Simrank++: Query Rewriting through Link Analysis of the Click Graph

Ioannis Antonellis
Computer Science Dept
Stanford University
antonell@cs.stanford.edu

Hector Garcia Molina
Computer Science Dept
Stanford University
hector@cs.stanford.edu

Chi Chao Chang
Yahoo! Inc.
Sunnyvale, CA, 94089
chichao@yahoo inc.com

ABSTRACT

We focus on the problem of query rewriting for sponsored search. We base rewrites on a historical click graph that records the ads that have been clicked on in response to past user queries. Given a query q, we first consider Simrank [7] as a way to identify queries similar to q, i.e., queries whose ads a user may be interested in. We argue that Simrank fails to properly identify query similarities in our application, and we present two enhanced versions of Simrank: one that exploits weights on click graph edges and another that exploits "evidence." We experimentally evaluate our new schemes against Simrank, using actual click graphs and queries from Yahoo!, and using a variety of metrics. Our results show that the enhanced methods can yield more and better query rewrites.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

sponsored search, link analysis, similarity metric, click graph

1. INTRODUCTION

In sponsored search, paid advertisements (ads) relevant to a user's query are shown above or along-side traditional web search results. The placement of these ads is in general related to a ranking score which is a function of the semantic relevance to the query and the advertiser's bid.

Ideally, a sponsored search system would appear as in Figure 1. The system has access to a database of available ads and a set of bids. Conceptually, each bid consists of a query q, an ad α, and a price p. With such a bid, the bidder offers to pay if the ad α is both displayed and clicked when a user issues query q. For many queries, there are not enough direct bids, so the sponsored search system attempts to find other ads that may be of interest to the user who submitted the query. Even though there is no direct bid, if the user clicks on one of these ads, the search engine will make some money (and the advertiser will receive a customer). The challenge is then to find ads related to incoming queries that may yield user click throughs.

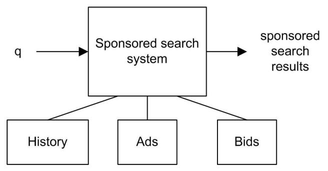

Figure 1: General sponsored search system architecture.

For a variety of practical and historical reasons, the sponsored search system is often split into two components, as shown in Figure 2. A front-end takes an input query q and periodically produces a list of *re-writes*, i.e., of other queries that are "similar" to q. For example, for query "camera," the queries "digital camera" and "photography" may be useful because the user may also be interested in ads for those related queries. The query "battery" may also be useful because users that want a camera may also be in the market for a spare battery. The query and its rewrites are then considered by the back-end, which displays ads that have bids for the query or its rewrites. The split approach reduces the complexity of the back-end, which has to deal with rapidly changing bids. The work of finding relevant ads, indirectly through related queries, is off-loaded to the front-end.

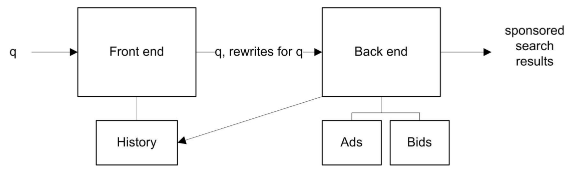

Figure 2: A common sponsored search system architecture.

At the front-end, queries can be rewritten using a variety of techniques (reviewed in our Related Work section) devel-

PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08

oped for document search. However, these techniques often do not generate enough useful rewrites. Part of the problem is that in our case "documents" (the ads) have little text, and queries are very short, so there is less information to work with, as compared with larger documents. Another part of the problem is that there are relatively few queries in the bid database, so even if we found all the textually related ones, we may not have enough. Thus, it is important to generate additional rewrites, using other techniques.

In this paper we focus on query rewrites based on the recent history of ads displayed and clicked on. The back-end generates a historical *click graph* that records the clicks that were generated by ads when a user inputs a given query. The click graph is a weighted bi-partite graph, with queries on one side and ads on the other (details in Section 3). The schemes we present analyze the connections in the click graph to identify rewrites that may be useful. Our techniques identify not only queries that are directly connected by an ad (e.g., users that submit either "mp3" or "i-tunes" click on an ad for "iPod.") but also queries that are more indirectly related (Section 4). Our techniques are based on the notion of *SimRank* [7], which can compute query similarity based on the connections in a bi-partite click-graph. However, in our case we need to extend SimRank to take into account the specifics of our sponsored search application. We present Simrank++ as a query rewriting technique and we compare it with existing query rewriting techniques. Since the query rewriting problem falls into the category of collaborative filtering problems, Simrank++ is essentially a new collaborative filtering technique.

Briefly, the contributions of this paper are as follows.

- We present a framework for query rewriting in a sponsored search environment.
- We identify cases where SimRank fails to transfer correctly the relationships between queries and ads into similarity scores.
- We present two SimRank extensions: one that takes into account the weights of the edges in the click graph, and another that takes into account the "evidence" supporting the similarity between queries.
- We experimentally evaluate these query rewriting techniques, using an actual click graph from Yahoo!, and a set of queries extracted from Yahoo! logs. We evaluate the resulting rewrites using several metrics. One of the comparisons we perform involves manual evaluation of query-rewrite pairs by members of Yahoo!'s Editorial Evaluation Team. Our results show that we can significantly increase the number of useful rewrites over those produced by SimRank and by another basic technique.

2. RELATED WORK

The query rewriting problem has been extensively studied in terms of traditional web search. In traditional web search, query rewriting techniques are used for recommending more useful queries to the user and for improving the quality of search results by incorporating users' actions in the results' ranking of future searches. Given a query and a search engine's results on this, the indication that a user clicked on some results can be interpreted as a vote that these specific results are matching the user's needs and thus are more relevant to the query. This information can then be used for improving the search results on future queries. Existing query rewriting techniques for traditional web search, include relevance feedback and pseudo-relevance feedback, query term deletion [8], substituting query terms with related terms from retrieved documents [15], dimensionality reduction such as Latent Semantic Indexing (LSI) [6], machine learning techniques [17, 16, 3] and techniques based on the analysis of the click graph [4].

Pseudo-relevance feedback techniques involve submitting a query for an initial retrieval, processing the resulting documents, modifying the query by expanding it with additional terms from the documents retrieved and then performing an additional retrieval on the modified query. However, pseudo-relevance feedback requires that the initial query retrieval procedure returns some results, something that is not always the case in sponsored search, as described before. In addition, pseudo-relevance has many limitations in effectiveness [13]. It may lead to query drift, as unrelated terms might be added to the query and is also computationally expensive. Query relaxation or deleting query terms leads to a loss of specificity from the original query.

In LSI, a collection of queries is represented by a terms queries matrix where each column corresponds to the vector space representation of a query. The column space of that matrix is approximated by a space of much smaller dimension that is obtained from the leading singular vectors of the matrix and then similarity scores between different queries can be computed. LSI is frequently found to be very effective even though the analysis of its success is not as straightforward [10]. The computational kernel in LSI is the singular value decomposition (SVD). This provides the mechanism for projecting both the queries on a lower-dimensional space spanned by the leading left singular vectors. In addition to performing dimensionality reduction, LSI captures hidden semantic structure in the data and resolves problems caused by synonymy and polysemy in the terms used. However, a well known difficulty with LSI is the high cost of the SVD for the large, sparse matrices appearing in practice.

3. PROBLEM DEFINITION

Let $\mathcal{Q}$ denote a set of n queries and $\mathcal{A}$ denote a set of m ads. A click graph for a specific time period is an undirected, weighted, bipartite graph $G = (\mathcal{Q}, \mathcal{A}, E)$ where E is a set of edges that connect queries with ads. G has an edge (q, α) if at least one user that issued the query q during the time period also clicked on the ad α. Each edge (q, α) has three weights associated with it. The first one is the number of times that α has been displayed as a result for q and is called the impressions of α given q. The second weight is the number of clicks that α received as a result of being displayed for the query q. This second weight is less than or equal to the first weight. The number of clicks divided by the number of impressions gives us the likelihood that a displayed ad will be clicked on. However, to be more accurate, this ratio needs to be adjusted to take into account the position where the ad was displayed. That is, an ad α placed near the top of the sponsored results is more likely to be clicked on, regardless of how good an ad it is for query q. Thus, the third weight associated with an edge (q, α) is the *expected click rate*, an adjusted clicks over impressions rate. The expected click rate is computed by the back-end (Figure 2), and we do not discuss the details here. The interested reader can find more information in the rich related literature (e.g. [11, 12]). Finally, for a node v in a graph, we denote by $E(v)$ the set of neighbors of v. We also define $N(v) = |E(v)|$ that is $N(v)$ denotes the number of v's total neighbors.

As discussed in the introduction, our goal is to find queries that are *similar*, in the sense that the ads clicked on for one query are likely to be clicked on when displayed for a user that entered the second query. We will predict similarity based on the information in the click graph: The intuition is that if an ad received clicks when displayed for both queries q_1 and q_2, then the queries are similar. Furthermore, if q_2 is related to q_3 in the same way but through some other ad, then q_1 and q_3 are also similar, although possibly to a lesser degree. We discuss our notion of similarity more in the following section.

Note that if the click graph does *not* contain an ad α that received clicks when q_1 and q_2 were issued, then we *cannot* infer that q_1 and q_2 are *not* similar. The queries could very well be similar (in our sense), but while the click-graph was collected, the back-end did not display ads that would have shown this similarity. (Perhaps there were no bids for those ads at the time.) As we will see later, even without the common ad α, we may still be able to discover the similarity of q_1 and q_2 through other similarity relationships in the click-graph.

Also note that in this paper we are not addressing problems of click or ad fraud. Fraud is a serious problem, where organizations or individuals generate clicks or place ads with the intent of defrauding or misleading the advertiser and/or the search engine. Query rewriting strategies may need to be adjusted to protect from fraud, but we do not consider such issues here.

Finally, notice that our query rewriting problem is a type of collaborative filtering (CF) problem. We can view the queries as "users" who are recommending "ads" by clicking on them. When we identify similar queries, we are finding queries that have similar recommendations, just like in CF, where one finds users that have similar tastes. In our setting, we are only trying to find similar queries (users), and not actually predicting recommended ads. Furthermore, as we will see, we are tuning our similarity metrics so they work well for sponsored search, as opposed to generic recommendations. However, we believe that the idea of using link analysis algorithms for the development of collaborative filtering techniques can be proven a powerful tool.

4. SIMILAR QUERIES

In this section we discuss the notion of query similarity that we are interested in. As we mentioned earlier, we will be saying that two queries are similar if they tend to make the search engine users to click on the same ads. Let us illustrate this with an example. Figure 3 shows a small click graph; for simplicity we have removed the weights from the edges and thus an edge indicates the existence of at least one click from a query to an ad. In this graph, the queries "pc" and "camera" are connected through a common ad and thus can be considered similar. Notice that this notion of similarity is not related to the actual similarity of the concepts described by the query terms. Now, we can observe that the queries "camera" and "digital camera" are connected through two common ads and thus can be considered similar. In contrast, queries "pc" and "tv" are not connected through any ad. However, both "pc" and "tv" are connected through an ad with the queries "digital camera" and "camera" which we already saw that are similar. Thus, we have a small amount of evidence that "pc" and "tv" are somehow similar, because they are both similar with queries that bring clicks to the same ads. In that case we will be saying that "pc" and "tv" are one hop away from queries that have a common ad. There might actually be cases where two queries will be two or more hops away from queries that bring clicks to the same ad. Finally, let us consider the queries "tv" and "flower". There is no path in the click graph that connects these two queries and thus we conclude that these queries are not similar.

Table 1: Query-query similarity scores for the sample click graph of Figure 3. Scores have been computed by counting the common ads between the queries

	pc	camera	digital camera	tv	flower
pc	-	1	1	0	0
camera	1	-	2	1	0
digital camera	1	2	-	1	0
tv	0	1	1	-	0
flower	0	0	0	0	-

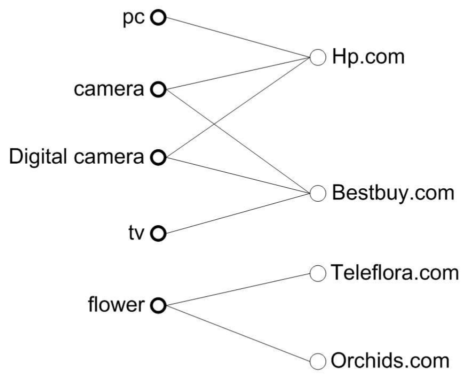

Figure 3: Sample unweighted click graph. An edge indicates the existence of at least one click from a query to an ad.

Thus, a naive way to measure the similarity of a pair of queries would be to count the number of common ads that they are connected to. Table 1 presents the resulting similarity scores for our sample click graph. As we can see there, "pc" has a similarity score 1 both with "camera" and "digital camera" but no similarity with "tv" and "flower". However, "camera" has a similarity score 2 with "digital camera" which indicates a stronger similarity. Also, "tv" has similarity 0 both with "pc" and "flower". Notice also that flower has similarity 0 with all the other queries. It is obvious that this naive technique cannot capture the similarity between "pc" and "tv" (as it does not look at the whole graph structure) and determines that their similarity is zero. In the following section we will see how we can compute similarity scores that take into account all the interactions appearing in the graph.

5. SIMRANK-BASED QUERY SIMILARITY

Simrank [7] is a method for computing object similarities, applicable in any domain with object-to-object relationships, that measures similarity of the structural context in which objects occur, based on their relationships with other objects. Specifically, in the case where there are two types of objects, bipartite Simrank is an iterative technique to compute the similarity score for each pair of objects of the same type. Bipartite Simrank is based on the underlying idea that two objects of one type are similar if they are related to similar objects of the second type. In our case, we can consider the queries as one type of objects and the ads as the other and use bipartite Simrank to compute similarity scores for each query-query pair.

Let $s(q, q')$ denote the similarity between queries q and q', and let $s(\alpha, \alpha')$ denote the similarity between ads α and α'. For $q \neq q'$, we write the equation:

$$s(q, q') = \frac{C_1}{N(q)N(q')} \sum_{i \in E(q)} \sum_{j \in E(q')} s(i, j) \qquad (1)$$

where C_1 is a constant between 0 and 1. For $\alpha \neq \alpha'$, we write:

$$s(\alpha, \alpha') = \frac{C_2}{N(\alpha)N(\alpha')} \sum_{i \in E(\alpha)} \sum_{j \in E(\alpha')} s(i, j) \qquad (2)$$

where again C_2 is a constant between 0 and 1.

If $q = q'$, we define $s(q, q') = 1$ and analogously if $\alpha = \alpha'$ we define $s(\alpha, \alpha') = 1$. Neglecting C_1 and C_2, equation 1 says that the similarity between queries q and q' is the average similarity between the ads that were clicked on for q and q'. Similarly, equation 2 says that the similarity between ads α and α' is the average similarity between the queries that triggered clicks on α and α'.

In the SimRank paper [7], it is shown that a simultaneous solution $s(*, *) \in [0, 1]$ to the above equations always exists and is unique. Also notice that the SimRank scores are symmetric, i.e. $s(q, q') = s(q', q)$.

In order to understand the role of the C_1, C_2 constants, let us consider a simple scenario where two ads α and α' were clicked on for a query q (which means that edges from q towards α and α' exist), so we can conclude some similarity between α and α'. The similarity of q with itself is 1, but we probably don't want to conclude that $s(\alpha, \alpha') = s(q, q) = 1$. Rather, we let $s(\alpha, \alpha') = C_2 \cdot s(q, q)$, meaning that we are less confident about the similarity between α and α' than we are between q and itself.

Let us look now at the similarity scores that Simrank computes for our simple click graph of Figure 3. Table 2 presents the similarity scores between all query pairs. If we compare these similarity scores with the ones in Table 1, we can make the following observations. Firstly, "camera" and "digital camera" have now the same similarity score with all other queries except for "flower". Secondly, "tv" has similarity 0.437 with "pc", 0.619 with "camera" and "digital camera" and zero with "flower". Notice that Simrank takes into account the whole graph structure and thus correctly produces a nonzero similarity score for the pair "tv" - "pc". Also notice that "camera" has two common ads with "digital camera" and only one common ad with "tv". However, Simrank does not produce different similarity scores for the "camera"-"digital camera" and "camera"-"tv" pairs. We will come back to this issue in detail in Section 6.

Table 2: Query-query similarity scores for the sample click graph of Figure 3. Scores have been computed by Simrank with $C_1 = C_2 = 0.8$

	pc	camera	digital camera	tv	flower
pc	-	0.619	0.619	0.437	0
camera	0.619	-	0.619	0.619	0
digital camera	0.619	0.619	-	0.619	0
tv	0.437	0.619	0.619	-	0
flower	0	0	0	0	-

We close our presentation of Simrank by presenting Algorithm 1 which computes the Simrank scores of a click graph. Appendix A explains the matrix notation used.

Algorithm 1 Simrank Computation

```
Require: transition matrix $P$, decay factor $C$, number of iterations $k$
Ensure: similarity matrix $S$
1: $[N,N]$ = size($P$);
2: $S = I_N$;
3: for $i = 1 : k$, do
4:     temp $= C * P^T * S * P$;
5:     $S$ = temp $+ I_N -$ Diag(diag(temp));
6: end for
```

6. SIMRANK IN COMPLETE BIPARTITE GRAPHS

Since we are trying to infer query similarities by analyzing the structure of the click graph (through Simrank), in this section we focus on a simple class of bipartite graphs; the complete, unweighted bipartite graphs. Our goal is to get insight of how well-suited Simrank scores are in this simple class, with respect to our notion of query similarity. Our focus here is two-fold. Firstly, Simrank scores in complete graphs can be easily computed and thus their appropriateness can be more easily analyzed. Actually, as we show in Appendix B (Lemma 1), the iterative equations 1,2 can be transformed to a closed-form formula. This simplifies the analysis. Secondly, since complete bipartite graphs appear as subgraphs in any click graph, the correctness of the Simrank scores on any click graph depends on the correctness of the scores that Simrank computes on the graph's complete subgraphs.

A complete bipartite graph is a special kind of bipartite graph where every vertex of the first node set is connected to every vertex of the second nodes set. In the click graph of Figure 3, the subgraphs consisting of the nodes "flower", "Teleflora.com", "orchids.com" and "camera", "digital camera", "hp.com", "bestbuy.com" are two examples of complete bipartite subgraphs. Formally, a complete bipartite graph $G = (V_1, V_2, E)$ is a bipartite graph such that for any two vertices $v_1 \in V_1$ and $v_2 \in V_2$, (v_1, v_2) is an edge in E. The complete bipartite graph with partitions of size $|V_1| = m$ and $|V_2| = n$, is denoted $K_{m,n}$. Figure 4(a) shows a $K_{2,2}$ graph from a click graph and Figure 4(b) shows a $K_{2,1}$ click graph.

Let us look at the similarity scores that Simrank computes for the pairs "camera" - "digital camera" and "pc" - "cam-

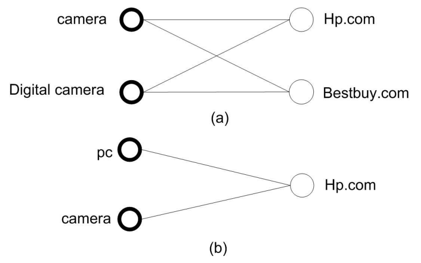

Figure 4: Sample complete bipartite graphs ($K_{2,2}$ and $K_{1,2}$) extracted from a click graph.

Table 3: Query-query similarity scores for the sample click graphs of Figure 4. Scores have been computed by Simrank with $C_1 = C_2 = 0.8$

Iteration	sim("camera", "digital camera")	sim("pc", "camera")
1	0.4	0.8
2	0.56	0.8
3	0.624	0.8
4	0.6496	0.8
5	0.65984	0.8
6	0.663936	0.8
7	0.6655744	0.8

era" from the graphs of Figure 4. Table 3 tabulates these scores for the first 7 iterations. As we can see sim("camera", "digital camera") is always less than sim("pc", "camera") although we observe that sim("camera", "digital camera") increases as we include more iterations. In fact, we can prove that sim("camera", "digital camera") becomes eventually equal to sim("pc", "camera") as we include more iterations. We can actually prove the following two Theorems for the similarity scores that Simrank computes in complete bipartite graphs (refer to Appendix B for the proofs).

Theorem 1. *Consider the two complete bipartite graphs $G = K_{1,2}$ and $G' = K_{2,2}$ with nodes sets $V_1 = \{a\}, V_2 = \{A, B\}$ and $V_1' = \{b, c\}$ and $V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite Simrank computes for the node pairs (A, B) and (C, D) after k iterations. Then, $sim^{(k)}(A, B) \geq sim^{(k)}(C, D), \ \forall\ k > 0$.* $\Box$

Theorem 2. *Consider the two complete bipartite graphs $G = K_{m,2}$ and $G' = K_{n,2}$ with $m < n$ and nodes sets $V_1, V_2 = \{A, B\}$ and $V_1', V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite Simrank computes for the node pairs (A, B) and (C, D) after k iterations. Then,*

(i) $sim^{(k)}(A, B) > sim^{(k)}(C, D), \ \forall\ k > 0$, and

(ii) $\lim_{k\to\infty}\ sim^{(k)}(A, B) = \lim_{k\to\infty}\ sim^{(k)}(C, D)$ if and only if $C_1 = C_2 = 1$, where C_1, C_2 are the decay factors of the bipartite Simrank equations. $\Box$

These Theorems provide us two pieces of evidence that Simrank scores are not intuitively correct in complete bipartite graphs. First, as in practice Simrank computations are limited to a small number of iterations, we would reach the conclusion that the pair "pc"-"camera" is more similar than the pair "camera" - "digital camera" which is obviously not correct. Second, even if we had the luxury to run Simrank until it converges, we would reach the conclusion that the similarity scores of the two pairs are the same. However, the fact that there are two advertisers that are connected with the queries "camera" and "digital camera" (versus the one that connects "pc" with "camera") is an indication that their similarity is stronger. We will try to fix such cases by introducing the notion of "evidence of similarity" in the following section.

7. REVISING SIMRANK

Consider a bipartite graph $G = (V_1, V_2, E)$ and two nodes $a, b \in V_1$. We will denote as evidence(a, b) the evidence existing in G that the nodes a, b are similar. The definition of evidence(a, b) we use is shown on Equation 3.

$$\text{evidence}(a, b) = \sum_{i=1}^{|E(a)\bigcap E(b)|} \frac{1}{2^i} \tag{3}$$

The intuition behind choosing such a function is as follows. We want the evidence score evidence(a,b) to be an increasing function of the common neighbors between a and b. In addition we want the evidence scores to get closer to one as the common neighbors increase.

We can now incorporate the evidence metric into the Simrank equations. We modify the equations 1 and 2 as follows:

For $q \neq q'$, we write the equation:

$$s_{\text{evidence}}(q, q') \quad = \quad \text{evidence}(q, q') \cdot s(q, q') \tag{4}$$

where $s(q, q')$ is the Simrank similarity between q and q'. For $\alpha \neq \alpha'$, we write:

$$s_{\text{evidence}}(\alpha, \alpha') \quad = \quad \text{evidence}(\alpha, \alpha') \cdot s(\alpha, \alpha') \tag{5}$$

where again $s(\alpha, \alpha')$ is the Simrank similarity between α and α'.

Notice that we could use k only iterations to compute the Simrank similarity scores and then multiply them by the evidence scores to come up with evidence-based similarities after k iterations. We will be loosely referring to these scores as evidence-based similarity scores after k iterations and we will be denoting them by $s^{(k)}_{\text{evidence}}(q, q')$.

Let us see now what the new Simrank equations compute for our sample click graphs. Table 4 tabulates these scores. As we can see sim("camera", "digital camera") is greater than sim("pc", "camera") after the first iteration. We can actually prove the following Theorem for the similarity scores that evidence-based Simrank computes in complete bipartite graphs (refer to [2] for the proof).

Theorem 3. *Consider the two complete bipartite graphs $G = K_{m,2}$ and $G' = K_{n,2}$ with $m < n$ and nodes sets $V_1, V_2 = \{A, B\}$ and $V_1', V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite evidence-based Simrank computes for the node pairs (A, B) and (C, D) after k iterations and let $C_1, C_2 > \frac{1}{2}$, where C_1, C_2 are the decay factors of the bipartite Simrank equations. Then,*

(i) $sim^{(k)}(A, B) < sim^{(k)}(C, D), \ \forall\ k > 1$, and

(ii) $\lim_{k\to\infty}\ sim^{(k)}(A, B) < \lim_{k\to\infty}\ sim^{(k)}(C, D)$. $\Box$

Table 4: Query-query similarity scores for the sample click graphs of Figure 4. Scores have been computed by the evidence-based Simrank with $C_1 = C_2 =$ 0.8

Iteration	sim("camera", "digital camera")	sim("pc", "camera")
1	0.3	0.4
2	0.42	0.4
3	0.468	0.4
4	0.4872	0.4
5	0.49488	0.4
6	0.497952	0.4
7	0.4991808	0.4

This Theorem indicates that the evidence-based Simrank scores in complete bipartite graphs will be consistent with the intuition of query similarity (as we discussed it in Section 4) even if we effectively limit the number of iterations we perform.

8. WEIGHTED SIMRANK

In the previous sections we ignored the information contained in the edges of a click graph and we tried to derive similarity scores for query pairs by just using the click graph's structure. In this section, we focus on weighted click graphs. We explore ways to derive query-query similarity scores that (i) are consistent with the graph's weights and (ii) utilize the edge weights in the computation of similarity scores.

8.1 Consistent similarity scores

We illustrate the notion of consistency between similarity scores and the graph's weights with the following two examples. First, consider the two weighted click graphs in Figure 5. Apparently the queries "flower"-"orchids" of the left graph are more "similar" than the queries "flower"-"teleflora" of the right graph. This observation is true because the number of clicks varies greatly for the second pair. If we now try to use Simrank (or even the evidence-based Simrank)we see that it will output the exact same similarity scores for both pairs. It is thus obvious that Simrank scores are not consistent with the weights on the graph. Now, consider the two graphs of Figure 6. The spread of

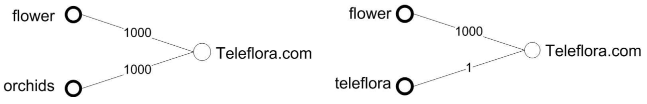

Figure 5: Sample weighted click graphs

values is the same in both graphs. However, it is also obvious that now the queries "flower-orchids" are more similar than the queries "flower-teleflora" since there are more clicks that connect the first pair with an ad. Again, Simrank or evidence-based Simrank will output the exact same similarity scores for both pairs.

In general, we define the notion of consistency as follows:

Definition 1. *[Consistent similarity scores] Consider a weighted bipartite graph $G = (V_1, V_2, E)$. Consider also*

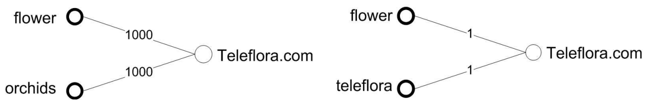

Figure 6: Sample weighted click graphs

four nodes $i_1, j_1, i_2, j_2 \in V_1$ and two nodes $v_1, v_2 \in V_2$. We now define the sets $W(v_1) = \{w(i_1, v_1), w(j_1, v_1)\}$ and $W(v_2) = \{w(i_2, v_2), w(j_2, v_2)\}$ and let variance(v_1) (variance(v_2)) denote a measure of $W(v_1)$'s ($W(v_2)$'s) variance respectively. We will be saying that a set of similarity scores $sim(i, j) \forall i, j \in V_1$ is consistent with the graph's weights if and only if $\forall i_1, j_1, i_2, j_2 \in V_1$ and $\forall v_1, v_2 \in V_2$ such that $\exists (i_1, v_1), (j_1, v_1), (i_2, v_2), (j_2, v_2) \in E$ both of the following are true:

(i) If variance$(v_1) <$ variance(v_2) and $w(i_1, v_1) > w(i_2, v_2)$) then $sim(i_1, j_1) > sim(i_2, j_2)$

(ii) If variance$(v_1) =$ variance(v_2) and $w(i_1, v_1) > w(i_2, v_2)$) then $sim(i_1, j_1) > sim(i_2, j_2)$

This definition utilizes the notion of the set variance which can be computed by averaging the squared distance of all the values belonging to the set from their mean value.

8.2 Revising Simrank

We can now modify the underlying random walk model of Simrank. Again we use the evidence scores as defined in Section 7, but now we will perform a different random walk. Remember that Simrank's random surfers model implies that a Simrank score sim(a, b) for two nodes a, b measures how soon two random surfers are expected to meet at the same node if they started at nodes a, b and randomly walked the graph. In order to impose the consistency rules in the similarity scores we perform a new random walk where its transition probabilities $p(\alpha, i)$, $\forall \alpha \in V_1, i \in E(\alpha)$ are defined as follows:

$$p(\alpha, i) = \text{spread}(i) \cdot \text{normalized\_weight}(\alpha, i), \forall i \in E(\alpha), \text{ and}$$
$$p(\alpha, \alpha) = 1 - \sum_{i \in E(\alpha)} p(\alpha, i)$$

where:

$$\text{spread}(i) = e^{-\text{variance}(i)}, \text{ and}$$
$$\text{normalized\_weight}(\alpha, i) = \frac{w(\alpha, i)}{\sum_{j \in E(\alpha)} w(\alpha, j)}$$

The value variance(i) corresponds to the variance of the weights of all edges that are connected with the node i.

The intuition behind choosing an exponential spread function is as follows. We want the evidence score evidence(a,b) to be an increasing function of the common neighbors between a and b. In addition we want the evidence scores to get closer to one as the common neighbors increase.

Notice how the new transition probability $p(\alpha, i)$ between two nodes $\alpha \in V_1, i \in V_2$ utilizes both the *spread*(i) value and the $w(\alpha, i)$ value in order to satisfy the consistency rules. Actually, we can prove the following Theorem that ensures us that weighted Simrank produces consistent similarity scores.

THEOREM 4. *Consider a weighted bipartite graph $G = (V_1, V_2, E)$ and let $w(e)$ denote the weight associated with an edge $e \in E$. Let also sim(i, j) denote the similarity score that weighted Simrank computes for two nodes $i, j \in V_1$. Then, $\forall i, j \in V_1$, sim(i, j) is consistent with the graph's weights.* $\square$

The actual similarity scores that weighted Simrank gives after applying the modified random walk are:

$$\begin{aligned} s_{\text{weighted}}(q,q') &= \text{evidence}(q,q') \cdot C_1 \cdot \\ &\sum_{i \in E(q)} \sum_{j \in E(q')} W(q,i) W(q',j) s_{\text{weighted}}(i,j) \\ s_{\text{weighted}}(\alpha,\alpha') &= \text{evidence}(\alpha,\alpha') \cdot C_2 \cdot \\ &\sum_{i \in E(\alpha)} \sum_{j \in E(\alpha')} W(\alpha,i) W(\alpha',j) s_{\text{weighted}}(i,j) \end{aligned}$$

where the factors $W(q, i)$ and $W(a, i)$ are defined as follows:

$$\begin{aligned} W(q,i) &= \text{spread}(i) \cdot \text{normalized\_weight}(q,i) \\ W(\alpha,i) &= \text{spread}(i) \cdot \text{normalized\_weight}(\alpha,i) \end{aligned}$$

Algorithm 2 (written using Matlab notation) computes the weighted Simrank. The entries of the weighted transition matrix P' correspond to the new transition probabilities and for the entries of the evidence matrix V we have $V(i, j) = \text{evidence}(i, j)$.

Algorithm 2 Simrank++ Computation

```
Require: weighted transition matrix $P'$, evidence matrix
    $V$, decay factor $C$, number of iterations $k$
Ensure: similarity matrix $S'$
1: $[N,N] = \text{size}(P')$;
2: $S' = I_N$;
3: for $i = 1 : k$, do
4:     $\text{temp} = C * P'^T * S' * P'$;
5:     $S' = \text{temp} + I_N - \text{Diag}(\text{diag}(\text{temp}))$;
6: end for
7: $S' = V . * S'$;
```

9. SCALABILITY OF OUR TECHNIQUES

The basic advantage of using query rewriting in sponsored search is that we offload the ad-serving system by computing offline and in batch the rewrites (by analyzing the click graph). Since the query rewrite work is separate, the ad-serving system can use all of its resources to find in real-time relevant ads for those rewritten queries. The query rewriting process usually has a window of several weeks to analyze the click graph.

Let us analyze the time and space requirements of Algorithm 2 for the computation of the Simrank++ similarity scores. Let $N = n + m$ denote the total number of query and ad nodes in the graph. The space required is $O(N^2)$ to store the results in the matrix S'. The time required is $O(kN^3)$, where k is the number of iterations. A more careful implementation (similar to one introduced in the original Simrank paper [7]) of Algorithm 2 can reduce the required time to $O(kN^2 d)$, where d is the average of $N(a)N(b)$ over all node-pairs (a, b) (both query and ad nodes). This result follows from the observation that at each iteration, the similarity score of every node-pair is updated only using values from its neighbor pairs, which will be d on average. Since the value of d corresponds to the square of the average node degree in the graph, d does not grow with n. Also, the space required can be reduced to $O(n^2 + m^2)$ by maintaining the results in two separate matrices, one for query and another for ad nodes. In addition, typical values for k are $k = 7$. Using such an implementation and a click graph with 15 million distinct queries, 14 million distinct ads and 28 million edges our Simrank++ method completes in 6 hours, on a single machine.

However, N^2 can be still prohibitively large in some applications. Distributed implementations of Algorithm 2 and pruning techniques (as the one presented in the original Simrank paper [7]) may be needed in this case; such implementations are beyond the scope of this paper. However, we mention that Algorithm 2 can become easily a distributed algorithm using programming paradigms like Map/Reduce [5].

Thus we believe that relatively large graphs can be analyzed in the 1-2 weeks available time period.

10. EXPERIMENTS

We conducted experiments to compare the performance of Simrank, evidence-based Simrank and weighted Simrank as techniques for query rewriting. Our baselines were three query rewriting technique based on the Pearson correlation, the Jaccard similarity and the cosine similarity respectively.

10.1 Baselines

The Pearson correlation between two queries q and q' is defined as: $\text{sim}_{\text{pearson}}(q, q') =$

$$\frac{\sum_{\alpha \in E(q) \bigcap E(q')} (w(q,\alpha) - \overline{w}_q)(w(q',\alpha) - \overline{w}_{q'})}{\sqrt{\sum_{\alpha \in E(q) \bigcap E(q')} (w(q,\alpha) - \overline{w}_q)^2 w(q',\alpha) - \overline{w}_{q'})^2}}$$

where $\overline{w}_q = \sum_{i \in E(q)} \frac{w(q,i)}{|E(q)|}$ is the average weight of all edges that have q as an endpoint. If $E(q) \bigcap E(q') = \emptyset$ then $\text{sim}_{\text{pearson}}(q, q') = 0$. The Pearson correlation indicates the strength of a linear relationship between two variables. In our case, we use it to measure the relationship between two queries. Notice, that $\text{sim}_{\text{pearson}}$ takes values in the interval $[-1, 1]$ and it requires that the two queries q and q' have at least one common neighbor in the click graph.

The Jaccard similarity between two queries q and q' is defined as the size of the intersection divided by the size of the union of the sets $E(q), E(q')$. Thus:

$$\text{sim}_{\text{Jaccard}}(q, q') = |E(q) \cap E(q')| / |E(q) \cup E(q')|$$

Finally, the cosine similarity between two queries q and q' corresponds to the angle between their vector representation. We represent each query q as a vector $v(q)$ where each coefficient corresponds to a node i of the graph and the coefficient takes the value 1 if the edge (q, i) exists or zero otherwise. We then have:

$$\text{sim}_{\text{cosine}}(q, q') = \arccos \frac{v(q) \cdot v(q')}{\|v(q)\| \|v(q')\|}$$

10.2 Dataset

We started from a two-weeks click graph from US Yahoo! search, containing approximately 15 million distinct queries, 14 million distinct ads and 28 million edges. An edge in this graph connects a query with an ad if and only if the ad had been clicked at least once from a user that issued the

Table 5: Dataset statistics

	# of Queries	# of Ads	# of Edges
subgraph 1	585,218	434,938	1,280,920
subgraph 2	530,797	374,243	1,130,314
subgraph 3	322,252	214,952	713,253
subgraph 4	313,951	243,406	703,747
subgraph 5	91,195	87,442	216,828
Total	1,843,413	1,354,981	4,045,062

query. In addition, each edge contains the number of clicks, the number of impressions, as well as the expected click rate. This graph consists of one huge connected component and several smaller subgraphs. In all our experiments that required the use of an edge weight we used the expected click rate.

To make the dataset size more manageable, we used the subgraph extraction method described in [1] to further decompose the largest component and we produced five smaller subgraphs. In summary, the algorithm in [1] is an efficient local graph partitioning algorithm that uses the PageRank vectors. Given a graph and an initial node, it tries to find a cut with small conductance [1] near that starting node. We started from different nodes and ran the algorithm iteratively in order to discover big enough, distinct subgraphs. Table 5 tabulates the total number of nodes (queries and ads) and edges contained in the five-subgraphs dataset. We also observed a number of power-law distributions, including ads-per-query, queries-per-ad and number of clicks per query-ad pair. We used this dataset as the input click graph for all query rewriting techniques we experimented with. The query set for evaluation is sampled, with uniform probability, from live traffic during the same two-weeks period. This traffic contains all queries issued at Yahoo! during that period; even the ones that did not bring any clicks on a sponsored search result. More specifically, we used a standardized 1200 query sample that has been generated by the above procedure and is currently being used as a benchmark at Yahoo!. We looked at these 1200 queries and extracted only the ones that actually appear in our five-subgraphs dataset as only for those our query rewriting methods would be able to provide rewrites. We found out that these are 120 queries and these are the queries that constitute our evaluation set. Using such an evaluation query selection procedure we made sure that queries issued rarely had a smaller probability of appearing in the evaluation set whereas more popular queries could appear with higher probability. We made this decision since we are interested in comparing the query rewriting techniques using a realistic query set. In other words, we prefer a rewriting technique that provides high quality rewrites for popular queries from another one that does the same only for rare queries.

10.3 Evaluation Method and Metrics

We run each method on the five-subgraphs dataset and recorded the top 100 rewrites for each query on our queries sample. We then use stemming to filter out duplicate rewrites (since such rewrites might appear in the click graph). In addition we perform *bid term filtering*, i.e., we remove queries that are *not* in a list of all queries that saw bids in the two-week period when the click graph was gathered. This list contains any query that received at least one bid at any point in the period; hence, if a query is not in the list it is unlikely to have bids currently. (Note that such queries with no bids may still be connected to ads in the click graph. These ads were displayed and clicked on because of query rewriting that took place when the query was originally submitted.)

The queries that remain after duplicate elimination and bid term filtering are considered for our evaluation. However, we limit ourselves to at most 5 rewrites per query per method because of the cost of the manual evaluation we describe next. Note that a method may generate fewer than 5 rewrites after filtering. We call the number of remaining rewrites the *depth* of a method.

To evaluate the quality of rewrites, we consider four methods. The first is a *manual evaluation*, carried out by professional members of Yahoo!'s editorial evaluation team. Each query – rewrite pair is considered by an evaluator, and is given a score on a scale from 1 to 4, based on their relevance judgment. (The scoring is the same as used in [9, 18]). The query rewrites that were more relevant with the original query assigned a score of 1, and the least related assigned a score of 4. Table 6 summarizes the interpretation of the four grades. The judgment scores are solely based on the evaluator's knowledge, and not on the contents of the click graph. We interpret the rewrites with scores 1-2 as relevant queries and the rewrites with scores 3-4 as irrelevant queries. We can then consider an IR task where we define the precision of method m for query q, precision(q, m) =

$$\frac{\text{relevant rewrites of } q \text{ that } m \text{ provides}}{\text{number of rewrites for } q \text{ that } m \text{ provides}}$$

and the recall(q, m) =

$$\frac{\text{relevant rewrites of } q \text{ that } m \text{ provides}}{\text{number of relevant rewrites for } q \text{ among all methods}}$$

The second evaluation metric is based on the *query coverage*. We are interested in the absolute number of queries (from our 120 query sample) for which each method manages to provide at least one rewrite. We call this number query coverage. In general, we prefer methods that cover as many as possible queries.

The third metric is the *query rewriting depth*. Here, we are interested in the total number of query rewrites that a method provides for a given query. This is called the depth of a query rewriting technique. Again, we are interested in methods that have larger rewriting depth.

Finally, our fourth evaluation method addresses the question of whether our methods made the "right" decision based on the evidence found in the click graph. Since the motivation behind using Simrank is to take into account the whole graph structure in defining the notion of query similarity, we want to quantify how well our techniques perform with respect to the graph structure and not the evaluation of a human expert. The basic idea is to remove certain edges from the click graph and to see if using the remaining data our schemes can still make useful inferences related to the missing data. Since, only techniques that look at the whole graph can be evaluated in this way, we

[1]The conductance is a way to measure how hard it is to leave a small set of a graph's nodes. If Φ_S is the conditional probability of leaving a set of nodes S given that we started from a node in S, then the conductance is defined as the minimal Φ_S over all sets S that have a total stationary probability of at most 1/2. More information can be found in [14].

do not include here our baselines and we only focus on the Simrank-based techniques. In particular, consider Figure 7, showing two queries q_2 and q_3 that share at least one common arc with a query q_1. In order to distinguish which query between q_2 and q_3 is a preferable rewrite for q_1, we define the desirability of query q_2 for query q_1 as $\text{des}(q_1, q_2) = \sum_{i \in E(q_1) \bigcap E(q_2)} \frac{1}{|E(q_2)|} \cdot w(q_2, i)$. By computing the desirability scores $\text{des}(q_1, q_2)$, $\text{des}(q_1, q_3)$ we can determine the most desirable rewrite for q_1. That is, given the evidence in the graph, if $\text{des}(q_1, q_2) > \text{des}(q_1, q_3)$ then q_2 would be a better rewrite for q_1 than q_3.

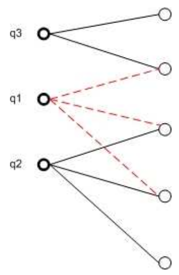

Figure 7: Sample setup for testing the ability of a rewriting method to compute correct query rewrites. By removing the red, dashed edges, we remove all direct similarity evidence between q_1 and q_2, q_3.

Given our definition of desirability, we can now conduct the following experiment. First, we remove the edges that connect q_1 to ads that are also connected with q_3 or q_2. In Figure 7 these are the red, dashed edges. Then, we run each variation of Simrank on the remaining graph and record the similarity scores $\text{sim}(q_1, q_2)$ and $\text{sim}(q_1, q_3)$ that the method gives. Finally, we test whether the ordering for q_2, q_3 that these similarity scores provide is consistent with the ordering derived from the desirability scores. In our example, if $\text{des}(q_1, q_2) > \text{des}(q_1, q_3)$ and $\text{sim}(q_1, q_2) > \text{sim}(q_1, q_3)$ then we would say that the similarity score was successful in predicting the desirable rewrite.

We repeated this edge removal experiment for 50 queries randomly selected from our five-subgraphs dataset. These queries played the role of query q_1 as described above. For each of those queries we identified all the queries from the dataset that shared at least one common ad with it and we randomly selected two of them. Those were the q_2 and q_3 queries. We then report the fraction of the 50 queries for which a method was able to correctly predict the desirability of q_2 (or q_3) over the other query. In order to make sure that a Simrank similarity score can be computed after the deletion of the edges in our experiment, we selected the queries q_2, q_3 after making sure that after edge removal there would still exist a path from q_2 to q_1 and from q_3 to q_1 through other edges in the graph.

11. RESULTS

11.1 Baseline comparison

We first evaluated our baselines. We found the technique that is based on the Pearson correlation consistently better compared to the other two (Jaccard, cosine). In particular, Figure 8 presents the precision/recall graphs for Pearson, Jaccard and cosine as well as the precision at 1-5 queries (P@X). For the computation of precision and recall the editorial scores were used in a binary classification manner; scores 1-2 were the positive class and scores 3-4 the negative class. For instance, in Figure 8 (bottom graph) we see that Pearson has 70% precision for 2 rewrites, meaning that 70% of its rewrites in the top two ranks were given scores of 1 or 2 by the evaluators.

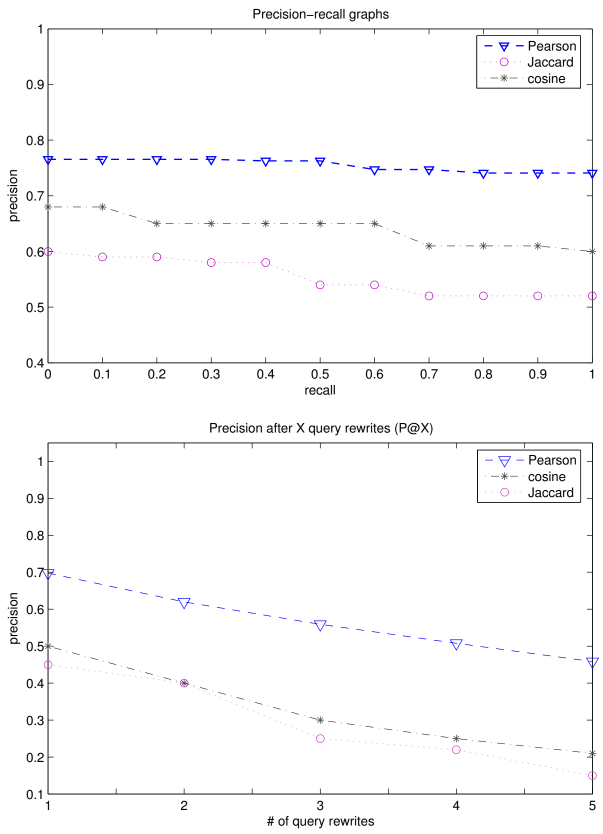

Figure 8: Precision at 11 standard recall levels (top) and precision after $X = 1, 2, \ldots, 5$ query rewrites (P@X) (bottom)

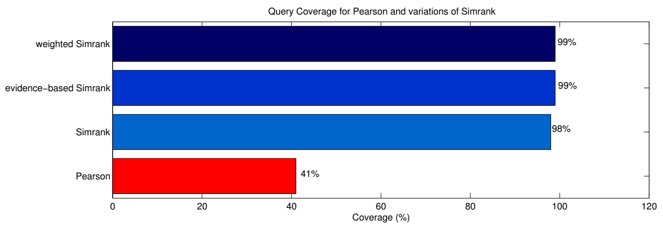

Figure 9: Comparing the query coverage of Pearson and Simrank

The query coverage, the query rewriting depth and the desirability prediction of all three baselines were identical. This is due to the fact that all of them require for two queries

Table 6: Editorial scoring system for query rewrites.

Score		Definition	Example (query - re-write)
1	Precise Match	near-certain match	corvette car - chevrolet corvette
2	Approximate Match	probable, but inexact match with user intent	apple music player - ipod shuffle
3	Marginal Match	distant, but plausible match to a related topic	glasses - contact lenses
4	Mismatch	clear mismatch	time magazine - time & date magazine

to have at least one common neighbor, in order to report a non-zero similarity value.

In what follows, we compare the Simrank based techniques with the Pearson-based baseline.

11.2 Query Coverage

Figure 9 illustrates the percentage of queries from the 120 queries sample that Pearson and Simrank provide rewrites for. Simrank provides rewrites almost for all queries (98%) when Pearson gives rewrites only for the 41% of the queries. This can be considered as expected, since Pearson can only measure similarity between two queries if they share a common ad, whereas Simrank takes into account the whole graph structure and does not require something similar. Also notice, that evidence-based Simrank further improves the coverage to 99%.

11.3 Precision-Recall

Figure 10 presents the precision/recall graphs for Pearson and Simrank as well as the precision at 1-5 queries (P@X). The values of the precision/recall we report here correspond to the average precision/recall among all the evaluation queries of our dataset. For the computation of precision and recall the editorial scores were used in a binary classification manner; scores 1-2 were the positive class and scores 3-4 the negative class. For instance, in Figure 10 (bottom graph) we see that Weighted Simrank has 93% precision for 2 rewrites, meaning that 93% of its rewrites in the top two ranks were given scores of 1 or 2 by the evaluators.

Also, we see that simple Simrank substantially improves the precision of the rewrites compared to Pearson. In addition, the use of the evidence score and the exploitation of the graph weights further boosts the precision, as expected.

11.4 Rewriting Depth

Figure 11 compares the rewriting depth of Pearson and the variations of Simrank. Note that our two enhanced schemes can provide the full 5 rewrites for over 85% of the queries. As mentioned earlier, the more rewrites we can generate, the more options the back-end will have for finding ads with active bids.

11.5 Desirability prediction

Figure 12 provides the results of our experiments for identifying the correct order of query rewrites as described in Section 10.3. Weighted Simrank is able to predict a surprisingly large number of the desirable rewrites (92% or 46 out of 50 cases). The methods that do not take edge weights into account (simple Simrank and evidence-based Simrank) do substantially worse (54% or 27 out of 50 cases). In spite of its low performance on this test, evidence-based Simrank is still able to generate many rewrites that the evaluators found relevant (previous results).

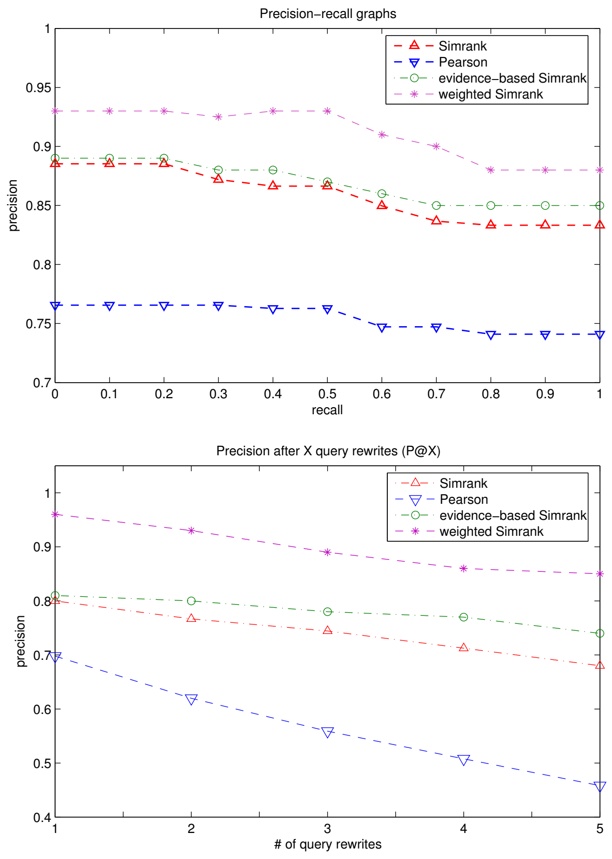

Figure 10: Precision at 11 standard recall levels (top) and precision after $X = 1, 2, \dots, 5$ query rewrites (P@X) (bottom)

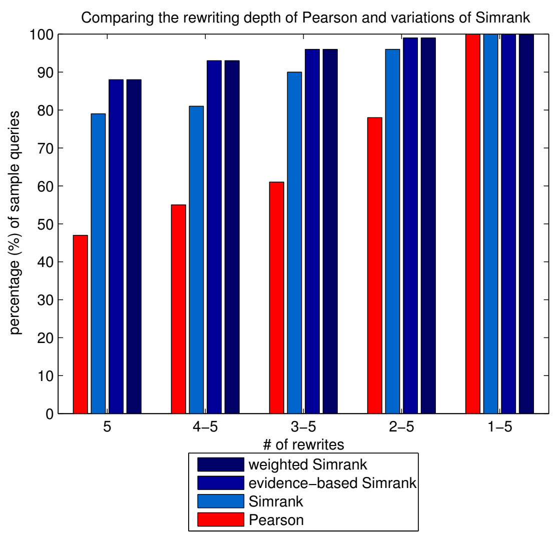

Figure 11: Comparing the rewriting depth

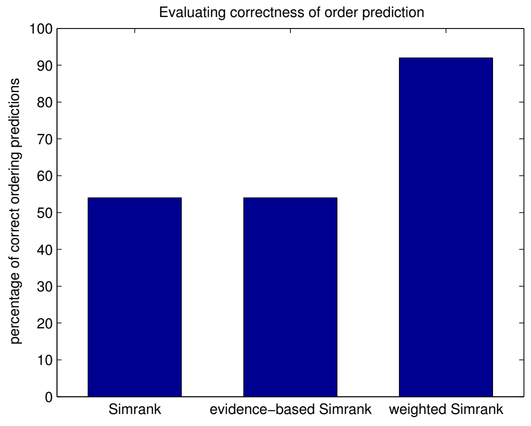

Figure 12: Comparing the ability of query rewriting methods to correctly predict the order of query rewrite candidates

11.6 Discussion

As we can see, simple Simrank outperforms Pearson both in query coverage, rewriting depth and precision/recall. Notice here that this version of Simrank does not utilize at all the qualitative information in the click graph, whereas Pearson does.

The introduction of evidence scores increases query coverage slightly (by 1%) and substantially improves the quality of the rewrites. For instance, the precision at 5 rewrites of simple Simrank is 75% whereas the precision after 5 rewrites of evidence-based Simrank is 80% (Figure 10). In addition, in the P@X rewrites diagram (Figure 10) the line corresponding to the precision of evidence-based Simrank is always above the one corresponding to the precision of simple Simrank. Finally, evidence-based Simrank increases the rewriting depth. For example, simple Simrank provides five rewrites for 79% of the queries, whereas evidence based Simrank gives five rewrites for the 89% of the queries (Figure 11).

Weighted Simrank builds upon evidence-based Simrank and utilizes the graph weights. It maintains the query coverage percentage of evidence-based Simrank at 99% (Figure 9) and substantially improves the quality of the rewrites. Figure 10 shows that the P@X line of weighted Simrank is always above the one of evidence-based Simrank. The precision at 5 rewrites of weighted Simrank goes from 80% (evidence-based Simrank) to 86%. Also, 96% of the queries have a high-quality top rewrite when we use weighted Simrank (P@1, Figure 10) when the corresponding percentages for evidence-based Simrank, simple Simrank and Pearson are 81%, 80% and 70%. In our desirability experiment, weighted Simrank predicted successfully the desirable rewrite for 92% of the cases (Figure 12). Finally, weighted-based Simrank maintains the rewriting depth of evidence-based Simrank (Figure 11).

12. CONCLUSIONS

In this paper we focused on the problem of query rewriting for sponsored search. We proposed Simrank to exploit the click graph structure and we introduced two extensions: one that takes into account the weights of the edges in the click graph, and another that takes into account the "evidence" supporting the similarity between queries. Our experimental results show that weighted-based Simrank is the overall best method for generating rewrites based on a click graph. In addition, the combination of increase in the query coverage and the rewriting depth that it brings can be interpreted as a huge potential of revenue increase for the search engine.

There are several query rewriting issues that we did not address in our analysis. Spam clicks can mislead our techniques and thus spam-resistant variations of our techniques would be useful. Also, methods for combining our similarity scores with semantic text-based similarities could be considered. Finally, techniques for updating the Simrank similarity scores when a click graph changes will be useful.

Even though our new schemes were developed and tested for query rewriting based on a click graph, we suspect that the weighted and evidence-based Simrank methods could be of use in other applications that exploit bi-partite graphs. We plan to experiment with these schemes in other domains, including collaborative filtering.

13. ACKNOWLEDGEMENTS

We thank Kevin Lang for providing us his code from [1] and for helping us with the subgraph extraction procedure. We also thank Yahoo! Editorial team for carrying out the evaluation of our techniques. We thank Panagiotis Papadimitriou, Zoltán Gyöngyi, Tasos Anastasakos and Tamás Sarlós for fruitful discussions. Finally, we thank the anonymous reviewers for insightful comments and Dmitry Lizorkin for his comments on Algorithm 1.

14. REFERENCES

[1] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *FOCS'06*.
[2] I. Antonellis, H. Garcia-Molina, and C. Chang. Simrank++: Query rewriting through link analysis of

the click graph. In *Technical Report, url: http://dbpubs.stanford.edu/pub/2007-32*, 2007.

[3] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD '00*.

[4] Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proc. SIGIR '07*.

[5] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*.

[6] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. 1990.

[7] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *KDD '02*.

[8] Rosie Jones and Daniel C. Fain. Query word deletion prediction. In *SIGIR '03*.

[9] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *WWW '07*.

[10] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: a probabilistic analysis. In *PODS '98*.

[11] M. Regelson and D. Fain. Predicting click-through rate using keyword clusters. In *Proc. 2nd Workshop on Sponsored Search Auctions*.

[12] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *WWW '07*.

[13] Ian Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *SIGIR '03*, pages 213–220.

[14] A. Sinclair. Algorithms for random generation and counting: A markov chain approach. In *Birkhauser, Boston-Basel-Berlin*, 1993.

[15] Egidio Terra and Charles L.A. Clarke. Scoring missing terms in information retrieval tasks. In *CIKM '04*.

[16] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 2002.

[17] Wei Vivian Zhang, Xiaofei He, Benjamin Rey, and Rosie Jones. Query rewriting using active learning for sponsored search. In *SIGIR '07*.

[18] Wei Vivian Zhang and Rosie Jones. Comparing click logs and editorial labels for training query rewriting. In *Query Log Analysis Workshop, WWW '07*.

APPENDIX

A. SIMRANK IN MATRIX NOTATION

Consider a click graph $G = (\mathcal{Q}, \mathcal{A}, E)$. Let A denote the vertex adjacency matrix of G, that is $A_{ij} = 1$ iff $(i, j) \in E$. Notice that A is a symmetric matrix and thus $A = A^T$. We also construct the transition matrix P for the graph G by normalizing the columns of A so that the elements of each column sum to 1. P is a stochastic matrix.

Let $e^{(k)} = [1\ 1\ 1\ \cdots\ 1]^T$ denote a vector with k coefficients; all equal to one. Let also $e_i^{(k)} = [0\ \cdots 0\ 1\ 0\ \cdots\ 0]^T$ denote the j-column of an identity matrix I_k, that is a vector with k coefficients that has its i-th coefficient set to 1 and all the rest equal to 0. In the rest of the paper, we will be neglecting the superscript of the e vectors when their size is perceived by the context they appear in. Using these two vectors we can express the elements of P in terms of the elements of A. Specifically, we have $P_{ij} = A_{ij}/(e_i^T Ae)$.

Finally, for a matrix X, $\mathrm{diag}(X)$ is a vector of diagonal entries in X; for a vector x, $\mathrm{Diag}(x)$ is the diagonal matrix such that $x = \mathrm{diag}(\mathrm{Diag}(x))$.

Let $S \in \mathcal{R}^{|\mathcal{Q}\cup\mathcal{A}|\times|\mathcal{Q}\cup\mathcal{A}|}$ denote the similarity matrix for the nodes in G, where $S_{i,j} = \mathrm{sim}(i, j)$, $\forall i \in \mathcal{Q} \cup \mathcal{A}$. S is symmetric since $\mathrm{sim}(i, j) = \mathrm{sim}(j, i)$. Also, since $\mathrm{sim}(i, i) = 1, \forall i \in \mathcal{Q} \cup \mathcal{A}$, all the diagonal elements of S are equal to 1. We will be denoting as $S^{(k)}$ the similarity matrix containing the similarity values after k Simrank iterations, where $k > 0$. We also define $S^{(0)} = I$, where I is an (appropriately sized) unary matrix.

Using this notation, equations 1, 2 become:

$$S^{(k)} = CP^T S^{(k-1)} P + I - \mathrm{Diag}(\mathrm{diag}(CP^T S^{(k-1)} P)) \quad (6)$$

Based on equation 6, Algorithm 1 computes the Simrank scores of a given click graph.

B. PROOF OF THEOREMS 1, 2 (SECT. 5)

LEMMA 1. *Consider the complete bipartite graph $K_{2,2}$ with nodes sets $V_1 = \{a, b\}$ and $V_2 = \{A, B\}$. Let $sim^{(k)}(A, B)$ denote the similarity between nodes A, B that bipartite Simrank computes after k iterations and let C_1, C_2 denote the Simrank decay factors. Then:*

(i) $sim^{(k)}(A, B) = \frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil}$

(ii) $\lim_{k\to\infty} sim^{(k)}(A, B) \le C_2$

PROOF. *(i) We omit the proof. It follows by induction from the Definition of Simrank. (ii) We know that $C_1 \le 1$ and $C_2 \le 1$. Thus:*

$$\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{i}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \le \frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}$$

Now we can write:

$$\begin{aligned}\lim_{k\to\infty} sim^{(k)}(A,B) &= \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \\ &\le \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}} = \frac{C_2}{2}\lim_{k\to\infty}\sum_{i=1}^{k}\frac{1}{i} = \\ &= \frac{C_2}{2}\cdot 2 = C_2\end{aligned}$$

□

THEOREM 1. *Consider the two complete bipartite graphs $G = K_{1,2}$ and $G' = K_{2,2}$ with nodes sets $V_1 = \{a\}, V_2 = \{A, B\}$ and $V_1' = \{b, c\}$ and $V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite Simrank computes for the node pairs (A, B) and (C, D) after k iterations. Then, $\forall\ k > 0$, $sim^{(k)}(A, B) \ge sim^{(k)}(C, D)$.*

PROOF. *From equations 1, 2 we have:*

$$sim^{(k)}(A,B) = \frac{C_2}{1\cdot 1}1 = C_2,\ \forall k > 0$$

Also, from Lemma 1(i), we have:

$$\begin{aligned}sim^{(k)}(C,D) &= \frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \le \\ &\le \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}} = \frac{C_2}{2}\cdot 2 = C_2\end{aligned}$$

□

LEMMA 2. *Consider the two complete bipartite graphs $G = K_{1,2}$ and $G' = K_{2,2}$ with nodes sets $V_1 = \{a\}, V_2 = \{A, B\}$ and $V_1' = \{b, c\}$ and $V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite Simrank computes for the node pais (A, B) and (C, D) after k iterations. Then, $\lim_{k\to\infty}\ sim^{(k)}(A, B) = \lim_{k\to\infty}\ sim^{(k)}(C, D)$ if and only if $C_1 = C_2 = 1$, where C_1, C_2 are the decay factors of the bipartite Simrank equations.*

PROOF. *Let us assume that $\lim_{k\to\infty}\ sim^{(k)}(A, B) = \lim_{k\to\infty}\ sim^{(k)}(C, D)$. This means that:*

$$\begin{aligned}\lim_{k\to\infty}\ sim^{(k)}(A,B) &= \lim_{k\to\infty}\ sim^{(k)}(C,D) \Leftrightarrow \\ C_2 &= \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \Leftrightarrow \\ C_2 &= \frac{C_2}{2}\lim_{k\to\infty}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \Leftrightarrow \\ 1 &= \frac{1}{2}\lim_{k\to\infty}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \Leftrightarrow \\ \lim_{k\to\infty}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} &= 2 \Leftrightarrow \\ C_1 &= C_2 = 1\end{aligned}$$

Now, let us assume that $C_1 = C_2 = 1$. We will have:

$$\begin{aligned}\lim_{k\to\infty}\ sim^{(k)}(C,D) &= \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}}C_1^{\lfloor\frac{i}{2}\rfloor}C_2^{\lceil\frac{i-1}{2}\rceil} \\ &= \lim_{k\to\infty}\frac{C_2}{2}\sum_{i=1}^{k}\frac{1}{2^{i-1}} = C_2 \\ &= \lim_{k\to\infty}\ sim^{(k)}(A,B)\end{aligned}$$

□

LEMMA 3. *Consider the two complete bipartite graphs $G = K_{1,2}$ and $G' = K_{2,2}$ with nodes sets $V_1 = \{a\}, V_2 = \{A, B\}$ and $V_1' = \{b, c\}$ and $V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores* *that bipartite Simrank computes for the node pais (A, B) and (C, D) after k iterations. Then, if for the decay factors of bipartite Simrank $C_1,\ C_2$ we know that $C_1 < 1$ or $C_2 < 1$, both of the following are true:*

(i) $sim^{(k)}(A, B) > sim^{(k)}(C, D),\ \forall\ k > 0$, *and*

(ii) $\lim_{k\to\infty}\ sim^{(k)}(A, B) > \lim_{k\to\infty}\ sim^{(k)}(C, D)$

Proof. *(i) It follows from Theorem 1 and Lemma 2.*

(ii) We have:

$$\begin{aligned}\lim_{k\to\infty}\ sim^{(k)}(C, D) &= \lim_{k\to\infty} \frac{C_2}{2} \sum_{i=1}^{k} \frac{1}{2^{i-1}} C_1^{\lfloor \frac{i}{2} \rfloor} C_2^{\lceil \frac{i-1}{2} \rceil} \\ &< \lim_{k\to\infty} \frac{C_2}{2} \sum_{i=1}^{k} \frac{1}{2^{i-1}} = C_2 \\ &= \lim_{k\to\infty}\ sim^{(k)}(A, B)\end{aligned}$$

□

Theorem 2. *Consider the two complete bipartite graphs $G = K_{m,2}$ and $G' = K_{n,2}$ with $m < n$ and nodes sets $V_1, V_2 = \{A, B\}$ and $V_1', V_2' = \{C, D\}$ correspondingly. Let $sim^{(k)}(A, B)$ and $sim^{(k)}(C, D)$ denote the similarity scores that bipartite Simrank computes for the node pairs (A, B) and (C, D) after k iterations. Then,*

(i) $sim^{(k)}(A, B) > sim^{(k)}(C, D),\ \forall\ k > 0$, *and*

(ii) $\lim_{k\to\infty}\ sim^{(k)}(A, B) = \lim_{k\to\infty}\ sim^{(k)}(C, D)$ *if and only if $C_1 = C_2 = 1$, where $C_1,\ C_2$ are the decay factors of the bipartite Simrank equations.*

Proof. *We omit the Proof. Similar arguments as in Lemma 3.* □