Efficient and Effective Link Analysis with Precomputed SALSA Maps

Marc Najork
Microsoft Research
Mountain View, CA, USA
najork@microsoft.com

Nick Craswell
Microsoft
Cambridge, UK
nickcr@microsoft.com

ABSTRACT

SALSA is a link-based ranking algorithm that takes the result set of a query as input, extends the set to include additional neighboring documents in the web graph, and performs a random walk on the induced subgraph. The stationary probability distribution of this random walk, used as a relevance score, is significantly more effective for ranking purposes than popular query-independent link-based ranking algorithms such as PageRank. Unfortunately, this requires significant effort at query-time, to access the link graph and compute the stationary probability distribution. In this paper, we explore whether it is possible to perform most of the computation off-line, prior to the arrival of any queries. The off-line phase of our approach computes a "score map" for each node in the web graph by performing a SALSA-like algorithm on the neighborhood of that node and retaining the scores of the most promising nodes in the neighborhood graph. The on-line phase takes the results to a query, retrieves the score map of each result, and returns for each result a score that is the sum of the matching scores from each score map. We evaluated this algorithm on a collection of about 28,000 queries with partially labeled results, and found that it is significantly more effective than PageRank, although not quite as effective as SALSA. We also studied the trade-off between ranking effectiveness and space requirements.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Storage and Retrieval—*search process, selection process*

General Terms

Algorithms, Measurement, Experimentation

Keywords

SALSA, link-based ranking, retrieval performance, web search

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

One of the fundamental problems in information retrieval is ranking – arranging the results of a query according to their relevance. In classical information retrieval, ranking relied primarily on textual features. Web search engines have a number of additional features at their disposal, including the hyperlinks leading from one web page to another. A hyperlink can be viewed as an endorsement by a web page's author of another web page. Link-based ranking algorithms can be broadly grouped into two classes: query-independent algorithms that estimate the quality of a web page, and query-dependent ones that estimate its relevance to a particular query. Recent large-scale evaluations have shown that some query-dependent link-based ranking algorithms (notably, the SALSA algorithm) are substantially more effective than well-known query-independent ones such as PageRank [11, 12].

However, this effectiveness comes at a significant expense: the first step in most query-dependent link-based ranking algorithms is to expand the result set of a query into a neighborhood graph. Given that the full web graph is enormous (containing on the order of hundreds of billions of edges), it does not fit into the main memory of any single machine. One way to overcome this problem would be to store the web graph on disk; however, computing SALSA scores for an average-sized result set would take minutes in such a setting, since it is necessary to first expand the result set to a neighborhood set (requiring a disk seek per result), and then to determine the linkage between documents in the neighborhood set (requiring another disk seek per neighbor). For a commercial search engine, latency on this scale is unacceptable, given that there is a strong correlation between fast response time and the engine's popularity, as measured by the number of submitted queries [8].

Alternatively, the web graph can be distributed over many servers, each of which keeps its portion of the graph in main memory. Computing SALSA using such infrastructure requires two rounds of remote procedure calls to each server (to expand the result set to a neighborhood set, and to determine the linkage between neighbors). In this scenario, it is possible to compute SALSA scores for the results in on the order of 100 milliseconds. However, this approach incurs substantial hardware expenditure, since storing the web graph in main memory requires many servers.

In this paper, we examine the question of whether it is possible to perform much of the computation ahead of query-time, while also reducing the hardware requirements that are required for computing SALSA at query-time. Our approach

has an off-line and an on-line phase. During the off-line phase, we compute a "score map" for each vertex in the web graph, which assigns scores to the vertex and some or all of its neighbors. During the on-line, query-time phase, we retrieve the score map for each result of the query, look up each result in all score maps and add up its scores, and use the sum as the total score of the result.

We evaluated this algorithm using the same data sets and the same retrieval performance measures that were used in earlier comparisons of PageRank, HITS and SALSA [3, 11, 12]. We found that our new algorithm is significantly more effective than PageRank and HITS, although not quite as effective as the best on-line version of SALSA known to us. We also explored the tradeoff between the size of the score map and retrieval performance. Given that any commercial deployment of our algorithm would require the score maps to be distributed over multiple servers and kept in main memory on each, the size of a score map governs the number of required servers. We found that it is possible to limit the size of the score map quite substantially without giving up too much in terms of effectiveness.

The remainder of this paper is structured as follows: Section 2 describes the data sets and the performance measures we used. Section 3 briefly reviews the on-line SALSA algorithm and its effectiveness, as measured against our data sets. Section 4 motivates the "score map" concept by demonstrating that singleton-seed SALSA is not very effective if we retain only the score of the seed itself. Section 5 defines a version of SALSA that starts from a single seed document and captures the scores of the seed and its neighbors in a score map; this algorithm outperforms PageRank and HITS. Section 6 shows that we can improve effectiveness further by using a wider definition of neighborhood inspired by Dean and Henzinger's Companion algorithm. Section 7 examines the efficiency-effectiveness tradeoff when retaining only the k highest scores in the score map. Section 8 studies the effectiveness of our approach for queries with varying degrees of specificity. Section 9 surveys related work. Finally, section 10 offers concluding remarks and avenues for future research.

2. EXPERIMENTAL SETUP

The evaluations presented in this paper were conducted on the same two data sets used in several earlier studies [3, 11, 12]. These two data sets are a large web graph and a substantial set of queries with associated results, some of which were labeled by human judges.

The web graph was obtained by performing a breadth-first search web crawl that retrieved 463,685,607 pages. These pages contain 17,672,011,890 hyperlinks (after eliminating duplicate links embedded in the same web page), which refer to a total of 2,897,671,002 distinct URLs. The mean out-degree of a crawled web page is 38.11; the mean in-degree of discovered pages (whether crawled or not) is 6.10.

The query set was produced by sampling 28,043 queries from the Live Search query log in a frequency-biased fashion, and retrieving a total of 66,846,214 result URLs for these queries, or about 2,838 results per query on average. It should be pointed out that our web graph covers only 9,525,566 pages or 14.25% of the result set. 485,656 of the results in the query set (about 17.3 results per query) were rated by human judges as to their relevance to the given query using a six point scale, the ratings being "definitive", "excellent", "good", "fair", "bad", and "detrimental". Results were selected for judgment based on their commercial search engine placement; in other words, the subset of labeled results is biased towards documents considered relevant by pre-existing ranking algorithms.

This study uses three alternative retrieval performance measures to quantify how effective a ranking algorithm is: the *normalized discounted cumulative gain* [4], the *mean average precision*, and the *mean reciprocal rank*.

In the following, given a rank-ordered vector of n results[1], let *rat(i)* be the rating of the result at rank i, with 5 being "definitive" and 0 being "detrimental" or "unlabeled". We define the *discounted cumulative gain* at document cut-off value k to be:

$$DCG@k = \sum_{i=1}^{k} \frac{1}{\log(1+i)} \left(2^{rat(i)} - 1\right)$$

The *normalized discounted cumulative gain* $NDCG@k$ of a scored result set is defined to be the $DCG@k$ of the result set rank-ordered according to the scores divided by the $DCG@k$ of the result set rank-ordered by an "ideal" scoring function, one that rank-orders results according to their rating.

Furthermore, let *rel(i)* be 1 if *rel(i)* >= 3. The *precision* $P@k$ at document cut-off value k is defined to be $\frac{1}{k}\sum_{i=1}^{k} rel(i)$, *i.e.* the fraction of relevant results among the k highest-ranking results. The *average precision* at document cut-off value k is defined to be:

$$AP@k = \frac{\sum_{i=1}^{k} rel(i)\, P@i}{\sum_{i=1}^{n} rel(i)}$$

The *reciprocal rank* at document cut-off value k is defined to be:

$$RR@k = \begin{cases} \frac{1}{i} & \text{if } \exists i \le k : rel(i) = 1 \wedge \forall j < i : rel(j) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Our implementations of NDCG, MAP, and MRR — described in [10] — handle results with tied scores in a principled manner. In earlier work [11, 12, 3], we used the same measures and the same tie-aware implementations.

3. THE SALSA ALGORITHM

Among the first link-based ranking algorithms is Jon Kleinberg's *Hypertext-Induced Topic Search* (HITS) algorithm [5]. The HITS algorithm takes the result set of a query as input, expands the result set to a *base set* by adding the immediate neighbors of each result, and constructs a neighborhood graph from the base set by including all edges in the full web graph that connect base set vertices. Kleinberg suggested to consider only neighbors that are non-affiliated, *e.g.* web pages that are on a different host or in a different domain than the result page they are connected to. Moreover, he suggested to limit the number of *ancestors* of each result (web pages linking to the result) to 50, by sampling the ancestor set uniformly if there are too many.

Once the neighborhood graph has been constructed, HITS computes two scores for each vertex in that graph: an *authority score* and a *hub score*. The authority score estimates how relevant a page is to the query that produced the result set; the hub score estimates whether a page contains valuable links to authoritative pages. Authority and hub scores

[1]We assume that the result vector contains all relevant results.

mutually enforce each other: a page receives a high authority score if it is linked to by pages with high hub scores, and a page receives a high hub score if it links to pages with high authority scores. Concretely, if A is the adjacency matrix of the neighborhood graph, then the authority scores of the base set are the principal eigenvector of the matrix $A^T A$, and the hub scores of the base set are the principal eigenvector of the matrix AA^T (modulo normalization). These eigenvectors can be computed using the standard power iteration method.

Lempel and Moran's *Stochastic Approach to Link-Sensitivity Analysis* [6, 7] is a variation of Kleinberg's algorithm. SALSA takes a result set R as input, and constructs a neighborhood graph from R in precisely the same way as HITS. Similarly, it computes an authority and a hub score for each vertex in the neighborhood graph, and these scores can be viewed as the principal eigenvectors of two matrices. However, instead of using the straight adjacency matrix that HITS uses, SALSA weighs the entries according to their in- and out-degrees. If we define the *inverse in-degree matrix* I such that $I_{u,v}$ is $in(v)^{-1}$ if the neighborhood graph contains an edge (u, v) and 0 otherwise, and we define the *inverse out-degree matrix* O such that $O_{u,v}$ is $out(u)^{-1}$ if the neighborhood graph contains an edge (u, v) and 0 otherwise, then the SALSA authority scores are the principal eigenvector of the matrix $I^T O$, and the SALSA hub scores are the principal eigenvector of the matrix OI^T.

The SALSA computation can be viewed as a stochastic process (hence the name), or more precisely as performing two independent random walks over the neighborhood graph, an *authority walk* and a *hub walk*. The authority walk starts out at any node with at least one incoming link; and each transition consists in choosing an incoming link at random, following it to reach some ancestor, and then selecting one of the ancestor's outgoing links at random and following it. Likewise, the hub walk starts out at any node with at least one outgoing link; and each transition consists in choosing an outgoing link at random, following it to reach some descendant, and then selecting one of the descendant's incoming links at random and following it. In this view, the SALSA authority vector is the stationary probability distribution of the authority walk, and the hub vector is the stationary probability distribution of the hub walk.

In earlier work [11, 12], we evaluated the effectiveness (the retrieval performance) of both HITS and SALSA, comparing them to BM25F [14] (a state-of-the art ranking algorithms based on textual features, including anchor text) and PageRank [13]. We found that HITS and SALSA hub scores are not particularly useful for ranking purposes, and that HITS authority scores are slightly more effective than PageRank scores, but that SALSA authority scores are substantially more effective than either. We note that the highest effectiveness is achieved by combining BM25F, our best single feature, with a link-based feature. We also found, quite surprisingly, that SALSA performed best when sampling between 3 and 8 ancestors per result (depending on the performance measure), as opposed to the 50 ancestors suggested by Kleinberg and Lempel & Moran.

In subsequent work [3], we discovered that the performance of SALSA can be improved further by selecting the neighbors of results using *consistent sampling* instead of random sampling. Consistent sampling is deterministic, and it preserves set overlap – if two sets overlap by a certain fraction, then (in expectation) consistent samples of both sets will overlap by the same fraction. Consistent sampling can be implemented using various methods, the best-known of which is min-wise hashing [1]. In the algorithms below, the function $\mathcal{C}_n(A)$ samples n elements consistently from set A; $\mathcal{C}_n(A) = A$ if $|A| \leq n$.

As it turns out, the performance of SALSA can be increased further by sampling the descendants of each result instead of including them all into the neighborhood graph. In order to present this variant of the SALSA algorithm, it is helpful to first introduce some notation.

Given a web graph (V, E) with vertex set V and edge set $E \subseteq V \times V$, a *link section predicate* P selects edges $(u, v) \in E$. Based on what we learned in previous studies, we consider only the *id* link section predicate in this paper:

$$id(u, v) \quad \Leftrightarrow \quad domain(u) \neq domain(v)$$

where $domain(u)$ denotes the domain of URL u. So, *id* is true only for inter-domain links.

The *descendant set* O^P of a vertex u with respect to a link-selection predicate P is defined to be:

$$O^P(u) = \{v \in V : (u, v) \in E \wedge P(u, v)\}$$

The *ancestor set* I^P of a vertex v with respect to a link-selection predicate P is defined to be:

$$I^P(v) = \{u \in V : (u, v) \in E \wedge P(u, v)\}$$

Given a web graph (V, E), a result set $R \subseteq V$, a link selection predicate P, and ancestor and descendant sampling parameters a and b, CS-SALSA (SALSA with consistent sampling of result ancestors and descendants) returns a scoring function $s : R \to \mathbb{R}$ mapping results to real-valued authority scores:[2]

1. Set the neighborhood vertex set:
   $B = \bigcup_{v \in R} \{v\} \cup \mathcal{C}_a(I^P(v)) \cup \mathcal{C}_b(O^P(v))$
2. Set the neighborhood edge set:
   $N = \{(u, v) \in E : u \in B \wedge v \in B \wedge P(u, v)\}$
3. Let B^A be $\{u \in B : in(u) > 0\}$.
4. For all $u \in B$: $s[u] := \begin{cases} \frac{1}{|B^A|} & \text{if } u \in B^A \\ 0 & \text{otherwise} \end{cases}$
5. Repeat until s converges:
   (a) For all $u \in B^A$:
   $$s'[u] := \sum_{(v,u) \in N} \sum_{(v,w) \in N} \frac{s[w]}{out(v)in(w)}$$
   (b) For all $u \in B^A$: $s[u] := s'[u]$

Table 1 shows the effectiveness of CS-SALSA for the *id* link selection predicate and ranging values of a and b. The evaluation was performed using the data sets described in section 2 and used in previous studies [3, 11, 12]. Note that effectiveness is highest for $a = 2$ and $b = 1$. This behavior is quite surprising, and we do not have a good explanation for it. However, the phenomenon is real; it manifests itself even when we evaluate against different web graphs and query sets. To put the CS-SALSA's NDCG@10 of 0.1820 into perspective, as reported in [11], the NDCG@10 of PageRank on the same data is 0.0917, and that of BM25F is 0.2209.

[2]Given that hub scores have been shown to be ineffective for ranking purposes, we ignore them in this exposition.

$a\backslash b$	0	1	2	3	4	5
0	0.1707	0.1713	0.1715	0.1705	0.1699	0.1689
1	0.1727	0.1803	0.1799	0.1793	0.1789	0.1781
2	0.1761	**0.1820**	0.1816	0.1811	0.1811	0.1806
3	0.1753	0.1810	0.1809	0.1804	0.1805	0.1801
4	0.1736	0.1797	0.1801	0.1801	0.1803	0.1799
5	0.1722	0.1793	0.1800	0.1801	0.1801	0.1799

NDCG@10

$a\backslash b$	0	1	2	3	4	5
0	0.0705	0.0711	0.0714	0.0706	0.0702	0.0695
1	0.0688	0.0743	0.0744	0.0741	0.0738	0.0734
2	0.0703	**0.0745**	0.0743	0.0741	0.0740	0.0737
3	0.0697	0.0737	0.0736	0.0732	0.0732	0.0729
4	0.0688	0.0726	0.0726	0.0726	0.0727	0.0723
5	0.0679	0.0720	0.0721	0.0721	0.0720	0.0719

MAP@10

$a\backslash b$	0	1	2	3	4	5
0	0.2499	0.2519	0.2518	0.2493	0.2480	0.2458
1	0.2456	**0.2570**	0.2563	0.2555	0.2546	0.2533
2	0.2480	0.2557	0.2554	0.2544	0.2542	0.2531
3	0.2454	0.2521	0.2520	0.2508	0.2505	0.2498
4	0.2421	0.2486	0.2488	0.2484	0.2483	0.2474
5	0.2395	0.2470	0.2473	0.2469	0.2470	0.2464

MRR@10

Table 1: Effectiveness of CS-SALSA, sampling (up to) a ancestors and b descendants of each result using consistent sampling.

$a\backslash b$	0	5	10	15	20	25
0	0.0342	0.0423	0.0434	0.0437	0.0438	0.0438
5	0.0366	0.0364	0.0364	0.0365	0.0365	0.0364
10	0.0285	0.0282	0.0282	0.0282	0.0282	0.0282
15	0.0304	0.0297	0.0297	0.0297	0.0297	0.0297
20	**0.0624**	0.0623	0.0623	0.0623	0.0623	0.0623
25	0.0599	0.0608	0.0609	0.0609	0.0608	0.0608

NDCG@10

$a\backslash b$	0	5	10	15	20	25
0	0.0069	0.0093	0.0096	0.0098	0.0098	0.0098
5	0.0078	0.0079	0.0079	0.0079	0.0079	0.0079
10	0.0054	0.0055	0.0055	0.0055	0.0055	0.0055
15	0.0059	0.0059	0.0059	0.0059	0.0059	0.0059
20	0.0161	**0.0165**	0.0165	0.0165	0.0165	0.0165
25	0.0151	0.0159	0.0160	0.0160	0.0159	0.0159

MAP@10

$a\backslash b$	0	5	10	15	20	25
0	0.0304	0.0418	0.0426	0.0431	0.0429	0.0430
5	0.0337	0.0338	0.0338	0.0338	0.0338	0.0338
10	0.0227	0.0228	0.0228	0.0228	0.0228	0.0228
15	0.0254	0.0251	0.0251	0.0251	0.0251	0.0251
20	0.0647	0.0676	0.0676	0.0676	0.0676	**0.0676**
25	0.0581	0.0639	0.0639	0.0639	0.0639	0.0639

MRR@10

Table 2: Effectiveness SS-SALSA-0, sampling (up to) a ancestors and b descendants of each result.

4. SINGLETON-SEED SALSA

The remainder of this paper introduces the *singleton-seed SALSA* algorithm and various refinements. The original SALSA has two stages at query time: neighborhood expansion for a results set and eigenvector calculation. SS-SALSA involves an offline calculation, finding a separate neighborhood expansion for each document in the corpus, giving it a score vector based on eigenvector calculation. Then at query time the score vectors are summed to give the overall SS-SALSA scores. It should be noted that the summed scores are not mathematically equivalent to the original SALSA scores. However, this SALSA-like algorithm has the advantage of query-time efficiency, because graph expansion and eigenvector calculation happen before query time.

The first variant of singleton-seed SALSA we present is a "strawman": it literally treats every vertex in the web graph as a singleton result set and applied CS-SALSA to it. This means that for each eigenvector calculation we keep only the score of the seed vertex. As we will see, this algorithm is quite ineffective; the main reason for presenting it is to motivate the concept of a *score map* later on.

The off-line phase of our strawman algorithm SS-SALSA-0 takes a web graph (V, E), a link selection predicate P, and ancestor and descendant sampling parameters a and b, and returns a global authority scoring function $g : V \to \mathbb{R}$ mapping vertices to real-valued scores:

For all $x \in V$:

1. Set the neighborhood vertex set:
   $B = \{x\} \cup \mathcal{C}_a(I^P(x)) \cup \mathcal{C}_b(O^P(x))$
2. Set the neighborhood edge set:
   $N = \{(u, v) \in E : u \in B \wedge v \in B \wedge P(u, v)\}$
3. Let B^A be $\{u \in B : in(u) > 0\}$.
4. For all $u \in B$: $s[u] := \begin{cases} \frac{1}{|B^A|} & \text{if } u \in B^A \\ 0 & \text{otherwise} \end{cases}$
5. Repeat until s converges:
   (a) For all $u \in B^A$:
   $$s'[u] := \sum_{(v,u)\in N} \sum_{(v,w)\in N} \frac{s[w]}{out(v)in(w)}$$
   (b) For all $u \in B^A : s[u] := s'[u]$
6. $g[x] := s[x]$

Given a result set $R \subseteq V$, the on-line phase of SS-SALSA-0 simply assigns the score $g[r]$ to each $r \in R$.

Table 2 shows the effectiveness of SS-SALSA-0 for the *id* link selection predicate and ranging values of a and b, using the same data sets and performance measure as were used in the previous section. It is quite apparent that SS-SALSA-0 performs much worse than CS-SALSA.

5. GOOD NEIGHBORS MATTER

Having shown that computing a single score per vertex in the web graph is ineffective, we now present a refined version of singleton-seed SALSA that computes a *score map* for each vertex that captures the score of the seed itself and all its neighbors. This means that the presence of a page in the results set can confer a score on some other page. We will use the notation $[[\star \mapsto x]]$ to denote a function that maps every key to x, and as before, we will write $s[k] := v$ to indicate that s is extended to map k to v.

The off-line phase of this algorithm SS-SALSA-1 takes a web graph (V, E), a link selection predicate P, and ancestor and descendant sampling parameters a and b, and returns a global authority scoring function $g : V \to (V \to \mathbb{R})$ mapping vertices to score maps, which in turn map vertices to real-valued scores:

For all $x \in V$:

1. Set the neighborhood vertex set:
   $B = \{x\} \cup \mathcal{C}_a(I^P(x)) \cup \mathcal{C}_b(O^P(x))$
2. Set the neighborhood edge set:
   $N = \{(u, v) \in E : u \in B \wedge v \in B \wedge P(u, v)\}$
3. Let B^A be $\{u \in B : in(u) > 0\}$.
4. $s := [[\star \mapsto 0]]$
5. For all $u \in B$: $s[u] := \begin{cases} \frac{1}{|B^A|} & \text{if } u \in B^A \\ 0 & \text{otherwise} \end{cases}$
6. Repeat until s converges:
   (a) For all $u \in B^A$:
   $$s'[u] := \sum_{(v,u) \in N} \sum_{(v,w) \in N} \frac{s[w]}{out(v)in(w)}$$
   (b) For all $u \in B^A : s[u] := s'[u]$
7. $g[x] := s$

Given a result set $R \subseteq V$, the on-line phase of SS-SALSA-1 assigns the score $\sum_{v \in R} g[v][r]$ to each $r \in R$. In a reasonable implementation, the score maps of the entire result set are retrieved first, and then the summation iterates for each result over the cached score maps.

Table 3 shows the effectiveness of SS-SALSA-1 for the *id* link selection predicate and ranging values of a and b, using the same data sets and performance measure as used previously. The SS-SALSA-1 algorithm is significantly more effective than SS-SALSA-0; in fact it outperforms both PageRank and HITS. The highest NDCG@10 value shown in the table is for the sampling parameters $a = 0, b = 5$. Under this parameter setting, each score map has 6 key-value pairs: the seed, no ancestors, and 5 descendants. If we represent score maps as association lists of 8-byte integer URL identifiers and 4-byte floating-point scores, then each map requires 72 bytes. It is also worth noting that the highest NDCG@10 value is in the rightmost column of the table; as we will see below, effectiveness continues to increase as we go beyond $b = 5$.

$a \backslash b$	0	1	2	3	4	5
0	0.0342	0.1095	0.1253	0.1328	0.1367	**0.1397**
1	0.0372	0.1006	0.1123	0.1181	0.1220	0.1248
2	0.0381	0.1003	0.1122	0.1181	0.1219	0.1248
3	0.0390	0.1006	0.1122	0.1182	0.1219	0.1247
4	0.0395	0.1008	0.1123	0.1182	0.1219	0.1247
5	0.0401	0.1011	0.1127	0.1186	0.1223	0.1250

NDCG@10

$a \backslash b$	0	1	2	3	4	5
0	0.0069	0.0409	0.0482	0.0518	0.0536	**0.0551**
1	0.0080	0.0376	0.0432	0.0461	0.0480	0.0495
2	0.0084	0.0374	0.0431	0.0461	0.0479	0.0494
3	0.0087	0.0376	0.0432	0.0461	0.0480	0.0495
4	0.0090	0.0377	0.0432	0.0461	0.0480	0.0494
5	0.0092	0.0379	0.0435	0.0464	0.0482	0.0495

MAP@10

$a \backslash b$	0	1	2	3	4	5
0	0.0304	0.1704	0.1922	0.2012	0.2060	**0.2093**
1	0.0351	0.1592	0.1769	0.1861	0.1915	0.1953
2	0.0374	0.1589	0.1768	0.1862	0.1914	0.1951
3	0.0395	0.1593	0.1770	0.1861	0.1914	0.1948
4	0.0407	0.1594	0.1769	0.1860	0.1915	0.1948
5	0.0425	0.1602	0.1777	0.1869	0.1922	0.1954

MRR@10

Table 3: Effectiveness of SS-SALSA-1, sampling (up to) a ancestors and b descendants of each result.

6. EXTENDING THE NEIGHBORHOOD

Our next variant of singleton-seed SALSA takes a more expansive view of what constitutes the neighborhood of a vertex, inspired by Dean and Henzinger's *Companion* algorithm [2]. The Companion algorithm included not only the ancestors and descendants of a vertex into its neighborhood, but also the descendants of ancestors and the ancestors of descendants.

The off-line phase of this algorithm SS-SALSA-2 takes a web graph (V, E), a link selection predicate P, ancestor and descendant sampling parameters a and b, and mate and sibling sampling parameters c and d,[3] and returns a global authority scoring function $g : V \to (V \to \mathbb{R})$ mapping vertices to score maps, which in turn map vertices to real-valued scores:

For all $x \in V$:

1. Set the neighborhood vertex set:
   $Anc = \mathcal{C}_a(I^P(x))$
   $Des = \mathcal{C}_b(O^P(x))$
   $Sib = \bigcup_{x \in Anc} \mathcal{C}_c(O^P(x))$
   $Mat = \bigcup_{x \in Des} \mathcal{C}_d(I^P(x))$
   $B = \{x\} \cup Anc \cup Des \cup Sib \cup Mat$
2. Set the neighborhood edge set:
   $N = \{(u, v) \in E : u \in B \wedge v \in B \wedge P(u, v)\}$
3. Let B^A be $\{u \in B : in(u) > 0\}$.

[3] *Siblings* are nodes with a shared ancestor, *mates* are nodes with a shared descendant.

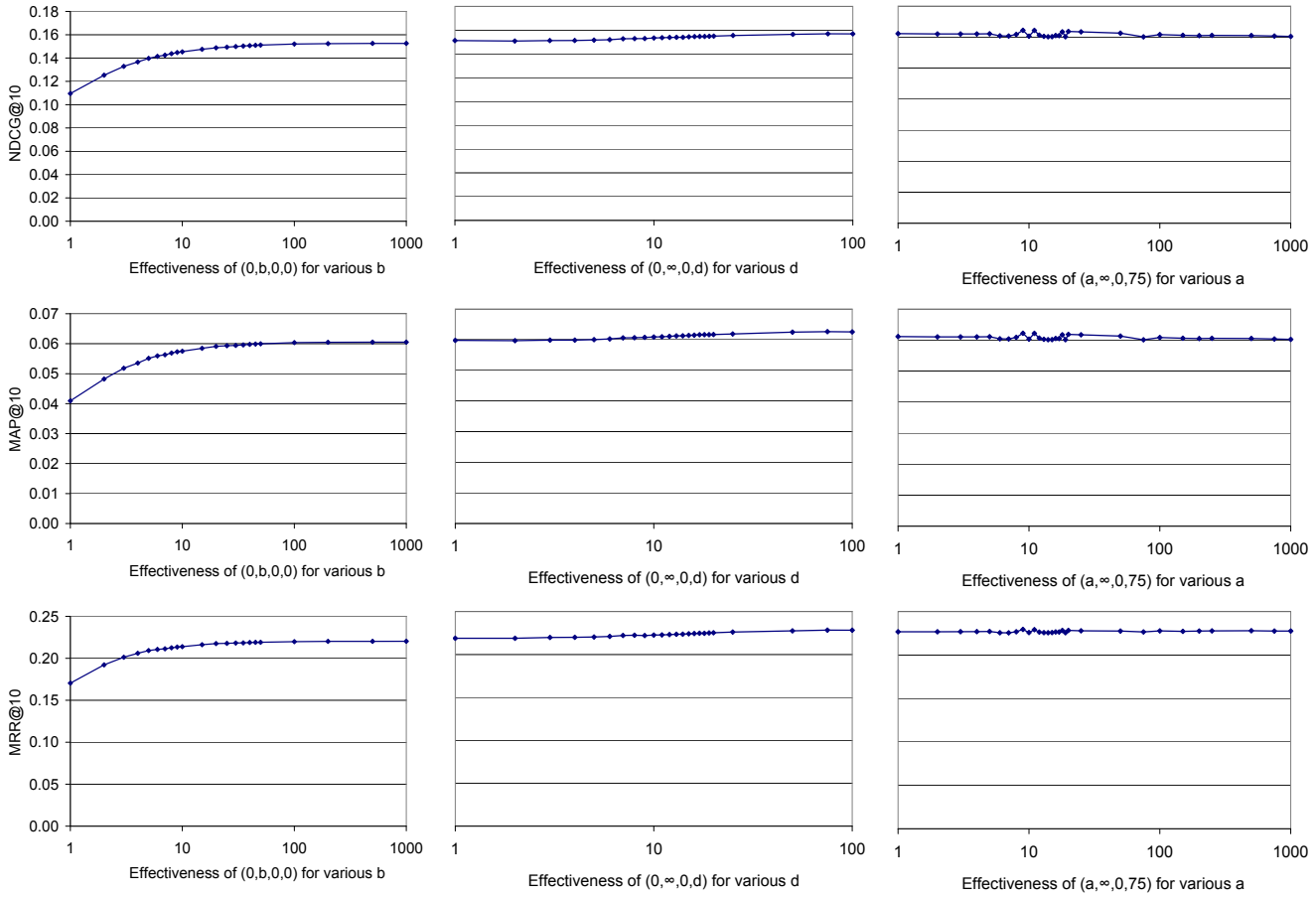

Figure 1: Effectiveness of SS-SALSA-2, sampling (up to) a ancestors, b descendants, c descendants of ancestors, and d ancestors of descendants of each result.

4. $s := [[\star \mapsto 0]]$
5. For all $u \in B$: $s[u] := \begin{cases} \frac{1}{|B^A|} & \text{if } u \in B^A \\ 0 & \text{otherwise} \end{cases}$
6. Repeat until s converges:
   (a) For all $u \in B^A$:
   $$s'[u] := \sum_{(v,u)\in N} \sum_{(v,w)\in N} \frac{s[w]}{out(v)in(w)}$$
   (b) For all $u \in B^A : s[u] := s'[u]$
7. $g[x] := s$

The on-line phase of SS-SALSA-2 is identical to that of SS-SALSA-1.

Note that SS-SALSA-2 has four free parameters. Evaluating any single parameter combination is quite expensive (multiple days on the hardware available to us), so we were not able to probe the entire space. Instead, we explored the effectiveness of the algorithm across one parameter dimension at a time, and fixed that parameter at the locally optimal setting when switching dimensions. It is possible and even likely that this approach did not locate the optimal parameter combination; we merely provide a lower bound on what is possible.

Figure 1 illustrates our efforts at tuning the parameters. The vertical axis shows effectiveness in terms of NDCG@10, the horizontal axis (plotted on a log scale) shows the value of one parameter with the three others remaining fixed; the line shows the effectiveness of SS-SALSA-2 at the given parameter settings. In the left graph, a, c, and d are fixed at 0, and b is probed in the range from 1 to 1000. Since the horizontal axis is plotted on a log scale, the graph does not show the extreme values for b, 0 (which does worst) and ∞ (which does best with an NDCG@10 of 0.1526). In the middle graph, b is fixed at ∞ ("include all descendants"), a and c are fixed at 0, and d ranges from 1 to 100. The NDCG@10 is maximal at 0.1569 for $d = 75$. In the right graph, b is fixed at ∞, d is fixed at 75, c is fixed at 0, and a ranges from 1 to 1000. The line in the right graph is more erratic than in the left and middle graphs, but no value of a improves on $a = 0$ ("include no ancestors"). The parameter c has no effect when $a = 0$, so it is not tuned here.

Our tuned parameters $(0, \infty, 0, 75)$ are striking in that they include no ancestors and all descendants of the seed. This differs from other algorithms; for example CS-SALSA works best with a small number of both ancestors and de-

k	bytes	NDCG@10	MAP@10	MRR@10
1	12	0.0342	0.0069	0.0304
2	24	0.1234	0.0475	0.1942
3	36	0.1369	0.0539	0.2090
4	48	0.1430	0.0565	0.2149
5	60	0.1470	0.0583	0.2186
10	120	0.1534	0.0612	0.2249
15	180	0.1549	0.0618	0.2269
20	240	0.1557	0.0621	0.2274
∞	n/a	0.1569	0.0626	0.2284
PR	4	0.0917	0.0271	0.0972
AS	379	0.1826	0.0759	0.2589

Table 4: Effectiveness and space requirements of SS-SALSA-3, sampling no ancestors, all descendants, no siblings, and 75 mates of each result, and using the k highest scores in the neighborhood of each result for ranking. For comparison, the table also shows the space requirements and effectiveness of PageRank and "approximate SALSA".

scendants from each seed (see Table 1). If the role of neighborhood expansion is to include certain nodes that are 'useful' for SALSA computation, it seems that finding useful hubs and authorities from a singleton seed requires a different strategy.

In the non-singleton case, an algorithm like CS-SALSA expands from a result set that might already contain multiple useful hubs and authorities. Then neighborhood expansion gives it multiple chances to find further useful nodes. By contrast SS-SALSA must find a graph with multiple useful hubs and authorities from a single seed. This explains why relatively aggressive expansion is required, taking many nodes via parameters $b = \infty$ and $d = 75$.

When expanding aggressively from a single seed, there is a greater risk of suffering from noise or drift. This might be the reason that ancestors and descendants are treated differently ($a = 0$ and $b = \infty$). Web authors create inter-domain links for a variety reasons, so the ancestors of a node may contain a mixture of hub pages and other types of page. Perhaps ancestor expansion yields useful pages at too low a rate, and the SALSA calculation suffers. It may be that single-seed expansion succeeds more reliably for a seed that is a useful hub, because a hub page has a coherent set of descendants yielding useful authorities at a sufficient rate for the SALSA computation to succeed. Overall, sampling of descendants might be a more reliable strategy.

We leave it to future work to design an experiment to test this explanation. This would involve identifying pages as hubs or authorities, then testing whether the score maps of hubs make a greater contribution to the overall effectiveness of SS-SALSA.

7. THE EFFECTIVENESS-SPACE TRADE-OFF

Increasing the number of sampled descendants per result and including its mates into the neighborhood graph leads to significant gains in effectiveness, but at a crippling cost: the more neighbors we include in each score map, the larger the size of the map. In order to be useful to a commercial search engine, ranking needs to be very fast; so if singleton-seed SALSA were to be used as a ranking feature, the score maps would have to be retrieved very fast, and thus would have to reside in main memory, distributed over a cluster of servers. The hardware cost of this cluster is directly proportional to the memory footprint of each score map. So, we should investigate whether it is possible to reduce the size of each score map without losing too much in terms of effectiveness, exploring the effectiveness-space tradeoff.

The SS-SALSA-3 algorithm is similar to SS-SALSA-2, but it caps the size of each score map. Given a mapping $s : V \to \mathbb{R}$, the function $\text{top}_k(s)$ returns a mapping that contains the k highest values along with their keys, and that maps every other key to 0.

The off-line phase of SS-SALSA-3 takes a web graph (V, E), a link selection predicate P, ancestor and descendant sampling parameters a and b, mate and sibling sampling parameters c and d, and a score map cap k, and returns a global authority scoring function $g : V \to (V \to \mathbb{R})$ mapping vertices to score maps, which in turn map vertices to real-valued scores:

For all $x \in V$:

1. Set the neighborhood vertex set:
   $Anc = \mathcal{C}_a(I^P(x))$
   $Des = \mathcal{C}_b(O^P(x))$
   $Sib = \bigcup_{x \in Anc} \mathcal{C}_c(O^P(x))$
   $Mat = \bigcup_{x \in Des} \mathcal{C}_d(I^P(x))$
   $B = \{x\} \cup Anc \cup Des \cup Sib \cup Mat$
2. Set the neighborhood edge set:
   $N = \{(u, v) \in E : u \in B \wedge v \in B \wedge P(u, v)\}$
3. Let B^A be $\{u \in B : in(u) > 0\}$.
4. $s := [[\star \mapsto 0]]$
5. For all $u \in B$: $s[u] := \begin{cases} \frac{1}{|B^A|} & \text{if } u \in B^A \\ 0 & \text{otherwise} \end{cases}$
6. Repeat until s converges:
   (a) For all $u \in B^A$:
   $$s'[u] := \sum_{(v,u) \in N} \sum_{(v,w) \in N} \frac{s[w]}{out(v)in(w)}$$
   (b) For all $u \in B^A$: $s[u] := s'[u]$
7. $g[x] := \text{top}_k(s)$

It is worth pointing out that the score map $g[x]$ computed by SS-SALSA-3 for a vertex x does not necessarily contain a (non-zero) score for x.

The on-line phase of SS-SALSA-3 is identical to that of SS-SALSA-1 and SS-SALSA-2.

Table 4 shows the effectiveness of SS-SALSA-3 for various values of k and fixed sampling parameters $a = 0$, $b = \infty$, $c = 0$, and $d = 75$. The second column shows the size in bytes of each feature vector, assuming that we represent score maps as association lists of 8-byte integer URL identifiers and 4-byte floating-point scores. The bottom row shows the performance of SS-SALSA-2 for the same sampling parameters; SS-SALSA-2 being a special case of SS-SALSA-3 with $k = \infty$. It is possible to get quite close to the effectiveness of SS-SALSA-2 for fairly small values of k, *i.e.* modest

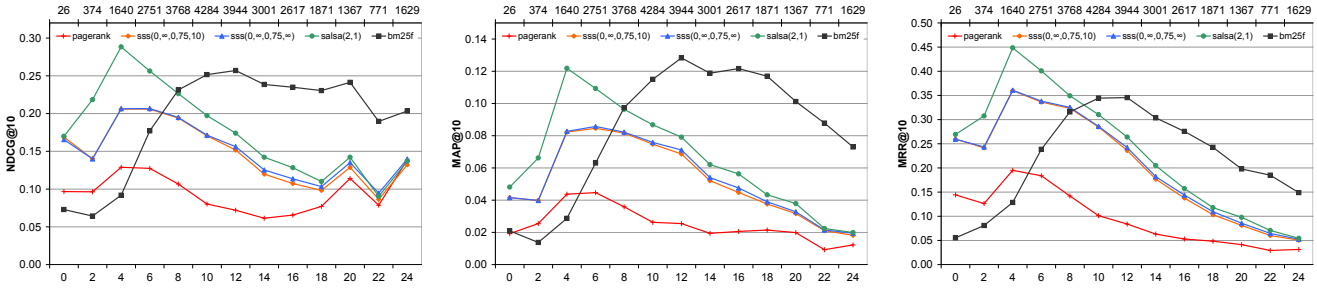

Figure 2: Effectiveness of different ranking features, broken down by query specificity.

score map sizes and therefore a manageable hardware expense. For example, a search engine could maintain a distributed in-memory table mapping 10 billion URLs to 120 byte score maps each using a cluster of around 80 computers with 16 GB of RAM each. By comparison, the approach described by Gollapudi et al. [3] produces an NDCG@10 of 0.1826 on the same web graph and query set, but associates a 379-byte feature vector with each URL in the web graph.

8. BREAKDOWN BY QUERY SPECIFICITY

Finally, we investigated the correlation between the specificity of a query and the effectiveness of SS-SALSA-2 and SS-SALSA3. Our definition of specificity is purely syntactic: we do not attempt to capture whether one query is more specific than another query in any semantic sense; rather, we just compare the cardinalities of the result sets produced by the various queries. We view queries with large result sets (for which good ranking algorithms are all the more important) as general, and queries with small result sets as specific.

Unfortunately, our query set does not contain the size of the entire result set. Therefore, we adopt our previous approach [11, 12] and approximate query specificity by the sum of inverse document frequencies of the individual query terms. The inverse document frequency of a query term t with respect to a document collection C is defined to be $\log \frac{|C|}{|C(t)|}$, where $C(t)$ is the subset of documents in C containing t. This approach assumes that the individual terms of a multi-word query occur independently of each other; the fact that this assumption is unwarranted means that we may over-estimate the specificity of a query. Although not perfect, using query term IDF as a measure of specificity is at least directionally accurate.

We broke our query set down into 13 subsets according to specificity, and ranked the queries in each subset using CS-SALSA with $a = 2$ and $b = 1$,[4] SS-SALSA-2 with $a = 0$, $b = \infty$, $c = 0$ and $d = 75$, and SS-SALSA-3 with the same sampling parameters and $k = 10$. For comparison, we also include the performance of PageRank [13] and BM25F [14].

Figure 2 shows the performance of each feature for each query subset. The figure shows three graphs, one per performance measure. The lower horizontal axis of each graph shows query specificity (the most general queries being on the far left); the upper horizontal axis shows the size of each of the 13 query subsets. The vertical axis denotes retrieval performance. Each graph contains five curves, one for each of the chosen features. As we already observed in previous work, BM25F performs best for medium-specific to fairly specific queries. This is not too surprising: BM25F measures the textual similarity between the query and the results, and is bound to work better in the presence of uncommon, specific query terms. Link-based features on the other hand perform best for very general queries. Among the link-based features, PageRank is the least effective and CS-SALSA is the most effective across all levels of query specificity. SS-SALSA-2 and SS-SALSA-3 fall in-between, and they perform quite similarly across the spectrum.

[4]Note that the CS-SALSA curve differs from the curve shown in [12] due to the use of consistent sampling and different and better sampling parameters.

9. RELATED WORK

The idea of leveraging hyperlinks in the ranking of web search results goes back to Marchiori [9], who proposed to view the number of links to a web page as a measure of its popularity with other web page authors. Page *et al.* refined this idea through the PageRank measure [13], which takes not only the number of referring hyperlinks into account, but also their "quality" (the PageRank score of the endorsing web page divided by the number of endorsements on that page).

Kleinberg introduced the concept of considering only the neighborhood of the result set in the web graph for ranking purposes. The underlying hypothesis is that result web pages are likely to be topically related (by virtue of each page containing the query terms), and that topically related pages tend to cite each other and be co-cited by "hub" pages. This concept is captured by the HITS algorithm, which computes a hub and an authority score for each page in a result set using a mutual recurrence relationship between the two score vectors [5]. Lempel and Moran's SALSA algorithm [6, 7] is a variant of HITS that takes the number of in- and out-links of each page in the neighborhood graph into account.

Two recent studies evaluated the effectiveness of the aforementioned ranking algorithms [11, 12]. The studies used a 17.7 billion edge web graph and 28,043 queries with partially labeled results (the same data sets that were used in this pager), and found that SALSA is substantially more effective than HITS, PageRank and in-degree.

One significant problem of SALSA and HITS is that they both are query-time algorithms, and that they are time-intensive to compute. In particular, determining the neighborhood graph of a result set is expensive, since it requires several rounds of remote procedure calls to hyperlink servers maintaining the web graph in main memory. Gollapudi *et al.* attempted to address this problem by proposing an algorithm for approximating SALSA that consists of an off-

line (indexing-time) and an on-line (query-time) stage [3]. The off-line stage computes a summary of the neighborhood of each web page; the summaries are stored in a summary server. The on-line stage retrieves the summaries of each result to a query (using a single round of remote procedure calls), uses them to construct an approximate neighborhood graph, and computes SALSA on that approximate graph.

The work described in this paper is similar to the *approximate SALSA* algorithm. The main difference is that in *approximate SALSA*, the eigenvector computation that is at the heart of SALSA is performed at query-time, whereas in our approach, it is performed off-line, further reducing the amount of computation that has to be performed at query time. Inherent to both algorithms is a tradeoff between effectiveness and space requirements. Compared to *approximate SALSA*, the algorithm described in this paper performs particularly well in the space-restrained portion of the spectrum; it is possible to achieve decent effectiveness at a storage expense of, say 120 bytes per web page.

10. CONCLUSION

This paper describes a novel approach to link-based ranking that represents an interesting trade-off between effectiveness and efficiency. The algorithm described in this paper is a more effective ranking feature than PageRank, HITS authority scores, or simple in-degree; but it is somewhat less effective than SALSA authority scores. The algorithm consists of an off-line stage and an on-line stage. Most of the computation happens during the off-line stage; the on-line stage simply consists in retrieving a "score map" for each result to a query, and adding up scores for each result from each score map. In this respect, the algorithm is similar to PageRank (the off-line phase of which requires a very expensive eigenvector computation over the entire web graph; whereas the on-line phase merely looks up a single score for each result). By contrast, the original SALSA algorithm is performed entirely on-line, and introduces a substantial latency into the query processing pipeline, most of it attributable to assembling the neighborhood graph of the result set. A recent variant [3], Approximate SALSA, eliminates some of that latency by summarizing the neighborhoods of each web page off-line. However, the summaries require 379 bytes per page and must be stored in memory for fast query-time access. This means that a large search engine would require a cluster of "summary lookup" servers to use the approximate SALSA technique. The SS-SALSA technique described in this paper also requires lookup servers, but the score maps are one-third the size, so hardware costs are also one third. This increase in efficiency is associated with a drop in effectiveness. Therefore SS-SALSA occupies a point on the efficiency-effectiveness tradeoff curve where efficiency is important, but effectiveness is maintained. The effectiveness of SS-SALSA is still better than PageRank or HITS.

11. REFERENCES

[1] A. Broder, M. Charikar, A. Frieze, M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences* 60(3):630–659, 2000.

[2] J. Dean and M. Henzinger. Finding related pages in the World Wide Web. In *Proc. of the 8th International World Wide Web Conference*, pages 389–401, 1999.

[3] S. Gollapudi, M. Najork and R. Panigrahy. Using Bloom filters to speed up HITS-like ranking algorithms. In *Proc. of the 5th Workshop on Algorithms and Models for the Web Graph*, pages 195–201, 2007.

[4] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[5] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[6] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks and ISDN Systems*, 33(1–6):387–401, 2000.

[7] R. Lempel and S. Moran. SALSA: The stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 19(2):131–160, 2001.

[8] G. Linden. Marissa Mayer at Web 2.0. Online at: *http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html*

[9] M. Marchiori. The quest for correct information on the Web: hyper search engines. In *Computer Networks and ISDN Systems*, 29(8–13):1225–1236, 1997.

[10] F. McSherry and M. Najork. Computing information retrieval performance measures efficiently in the presence of tied scores. In *30th European Conference on Information Retrieval*, pages 414–421, 2008.

[11] M. Najork, H. Zaragoza and M. Taylor. HITS on the Web: how does it compare? In *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 471–478, 2007.

[12] M. Najork. Comparing the effectiveness of HITS and SALSA. In *16th ACM Conference on Information and Knowledge Management*, pages 157–164, 2007.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[14] H. Zaragoza, N. Craswell, M. Taylor, S. Saria, and S. Robertson. Microsoft Cambridge at TREC–13: Web and HARD tracks. In *Proc. of the 13th Text Retrieval Conference*, 2004.