4 Page Selection

4.1 Introduction

The design of a good Web Information Retrieval System (WIRS) presents many challenges. In particular, the WIRS must deal with huge volumes of data. Unless it has unlimited computing resources and unlimited time, it must carefully decide what URLs to download and in what order. In this chapter we address this important challenge: How should a WIRS select URLs to download from its list of known URLs? If a WIRS intends to perform a single scan of the entire Web, and the load placed on target sites is not an issue, then any URL order will suffice. That is, eventually every single known URL will be visited, so the order is not critical. However, most WIRS will not be able to visit every possible page for two main reasons:

- The WIRS or its client may have limited storage capacity, and may be unable to index or analyze all pages. Currently the Web is believed to have several terabytes of textual data and is growing rapidly, so it is reasonable to expect that most clients will not want or will not be able to cope with all that data [Kah97].

- Crawling for WIRS takes time, so at some point the WIRS may need to start revisiting previously retrieved pages, to check for changes. This means that it may never get to some pages. It is currently estimated that there exist more than one billion pages available on the Web [BYBCW00, LG99, LG98] and many of these pages change at rapid rates [PP97, WM99, DFK99, CGM01].

In either case, it is important for the WIRS to visit “important” pages first, so that the fraction of the Web that is visited (and kept up to date) is more meaningful. In the following sections, we present several different
useful definitions of importance, and develop crawling processes of the
Information Retrieval priorities so that important pages have a higher
probability of being visited first. We also present experimental results from
crawling the Savitribai Phule Pune University Web pages that show how
effective the different crawling strategies are. (Please Note: elsewhere in this
dissertations Savitribai Phule Pune University is also mentioned by the name
of Pune University or University of Pune for the sake of convenience).

4.2 Importance metrics

Not all pages are necessarily of equal interest to a WIRS client. For
instance, if the client is building a specialized database on a particular topic,
then pages that refer to that topic are more important, and should be visited as
early as possible. Similarly, a search engine may use the number of Web
URLs that point to a page, the so-called backlink count, to rank user query
results. If the crawler cannot visit all pages, then it is better to visit those with
a high backlink count, since this will give the end-user higher ranking results.

Given a Web page \( p \), we can define the importance of the page, \( I(p) \), in
one of the following ways. (These metrics can be combined, as will be
discussed later.)

1. **Similarity to a Driving Query Q**: A query \( Q \) drives the crawling
process, and \( I(p) \) is defined to be the textual similarity between \( p \) and \( Q \).
Similarity has been well studied in the Information Retrieval (IR)
community [Sal83] and has been applied to the Web environment
[YLYL95]. We use IS\((p)\) to refer to the importance metric in this case.
We also use IS (\( p, Q \)) when we wish to make the query explicit. To
compute similarities, we can view each document (\( p \) or \( Q \)) as an \( n \)-
dimensional vector \( w_1, \ldots, w_n \). The term \( w_i \) in this vector represents the
ith word in the vocabulary. If \( w_i \) does not appear in the document, then \( w_i \)
is zero. If it does appear, \( w_i \) is set to represent the significance of the
word. One common way to compute the significance \( w_i \) is to multiply the
number of times the ith word appears in the document by the inverse
document frequency (idf) of the ith word. The idf factor is one divided by
the number of times the word appears in the entire “collection,” which in this case would be the entire Web. The idf factor corresponds to the content discriminating power of a word: A term that appears rarely in documents (e.g., “queue”) has a high idf, while a term that occurs in many documents (e.g., “the”) has a low idf. (The wi terms can also take into account where in a page the word appears. For instance, words appearing in the title of an HTML page may be given a higher weight than other words in the body.) The similarity between p and Q can then be defined as the inner product of the p and Q vectors. Another option is to use the cosine similarity measure, which is the inner product of the normalized vectors. Note that if we do not use idf terms in our similarity computation, the importance of a page, IS(p), can be computed with “local” information, i.e., just p and Q. However, if we use idf terms, then we need global information. During the crawling process we have not seen the entire collection, so we have to estimate the idf factors from the pages that have been crawled, or from some reference idf terms computed at some other time. We use IS (p) to refer to the estimated importance of page p, which is different from the actual importance IS (p) that is computable only after the entire Web has been crawled. If idf factors are not used, then IS (p) = IS (p) 2. Backlink Count: The value of IB(p) is the number of links to p that appear over the entire Web. We use IB(p) to refer to this importance metric. Intuitively, a page p that is linked to by many pages is more important than one that is seldom referenced. This type of “citation count” has been used extensively to evaluate the impact of published papers. On the Web, IB(p) is useful for ranking query results, giving end-users pages that are more likely to be of general interest. Note that evaluating IB(p) requires counting backlinks over the entire Web. A crawler may estimate this value with IB (p), the number of links to p that have been seen so far.

3. **PageRank**: The IB(p) metric treats all links equally. Thus, a link from the Yahoo home page counts the same as a link from some individual’s home page. However, since the Yahoo home page is more important (it has a much higher IB count), it would make sense to value that link more highly. The PageRank backlink metric, IR(p), recursively defines the importance of a page to be the weighted sum of the importance of the pages that have backlinks to p. Such a metric has been found to be very useful in ranking results of user queries [PB98, Goo].

66
We use IR (p) for the estimated value of IR(p) when we have only a subset of pages available. More formally, if a page has no outgoing link, we assume that it has outgoing links to every single Web page. Next, consider a page p that is pointed at by pages t₁, . . . , tn. Let cᵢ be the number of links going out of page tᵢ. Also, let d be a damping factor (whose intuition is given below). Then, the weighted backlink count of page p is given by:

\[ IR(p) = (1 - d) + d \left[ \frac{IR(t_1)}{c_1} + \cdot \cdot \cdot + \frac{IR(t_n)}{c_n} \right] \]

This leads to one equation per Web page, with an equal number of unknowns. The equations can be solved for the IR values. They can be solved iteratively, starting with all IR values equal to 1. At each step, the new IR(p) value is computed from the old IR(tᵢ) values (using the equation above), until the values converge. This calculation corresponds to computing the principal eigenvector of the link matrices.

One intuitive model for PageRank is that we can think of a user “surfing” the Web, starting from any page, and randomly selecting from that page a link to follow. When the user reaches a page with no outlinks, he jumps to a random page. Also, when the user is on a page, there is some probability d, that the next visited page will be completely random. This damping factor d makes sense because users will only continue clicking on one task for a finite amount of time before they go on to something unrelated. The IR(p) values we computed above give us the probability that our random surfer is at p at any given time.

4. **Forward Link Count:** For completeness we may want to consider a metric IF (p) that counts the number of links that emanate from p. Under this metric, a page with many outgoing links is very valuable, since it may be a Web directory. This metric can be computed directly from p, so IF (p) = IF (p). This kind of metric has been used in conjunction with other factors to reasonably identify index pages [PPR96]. We could also define a weighted forward link metric, analogous to IR(p), but we do not consider it here.
5. **Location Metric:** The IL(p) importance of page p is a function of its location, not of its contents. If URL u leads to p, then IL(p) is a function of u. For example, URLs ending with “.com” may be deemed more useful than URLs with other endings, or URLs containing the string “home” may be more of interest than other URLs. Another location metric that is sometimes used considers URLs with fewer slashes more useful than those with more slashes. All these examples are local metrics since they can be evaluated simply by looking at the URL u. As stated earlier, our importance metrics can be combined in various ways. For example, we may define a metric IC(p) = k_1 \cdot IS(p, Q) + k_2 \cdot IB(p), for some constants k_1, k_2. This combines the similarity metric (under some given query Q) and the backlink metric. Pages that have relevant content and many backlinks would be the highest ranked. (Note that a similar approach was used to improve the effectiveness of a search engine [Mar97].)

4.3 **Problem definition**

Our goal is to design a WIRS that if possible visits high I(p) pages before lower ranked ones, for some definition of I(p). Of course, the crawler will only have available I(p) values, so based on these it will have to guess what are the high I(p) pages to fetch next. Our general goal can be stated more precisely in three ways, depending on how we expect the crawler to operate. (In our evaluations of Section 4.5 we use the second model in most cases, but we do compare it against the first model in one experiment. Nevertheless, we believe it is useful to discuss all three models to understand the options.)

**Crawl & Stop:** Under this model, the crawler C of WIRS starts at its initial page p_0 and stops after visiting K pages. At this point an “ideal” crawler would have visited pages r_1, . . . , r_K, where r_i is the page with the highest importance value, r_2 is the next highest, and so on. We call pages r_1 through r_K the “hot” pages. The K pages visited by our
real crawler will contain only $M$ pages with rank higher than or equal to $I(rK)$. We define the performance of the crawler $C$ of WIRS to be $PCS(C) = M/K$. The performance of the ideal WIRS is of course 1. A WIRS that somehow manages to visit pages entirely at random, and may revisit pages, would have a performance of $K/T$, where $T$ is the total number of pages in the Web. (Each page visited is a hot page with probability $K/T$. Thus, the expected number of desired pages when the crawler of the WIRS stops is $K^2/T$.)

**Crawl & Stop with Threshold:** We again assume that the crawler of the WIRS visits $K$ pages. However, we are now given an importance target $G$, and any page with $I(p) \geq G$ is considered hot. Let us assume that the total number of hot pages is $H$. The performance of the crawler of the WIRS, $PST(C)$, is the fraction of the $H$ hot pages that have been visited when the crawler stops. If $K < H$, then an ideal WIRS will have performance $K/H$. If $K \geq H$, then the ideal WIRS has the perfect performance 1. A purely random WIRS that revisits pages is expected to visit $(H/T) \cdot K$ hot pages when it stops. Thus, its performance is $K/T$. Only when the random crawler of the WIRS visits all $T$ pages is its performance expected to be 1.

**Limited Buffer WIRS:** In this model we consider the impact of limited storage on the WIRS process. We assume that the WIRS can only keep $B$ pages in its buffer. Thus, after the buffer fills up, the WIRS must decide what pages to flush to make room for new pages. An ideal WIRS could simply drop the pages with lowest $I(p)$ value, but a real WIRS must guess which of the pages in its buffer will eventually have low $I(p)$ values. We allow the WIRS to visit a total of $T$ pages, equal to the total number of Web pages. At the end of this process, the
fraction of the buffer pages that are hot gives us the performance PBC (C). We can define hot pages to be those with \( I(p) \geq G \), where \( G \) is a target importance, or those with \( I(p) \geq I(r_B) \), where \( r_B \) is the page with the \( B^{th} \) highest importance value. The performance of an ideal and a random WIRS are analogous to those in the previous cases.

Note that to evaluate a WIRS under any of these metrics, we need to compute the actual \( I(p) \) values of pages, and this involves crawling the “entire” Web. To keep our experiments (Section 4.5) manageable, we imagine that the Pune University pages form the entire Web, and we only evaluate performance in this context. That is, we assume that all pages outside of Pune University have \( I(p) = 0 \), and that links to pages outside of Pune University or links from pages outside of Pune University do not count in \( I(p) \) computations. In Section 4.5.2 we study the implications of this assumption by also analyzing a smaller Web within the Stanford domain, and seeing how Web size impacts performance.

### 4.4 Ordering metrics

A WIRS (Web Information Retrieval System) keeps a queue of URLs it has seen during a crawl, and must select from this queue the next URL to visit. The ordering metric \( O \) is used by the WIRS for this selection, i.e., it selects the URL \( u \) such that \( O(u) \) has the highest value among all URLs in the queue. The \( O \) metric can only use information seen (and remembered if space is limited) by the WIRS.

The \( O \) metric should be designed with an importance metric in mind. For instance, if we are searching for high \( IB(p) \) pages, it makes sense to use \( O(u) = IB(p) \), where \( p \) is the page \( u \) points to. However, it might also make sense to use \( O(u) = IR(p) \), even if our importance metric is not weighted. In our experiments, we explore the types of ordering metrics that are best suited for
either IB(p) or IR(p). For a location importance metric IL(p), we can use that metric directly for ordering since the URL of p directly gives the IL(p) value. However, for forward link IF (p) and similarity IS (p) metrics, it is much harder to devise an ordering metric since we have not seen p yet. As we will see, for similarity, we may be able to use the text that anchors the URL u as a predictor of the text that p might contain. Thus, one possible ordering metric O(u) is IS(A, Q), where A is the anchor text of the URL u, and Q is the driving query.

4.5 Experiments

To avoid network congestion and loads on the server, we did our experimental evaluation in two steps. In the first step, we physically crawled all Pune University Web pages and built a local repository of the pages. This was done with the SEOChat tool mentioned earlier. After we built the repository, we ran our virtual WIRS on it to evaluate different crawling schemes. Note that even though we had the complete image of the Pune University domain in the repository, our virtual crawler based its crawling decisions only on the pages it saw for itself. In this section we briefly discuss how the particular database was obtained for our experiments.

4.5.1 Description of dataset

To download an image of the Pune University Web pages, we started SEOChat with an initial list of “unipune.ac.in” URLs. These URLs were obtained from an earlier crawl. During the crawl, non-Pune University URLs were ignored. Also, we limited the actual data that we collected for two reasons. The first is that many heuristics are needed to avoid automatically generated, and potentially infinite, sets of pages. For example, any URLs containing “/cgi-bin/*” are not crawled, because they are likely to contain programs which generate infinite sets of pages, or produce other undesirable
side effects such as an unintended box for choosing options. We used similar heuristics to avoid downloading pages generated by programs. Another way the data set is reduced is through the robots exclusion protocol [Rob], which allows Webmasters to define pages they do not want crawled by automatic systems. To keep our experiment simple, at the end of the process, we kept the downloaded web pages limited to 1000 pages and had 62 known URLs to Pune University pages. The crawl was stopped before it was complete, but most of the uncrawled URLs were on only a few servers, so we believe the dataset we used to be a reasonable representation of the unipune.ac.in Web. In particular, it should be noted that 62 of the known URLs were on one server, http://www.unipune.ac.in/, which has a program that could generate an unlimited number of Web pages. Since the dynamically-generated pages on the server had links to other dynamically generated pages, we would have downloaded an infinite number of pages if we naively followed the links. We confined our dataset consisting of the total number of pages for our experiments to 1,000 web pages.

We should stress that the virtual crawlers of our WIRS that will be discussed next do not use SEOChat directly but use the dataset collected by the SEOChat, and do their own crawling on it. The virtual crawlers are simpler than the SEOChat tool. For instance, they can detect if a URL is invalid simply by seeing if it is in the dataset. Similarly, they do not need to distribute the load to visited sites. These simplifications are fine, since the virtual crawlers are only used to evaluate ordering schemes, and not to do real crawling.

### 4.5.2 Backlink-based WIRS

In this section we study the effectiveness of various ordering metrics, for the scenario where importance is measured through backlinks (i.e., either the IB(p) or IR(p) metrics). We start by describing the structure of the virtual
crawler of WIRS, and then consider the different ordering metrics. Unless otherwise noted we use the Pune University dataset described in Section 4.5.1, and all crawls are started from the Pune University homepage. For the PageRank metric we use a damping factor $d$ of 0.9 (for both IR(p) and IR (p)) for all of our experiments. Figure 4.1 shows our basic virtual crawler of WIRS. The crawler manages three main data structures. Queue url queue contains the URLs that have been seen and need to be visited. Once a page is visited, it is stored (with its URL) in crawled pages. links holds pairs of the form $(u_1, u_2)$, where URL $u_2$ was seen in the visited page with URL $u_1$. The crawler’s ordering metric is implemented by the function reorder queue(), shown in Figure 4.2. We used three ordering metrics: (1) breadth-first (2) backlink count IB (p), and (3) PageRank IR (p). The breadth-first metric places URLs in the queue in the order in which they are discovered, and this policy makes the crawler visit pages in breadth-first order.

**Input:** starting_url: seed URL

**Procedure:**

```
[1] enqueue(url_queue, starting_url)
[2] while (not empty(url_queue))
[3]    url = dequeue(url_queue)
[4]    page = crawl_page(url)
[6]    url_list = extract_urls(page)
[7]    foreach u in url_list
[8]        enqueue(links, (url, u))
[9]        if (u∉url_queue and (u,-)∉crawled_pages)
[10]           enqueue(url_queue, u)
```

**Function description:**

- enqueue(queue, element): append element at the end of queue
- dequeue(queue): remove the element at the beginning of queue and return it
- reorder_queue(queue): reorder queue using information in links

Figure 4.1 Basic crawling algorithm
(1) breadth first
do nothing (null operation)

(2) backlink count, \( IB'(p) \)
foreach \( u \) in url_queue
   backlink_count\( [u] \) = number of terms \( (-,u) \) in links
sort url_queue by backlink_count\( [u] \)

(3) PageRank \( IR'(p) \)
solve the following set of equations:
   \[
   IR[u] = (1 - 0.9) + 0.9 \sum_i \frac{IR[v_i]}{c_i},
   \]
   where \( (v_i, u) \in \text{links} \) and \( c_i \) is the number of links in the page \( v_i \)
sort url_queue by \( IR(u) \)

Figure 4.2 Description of reorder queue() of each ordering metric

Figure 4.3 Fraction of Pune University Web crawled vs. PST \( I(p) = IB(p) \); \( O(u) = IB(p) \).

We start by showing in Figure 4.3 the WIRS performance with the backlink ordering metric. In this scenario, the importance metric is the number of backlinks to a page \( (I(p) = IB(p)) \) and we consider a Crawl & Stop with Threshold model in Section 4.3 with \( G \) either 3, 10, or 100. Recall that a page with \( G \) or more backlinks is considered important, i.e., hot. In Figure 4.3, the horizontal axis is the fraction of the Pune University Web pages that has been crawled over time. At the right end of the horizontal axis, all 1,000 pages have been visited. The vertical axis represents \( P_{ST} \), the fraction of the total hot pages that has been crawled at a given point. The solid lines in the figure show the results from our experiments. For example, when the WIRS in our experiment visited 0.2 (20%) of the Pune University pages, it crawled 0.5
(50%) of the total hot pages for \( G = 100 \). The dashed lines in the graph show the expected performance of ideal WIRS. An ideal WIRS reaches performance 1 when \( H \) pages have been crawled. The dotted line represents the performance of a random crawler, and it increases linearly over time.

The graph shows that as our definition of a hot page becomes more stringent (larger \( G \)), the faster the WIRS can locate the hot pages. This result is to be expected, since pages with many backlinks will be seen quickly after the crawl starts. Figure 4.3 also shows that even if \( G \) is large, finding the “last” group of hot pages is always difficult. That is, to the right of the 0.8 point on the horizontal axis, the WIRS finds hot pages at roughly the same rate as a random crawler.

![Graph showing performance of ideal WIRS and random crawler](image)

Figure 4.4 Fraction of Pune University Web crawled vs. PST. \( I(p) = IB(p); G = 100 \).

In our next experiment we compare three different ordering metrics: 1) breadth-first 2) backlink-count and 3) PageRank (corresponding to the three functions of Figure 4.2). We continue to use the Crawl & Stop with Threshold model, with \( G = 100 \), and a \( IB(p) \) importance metric. Figure 4.4 shows the results of this experiment. The results are rather counterintuitive. Intuitively one would expect that a crawler using the backlink ordering metric \( IB(p) \) that matches the importance metric \( IB(p) \) would perform the best. However, this is not the case, and the PageRank metric \( IR(p) \) outperforms the \( IB(p) \) one. To understand why, we manually traced the WIRS operation. We noticed that
often the IB (p) crawler of the WIRS behaved like a depth-first one, frequently visiting pages in one “cluster” before moving on to the next. On the other hand, the IR (p) crawler of the WIRS combined breadth and depth in a better way.

To illustrate, let us consider the Web fragment of Figure 4.5. With IB (p) ordering, the crawler of WIRS visits a page like the one labeled p₁ and quickly finds a cluster A of pages that point to each other. The A pages temporarily have more backlinks than page p₂, so the visit of page p₂ is delayed even if page p₂ actually has more backlinks than the pages in cluster A. On the other hand, with IR (p) ordering, page p₂ may have higher rank (because its link comes from a high ranking page) than the pages in cluster A (that only have pointers from low ranking pages within the cluster). Therefore, page p₂ is reached faster.

In summary, during the early stages of a crawl, the backlink information is biased by the starting point. If the WIRS bases its decisions on this skewed information, it tries getting locally hot pages instead of globally hot pages, and this bias gets worse as the crawl proceeds. On the other hand, the IR (p) PageRank crawler of WIRS is not as biased towards locally hot pages, so it gives better results regardless of the starting point. Figure 4.6 shows that this conclusion is not limited to the Crawl & Stop with Threshold model. In the figure we show the performance of the WIRS under the Crawl & Stop model (Section 4.3). Remember that under the Crawl & Stop model, the definition of hot pages changes over time. That is, the WIRS does not have a predefined notion of hot pages, and instead, when the WIRS has visited, say, 30% of the entire Web under consideration, it considers the top 30% pages as hot pages. Therefore, an ideal WIRS would have performance 1 at all times because it would download pages in the order of their importance. Figure 4.6 compares 1) breadth-first 2) backlink and 3) PageRank ordering metrics for the IB(p) importance metric under this model. The vertical axis represents PCS, the
crawled fraction of hot pages at each point under the varying definition of hot pages.

The figure shows that the results of the Crawl & Stop model are analogous to those of the Crawl & Stop with Threshold model: The PageRank ordering metric shows the best performance.

Returning to the Crawl & Stop with Threshold model, Figure 4.7 shows the results when we use the IR(p) PageRank importance metric with G = 13.2. Again, the PageRank ordering metric shows the best performance. The backlink and the breadth-first metrics show similar performance. Based on
these results, we recommend using the PageRank ordering metric for both the IB(p) and the IR(p) importance metrics.

Figure 4.7 Fraction of Pune University Web crawled vs. PST. I (p) = IR(p); G = 13

4.6 Conclusion

In this chapter we addressed the general problem of ordering URLs for WIRS. We defined several different kinds of importance metrics, and built models to evaluate WIRS. We experimentally evaluated several combinations of importance and ordering metrics, using the Savitribai Phule Pune University Web pages. In general, our results show that PageRank, IR (p), is an excellent ordering metric when either pages with many backlinks or with high PageRank are sought. In addition, if the similarity to a driving query is important, it is useful to visit earlier URLs that:

- Have anchor text that is similar to the driving query;
- Have some of the query terms within the URL itself; or
- Have a short link distance to a page that is known to be hot.

With a good ordering strategy, we can build WIRS that can obtain a significant portion of the hot pages relatively early. This property can be extremely useful when we are trying to crawl a fraction of the Web, when our resources are limited, or when we need to revisit pages often to detect changes.
One limitation of our experiments is that they were run only over the Savitribai Phule Pune University Web pages. We believe that the Savitribai Phule Pune University pages are a reasonable sample: For example, they are managed by many different people who structure their pages in a variety of ways. They include many individual home pages, and also many clusters that are carefully managed by organization. Nevertheless, if resources and time permit, it will be interesting to investigate in the future non-Pune University Web pages to analyze structural differences and their implication for crawling.