1

Combining Link and Content Information in Web Search

Matthew Richardson and Pedro Domingos

Department of Computer Science and Engineering, University of Washington, Seattle, WA 98115, USA
email: mattr,pedrod@cs.washington.edu

Summary. As the World-Wide Web has grown, search engines have become the preferred method for finding information on the Web; it is now almost impossible to find specific information without them. This has given rise to a problem: How can we automatically determine the quality and relevance of a Web page to a particular query? The original search engines used the content of a page to determine its relevance, but recently it was found that results could be greatly improved by incorporating information gleaned from the link structure as well. In this chapter, we describe two of the most well known algorithms which do this: *HITS* [17] and *PageRank* [18], and also survey some of their improvements. We then introduce our algorithm, *Query-Dependent PageRank*, which maintains query-time efficiency while alleviating the problem of topic drift. Experiments on two large subsets of the Web indicate that our algorithm significantly outperforms PageRank in the (human-rated) quality of the pages returned, while remaining efficient enough to be used in today's large search engines. After presenting these results and a discussion of scalability, we leave the reader with some open questions and possible directions for future work.

1.1 Introduction

Traditional information retrieval measures, such as TFIDF [20], rate a document highly if the query terms occur frequently in it. These can give poor results on the Web, with its vast scale and highly variable content quality. Recently, however, it was found that search results can be greatly improved by utilizing the information contained in the link structure between pages.

The two best-known algorithms which do this are HITS [17] and PageRank [18]. The latter is used in the highly successful Google search engine [3]. The heuristic underlying both of these approaches is that pages with many inlinks are likely to be of higher quality than pages with few inlinks. This is based on the assumption that the author of a page will only include links to pages that s/he believes are of high quality. But both HITS and PageRank have

downsides. HITS is too slow at query time to be used in practice in a large-scale search engine. PageRank (and, to a lesser extent, HITS) suffers from *topic drift*, whereby it may return pages that, although they are of high quality, are only peripherally related to the query. This occurs because only the link structure between pages is considered by the computation — the content of pages is ignored.

We first discuss the information contained in the link structure of the Web. We review HITS and some of its recent improvements. We then present the PageRank algorithm and show how it is prone to topic drift. In the balance of the chapter we present our query-dependent version of PageRank, which tackles the problem of topic drift with clean probabilistic semantics. Further, by incorporating query relevance directly into PageRank, we eliminate the ad hoc posterior step of merging a page's PageRank and query relevance scores. Our results show that a query-dependent PageRank can be implemented efficiently, and returns better search results than the traditional (query independent) PageRank. We conclude with some open questions and ideas for future research.

1.2 Links Contain Information

An important characteristic that differentiates Web pages from simple documents is that they are interconnected. Initial attempts at Web search considered each page to be independent, but in fact, the hyperlinks between pages contain useful information on the topical content and quality of pages.

Topical content

A Web page typically links to other pages on similar or related topics [11]. Hence, when inferring the topic of a page, it is useful to consider not only the contents of the page itself, but also the contents of the pages it links to and those which link to it (a page's *neighbors*). One simple approach is to include the contents of the neighboring pages when computing the topic of a page. Chakrabarti *et al.* [6] found that this actually *decreased* classification accuracy, which they attributed to the fact that links are noisy — a page tends to link to pages on a similar topic, but will also typically contain a few links to pages on completely different topics ("Free Speech Online" or "Yahoo" for example). Further, this does not take advantage of the (albeit weaker) correlation between the topic of a page and its neighbors' neighbors, or their neighbors, and so on. Notice that this is a circular problem: the topic of a page depends on its neighbors, but their topics depend on it. Chakrabarti *et al.* thus use a Markov random field in which each node represents a page, the state of a node is the page's topic, and edges connect any two nodes whose pages are neighbors. The best assignment of topics to pages can be solved using Markov random field methods such as relaxation labeling. With this technique, they

were able to reduce the error rate on a Web page classification task from about 36% to about 26%.

For the remainder of the chapter, we will focus on another piece of information provided by links, that of quality.

Quality

The problem when searching the Web is that it is too vast, and queries are too underspecified. A typical query can easily match thousands or even hundreds of thousands of pages. A good search engine needs to order these results by some measure of quality and relevance to the query. Methods such as TFIDF help provide the relevance of a page to a query, but do not provide information on its quality.

In general, links are created by people. As such, they are indicative of the quality of the pages to which they point — when creating a page, an author presumably chooses to link to pages which s/he deems to be of good quality. By taking advantage of this information we can estimate the quality of each page based on the link structure around it. In the following two sections we introduce HITS and PageRank, two algorithms for doing this. Further introduction to these methods, as well as common techniques for modeling and mining textual content, can be found in [7] and [8].

1.3 HITS

HITS [17] is based on the notion that pages can be primarily categorized into two types: *hubs* (pages that point to many pages of high quality) and *authorities* (pages of high quality). Given a query, HITS first invokes a traditional search engine to obtain a set of pages relevant to it (the *root set*). This set is then expanded to include all pages which link to or are linked to by one of its pages[1]

Each page x is assigned a hub (or index) score, $h(x)$, and authority score, $a(x)$ (initially set to 1) which are updated according to the equations:

$$a(i) = \sum_{j \in \mathbf{B}_i} h(j) \qquad h(i) = \sum_{j \in \mathbf{F}_i} a(j) \tag{1.1}$$

where $\mathbf{F}_i$ is the set of pages page i links to, and $\mathbf{B}_i$ is the set of pages which link to page i. The equations are iterated until they converge. Notice that, from the equations, hubs and authorities are defined recursively; a hub is a

[1] Typically, the pages which link to a URL are found by querying a search engine. Many search engines, such as Altavista (www.altavista.com), allow queries of the form link:URL, which returns all (up to a limit) pages which link to a given URL. The pages which a page links to are typically found by retrieving the page and parsing it.

page which points to many authorities, and an authority is a page that is pointed to by many hubs.

This computation takes advantage of the fact that (typically) links are human-generated, and the choice of pages to link to reflects an author's opinion of their quality. It also uses the fact that many pages on the WWW are organized into this hub and authority structure.

1.3.1 Topic drift in HITS

The HITS computation for a given query only depends on it via the initial root set. The pages used to expand the root set may or may not be related to the query, and the computation of good hubs and authorities depends only on the link structure among those pages, not their content. As a result, HITS suffers from *topic drift*, a drift away from the query topic because some peripheral topic is more popular and/or has better connected pages.

A variety of research has been conducted on combating this problem. One common technique is to weigh each edge according to some function of the query terms and the pages it connects. This is the approach taken by IBM's Clever system [5][4], in which an edge is given a higher weight if the text near the link contains the query terms. The rationale behind this is that, for a particular topic, a hub is really only conferring authority to pages that it specifically points to in reference to that topic. With weighted edges, equation 1.1 simply becomes:

$$a(i) = \sum_{j \in \mathbf{B}_i} w(j \rightarrow i) h(j) \qquad h(i) = \sum_{j \in \mathbf{F}_i} w(i \rightarrow j) a(j) \qquad (1.2)$$

where $w(i \rightarrow j)$ is the weight assigned to the link from page i to page j.

Clever further reduces topic drift by breaking large hub pages into *pagelets*. A hub may cover multiple topics (e.g. A professor's "favorite links" page which lists pages on machine learning as well as bicycling), or varying specialties of a broad topic (e.g. A hub about international travel, with separate sections devoted to Asia and Europe), so dividing it into sections allows the hub scores for each section to be more accurate and query specific [4]. Similar work includes that of Bharat and Henzinger [1], who propose heuristic methods for weighting links based on pages' relevance to the root set. Cohn and Hofmann [9] introduce a joint probabilistic model which combines the content of pages (using probabilistic latent semantic analysis [15]) and the connectivity between them (using a probabilistic version of HITS called PHITS [10]). A joint model of content and connectivity significantly reduces the problems of topic drift, and also opens up opportunities for a variety of interesting new mining tasks.

1.3.2 Speed of HITS

HITS has an additional drawback: the hub and authority scores are computed at query time. Though the iterative computation (equation 1.1) typically takes only about one second [5], the retrieval of the relevant pages (the root set) can take up to 30 minutes [1], which would be infeasible for large-scale Web search engines.

As of this writing, Google, a popular search engine, requires approximately 0.1 seconds to answer a given query. Yet even at this speed, the company requires over 10000 computers to serve the 150 million search requests per day it receives.[2] It would be prohibitively expensive for a popular search engine to require even seconds per query. If used "within" the search engine, HITS may have access to the contents of crawled pages, making it much faster, but still an order of magnitude slower than Google. Further, to be efficient, the set of pages upon which HITS is performed must be kept small. This limits its efficient use to queries that return a small number of pages and have relatively local link structure.

Bharat *et al.* propose using a *connectivity server* [2], which crawls and serves page linkage information. Using such a server, HITS may simply retrieve the link structure of its root set, rather than having to retrieve every page, thus making it much more responsive. However, using such a server is incompatible with the techniques outlined in the previous section for preventing topic drift, which require knowing the content of each page (unless the computation is being performed "within" the search engine).

In the next section, we introduce PageRank, another method for exploiting the information contained in the links between pages. In contrast with HITS, PageRank computes a single measure of quality for each page before any query is issued. This measure is then combined with a traditional information retrieval score at query time. This has the advantage of much greater efficiency, but has the disadvantage that there is no clear and principled way to combine PageRank with a measure of query relevance,[3] leaving it open to the same problems of topic drift that were encountered by HITS.

1.4 PageRank: The Random Surfer

Imagine a Web surfer who jumps from Web page to Web page, choosing with uniform probability which link to follow at each step. In order to reduce the effect of dead-ends or endless cycles the surfer will occasionally jump to a random page with some small probability $1 - \beta$, or when on a page with no out-links. To reformulate this in graph terms, consider the Web as a directed

[2] http://www.google.com/press/highlights.html

[3] As we will see later in this chapter, the method for combining PageRank with text-based query relevance has a large effect on the results. Exactly how Google does this combination is, so far, an unpublished trade secret.

graph, where nodes represent Web pages, and edges between nodes represent links between Web pages. Let $\mathbf{W}$ be the set of nodes, N=$|\mathbf{W}|$, $\mathbf{F_i}$ be the set of pages page i links to, and $\mathbf{B_i}$ be the set pages which link to page i. For pages which have no outlinks we add a link to all pages in the graph[4]. In this way, probability mass which is lost due to pages with no outlinks is redistributed uniformly to all pages. If averaged over a sufficient number of steps, the probability the surfer is on page j at some point in time is given by the formula:

$$P(j) = \frac{(1-\beta)}{N} + \beta \sum_{i \in \mathbf{B_j}} \frac{P(i)}{|\mathbf{F_i}|} \tag{1.3}$$

The PageRank score for node j is defined as this probability: $PR(j) = P(j)$. Because equation 1.3 is recursive, it must be iteratively evaluated until $P(j)$ converges. Typically, the initial distribution for $P(j)$ is uniform. PageRank is equivalent to the primary eigenvector of the transition matrix $\mathbf{Z}$:

$$\mathbf{Z} = (1-\beta)\left[\frac{1}{N}\right]_{NxN} + \beta\mathbf{M} \tag{1.4}$$

with

$$M_{ji} = \begin{cases} \frac{1}{|\mathbf{F_i}|} & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

One iteration of equation 1.3 is equivalent to computing $x^{k+1} = \mathbf{Z}x^k$, where $x_j^k = P(j)$ at iteration k. After convergence, we have $x^{K+1} = x^K$, or $x^K = \mathbf{Z}x^K$, which means x^K is an eigenvector of $\mathbf{Z}$. Furthermore, since the columns of $\mathbf{Z}$ are normalized, x has an eigenvalue of 1.

1.4.1 Topic drift in PageRank

PageRank is query-independent, which allows it to be computed offline. At query-time, the PageRank may simply be looked up, which allows for very quick query-time response, but it also means that the PageRank of a page is completely independent of its content and the query itself. Content is brought back into the picture by combination of PageRank with some measure of query relevance. As a result, PageRank suffers from topic drift.

Figure 1.1 demonstrates this problem. A page containing the word "dog" which is linked to from a page with very high PageRank, such as Yahoo, can end up with a very high PageRank. Instead, we would expect that the best page about dogs would be one that is linked to from many other pages in the "dog community": a collection of pages about dogs that are interlinked with each other.

[4] For each page s with no outlinks, we set $\mathbf{F_s}$ = {all N nodes}, and for all other nodes augment $\mathbf{B_i}$ with s. ($\mathbf{B_i} \cup \{s\}$)

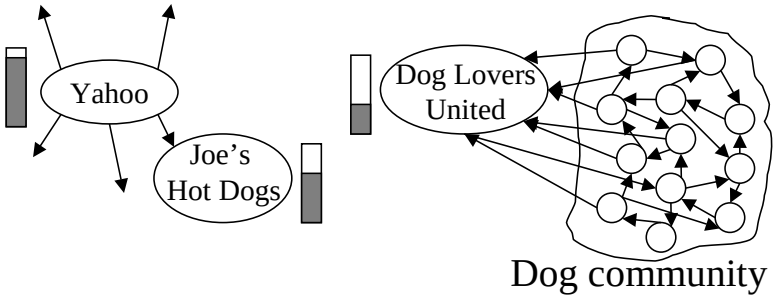

Fig. 1.1. Topic drift. The bars represent the PageRank of each page.

Some work has been done to combat topic drift in PageRank. Haveliwala has introduced the *topic-sensitive PageRank* [13], which has some similarities to our own *query-dependent PageRank*, which we describe below. In topic-sensitive PageRank, one PageRank is calculated for each of a number of topics, which are then combined at query time depending on how the query relates to the topics. When calculating a particular topic-specific PageRank, the random surfer will occasionally jump to one of a set of pages on a particular topic (taken from the Open Directory Project[5]) rather than any page. This has the effect of "grounding" the PageRank to that topic. Recent work by Jeh and Windom [16] improves the scalability of this approach. In the next section, we introduce our method for combating topic drift, *query-dependent PageRank*.

Tables 1.1 and 1.2 summarize some of the main advantages, disadvantages and characteristics of the various link-based Web search methods surveyed in this paper.

1.5 Directed Surfer Model

PageRank rates a page highly if it is at the center of a large sub-web (i.e., if many pages point to it, many other pages point to those, etc.). Intuitively, however, the best pages should be those that are at the center of a large sub-web *relevant to the query*. This section introduces our algorithm, which formalizes this intuition while, like PageRank, does most of its computations at crawl time[19].

The PageRank of a page can be viewed as the rate at which a surfer would visit that page, if it surfed the Web indefinitely, blindly jumping from page to page. We propose a more intelligent surfer that, more like a human surfer, probabilistically hops from page to page, depending on the content of the pages and the query terms the surfer is looking for. The resulting probability distribution over pages is:

[5] http://www.dmoz.org

Table 1.1. Summary of some methods for using link structure to improve Web search

Algorithm	Pros	Cons
HITS [17]	Seeded by query/content	Computation ignores content Slow at query time
Auto. Resource Compilation [5]	Weigh links by relevance of anchor text to query	Ad hoc
The Missing Link [9]	Consistent prob. semantics to incorporate content	Lack of scalability
Connectivity Server[2]	Retrieve link information quickly	Still too slow for search engine Incompatible with link weighting by content
PageRank [18]	Fast at query time	Combination with content is problematic
TS-PageRank [13]	Incorporates content in part of calculation	Coarse notion of content
QD-PageRank [19]	Sound, fine-grained incorporation of content	More computation at crawl time

Table 1.2. Summary of computations performed by link-based Web search methods.

Algorithm	Offline	Query Time
HITS	-	Crawl and computation
HITS + Connectivity Server	Crawl	Retrieve graph and compute
PageRank	Computation	Retrieve score
QD-PageRank	Computation	Retrieve score per query term and add

$$P_q(j) = (1-\beta)P'_q(j) + \beta \sum_{i \in \mathbf{B_j}} P_q(i)P_q(i \rightarrow j) \tag{1.6}$$

where $P_q(i \rightarrow j)$ is the probability that the surfer transitions to page j given that he is on page i and is searching for the query q. $P'_q(j)$ specifies where the surfer chooses to jump when not following links. $P_q(j)$ is the resulting probability distribution over pages and corresponds to the *query-dependent PageRank* score (QD-PageRank$_q(j) \equiv P_q(j)$). As with PageRank, QD-PageRank is determined by iterative evaluation of equation 1.6 from some initial distribution, and is equivalent to the primary eigenvector of the transition matrix $\mathbf{Z_q}$, where $Z_{q_{ji}} = (1-\beta)P'_q(j) + \beta P_q(i \rightarrow j)$. Although $P_q(i \rightarrow j)$ and $P'_q(j)$ are arbitrary distributions, we will focus on the case where both probability distributions are derived from $R_q(j)$, a measure of relevance of page j to query q:

$$P'_q(j) = \frac{R_q(j)}{\sum\limits_{k \in \mathbf{W}} R_q(k)} \tag{1.7}$$

$$P_q(i \to j) = \frac{R_q(j)}{\sum\limits_{k \in \mathbf{F_i}} R_q(k)} \tag{1.8}$$

In other words, when choosing among multiple out-links from a page, the directed surfer tends to follow those which lead to pages whose content has been deemed relevant to the query (according to R_q). Similarly to PageRank, when a page has no outlinks (or the outlinks all have zero relevance), we implicitly add links from that page to all other pages in the network. On such a page, the surfer thus chooses a new page to jump to according to the distribution $P'_q(j)$.

When given a multiple-term query, Q=$\{q_1, q_2, \ldots\}$, the surfer selects a q according to some probability distribution, $P(q)$ and uses that term to guide its behavior (according to equation 1.6) for a large number of steps[6]. It then selects another term according to the distribution to determine its behavior, and so on. The resulting distribution over visited Web pages is QD-PageRank$_Q$ and is given by

$$\text{QD-PageRank}_\text{Q}(j) \equiv P_Q(j) = \sum_{q \in Q} P(q) P_q(j) \tag{1.9}$$

For standard PageRank, the PageRank vector is equivalent to the primary eigenvector of the matrix $\mathbf{Z}$. The vector of single-term QD-PageRank$_q$ is again equivalent to the primary eigenvector of the matrix $\mathbf{Z}_q$. An interesting question that arises is whether the QD-PageRank$_Q$ vector is equivalent to the primary eigenvector of a matrix (corresponding to the combination performed by equation 1.9). In fact, this is not the case. Instead, the primary eigenvector of $\mathbf{Z}_Q$ corresponds to the QD-PageRank obtained by a random surfer who, *at each step*, selects a new query according to the distribution $P(q)$. However, QD-PageRank$_Q$ is approximately equal to the PageRank that results from this single-step surfer, for the following reason.

Let $\mathbf{x}_q$ be the L2-normalized primary eigenvector for matrix $\mathbf{Z}_q$ (note that element j of $\mathbf{x}_q$ is QD-PageRank$_q(j)$). Since $\mathbf{x}_q$ is the primary eigenvector for $\mathbf{Z}_q$, we have [12]: $\forall q, r \in Q : \|\mathbf{Z}_q \mathbf{x}_q\| \geq \|\mathbf{Z}_q \mathbf{x}_r\|$. Thus, to a first degree of approximation, $\mathbf{Z}_q \sum\limits_{r \in Q} \mathbf{x}_r \approx \kappa \mathbf{Z}_q \mathbf{x}_q$. Suppose $P(q) = 1/|Q|$. Consider $\mathbf{x}_Q = \sum\limits_{q \in Q} P(q) \mathbf{x}_q$ (see equation 1.9). Then

$$\mathbf{Z}_Q \mathbf{x}_Q = \left(\sum_{q \in Q} \frac{1}{|Q|} \mathbf{Z}_q \right) \left(\sum_{q \in Q} \mathbf{x}_q \right) = \frac{1}{|Q|} \sum_{q \in Q} \left(\mathbf{Z}_q \sum_{r \in Q} \mathbf{x}_r \right) \tag{1.10}$$

[6] However many steps are needed to reach convergence of equation 1.6

$$\approx \frac{1}{|Q|}\sum_{q\in Q}(\kappa \mathbf{Z}_q \mathbf{x}_q) = \frac{\kappa}{|Q|}\sum_{q\in Q}\mathbf{x}_q = \frac{\kappa}{n}\mathbf{x}_Q$$

and thus $\mathbf{x}_Q$ is approximately an eigenvector for $\mathbf{Z}_Q$. Since $\mathbf{x}_Q$ is equivalent to QD-PageRank$_Q$, and $\mathbf{Z}_Q$ describes the behavior of the single-step surfer, QD-PageRank$_Q$ is approximately the same PageRank that would be obtained by using the single-step surfer. The approximation has the least error when the individual random surfers defined by $\mathbf{Z}_q$ are very similar, or are very dissimilar.

The choice of relevance function $R_q(j)$ is arbitrary. In the simplest case, $R_q(j) = R$ is independent of the query term and the document, and QD-PageRank reduces to Page-Rank. One simple content-dependent function could be $R_q(j) = 1$ if the term q appears on page j, and 0 otherwise. Much more complex functions could be used, such as the well-known TFIDF information retrieval metric, a score obtained by latent semantic indexing, or any heuristic measure using features such as text size and positioning. It is important to note that most current text ranking functions could be easily incorporated into the directed surfer model.

1.6 Scalability

The difficulty with calculating a query-dependent PageRank is that a search engine cannot perform the computation, which can take hours, at query time, when it is expected to return results in seconds (or less). We surmount this problem by precomputing the individual term rankings QD-PageRank$_q$, and combining them at query time according to equation 1.9. We show that the computation and storage requirements for QD-PageRank$_q$ for hundreds of thousands of words is only approximately 100-200 times that of a single query independent PageRank.

Let $\mathbf{L} = \{q_1, q_2, \ldots, q_m\}$ be the set of words in our lexicon. That is, we assume all search queries contain terms in $\mathbf{L}$, or we are willing to use plain PageRank for those terms not in $\mathbf{L}$. Let d_q be the number of documents which contain the term q. Then $S = \sum_{q\in\mathbf{L}} d_q$ is the number of unique document-term pairs.

1.6.1 Disk storage

For each term q, we must store the results of the computation. We add the minor restriction that a search query will only return documents containing all of the terms[7]. Thus, when merging QD-PageRank$_q$'s, we need only to know the QD-PageRank$_q$ for documents that contain the term. Each QD-PageRank$_q$ is

[7] Google has this "feature" as well. See http://www.google.com/technology/why-use.html.

a vector of d_q values. Thus, the space required to store all of the PageRanks is S, a factor of S/N times the query independent PageRank alone (recall N is the number of Web pages). Further, note that the storage space is still considerably less than that required for the search engine's reverse index, which must store information about all document-term pairs, as opposed to our need to store information about every *unique* document-term pair.[8]

1.6.2 Time requirements

If $R_q(j) = 0$ for some document j, the directed surfer will never arrive at that page. In this case, we know QD-PageRank$_q(j) = 0$, and thus when calculating QD-PageRank$_q$, we need only consider the subset of nodes for which $R_q(j) > 0$. We add the reasonable constraint that $R_q(j) = 0$ if term q does not appear in document j, which is common for many information retrieval relevance metrics, especially those used by commercial search engines today. The computation for term q then only needs to consider d_q documents. Because it is proportional to the number of documents in the graph, the computation of QD-PageRank$_q$ for all q in W will require O(S) time, a factor of S/N times the computation of the query independent PageRank alone. Furthermore, we have noticed in our experiments that the computation converges in fewer iterations on these smaller sub-graphs, empirically reducing the computational requirements to $0.75 * S/N$. Additional speedup may be derived from the fact that for most words, the sub-graph will completely fit in memory, unlike PageRank which (for any large corpus) must repeatedly read the graph structure from disk during computation.

1.6.3 Empirical scalability

The fraction S/N is critical to determining the scalability of QD-PageRank. If every document contained vastly different words, S/N would be proportional to the number of search terms, m. However, this is not the case. Instead, there are a very few words that are found in almost every document, and many words which are found in very few documents[9]; in both cases the contribution to S is small.

In our database of 1.7 million pages (see section 1.7), we let **L** be the set of all unique words (the lexicon), and removed the 100 most common words[10].

[8] Storing every document-term pair is necessary for certain functionality like term proximity calculation. Most major search engines provide this functionality. If one were to choose not to, it would only need to store every unique document-term pair in its reverse index.

[9] This is because the distribution of words in text tends to follow an inverse power law [21]. We also verified experimentally that the same holds true for the distribution of the number of documents a word is found in.

[10] It is common to remove "stop" words such as *the*, *is*, etc., as they do not affect the search.

This results in $|\mathbf{L}|$=2.3 million words, and the ratio S/N was found to be 165. S/N is essentially the average number of unique words per page, which should remain roughly constant once it has been measured over a large sample of pages. Thus, we expect that S/N will be approximately 165 even for much larger sets of Web pages. This means QD-PageRank requires approximately 165 times the storage space and 124 times the computation time to allow for arbitrary queries over any of the 2.3 million words (which is still less storage space than is required by the search engine's reverse index alone).

1.7 Results

We give results on two data sets: *educrawl*, and *WebBase*. *Educrawl* is a crawl of the Web, restricted to .edu domains. The crawler was seeded with the first 18 results of a search for "University" on Google (www.google.com). Links containing "?" or "cgi-bin" were ignored, and links were only followed if they ended with ".html". The crawl contains 1.76 million pages over 32,000 different domains. *WebBase* is the first 15 million pages of the Stanford WebBase repository [14], which contains over 120 million pages. For both datasets, HTML tags were removed before processing.

We calculated QD-PageRank as described above, using $R_q(j)$ = the fraction of words equal to q in page j, and $P(q) = 1/|Q|$. We compare our algorithm to our implementation of the standard PageRank algorithm. For content ranking, we used the same $R_q(j)$ function as for QD-PageRank, but, similarly to TFIDF, weighted the contribution of each search term by the log of its inverse document frequency. As there is nothing published about merging PageRank and content rank into one list, the approach we followed was to normalize the two scores and add them. This implicitly assumed that PageRank and content rank were equally important. This resulted in poor PageRank performance, which we found was because the distribution of PageRanks was much more skewed than the distribution of content ranks; normalizing the vectors resulted in PageRank primarily determining the final ranking. To correct this problem, we scaled each vector to have the same average value in its top ten terms before adding the two vectors. This dramatically improved PageRank.

For *educrawl*, we requested a single word and two double word search queries from each of three volunteers, resulting in a total of nine queries. For each query, we randomly mixed the top 10 results from standard PageRank with the top 10 results from QD-PageRank, and gave them to four volunteers, who were asked to rate each search result as a 0 (not relevant), 1 (somewhat relevant, not very good), or 2 (good search result) based on the contents of the page it pointed to. In Table 1.3, we present the final rating for each method, per query. This rating was obtained by first summing the ratings for the ten pages from each method for each volunteer, and then averaging the individual ratings. A similar experiment for *WebBase* is given in Table 1.4.

Table 1.3. Results on *educrawl*

Query	QD-PR	PR
chinese association	10.75	6.50
computer labs	9.50	13.25
financial aid	8.00	12.38
intramural	16.50	10.25
maternity	12.50	6.75
president office	5.00	11.38
sororities	13.75	7.38
student housing	14.13	10.75
visitor visa	19.25	12.50
Average	**12.15**	**10.13**

Table 1.4. Results on *WebBase*

Query	QD-PR	PR
alcoholism	11.50	11.88
architecture	8.45	2.93
bicycling	8.45	6.88
rock climbing	8.43	5.75
Shakespeare	11.53	5.03
stamp collecting	9.13	10.68
vintage car	13.15	8.68
Thailand tourism	16.90	9.75
Zen Buddhism	8.63	10.38
Average	**10.68**	**7.99**

For *WebBase*, we randomly selected the queries from Bharat and Henzinger [1]. The four volunteers for the *WebBase* evaluation were independent from the four for the *educrawl* evaluation, and none knew how the pages they were asked to rate were obtained.

QD-PageRank performs better than PageRank, accomplishing a relative improvement in relevance of 20% on educrawl and 34% on WebBase. The results are statistically significant ($p < .03$ for educrawl and $p < .001$ for WebBase using a two-tailed paired t-test, one sample per person per query). Averaging over queries, every volunteer found QD-PageRank to be an improvement over PageRank, though not all differences were statistically significant.

One item to note is that the results on multiple word queries are not as positive as the results on single word queries. As discussed in section 1.5, the combination of single word QD-PageRanks to calculate the QD-PageRank for a multiple word query is only an approximation, made for practical reasons. This approximation is worse when the words are highly dependent. Further, some queries, such as "financial aid" have a different intended meaning as

a phrase than simply the two words "financial" and "aid". For queries such as these, the words are highly dependent. We could partially overcome this difficulty by adding the most common phrases to the lexicon, thus treating them the same as single words.

1.8 Open Questions and Future Work

There are several interesting directions to pursue in this area. Initially, search engines used only the contents of a page to measure its quality and relevance to a query. Newer search engines, such as Google, take advantage of the implicit information held in the links between pages. What other sources of information might prove to be useful for future search engines? One remaining source of information is the URL itself. The name of the server, the filename, and the directory structure of the URL are all potentially informative. The IP address of the server may help identify its quality, based on how close it is to a major Internet backbone, who is providing the server, etc. Information about the user him/herself, if accessible, may also be useful (e.g., the user's bookmark/favorite list contains information about his/her interests, and may be useful in disambiguating queries).

One of the drawbacks of our approach is that, in order retain computational efficiency mentioned, links to pages that do not contain the query term must have zero weight. This may divide the relevant pages into many small, isolated sub-graphs. Is there a way that we can allow the surfer to transition through these pages, yet still perform the computation efficiently? What if we always use the same, small constant for the probability of transitioning to a page that does not contain the query term? What might the benefits be in combining QD-PageRank with traditional PageRank?

Another approach is to generate QD-PageRanks for clusters of words, instead of for each individual one. For example, we may compute a "pet" QD-PageRank, which prefers pages containing any pet-related term (dog, cat, fish, etc.). To order results for a query about dogs, we would use this pet-specific PageRank. Using this technique, rare words will not face the problem mentioned in the previous paragraph where pages with the query term are isolated from each other. If we were to use approximately 100 clusters, the computation cost of the calculation will be similar to that of QD-PageRank. Which should we expect to give better results — QD-PageRank for individual terms or for clusters of terms?

Define $\mathbf{W}(x)$ to be the set of Web pages which contain the term x. If $\mathbf{W}(q) \subseteq \mathbf{W}(r)$ then does knowing QD-PageRank$_r$ assist in the calculation of QD-PageRank$_q$? What if $\mathbf{W}(q) \subseteq \{\mathbf{W}(r_1) \cup \mathbf{W}(r_2)\}$? What if all we know is that $\frac{\mathbf{W}(q) \cap \mathbf{W}(r)}{\mathbf{W}(q) \cup \mathbf{W}(r)}$ is close to unity (i.e. there is significant overlap between $\mathbf{W}$(q) and $\mathbf{W}$(r))?

The QD-PageRank$_Q$ for a multiple word query is a combination of the QD-PageRank$_q$'s over $q \in Q$. As mentioned, this is only an approximation. Is

there a better approximation for QD-PageRank$_Q$ which may still be computed quickly at query-time? What if, offline, we calculate (and store) cluster specific QD-PageRanks: QD-PageRank$_q^c$ where q is a query term, and c is one of a small number (100) of query term clusters which represent the other "missing" query terms. At query-time, a cluster (or number of clusters) could be chosen, depending on the query, and that subset of QD-PageRank$_q$'s used in the combination. This may result in a better approximation for QD-PageRank$_Q$, if a suitable definition of QD-PageRank$_q^c$ is found. Jeh and Windom's work [16] may also be applicable here.

1.9 Conclusions

In this chapter, we explored the idea that links carry useful information for Web search. We introduced the two most common algorithms for extracting this information (HITS and PageRank), and discussed some of the drawbacks of each. In particular, HITS is not efficient enough at query time, and PageRank (and to a lesser extent HITS) suffers from topic drift. We then introduced a model that overcomes these problems by probabilistically combining page content and link structure in the form of an intelligent random surfer. The model accommodates most query relevance functions in use today, and produces higher-quality results than PageRank, while having time and storage requirements that are within reason for today's large-scale search engines.

1.10 Acknowledgements

We would like to thank Gary Wesley and Taher Haveliwala for their help with WebBase, Frank McSherry for eigen-help, and our experiment volunteers for their time. This work was partially supported by NSF CAREER and IBM Faculty awards to the second author.

References

1. K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, 1998. ACM Press.
2. Krishna Bharat, Andrei Z. Broder, Monika Rauch Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. *WWW7 / Computer Networks*, 30(1-7):469–477, 1998.
3. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, 1998. Elsevier.

4. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the web's link structure. *IEEE Computer*, August 1999.
5. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference*, pages 65–74, Brisbane, Australia, 1998. Elsevier.
6. S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 307–318, Seattle, WA, 1998. ACM Press.
7. Soumen Chakrabarti. Data mining for hypertext: A tutorial survey. *SIGKDD Explorations*, 1(2):1–11, 2000.
8. Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.
9. D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2001.
10. David Cohn and Huan Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th International Conf. on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000.
11. Brian D. Davison. Topical locality in the web. In *Twenty-third Annual International Conference on Research and Development in Information Retrieval*, pages 272–279, Athens, Greece, 2000.
12. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
13. Taher Haveliwala. Topic-sensitive PageRank. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, HI, USA, May 2002.
14. Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, and Andreas Paepcke. WebBase: A repository of web pages. In *Proceedings of the 9th International World Wide Web Conference (WWW9)*, 2000.
15. Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.
16. Glen Jeh and Jennifer Windom. Scaling personalized web search. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
17. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
18. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, 1998.
19. M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1441–1448. MIT Press, Cambridge, MA, 2002.
20. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.
21. George Kingsley Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.