Using Bloom Filters to Speed Up HITS-like Ranking Algorithms

Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy

Microsoft Research, Mountain View CA 94043, USA

Abstract. This paper describes a technique for reducing the query-time cost of HITS-like ranking algorithm. The basic idea is to compute for each node in the web graph a summary of its immediate neighborhood (which is a query-independent operation and thus can be done off-line), and to approximate the neighborhood graph of a result set at query-time by combining the summaries of the result set nodes. This approximation of the query-specific neighborhood graph can then be used to perform query-dependent link-based ranking algorithms such as HITS and SALSA. We have evaluated our technique on a large web graph and a substantial set of queries with partially judged results, and found that its effectiveness (retrieval performance) is comparable to the original SALSA algorithm, while its efficiency (query-time speed) is substantially higher.

1 Introduction

One of the fundamental problems in Information Retrieval is the *ranking problem*: how to arrange the documents that satisfy a query into an order such that the documents most relevant to the query rank first. Traditional ranking algorithms proposed by the pre-web IR community were mostly based on similarity measures between the terms (words) in the query and the documents satisfying the query.

In addition to structured text, web pages also contain hyperlinks between web pages, which can be thought of as peer endorsements between content providers. Marchiori suggested early on to leverage incoming hyperlinks as another feature in ranking algorithms [9], and the simplistic idea of merely counting in-links quickly evolved into more sophisticated link-based ranking algorithms that take the quality of an endorsing web page into account.

Link-based ranking algorithms can be grouped into two classes: query-independent ones such as in-link count or Google's famous PageRank [12], and query-dependent ones such as Kleinberg's HITS [4, 5] and Lempel & Moran's SALSA [6, 7]. The aforementioned algorithms were described in seminal papers that inspired a great deal of subsequent work; however, there has been little published work on the effectiveness (that is, the accuracy of the ranking) of these algorithms.

A recent study using a 17-billion edge web graph and a set of 28,043 queries with partially judged results concluded that SALSA, a query-dependent link-based ranking algorithm, is substantially more effective than HITS, PageRank

and in-degree, although it is not as effective as the state-of-the-art textual ranking algorithm.

Unfortunately, SALSA in particular and HITS-like algorithms in general require a substantial amount of expense at query-time. The vast fraction of this expense is devoted to determining the *neighborhood graph* of the results to a query; the subsequent computation of scores for the nodes in the neighborhood graph is cheap in comparison, despite the fact that most HITS-like algorithms use power iteration to compute the fixed-points of the score vectors. The fact that HITS-like algorithms incur substantial computational cost at query-time puts them at a crippling disadvantage to query-independent algorithms such as PageRank: according to Marissa Mayer, Google's VP of Search Products & User Experience, delaying Google's response time by half a second led to a 20% drop in query traffic (and revenue) from the user population subjected to the delay [8].

This paper describes a technique that dramatically lowers the query-time cost of HITS-like ranking algorithms, *i.e.* algorithms that perform computations on the distance-one neighborhood graph of the results to a query. The basic idea is to compute a *summary* of the neighborhood of each page on the web (an operation that is query-independent and thus can be done off-line, say at index construction time), and to use these summaries at query time to *approximate* the neighborhood graph of the result set and to compute scores using the approximate graph. We have evaluated this approach using the same methodology and the same data sets that were used in the earlier comparisons of in-degree, PageRank, HITS, and SALSA, and found that applying our technique to SALSA does not impair its effectiveness and at the same substantially improves its efficiency.

The remainder of this paper is structured as follows: section 2 reviews the HITS and SALSA algorithms; section 3 explains our technique for summarizing the neighborhood of each web page; section 4 describes the experimental validation of our technique; and section 5 offers concluding remarks.

2 HITS and SALSA

Both HITS and SALSA are query-dependent link-based ranking algorithms: given a web graph (V, E) with vertex set V and edge set $E \subseteq V \times V$ (where edges/links between vertices/pages on the same web server are typically omitted), and the set of result URLs to a query (called the *root set* $R \subseteq V$) as input, both compute a *base set* $B \subseteq V$, defined to be:

$$B = R \cup \bigcup_{u \in R} \{v \in V : (u, v) \in E\} \cup \bigcup_{v \in R} \mathcal{S}_n[\{u \in V : (u, v) \in E\}]$$

where $\mathcal{S}_n[X]$ denotes a uniform random sample of n elements from set X ($\mathcal{S}_n[X] = X$ if $|X| < n$). The sampling parameter n will typically be below 100, and its choice has a significant impact on the effectiveness of SALSA in particular [11]. The neighborhood graph (B, N) consists of base set B and edge set $N = \{(u, v) \in E : u \in B \wedge v \in B\}$.

Both HITS and SALSA compute two scores for each node $u \in B$: an authority score $A(u)$, estimating how authoritative u is on the topic induced by the query, and a hub score $H(u)$, indicating whether u is a good reference to many authoritative pages. In the case of HITS, hub scores and authority scores are computed in a mutually recursive fashion:

1. For all $u \in B$ do $H(u) := \sqrt{\frac{1}{|B|}}, A(u) := \sqrt{\frac{1}{|B|}}$.
2. Repeat until H and A converge:
   (a) For all $v \in B$ do $A'(v) := \sum_{(u,v)\in N} H(u)$
   (b) For all $u \in B$ do $H'(u) := \sum_{(u,v)\in N} A(v)$
   (c) For all $u \in B$ do $H(u) := \frac{1}{\|H'\|_2} H'(u), A(u) := \frac{1}{\|A'\|_2} A'(u)$

where $\|X\|_2$ is the euclidean norm of vector X.

By contrast, SALSA authority scores can be computed independently of hub scores, which is interesting insofar as that SALSA (and HITS) hub scores are poor ranking features. The algorithm for computing SALSA authority scores is:

1. Let B^A be $\{u \in B : in(u) > 0\}$
2. For all $u \in B$ do $A(u) := \begin{cases} \frac{1}{|B^A|} \text{ if } u \in B^A \\ 0 \quad \text{otherwise} \end{cases}$
3. Repeat until A converges:
   (a) For all $u \in B^A$ do $A'(u) := \sum_{(v,u)\in N} \sum_{(v,w)\in N} \frac{A(w)}{out(v)in(w)}$
   (b) For all $u \in B^A$ do $A(u) := A'(u)$

For reasons of space, we omit the algorithm for computing SALSA hub scores; the interested reader is referred to [6] or [11]. The latter paper compares the effectiveness of HITS and SALSA, and finds that SALSA authority scores are a substantially better ranking feature than HITS authority scores. Our experimental validation in section 4 employs the same methodology and data sets.

When performed on a web-scale corpus, both HITS and SALSA require a substantial amount of query time processing. Much of this processing is attributable to the computation of the neighborhood graph. The reason for this is that the entire web graph is enormous. A document collection of five billion web pages induces a set of about a quarter of a trillion hyperlinks. Storing this web graph on disk would make lookup operations unacceptably slow due to the inherent seek time limitations of hard drives. On the other hand, the graph is too big to be stored in the main memory of any single machine; therefore, it has to be partitioned across many machines. In such a setup, the cost of a link lookup is governed by the cost of a remote procedure call (RPC). A sophisticated implementation of the SALSA algorithm against a distributed link database will batch many lookup operations into a single RPC request to reduce latency and will query all link servers in parallel, but even so it will require four rounds of concurrent requests: one round to map the root set URLs to short representations; the second round to determine the pages linking to the root set; the third round to determine the pages pointed to by the root set; and the fourth round to determine the edges induced by the base set. The SALSA implementation

used to perform the experiments described in [11] required 235 milliseconds per query for the most effective parametrization of SALSA, and 2.15 seconds per query for the most expensive parametrization. Over 90% of the time spent was spent on performing the RPC calls to the link servers in order to assemble the neighborhood graph, as opposed to computing scores on that graph.

3 Summarizing Neighborhood Graphs

In this paper, we present a technique to substantially lower the query-time cost of HITS and SALSA. We do so by moving the most expensive part of the computation off-line. At index-construction time, we build a database mapping web page URLs to summaries of their neighborhoods. At query time, we rank the results satisfying a query by looking up each result in the summary database (an operation that requires only one round of RPCs, as opposed to four), approximating the neighborhood graph of the result set based on the neighborhood summaries of the constituent results, and computing SALSA (or HITS) scores using that approximation of the neighborhood graph. In the experimental evaluation below, we will show that this approximation has no detrimental effect on the quality of the ranking.

As outlined above, our summary of the neighborhood graph of a web page u consists of a summary of the *ancestors* (the pages that link to u) and a summary of the *descendants* (the pages that u links to), each consisting of a Bloom filter containing a limited-size subset of ancestors or descendants plus a much smaller subset containing explicit web page identifiers (64-bit integers). A Bloom filter is a space-efficient probabilistic data structure that can be used to test the membership of an element in a given set; the test may yield a false positive but never a false negative. A Bloom filter represents a set using an array A of m bits (where $A[i]$ denotes the ith bit), and uses k hash functions h_1 to h_k to manipulate the array, each h_i mapping some element of the set to a value in [1,m]. To add an element e to the set, we set $A[h_i(e)]$ to 1 for each $1 \leq i \leq k$; to test whether e is in the set, we verify that $A[h_i(e)]$ is 1 for all $1 \leq i \leq k$. Given a Bloom filter size m and a set size n, the optimal (false-positive minimizing) number of hash functions k is $\frac{m}{n} \ln 2$; in this case, the probability of false positives is $(\frac{1}{2})^k$. For an in-depth description of Bloom filters, the reader is referred to [1, 3]. In the following, we will write $BF[X]$ to denote the Bloom filter representing the set X.

While the original SALSA algorithm samples the neighborhood (and in particular the ancestors) uniformly *at random*, we use consistent sampling [2]. Let $\mathcal{C}_n[X]$ denote a consistent unbiased sample of n elements from set X, where $\mathcal{C}_n[X] = X$ if $|X| < n$. Consistent sampling is *deterministic*; that is, when sampling n elements from a set X, we will always draw the same n elements. Moreover, any element x that is sampled from set A is also sampled from subset $B \subset A$ if $x \in B$. We define the set $I_n(u)$ to be a consistent sample $\mathcal{C}_n[\{v \in V : (v,u) \in E\}]$ of (at most) n of the ancestors of u; and the set $O_n(u)$ to be a consistent sample $\mathcal{C}_n[\{v \in V : (u,v) \in E\}]$ of n of the de-

scendants of u. For each page u in the web graph, we compute a summary $(BF[I_a(u)], I_b(u), BF[O_c(u)], O_d(u))$.

At query time, given a result set R, we first look up the summaries for all the results in R. Having done so, we compute a cover set

$$C = R \cup \bigcup_{u \in R} I_b(u) \cup \bigcup_{u \in R} O_d(u)$$

Next, we construct a graph consisting of the vertices in C. We fill in the edges as follows: For each vertex $u \in R$ and each vertex $v \in C$, we perform two tests: If $BF[I_a(u)]$ contains v, we add an edge (v, u) to the graph; if $BF[O_c(u)]$ contains v, we add an edge (u, v) to the graph. This graph serves as our approximation of the neighborhood graph of R; we use it to compute SALSA (or HITS) scores using the same algorithm as described above in section 2.

Observe that the approximate neighborhood graph differs from the exact neighborhood graph in three ways:

- In the exact graph, we do not sample the vertices directly reachable from the root set, but rather include them all.
- The approximate graph only contains edges from $C \cap I_a(u)$ to $u \in R$ and from $u \in R$ to $C \cap O_c(u)$. In other words, it excludes edges between nodes in C that are not part of the root set.
- We do not use exact set representations for $I_a(u)$ and $O_c(u)$, but approximate them by using Bloom filters. This introduces additional edges whose number depends on the false positive probability of the Bloom filter. Using k hash functions, we add about $2^{-k+1}|C||R|$ spurious edges in the graph.

At first glance, it is non-obvious why this approximation of the neighborhood graph should preserve any of the properties relevant to ranking algorithms. After all, the approximation may exclude actual edges due to the sampling process, and add phantom edges due to the potential for false positives inherent to Bloom filters. However, it is worth noting that consistent sampling preserves co-citation relationships between pages in the result set. The experimental validation described in the following section confirms that the summarization algorithm indeed preserves properties relevant to link-based ranking.

4 Experimental Validation

Our experimental validation is based on the two data sets used in [10, 11]: a large web graph with 2.9 billion nodes and 17.7 billion edges, and a set of 28,043 queries, each with 2,838 results on average, 17.3 of which were rated by human judges on a six-point scale. For more details on how the graph was collected and how results were judged, the reader is referred to the earlier papers. We use three popular measures of effectiveness (or *retrieval performance*): the *normalized discounted cumulative gain* (NDCG), the *mean average precision* (MAP), and the *mean reciprocal rank* (MRR) measures. All three measures are normalized

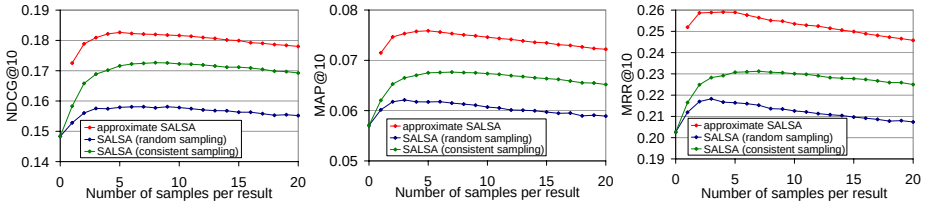

Fig. 1. Effectiveness of authority scores computed using different parameterizations of original and approximate SALSA; measured in terms of NDCG, MAP and MRR.

to range between 0 and 1; higher values indicate better performance. Again, the reader is referred to [10, 11] for the full definitions of these measures.

In our experiments, we used $k = 10$ hash functions, and we fixed the parameters a and c at 1000; that is, we included a sample of (up to) 1000 ancestors or descendants into each Bloom filter. The Bloom filters averaged 227 bytes for ancestor sets and 72 bytes for descendant sets. We measured the effectiveness of SALSA using our summarization technique for b and d ranging from 2 to 20, and compared it to original SALSA with the same sampling values. Figure 1 depicts the results. The figure contains three graphs, one for each performance measure (NDCG, MAP, and MRR). The horizonal axis shows the number of samples (the b and d parameters of our summarization-based SALSA, and the n parameter of the original SALSA); the vertical axis shows the retrieval performance. Each graph contains three curves; the blue (dark) curve showing the performance of the original SALSA; the green (medium) curve shows the performance of the original SALSA with consistent instead of random sampling; and the red (light) curve showing that of our summarization-based version. Using consistent instead of random sampling substantially improves the performance under all measures. However, our new approximate version of SALSA outperforms the original SALSA algorithm under all measures. Performance is maximal for b and d between 4 and 5, depending on the measure. For $b = d = 5$, each summary is 379 bytes in size (227 bytes for $BF[I_{1000}]$, 40 bytes for I_5, 72 bytes for $BF[O_{1000}]$, and 40 bytes for O_5).

Our current implementation of summarization-based SALSA does not yet employ a distributed summary server; we use our distributed link server to compute summaries. However, since a summary server is similar to, and indeed simpler than, a link server, we can draw on our experiences with the latter to estimate what the query-time performance of a production system would be. We measured the performance of our current implementation, subtracted the time required to compute the summaries, and added the time we have observed for retrieving a vector of links of the size of an average summary from our link server. These measurements suggest that it would take us an average of 171 milliseconds per query to compute approximate SALSA scores. This compares favorably to the 235 milliseconds per query of our implementation of the original SALSA algorithm. Moreover, we have not spent any time on optimizing

the code for constructing approximate neighborhood graphs, and believe that further speedups are possible.

5 Conclusion

This paper describes a technique for reducing the query-time cost of query-dependent link-based ranking algorithms that operate on the distance-one neighborhood graph of the result set, such as HITS and SALSA. Our technique computes a summary of each page on the web ahead of query-time, and combines these summaries at query-time into approximations of the neighborhood graph of the result set. We tested our technique by applying it to a large real web graph and a sizable collection of real queries with partially assessed results, and were able to demonstrate that the approximate nature of our technique does not have any negative impact on the effectiveness (retrieval performance). Future work includes implementing a distributed summary server and verifying that our technique is indeed faster than the original (all-online) implementation.

References

1. B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13(7):422–426, 1970.
2. A. Broder, M. Charikar, A. Frieze, M. Mitzenmacher. Min-Wise Independent Permutations. *Journal of Computer and System Sciences* 60(3):630–659, 2000.
3. A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics* 1(4):485–509.
4. 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.
5. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
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. 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.
11. M. Najork. Comparing the Effectiveness of HITS and SALSA To appear in *16th ACM Conference on Information and Knowledge Management*, 2007.
12. 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.