Querying the Web Graph
(Invited Talk)

Marc Najork

Microsoft Research, 1065 La Avenida, Mountain View, CA, USA
najork@microsoft.com
http://research.microsoft.com/~najork/

Abstract. This paper focuses on using hyperlinks in the ranking of web search results. We give a brief overview of the vast body of work in the area; we provide a quantitative comparison of the different features; we sketch how link-based ranking features can be implemented in large-scale search engines; and we identify promising avenues for future research.

Keywords: Web graph, link analysis, ranking, PageRank, HITS, SALSA.

1 The Ranking Problem

One of the fundamental problems of information retrieval is the ranking problem: given a corpus of documents and a query reflecting a user's information need, and having drawn from the corpus all the "result" documents that satisfy the query, order the results by decreasing relevance with respect to the user's information need. The aim of ranking algorithms is to maximize the utility of the result list to the user; or (more subjectively) to maximize the user's satisfaction.

Ranking algorithms deployed in commercial search engines typically draw on a multitude of individual features, where each feature manifests itself as a numerical score, and combine these features in some way, e.g. by weighted linear combination. Features can be classified across many possible dimensions. Figure 1 shows one possible taxonomy incorporating two dimensions: when a feature is computed, and what it is based on. Query-dependent or "dynamic"

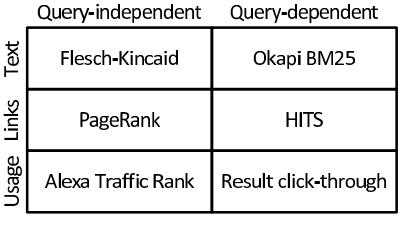

Fig. 1. A two-dimensional classification of ranking features

E. Chavez and S. Lonardi (Eds.): SPIRE 2010, LNCS 6393, pp. 1–12, 2010.

features take the query into account and thus can only be computed at query time, whereas query-independent or "static" features do not rely on the query and thus can be computed ahead of time. Loosely speaking, dynamic features estimate the relevance of a document with respect to a query, while static features estimate its general quality.

Much of the classic research in information retrieval focused on small- to medium-sized curated corpora of high-quality documents, and assumed that queries are issued by trained library scientists. Consequently, early ranking algorithms leveraged textual features, based on the words or "terms" contained in documents and queries. Examples of query-independent textual features are readability measures such as Flesch-Kincaid [15]; examples of query-dependent textual features include BM25 [30]. As information retrieval broadened to web search, some of these assumptions became less tenable (the web is not curated; pages are more variable in length and quality; and users of web search engines are less experienced and tend to frame fairly short queries), but the fact that documents are interconnected by hyperlinks made new features available. Examples of query-independent link-based features include PageRank [28]; examples of query-dependent link-based features include HITS [16] and its many descendants. Finally, the explosive growth in the popularity of the web itself and of commercial web search engines generates an enormous amount of data on user activity both on search engines and the web at large, which can be harnessed into usage-based features. An example of a query-independent usage-based feature is Alexa Traffic Rank [2]; an example of a query-dependent usage-based feature is result click-through rate [31].

In the remainder of this paper, we will focus on link-based features. However, it is worth pointing out that all features – whether text-, link- or usage-based – are generated by human activity: by authors creating documents and endorsing other authors' documents through hyperlinks, and by users surfing the web, framing queries, and clicking on results. So, the ranking problem is about identifying features generated by observable human activities that are well-correlated with human satisfaction with the ordering (or more generally the selection and presentation) of the results of a query. In short, at its very core ranking is neither a mathematical nor an algorithmic problem, but a social one, concerned with the behaviors and interactions of people. Given that search is ultimately a social activity and that so far we are not not able to model human behavior well enough to accurately predict the effectiveness of any new ranking algorithm, the importance of experimental studies cannot be overemphasized.

2 Using Hyperlinks to Rank Web Search Results

Marchiori was the first to propose leveraging hyperlinks for ranking web search results. Specifically, he suggested computing a conventional test-based score for the result pages of a query, and then to propagate these scores (appropriately attenuated) to neighboring pages in the web graph [20]. This work inspired other researchers to consider hyperlinks as ranking features in their own right.

Hyperlinks can be viewed as endorsements made by web page authors of other web pages, and they can be classified along several dimensions. First, we can distinguish between "egotistic" and "altruistic" hyperlinks[1] – links that endorse pages in which the author has a vested interest vs. those that endorse other pages. It is very hard to identify altruistic links with high confidence, but there are some easy-to-determine indicators of egotistic links: links that point to pages on the same web server, on a web server in the same domain, on a server with the same IP address or in the same IP subnet, or registered to the same entity. Second, we can distinguish between "topical" and "templatic" links[2] – links that point to pages that are on the same topic as the linking page vs. those that point to topically dissimilar pages and often are part of a design template applied to the entire web site (for example, links to the site's privacy policy). In this world view, altruistic topical links provide the most meaningful endorsements.

2.1 PageRank

Page (together with Brin, Motwani and Winograd) proposed a purely link-based ranking function that went beyond the straightforward counting of endorsing hyperlinks in two respects: First, he suggested that the reputation of a page should be dependent on the reputation of its endorsing pages, and second, he suggested that the endorsement ability of a page should be divided among the pages it endorses. Furthermore, to prevent degenerate behavior in the presence of cycles and sink pages, he suggested affording each page a guaranteed minimal score. The resulting ranking function is the now-famous PageRank [28]. In order to define it formally, we will need some notation.

Web pages and hyperlinks induce a graph with vertex (page) set V and edge (link) set $E \subseteq V \times V$. We write $I(v, E)$ to denote the set of pages $\{u : (u, v) \in E\}$ that link to v, and $O(u, E)$ to denote the set of pages $\{v : (u, v) \in E\}$ that u links to. The teleportation vector $t : V \to [0, 1]$ controls the guaranteed minimum score of each page, the link weight matrix $W : V \times V \to [0, 1]$ is typically informed by the graph's adjacency matrix and controls the endorsement power of each hyperlink, and the damping factor λ balances the influence of either component. Using this notation, the PageRank $p(v)$ of a page v is defined in its most general form as the fixed point of the following recurrence relation:

$$p(v) = \lambda t(v) + (1 - \lambda) \sum_{u \in V} p(u) W(u, v)$$

or, using linear algebra notation, as:

$$p = \lambda t + (1 - \lambda) pW$$

Yet more abstractly, p is the principal eigenvector of the transition matrix T (i.e. the fixed point of the recurrence relation $p = pT$) defined as $T(u, v) =$

[1] Egotistic links are called "nepotistic" by Davison [9] and "intrinsic" by Kleinberg [16].

[2] Qi et al. refer to topical links as "qualified links" [29].

$\lambda t(v) + (1 - \lambda)W(u, v)$, assuming that p adds up to 1. In the classical definition of PageRank, $t(v) = \frac{1}{|V|}$ for all $v \in V$ (i.e. each page has the same guaranteed minimum score), and $W(u, v) = \frac{1}{|O(u,E)|}$ for all $v \in O(u, E)$ and 0 otherwise (i.e. page u uniformly endorses all pages it links to).

p, t and the rows of W are typically constrained to add up to 1, such that they can be viewed as probability distributions. That raises the question of how to deal with a "terminal" or "dangling" web page u that does not contain any hyperlinks, i.e. $\sum_{v \in V} W(u, v) = 0$. Page et al. proposed pruning such terminal vertices from the web graph (which may require multiple iterations, since pruning such terminal vertices and their incoming links may leave other vertices without outgoing links), computing PageRank on the core graph, and finally adding the pruned vertices and edges back in and propagating scores to them [28]. As it turns out, we can ignore the problem. To understand why, let us assume that we add a "phantom vertex" ϕ to the graph, plus a reflexive edge (ϕ, ϕ) and a phantom edge from each terminal to ϕ. The modified graph is free of terminal vertices. We set $t(\phi) = 0$, $W(u, \phi) = 1$ iff $O(u, E) = \emptyset$ and 0 otherwise, $W(\phi, \phi) = 1$, and $W(\phi, v) = 0$ for all $v \in V$. If we compute p on this graph using power iteration, $\|p\|_1$ (the ℓ_1-norm of p) will not change over iterations. Significantly, the $p(v)$ score of any non-phantom vertex is the same as it would be had we computed p using power iteration on the original graph while resigning ourselves that zero-sum rows in W will cause $\|p\|_1$ to shrink. The score mass that simply disappears when computing PageRank on the original graph can be found in the phantom node of the augmented graph. In other words, adding the phantom node and phantom edges to the graph is a useful intellectual device, but not actually needed for computing PageRank.

One very popular interpretation of the PageRank formula is the "random surfer model". In this model, a "surfer" traverses the web graph, visiting a new vertex at each transition. The surfer either "steps" with probability $1 - \lambda$ or "jumps" with probability λ. The destination vertex of a step is conditioned on the departure vertex, while that of a jump is not. Assuming that W is based on the web graph's adjacency matrix, taking a step means following a link. The surfer transitions from vertex u to vertex v with probability $T(u, v) = \lambda t(v) + (1-\lambda)W(u, v)$. In this interpretation, $p(v)$ is the probability that the surfer is at vertex v at any given point in time. The random surfer model is very intuitive, and it relates PageRank to Markov random walks. Unfortunately, it also has led to a fair amount of confusion in the community: the random surfer model suggests that PageRank is modeling the web surfing behavior of users, i.e. of consumers of web content; the endorsement model described above suggests that PageRank is modeling the cross-reference behavior of authors, i.e. of producers of web content. We subscribe to the latter interpretation. Incidentally, commercial search engines have fairly direct ways of observing user behavior (e.g. through browser toolbars) and thus little need to model it.

Since PageRank is query-independent, it can be computed off-line. The PageRank vector p is typically computed using power iteration. Major commercial search engines maintain web graphs with tens of billions of vertices, and thus

compute p in a distributed setting. There are two standard approaches of doing this. The first approach uses data-parallel frameworks such as MapReduce [10], Hadoop [13] or DryadLINQ [32]. In this setting, the link weight matrix W, the teleportation vector t, and the old and new score vectors p reside on disk and are processed in a streaming fashion. Multiplying W and p corresponds to a join operation in relational algebra followed by a group-by, and the data streams to be joined must be sorted on the join key. W can be sorted ahead of time, while p has to be re-sorted every iteration. The second, ad-hoc approach partitions p, t and the rows of W such that all entries corresponding to pages on the same web server fall into the same partition. The new score vector is kept in memory (partitioned over many machines). W, t and the old score vector, which all reside on disk, are read in a streaming fashion, and fractions of the old scores are used to increment entries of the new score vector. These updates have a random-access pattern, but most will affect the local partition of the score array due to link locality – typically, most hyperlinks in a web page refer to other pages on the same web server. After every iteration, the new score vector is written to disk and becomes the old vector in the next iteration. The advantage of the ad-hoc approach over the MapReduce-approach is that it does not require any sorting; the two disadvantages are that it requires more engineering (e.g. to provide for fault-tolerance) and that computing scores for a larger graph requires more memory (typically by provisioning additional machines).

PageRank scores are computed off-line, and used at query-time to score results. As stated above, major search engines maintain corpora of tens of billions of documents. At the same time, they typically aim to answer queries within a fraction of a second [19]. In order to answer queries over such a large corpus and with such tight latency constraints, engines maintain all or at least the "hot" part of the index in main memory, partitioned across many machines, with each partition replicated multiple times across machines to achieve fault tolerance and increase throughput. The web corpus is commonly partitioned by document, i.e. all the terms of a given document are stored in the same sub-index. When a query arrives at the search engine, it is distributed to a set of index-serving machines that together hold the full index. Each machine finds the set of results in its sub-index that satisfy the query, scores these results using locally available features, and sends the top-k results to a result aggregation machine, which integrates them into the overall result set, possibly performing a more in-depth scoring. Since PageRank is a query-independent document score, the PageRank vector can be partitioned in the same way as the web corpus, and index-serving nodes can determine the PageRank scores of results using local table lookups. In other words, the query-time portion of PageRank is both extremely efficient and extremely scalable.

PageRank's elegance and intuitiveness, combined with the fact that it was credited for much of Google's extraordinary success, led to a plethora of research. Broadly speaking, this research falls into four classes: PageRank's mathematical properties (e.g. [17]); variations of the PageRank formula (e.g. [3]); computing PageRank efficiently (e.g. [14,21]); and adversarial attacks against PageRank.

Table 1. Effectiveness of various ranking features, in isolation

Feature	Class	NDCG@10	Source
PageRank	link/q-ind	0.092	[23]
HITS	link/q-dep	0.104	[23]
inter-domain indegree	link/q-ind	0.106	[23]
SS-SALSA-3	link/q-dep	0.153	[25]
SALSA	link/q-dep	0.158	[24]
SALSA-SETR	link/q-dep	0.196	[26]
BM25F	text+link/q-dep	0.221	[23]

Alas, there is a paucity in experimental validations of its effectiveness as a ranking function. Part of this is due to the fact that it is hard to assemble a large web corpus, and expensive to compile human judgments required in Cranfield-style evaluations of ranking effectiveness. Early studies tried to overcome these obstacles by leveraging existing search engines to obtain linkage information [4]; more recent work by our group performed substantial web crawls and used a test set compiled by a commercial search engine [23]. To our surprise, we found that PageRank in its classic form (with uniform teleportation and link bias, and $\lambda = 0.15$) is even less effective than simply counting altruistic (inter-domain) hyperlinks. Table 1 summarizes the results of these studies, which were all based on the same data sets.

At first glance, it is surprising that PageRank does not outperform inter-domain link-counting – after all, link-counting ignores the reputation of the linking page, and considers only the immediate neighborhood of each web page. We believe that there are two reasons why classic PageRank fares so poorly: First, it treats all links the same; it ignores whether they are egotistic or altruistic, topical or templatic. Inter-domain link-counting on the other hand will discard links that are obviously egotistic. Second, being credited as an important factor in Google's ranking algorithm, PageRank is under attack by legions of search engine optimizers. In its classic formulation, each page receives a guaranteed minimum score, so the obvious attack is to publish millions of pages that all endorse a single landing page, which will receive the (slightly dampened) sum of the scores of the endorsing pages. The key enabler for spammers is that web pages can be automatically generated on the fly, so publishing even a very large collection of pages is virtually free. This technique is known as link spam or link bombs, and there are several studies on the most effective shape of such link bombs [1,12].

Given that the key enabler of link spam is the low cost of publishing, the appropriate countermeasure is to correlate the teleportation vector with a feature that has actual economic cost. Examples of features that have non-zero cost are domain names (since it requires payment to a registrar), IP addresses (IPv4 addresses in particular are becoming quite scarce), and of course actual traffic on a page [22]. For example, using visits(u) to denote the number of visits to page u in a given amount of time (optionally weighted by the dwell-time), we can define the teleportation vector to discount unpopular pages:

$$t(u) = \frac{\text{visits}(u)}{\sum_{v \in V} \text{visits}(v)}$$

As a second example, using domain(u) to denote the domain of web page u, we can define the teleportation vector to discount domains with many web pages:

$$t(u) = \frac{1}{|\bigcup_{v \in V} \text{domain}(v)| \; |\{v \in V : \text{domain}(u) = \text{domain}(v)\}|}$$

Using the random-surfer analogy, in the probability-λ event of a jump, the surfer does not choose a page uniformly at random, but instead first chooses a domain and then a page within that domain, thereby giving equal minimum-endorsement ability to each (nonzero-cost) domain instead of each (zero-cost) page. As a third example, using addresses(u) to denote the IP address(es) of the web server(s) serving page u, we can define the teleportation vector to discount servers with many web pages while giving credit to multi-hosted pages:

$$t(u) = \frac{1}{|\bigcup_{v \in V} \text{addresses}(v)|} \sum_{a \in \text{addresses}(u)} \frac{1}{|\{v \in V : a \in \text{addresses}(v)\}|}$$

Again using the random-surfer analogy, in the event of a jump the surfer first chooses a web server IP address and then chooses a page served by that machine.

In addition to adjusting the teleportation vector to dilute the ability of link farms to endorse target pages, we could adjust the link weight matrix to prefer altruistic or topical links. As we said above, any altruistic link classifier suffers from a non-negligible false-positive rate: it is impossible to rule out the possibility that two web publishers are colluding. On the other hand, the topicality of a link from u to v can be captured by straightforward means, e.g. by quantifying the textual similarity between u and v, using, say, the cosine similarity between their tf.IDF-weighted term vectors [29]. Given a similarity measure $\sigma : V \times V \to \mathbb{R}$ where higher values indicate higher similarity, we can define W as follows:

$$W(u,v) = \frac{\sigma(u,v)}{\sum_{w \in O(u,E)} \sigma(u,w)} \quad \text{if } (u,v) \in E;\ 0 \text{ otherwise}$$

There are many other possible techniques for distinguishing topical from templatic links, for example segmenting the page into "blocks" and discounting links contained in navigational or advertising blocks [6].

Hopefully the above examples will convince you that there is room for improving PageRank, by finding ways to make it a more effective ranking function and more resilient to link-spam. Any such research should be data-driven: given the availability of large web corpora (e.g. the billion-page ClueWeb09 crawl [8]) and associated test sets (e.g. the TREC 2009 web track judgments [7]), it is possible to evaluate ranking functions in a repeatable fashion. As a corollary, studies of PageRank's mathematical properties and efforts to speed up its computation have more impact if they don't assume t and W to be uniform in any way.

2.2 HITS and Its Variants

At about the same time as Page et al. proposed PageRank, Jon Kleinberg suggested a different link-based ranking algorithm called "Hyperlink-Induced Topic Search" [16]. HITS takes the result set of a query as its input, expands the result set to include immediately neighboring pages in the web graph, projects this expanded vertex set onto the full web graph to obtain a neighborhood graph, and computes scores for each vertex in that neighborhood graph. Since it takes a query's result set as input, HITS is query-dependent, and the latency introduced by computing it at query-time is a major concern to commercial search engines, given the high correlation between response time and audience engagement [19].

Fundamentally, HITS consists of two distinct steps: First, given a result set, compute a neighborhood graph; and second, compute scores for each vertex in the neighborhood graph. Kleinberg suggested that the neighborhood graph should be computed by extending the result set to include all vertices that are within distance 1 in the link graph, ignoring "intrinsic" (egotistic by our terminology) links. In order to keep high-in-degree result vertices from inflating the neighborhood vertex set too much, he suggested including just (say) 50 randomly chosen endorsing vertices of each such high-indegree result. The neighborhood graph consists of the neighborhood vertices and those edges of the full web graph that connect neighborhood vertices and are not intrinsic. To formalize this, we write $\mathcal{R}_n(A)$ to denote a uniform random sample of n elements from set A ($\mathcal{R}_n(A) = A$ iff $|A| \leq n$), and we assume that all intrinsic edges have been removed from the web graph (V, E). Using this notation and given a result set $R \subseteq V$ to query q, Kleinberg's neighborhood graph consists of a neighborhood vertex set V_R and a neighborhood edge set E_R:

$$V_R = \bigcup_{u \in R} \{u\} \cup \mathcal{R}_{50}(I(u, E)) \cup O(u, E)$$

$$E_R = \{(u, v) \in E : u \in V_R \wedge v \in V_R\}$$

Kleinberg furthermore suggested computing two scores for each $v \in V_R$: an authority score $a(v)$ and a hub score $h(v)$, the former indicating whether v is a good authority (i.e. how relevant v is with respect to q), and the latter indicating whether v is a good hub (i.e. if v links to pages that are good authorities). Kleinberg defined a and h in a mutually recursive fashion, as $a(v) = \sum_{u \in I(v, E_R)} h(u)$ and $h(u) = \sum_{v \in O(u, E_R)} a(v)$, and suggested computing the fixed point of this recurrence by performing power iteration and normalizing a and h to unit length after each step. Using linear algebra notation leads to a more concise definition. If we define the neighborhood graph's adjacency matrix as $A(u, v) = 1$ iff $(u, v) \in E_R$ and 0 otherwise, then the authority score vector is the principal eigenvector of the matrix $A^T A$, and the hub score vector is the principal eigenvector of the matrix AA^T. Based on experimental studies [23], authority scores are useful ranking features, while hub scores carry virtually no signal.

Lempel and Moran suggested a variant of HITS called the Stochastic Approach to Link Sensitivity Analysis, or SALSA for short [18]. SALSA combines

ideas from HITS and PageRank: The SALSA authority score vector is the stationary probability distribution of a random walk over the neighborhood graph, where each transition consists of choosing an incoming link and traversing it backwards, and then choosing an outgoing link and traversing it forwards. Using linear algebra notation, the authority score vector is the principal eigenvector of the matrix $I^T O$, where $I(u,v) = \frac{1}{|I(v,E_R)|}$ iff $(u,v) \in E_R$ and 0 otherwise, and $O(u,v) = \frac{1}{|O(u,E_R)|}$ iff $(u,v) \in E_R$ and 0 otherwise. Despite their similarity, SALSA scores are significantly better ranking features than HITS scores [24].

In the course of performing the aforementioned experimental studies [23,24] into the effectiveness of HITS and SALSA as ranking features, we found that the choice of neighborhood graph has a significant impact on effectiveness. We discovered four modifications to the neighborhood selection algorithm that each increase effectiveness: first, using consistent instead of random sampling; second, sampling both the endorsed as well as the endorsing vertices of each result; third, omitting edges that don't touch results; and fourth, sampling the eligible edges [26]. Using $\mathcal{C}_n(A)$ to denote an unbiased consistent sample [5] of n elements from set A, we select the neighborhood graph of result set R as follows:

$$V_R = \bigcup_{u \in R} \{u\} \cup \mathcal{C}_a(I(u,E)) \cup \mathcal{C}_b(O(u,E))$$

$$E_R = \{(u,v) \in E : (v \in R \wedge u \in V_R \cap \mathcal{C}_c(I(v,E))) \vee (u \in R \wedge v \in V_R \cap \mathcal{C}_d(O(u,E)))\}$$

The free variables a, b, c, and d in the above formulas determine how many adjacent vertices and edges are sampled for each result vertex. In our experiments, effectiveness was maximal for a, b in the mid-single digits and c, d around 1000. The *SALSA-SETR* row of Table 1 provides effectiveness numbers.

Experimenting with HITS-like ranking algorithms requires fast access to vertices and edges in the web graph. While it would be possible to use a standard relational database to store the graph, extracting the neighborhood graph of a given result set exhibits a very random access pattern, such that disk latency would become a serious bottleneck. For this reason, we developed the Scalable Hyperlink Store [27], a bespoke system that maintains a web graph in main memory, distributed over multiple machines and employing compression techniques that leverage structural properties of web graphs, e.g. link locality. This infrastructure enables us to get one-minute turnarounds when scoring 100 TREC queries; however, the time required for scoring an individual query is pushing the boundary of what is acceptable for a commercial search engine, where the aim is to keep overall query latencies within fractions of a second.

In order to overcome this problem, we experimented with techniques for performing as much of the SALSA computation off-line as possible. We explored two approaches. The first and more radical approach is to assume that each page in the web graph is a singleton result set (to some unspecified query), and to off-line perform a separate SALSA computation for each page. More specifically, for each page u, we extract the neighborhood graph $(V_{\{u\}}, E_{\{u\}})$ around u, compute SALSA authority scores for all vertices in $V_{\{u\}}$, and store a mapping from u to the k highest-scoring $v \in V_{\{u\}}$ together with their scores. At query time,

having determined the result set R for the given query, we retrieve the entries for each $v \in R$ from the mapping (yielding a multi-set S of $k|R|$ vertex-score pairs), and for each $v \in R$ we find all occurrences of v in S and sum up their scores to produce an overall score for v. The parameter k controls a trade-off between ranking effectiveness and required space. The *SS-SALSA-3* row in Table 1 shows the effectiveness for $k = 10$ (i.e. using 120 bytes per page in the web corpus). The effectiveness is below that of Lempel and Moran's "classic" SALSA, but the query-time portion of the computation is much cheaper, simply requiring a table lookup for each result. The table can be distributed across multiple machines in the same way the index is in a modern search engine; the score multi-set of each result can be looked up by the same index-serving machine that held the result, but the final scoring has to be performed by the result aggregator.

The second, less radical approach is to move the neighborhood graph extraction off-line while keeping the score computation on-line [26]. The basic idea is to compute and store a short summary for each node in the web graph, consisting of two small samples of endorsing and endorsed vertices, and two Bloom filters containing larger samples of endorsing and endorsed vertices. At query time, we look up the summary of each vertex in the result set, we use these summaries to construct an approximation of the neighborhood graph, and we compute SALSA authority scores on this approximate graph. The size of a summary is governed by how many neighbors are sampled and how many hash functions are used by the Bloom filter, and (as one might expect) there is a trade-off between size and ranking effectiveness. Choosing parameters that lead to 500 byte summaries makes this algorithm as effective as the (purely online) SETR variant of SALSA described above. The summary table can be distributed across the index-serving machines and summaries can be passed along with results to the result aggregator; however, the construction of the approximate neighborhood graph and the subsequent scoring requires all summaries, and therefore has to be performed on the machine responsible for query distribution and result aggregation.

Much of the design space in HITS-like ranking algorithms remains to be explored. For example, HITS considers only the distance-one neighborhood of the result set. In the Companion algorithm [11], Dean and Henzinger suggested including more-distant neighbors; how does this impact ranking effectiveness? As another example, HITS and SALSA both discard egotistic hyperlinks, but make no attempts to incorporate link topicality. Could effectiveness be improved by weighing each edge according to the textual similarity of the linked pages, in a fashion similar to what we suggested above for PageRank? And as a final example, how affected are HITS and SALSA by link spam, and could we improve their resiliency by using ideas similar to what was described above, e.g. by sampling endorsing and endorsed vertices in a traffic-biased fashion?

3 Conclusion

This paper gives a high-level overview of the two main approaches to leveraging hyperlinks as ranking features: query-independent features as exemplified

by the PageRank algorithm, and query-dependent features as exemplified by the HITS algorithm. The main research challenge in this space is to identify features that (in combination with many other features) improve ranking effectiveness, but that at the same are very efficient to determine at query time. For PageRank-style features, query-time efficiency is not a significant issue, but ranking effectiveness is, due to the increase in link spam as well as the decreasing fraction of on-topic links. Effectiveness can be improved by biasing PageRank towards on-topic links and against "spammy" web pages. On the other hand, various descendants of HITS produce highly effective ranking features, but have a high query-time cost. Two approaches to containing this expense are to implement highly-optimized infrastructure for executing HITS-like computations, or to devise ways to to move as much of the computation as possible off-line.

The ranking problem is about identifying and leveraging observable human activities (production by authors and consumption by readers) that correlate well with human satisfaction with the performance of an IR system. In order to truly solve this problem, we need a model of these authors and readers. But at present, we do not possess any model with predictive abilities – i.e. a model that would allow us to *a priori* predict the performance of a new ranking feature and that would stand up to subsequent experimental validation. In the absence of such a model, experimental validation is of paramount importance!

References

1. Adali, S., Liu, T., Maddon-Ismail, M.: Optimal link bombs are uncoordinated. In: 1st Intl. Workshop on Adversarial Information Retrieval on the Web, pp. 58–69 (2005)
2. Alexa Traffic Rank, `http://www.alexa.com/help/traffic-learn-more`
3. Baeza-Yates, R., Boldi, P., Castillo, C.: Generalizing PageRank: damping functions for link-based ranking algorithms. In: 29th Annual ACM Conference on Research and Development in Information Retrieval, pp. 308–315 (2006)
4. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. ACM Trans. Internet Technology 5, 231–297 (2005)
5. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. In: 30th Annual ACM Symposium on Theory of Computing, pp. 327–336 (1998)
6. Cai, D., He, X., Wen, J.-R., Ma, W.-Y.: Block-level link analysis. In: 27th Annual ACM Conference on Research and Development in Information Retrieval, pp. 440–447 (2004)
7. Clarke, C.L.A., Craswell, N., Soboroff, I.: Overview of the TREC 2009 Web track. In: 18th Text REtrieval Conference (2009)
8. The ClueWeb09 dataset, `http://boston.lti.cs.cmu.edu/Data/clueweb09/`
9. Davison, B.D.: Recognizing nepotistic links on the Web. In: AAAI Workshop on Artificial Intelligence for Web Search, pp. 23–28 (2000)
10. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: 6th Symposium on Operating Systems Design & Implementation, pp. 137–149 (2004)

11. Dean, J., Henzinger, M.: Finding related pages in the World Wide Web. In: 8th Intl. World Wide Web Conference, pp. 389–401 (1999)
12. Gyöngyi, Z., Garcia-Molina, H.: Link spam alliances. In: 31st Intl. Conference on Very Large Data Bases, pp. 517–528 (2005)
13. Hadoop, `http://hadoop.apache.org/`
14. Kamvar, S.D., Haveliwala, T.H., Manning, C.D., Golub, G.H.: Extrapolation methods for accelerating PageRank computations. In: 12th Intl. World Wide Web Conference, pp. 261–270 (2003)
15. Kincaid, J.P., Fishburn, R.P., Rogers, R.L., Chissom, B.S.: Derivation of new readability formulas for Navy enlisted personnel. Research Branch Report 8-75, U.S. Naval Air Station, Memphis (1975)
16. Kleinberg, J.: Authoritative sources in a hyperlinked environment. J. ACM 46, 604–632 (1999)
17. Langville, A.N., Meyer, C.D.: Deeper inside PageRank. Internet Mathematics 1, 335–380 (2004)
18. Lempel, R., Moran, S.: SALSA: The stochastic approach for link-structure analysis. ACM Transactions on Information Systems 19, 131–160 (2001)
19. Linden, G.: Marissa Mayer at Web 2.0, `http://glinden.blogspot.com/2006/11/marissa-mayer-atweb-20.html`
20. Marchiori, M.: The quest for correct information on the Web: hyper search engines. In: 6th Intl. World Wide Web Conference, pp. 265–274 (1997)
21. McSherry, F.: A uniform approach to accelerated PageRank computation. In: 14th Intl. World Wide Web Conference, pp. 575–582 (2005)
22. Najork, M.: Systems and methods for ranking documents based upon structurally interrelated information. US Patent 7,739,281 (filed 2003, issued 2010)
23. Najork, M., Zaragoza, H., Taylor, M.: HITS on the Web: how does it compare? In: 30th Annual ACM Conference on Research and Development in Information Retrieval, pp. 471–478 (2007)
24. Najork, M.: Comparing the effectiveness of HITS and SALSA. In: 16th ACM Conference on Information and Knowledge Management, pp. 157–164 (2007)
25. Najork, M., Craswell, N.: Efficient and effective link analysis with precomputed SALSA maps. In: 17th ACM Conference on Information and Knowledge Management, pp. 53–61 (2008)
26. Najork, M., Gollapudi, S., Panigrahy, R.: Less is more: sampling the neighborhood graph makes SALSA better and faster. In: 2nd ACM Intl. Conference on Web Search and Data Mining, pp. 242–251 (2009)
27. Najork, M.: The Scalable Hyperlink Store. In: 20th ACM Conference on Hypertext and Hypermedia, pp. 89–98 (2009)
28. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the Web. Technical Report, Stanford InfoLab (1999)
29. Qi, X., Nie, L., Davison, B.: Measuring similarity to detect qualified links. In: 3rd Intl. Workshop on Adversarial Information Retrieval on the Web, pp. 49–56 (2007)
30. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: 3rd Text REtrieval Conference (1994)
31. Xue, G.-R., Zeng, H.-J., Chen, Z., Yu, Y., Ma, W.-Y., Xi, W., Fan, W.: Optimizing web search using web click-through data. In: 13th ACM Intl. Conference on Information and Knowledge Management, pp. 118–126 (2004)
32. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, Ú., Gunda, P.K., Currey, J.: DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In: 8th Symposium on Operating Systems Design & Implementation, pp. 1–14 (2008)