random walks starting at q_l). For the sum $\sum_{q_l\in Q}\rho_{jl}[(\mathbf{I}-\tilde{\mathbf{P}})^{-1}]_{lj}$ appearing in the denominators of the second and fourth terms, note that $[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{lj}$ equals $\pi_j^l/\mathbf{P}_{jn}$, with π_j^l the overall probability that a random walk starting at q_l terminates at q_j. We can thus write:

$$\begin{aligned}
&h(Q) = \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_j))] \\
&+\frac{[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj} \sum_{q_l \in Q} \rho_{jl} \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_l))]}{1 - x(Q)} \\
&-[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj} \sum_{q_l \in Q} \rho_{jl} \cdot w(j) - \frac{[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj} x(Q)}{1 - x(Q)} \sum_{q_l \in Q} \rho_{jl} \cdot w(j) \\
&= \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_j))] \\
&+\frac{[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj}}{1 - x(Q)} \sum_{q_l \in Q} \rho_{jl} (\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_l))] - w(j)),
\end{aligned}$$

where we set $x(Q) = \frac{1}{\mathbf{P}_{jn}} \sum_{q_l \in Q} \rho_{jl} \pi_j^l$. The second equality follows by rearranging terms. To complete the proof, set

$$\begin{aligned}
&f(Q) = \mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_j))] \\
&+[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj} \sum_{q_l \in Q} \rho_{jl} (\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_l))] - w(j)),
\end{aligned}$$

and note that Algorithm `SSQR-Greedy` maximizes $f(Q)$, since $\mathbf{E}^{\mathbf{P}}[U(\mathbf{path}(\mathbf{e}_j))]$ and $[(\mathbf{I} - \tilde{\mathbf{P}})^{-1}]_{jj}$ only depend on $\tilde{\mathbf{P}}$ and not on the perturbation. Let $x = \max_Q x(Q)$ and $\alpha = 1/(1 - x)$. We then have:

$$h(Q_{ALG}) > f(Q_{ALG}) \geq f(Q_{OPT}) \geq \frac{1}{\alpha} h(Q_{OPT}).$$

Here, the third inequality follows since $h(Q)$ and $f(Q)$ have respectively the form $a + b/(1 - x)$ and $a + b$, with a and b non-negative in our case. Since $1 - x$ is strictly less than 1, it follows straightforwardly that their ratio is less than $1/(1 - x)$, thus proving the claim. $\square$

Note that, in practice, x should be small for most non-pathological instances, (possibly, $x << 1$). This follows since, considered a generic Q, $x(Q) = \frac{1}{\mathbf{P}_{jn}} \sum_{q_l \in Q} \rho_{jl} \pi_j^l$ is, up to the factor $1/\mathbf{P}_{jn}$, the probability that a random walk starting at q_j follows one of the recommendations as a first step and then proceeds for an arbitrary number of steps along a possibly non simple path, terminating at q_j itself. On the other hand, $\mathbf{P}_{jn}$ is pretty large in practice, (0.5 or larger).

The case of no incoming links. Note that, when q_j has no incoming links in the unperturbed Markov chain, the solution provided by Algorithm `SSQR-Greedy` is optimal. This trivially follows by observing that, no matter which is the set Q of recommendations chosen by the algorithm, we have $x(Q) = 0$ in this case. This in turn is a consequence of the considerations in the last paragraph above. 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 when recommendations added at q_j take the user to a different search engine.