Focused Crawling: A New Approach to Topic-Specific Resource Discovery*

Soumen Chakrabarti
soumen@cs.berkeley.edu

Martin van den Berg
vdberg@let.uva.nl

Byron Dom
dom@almaden.ibm.com

IBM Almaden Research Center

Abstract

The rapid growth of the world-wide web poses unprecedented scaling challenges for general-purpose crawlers and search engines. In this paper we describe a new hypertext information management system called a *Focused Crawler*. The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of *topics*. The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible hypertext documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl. It avoids irrelevant regions of the web. This leads to significant savings in hardware and network resources, and helps keep the crawl more up-to-date.

We describe a prototype implementation that is comprised of three programs integrated via a relational database: a *crawler* that stores metadata for documents and links in relational tables and is guided by various attributes in these tables, a *hypertext classifier* that updates the metadata with topic information from a large taxonomy (such as Yahoo!), and a *rating system* that updates metadata fields signifying the value of a page as a access point for a large number of relevant pages. The two "mining" modules guide the crawler away from unnecessary exploration and focus its efforts on web regions of interest.

We report on extensive focused-crawling experiments using several topics at different levels of specificity. Focused crawling acquires relevant pages steadily while standard crawling quickly loses its way, even though they are started from the same root set. Focused crawling is robust against large perturbations in the starting set of URLs. It discovers largely overlapping sets of resources in spite of these perturbations. It is also capable of exploring out and discovering valuable resources that are dozens of links away from the start set, while carefully pruning the millions of pages that may lie within this same radius. Our anecdotes suggest that focused crawling is very effective for building high-quality collections of web documents on specific topics, using modest desktop hardware.

1 Introduction

The world-wide web, having over 300 million pages, continues to grow rapidly at a million pages per day [2]. Such growth poses basic limits of scale for today's generic crawlers and search engines. At the time of writing[1], Alta Vista's crawler, called the *Scooter*, runs on a 1.5 GB memory, 30 GB RAID disk, 4x533MHz AlphaServer 4100 5/300 with 1 GB/s I/O bandwidth. Scooter connects to the indexing engine *Vista*, which is a 2 GB memory, 180 GB RAID disk, 2x533MHz AlphaServer 4100 5/300. (The query engine is even more impressive, but is not relevant to our discussion.) Other powerful web crawlers use similar fire-power, although in somewhat different forms, e.g., Inktomi uses a cluster of hundreds of Sun Sparc workstations[2] with 75 GB of RAM and over 1 TB of spinning disk, and it crawls over 10 million pages a day!

*Submitted to the *Eighth World Wide Web Conference*, Toronto, Canada, 1999.

[1] http://www.altavista.com/av/content/about_our_technology.htm

[2] http://www.inktomi.com/Tech/CoupClustWhitePap.html

In spite of these heroic efforts with high-end multiprocessors and clever crawling software, the largest crawls cover only 30–40% of the web, and refreshes take weeks to a month [2, 19]. The overwhelming engineering challenges are in part due to the one-size-fits-all philosophy: Alta Vista and Inkomi try to cater to *every possible query* that might be made on the web. Although services like Alta Vista or Inktomi are invaluable because of their broad coverage, the resulting diversity of content often snares all but the most craftily constructed queries in thousands of responses of little relevance or quality. Furthermore, the imminent explosion of web publication beyond North America and Europe, and beyond academic and corporate sites, will challenge even the most scalable solutions.

Compared to the web, development of the human brain has been tardy: it has grown "only linearly" from 400 to 1400 cubic centimeters in the last 3.5 million years. How do people avoid information overload? Serious web users adopt the strategy of filtering by *relevance* and *quality*. It does not matter to a physicist that the web, with 330 million pages, grows at a million pages a day; only a few dozen pages dealing with quantum electrodynamics will be added or updated in one week. Seasoned users also rarely roam aimlessly; they have bookmarked sites important to them, and their primary need is to expand and maintain this set of sites while preserving the quality.

The importance of such "starting points" is evident in that following Yahoo!, almost all search engines have added topic hierarchies with hand-picked, popular starting points. Unstructured ad-hoc search over the whole web is rapidly giving way to the formation of special-interest sites and "web-rings" supported by a community of people who contribute pages and links relevant to the their topic of interest. They are interested in expanding their collection while remaining strictly relevant.

For broad keyword queries that retrieve a large number of responses on the web, various systems [5, 3, 7, 17] have been designed to distill, from the responses and their link neighborhood, the most popular pages based on bibliometric measures [18] of citation. In order to answer arbitrary ad-hoc queries, these systems depend upon a key-word and linkage index of a substantial portion of the web, which must be generated by a crawler that is at least of the same scale as Alta Vista or Inktomi.

If such a system for ad-hoc topic distillation were somehow applied to the more specific problem of discovering resources related to a few select categories, the resulting system would not be *output sensitive*; e.g., to distill fifty good resource links on the topic of recreational bicycling, one would need to access a term and link index over hundreds of millions of documents. Much of this index would never be used, but, burdened by the responsibility of maintaining this huge index, the crawler would not be able to preferentially and frequently refresh the relevant regions of the index.

In this paper, we describe a crawler that will seek, acquire, index, and maintain pages on a specific set of topics that represent a relatively narrow segment of the web. Such a *Focused Crawler* entails a very small investment in hardware and network resources and yet achieves respectable coverage at a rapid rate, simply because there is relatively little to do. Thus, web content can be managed by a distributed team of focused crawlers, each specializing in one or a few topics. Each focused crawler will be far more nimble in detecting changes to pages within its focus than a crawler that is crawling the entire web. Such a need is increasingly expressed on the web today (Jim White in *Internet World* [28]):

> "We have an impersonal view of that sea of information. I want a personal view that zeros me in on the one percent of the Internet that I care about."

Focused crawling is more difficult than standard crawling. A standard ("unfocused") crawler's basic task is to fetch a page, parse out-links, and repeat. With sufficient concurrency via multi-threading, the only bottleneck will be disk I/O. Indexing the pages leads to an "inverted" index, mapping terms to document ID's or URL's. Hardly any other document attribute needs to be computed or updated. A focused crawler's task is more difficult. It has to maintain with each document some attributes reflecting its relevance to the focus topics and its quality and it has to use these to locate the pages most worthy of fetching or refreshing. This involves computation over various aggregates of pages, links, terms, and sites. Whereas custom disk data structures are used in standard crawlers (for the sake of speed) a focused crawler benefits greatly from the structured-query interface of a database. The transaction and crash-recovery support from a *relational database management system* (RDBMS) also assists rapid exploration of the space of possible

focused-crawling strategies. Balancing this convenience is the care needed to plan the interaction between the crawler, the topic classifier, and the quality rating system in a way that does not jeopardize performance.

Benefits of a semi-structured view: Here are some compelling examples of semi-structured queries that are enabled by our RDBMS representation (details later):

Site rating: We can order *sites* according to the density or quality of relevant pages found there. E.g., we can find the top five sites mostly specializing in mountain biking, using simple `group by` queries.

Linkage: Using a focused crawl with multiple topics, e.g., *traffic radar makers* and *auto insurance companies*, we can significantly narrow down the search for a possible suspicious alliance.

Community behavior: Simple statistics about the link graphs reveal important information about the community of the focused topic, e.g., whether it is competitive or collaborative, the typical time taken by a good resource to become popular, etc.

Semi-supervised learning: Human-supervised topic learning yields very high-quality filtering, but needs labor-intensive training. Focused crawling helps identify websites that are the "purest" by topic; all crawled pages for these sites (after vastly reduced human scrutiny per site) can then be fed back to improve the classification system.

Lifetime: Simple queries can identify topical regions of the web that grow or change dramatically as against those that are relatively stable. This can be of great value to web ontologists as in Yahoo! or the Mining Company (`www.miningco.com`).

Our work strikes an interesting symbiosis regarding the performance issue: the RDBMS lets us do ad-hoc structured analysis of the growing web graph so we can focus our attention on useful regions of the web; in turn, *focusing* limits the crawling scale (unlike the largest search engines) which enables the use of an RDBMS on modest desktop hardware.

2 Focused crawler administration

The focused crawler has to be given initial human input in two forms. First, a taxonomy of the most basic and prominent web topics has to be constructed. Second, the user has to specify which of these categories and their subtrees are of interest to the user. In this section we give a user's view of the system.

2.1 Topics and categories

Taxonomies have been used to simplify studying the world by stratifying and partitioning it since ancient times. Well-known examples are taxonomies in biology and the Dewey decimal system. More recent examples are Yahoo![3], the Mining Company[4] and the Virtual Library[5]. Almost every web-search site today has teams of ontologists who organize web information into hierarchies of categories. The focused-crawler administrator starts with such a *master category tree*[6].

The purpose of this starting set of categories is to provide a kind of *basis* (in the mathematical sense) onto which the user maps her interests (the focus *topics*). Eventually, the focused crawler will assign relevance scores to each visited document based on how well it matches the categories associated with the focus topics. Why are we not setting up a simple two-class learning problem for each focus topic rather than doing this mapping onto a fixed hierarchy of categories? There are three major reasons for this design decision:

[3] http://www.yahoo.com
[4] http://www.miningco.com
[5] http://www.vlib.org
[6] For this paper we will use a taxonomy with 1775 leaves derived from Yahoo!.

Reuse classifier-training effort: Learning to recognize text documents is made difficult by the large dimensionality of the problem. It may be burdensome for every system administrator to prepare enough sample documents to have an adequate number (for learning) of positive and negative examples for his or her interest profile. The work of mapping interest topics onto a predefined set of categories will usually be significantly less than finding an adequate number of these positive and negative examples. The idea here is that the classifier is trained once for the categories in the taxonomy and then, each new application (set of interest topics) can be implemented by this simple topic-to-category mapping.

By packaging broad web topics as a starting point in their administration, the focused crawler relieves them of a large part of the work. One may even envisage that a standards organization will design many "backbone taxonomies" for smaller groups to fill in and refine. (Similar models are espoused in the Mining Company and the Open Directory Project[7].

Model negative class better: When doing a binary (positive/negative) classification characterization of the the negative class (e.g. "not about mutual funds") is often problematic. Using a taxonomy as we do, deals with this problem by describing the negative class as a union of positive classes. This is not only a mental artifice, but it also affects the accuracy of learning algorithms significantly [23], because commonly used statistical models have large estimation errors on the diverse negative class.

Discover related classes: The framework of a master taxonomy helps the user detect additional regions of the web that are topically related to her interest which were not naturally connected with her start set. As we shall see later, the focused crawler is quick to suggest that crawling only on *mutual funds* while forbidding *investment* in general will not work well, because the neighborhoods of these two topics comingle. It is able to do this because is can classify web pages encountered into other categories from the taxonomy, which would not be possible if the binary-classification approach were used.

Why is this significant? In addition to teaching the user about the relationship of her interests to other topics on the web, this capability is important for diagnostic purposes. In some cases (e.g. *mutual funds* vs *investment*) such as narrow topics, it is better to broaden the set of categories from those providing a minimal covering of the interest topics, because doing so provides a higher degree of linkage, which means many more available paths for finding relevant pages. In such a scenario the crawling-priority relevance score and the final (for displaying to the user) relevance score will be determined differently. A natural way to expand the topic set for this purpose is to add some of the *parents* and/or *siblings* of the relevant topics from the taxonomy - another advantage over binary classification.

2.2 Mapping and marking good categories

The master taxonomy is used to guide the user in marking suitable nodes as relevant. Whereas we describe topics informally in a few words, a topic is not to be interpreted syntactically. The first task is to map the topic to a set of nodes in the master category tree. This is started by the user submitting the sample starting points to a classifier associated with the topic taxonomy. (The applet shown monitors the web page being rendered by the web browser. The user submits a page for classification using the *Classify* menu.) The classifier routes the submitted documents to the best matching nodes in the category tree, marking all nodes on the way. All marked nodes are presented to the user as candidates for focused crawling. The user selects some nodes and selects them, thereby marking them *good* in the parlance of this paper. These are shown highlighted in the tree view in Figure 1.

For the topic of "recreational bicycling," two category nodes, `/Business_and_Economy/Companies/Sports/Cycling` and `/Recreation/Sports/Cycling`, were found to be good choices, arranged as:

```
/Business_and_Economy/Companies/Sports/Cycling/
    Bike_Shops/
    Others/
/Recreation/Sports/Cycling/
    Clubs/
```

[7] http://directory.mozilla.org/

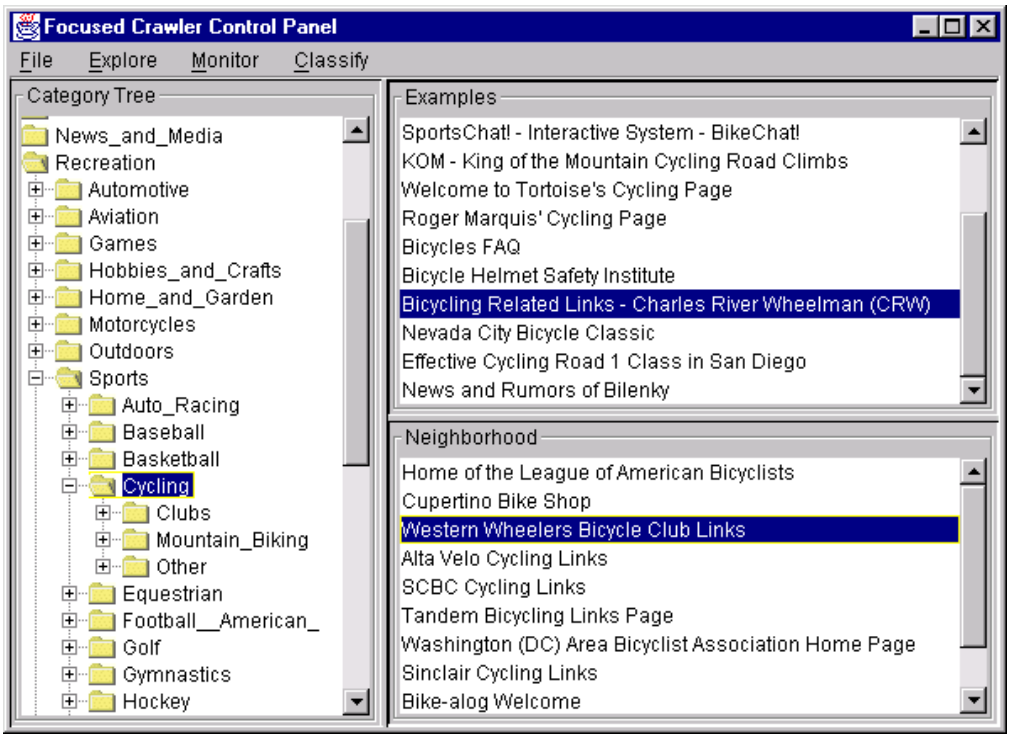

Figure 1: Focused crawler administration. A sample session for configuring a crawl for "recreational bicycling" resources. The tree view is used to show the user some of the categories that best match her examples, and get a selection of category nodes to be marked "good" (shown selected). The right upper panel shows the exemplary documents that have been used to derive the category models for the classifier. The right lower panel shows URL's in the vicinity of the example pages that could be good additional starting points for the crawler. The user can drag these into the examples panel as desired.

```
Mountain_Biking/
Others/
```

Goodness is inherited by descendants of a node. Example sites that the master category system already knows about are shown in the upper right panel and can be viewed through a browser by clicking. When such a page is visited, the applet shows URLs of pages in the neighborhood of the example whose titles have many words in common with the most distinctive words of the topic (some means of computing these are discussed in an earlier paper [6]). Any pages thus found useful can also be added to the examples by dragging and dropping.

This is the only manual part in the process, where some experience with the web is desirable. A topic may or may not map to a single subtree in the category tree. In our example above it did not. In another example we will use later (`/Health/Diseases_and_Conditions/AIDS_HIV`) it did. The taxonomy provides a natural mechanism to control the recall-precision trade-off. Picking nodes near the leaves makes the crawl very sharply focused, but it may miss relevant sites because it is too stringent. For more coverage, more general categories must be used, but more crawling has to be done. (In particular, it is a bad idea for the root node to be good unless one wants a crawl of the whole web.)

2.3 Extending and editing categories

The administrator potentially provides feedback in three ways: correcting classifications, eliminating classes, and refining classes. Sometimes the administrator will feel that the leaf nodes to which her examples have been assigned are still too broad and need to be refined. The tree view interface lets her create new directories

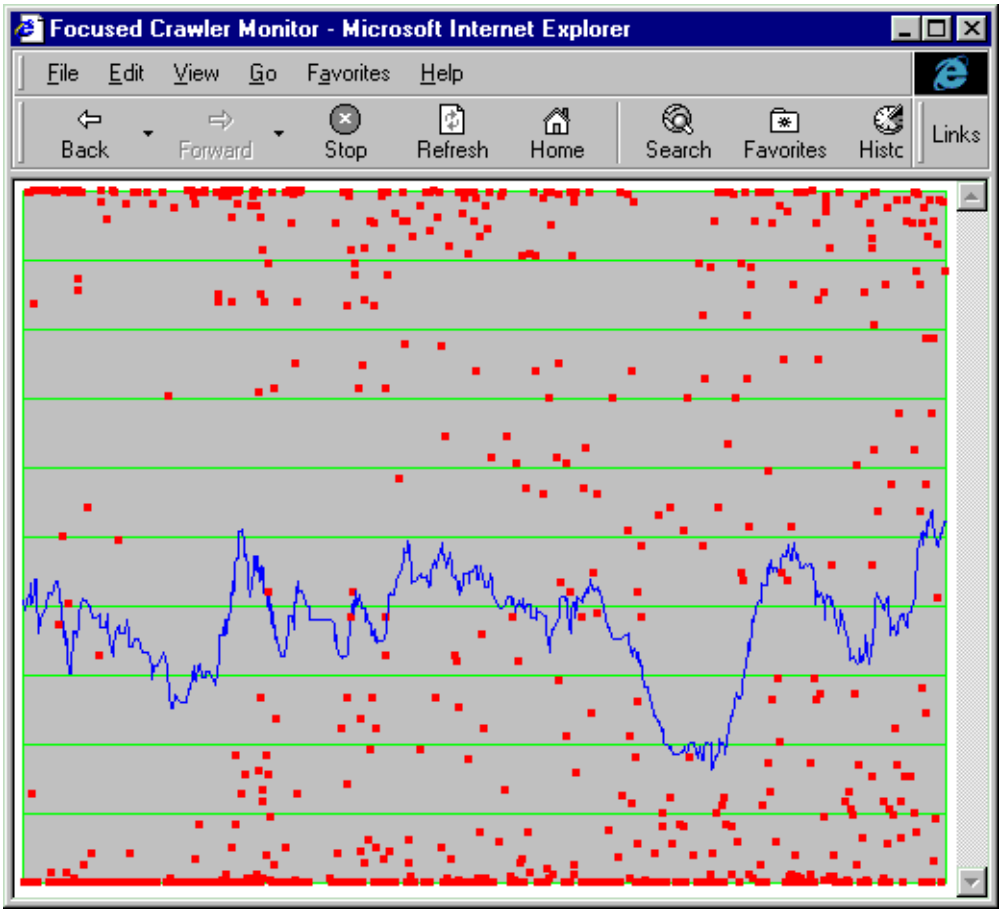

Figure 2: Monitoring the progress of a focused crawler. This applet monitors the crawl database and plots a continually scrolling chart of the recent relevant page acquisition rate (blue line). The x-axis represents time and the y-axis is a relevance probability score between 0 and 1. Each red dot is a crawled web page, which can be visited by clicking.

(and move directories) and populate them with examples. If major changes are made to the master category tree, some time is needed for the classifier to integrate the new structure into its models [6]. For our testbed with about 260,000 documents from Yahoo!, this takes a few hours. Smaller changes, such as moving of documents while keeping the tree unchanged, are interactive.

2.4 Running and monitoring the crawler

At this stage, the focused crawler can be started. A focused crawler is a complex piece of software which not only crawls tens of thousands of pages per hour, but makes decisions based on millions of arithmetic calculations per page. It is thus quite helpful for diagnostic purposes to visualize the status of the crawl graphically.

We have developed an applet that runs inside a browser and contacts the crawler's database over JDBC. It then maintains a plot of page relevance against time. In the figure, each red dot is a web page, which may be viewed in a browser window by clicking on the dot. The x-axis represents time. The y-axis is the relevance score (a probability value) between zero and one. The blue line is a smoothed moving average over a recent window of pages fetched. Continual refreshing introduces new points at the right edge, and the display scrolls the leftmost points off the screen (see Figure 2).

If the page acquisition rate suddenly lowers, the right-to-left scrolling slows down. This can be made to

raise an alarm (not implemented). Alternatively, the crawler may be getting many pages, but their relevance will be very low. The blue line will go down without significant recovery. This too can be made to raise an explicit alarm if necessary.

3 System architecture

The focused crawler has the following modes of operation:

Learning from example: We envisage that a focused crawl starts with a master category tree which is typically customized for the application. In the learning phase, the user submits the example starting sites to the classifier and refines the category taxonomy as desired, furnishing examples for new categories. The system then analyzes the examples to learn the categories.

Resource discovery: In the second phase the system crawls out from the starting pages provided, containing the crawl to relevant pages as far as possible, and discovering new resources (sites and pages) as quickly as possible.

Reconnaissance: Interleaved with the second phase is the task of identifying from the current crawl database some pages from which large numbers of highly relevant pages are accessible. This list has two purposes. It is a compact output of the crawl for human inspection (similar to a Yahoo! directory), and it also helps to further guide the crawler priorities.

Maintenance: Once the set of relevant sites known to the crawler becomes mature, the focused crawler can be instructed to allocate a specified part of its time for checking back on pages crawled earlier for new material and links to crawl. This can also be priortized by relevance or popularity score.

In this section we will describe the main modules of the focused crawling system.

3.1 The relevance rating system

Relevance is enforced on the focused crawler using a hypertext classifier [6, 10]. We assume that the category taxonomy imposes a hierarchical partition of web documents. (In real life, documents are often judged to belong to multiple categories. The glitch in our model is notional and appears not to affect the system. The main problem in extending the taxonomy to a directed acyclic graph is that various statistical estimations will need to be computed over exponential number of paths.) Categories in the taxonomy tree, also called nodes, are denoted c. The predicate good(c) denotes whether a node c has been marked as good. By definition, for any document d, the probability that it was generated from the category c_0 corresponding to the root node, denoted $\Pr[c_0|d]$, is 1. In general $\Pr[c|d] = \Pr[\text{parent}(c)|d]\,\Pr[c|d, \text{parent}(c)]$. $\Pr[\text{parent}(c)|d]$ is computed recursively. $\Pr[c|d, \text{parent}(c)]$ is computed using Bayes rule as $\Pr[c|\,\text{parent}(c)]\,\Pr[d|c]/\Sigma_{c'}\,\Pr[c'|\,\text{parent}(c')]\,\Pr[d|c']$, where the sum ranges over all siblings c' of c. Finally, the probablity that a page is relevant to the focused crawl is $\Sigma_{\text{good}(c)}\,\Pr[c|d]$. This quantity, denoted $R(d)$, is typically very small in absolute value, so for each document we store the logarithm of $R(d)$ in our tables.

To model the way a document is generated by a category we use the followin Bernoulli document model for text. First pick a category for the document and pick a length for it. Associated with each category is a many-sided coin. Each face of the coin represents a word; the probability that the face comes up corresponds with the probability that the corresponding word occurs in a document of this particular category. This coin is repeatedly tossed to write out the document until the length is reached. A document is thus seen as a bag of words.

The category taxonomy is represented in a relational table as follows:

```
kcid            smallint        not null        primary key,
pcid            smallint        not null,
kcname          varchar(250)    not null,
good            smallint        not null
```

`kcid` is the ID of the current node ("kid" class ID), `pcid` is the ID of the parent class, `kcname` is the textual name, and `good` is a bit set to express if the node is good for the current focus or not. The document table has two fields coupling it with the taxonomy table: `relevance`, which is set to $\log \Sigma_{\text{good}(c)} \Pr[c|d]$; and `cid`, which represents the *best matching leaf class* for the document.

3.2 The popularity rating system

Relevance is not the only attribute used to evaluate a page while crawling. A long essay very relevant to the topic but wthout links is only a finishing point in the crawl. A good strategy for the crawler is to identify pages that are almost exclusively a collection of resource links on the topic. These are similar in spirit to Kleinberg's *hubs* [17], except that they are computed not from the result of a keyword query on an index of the whole web, but from the portion of the web we have marked as relevant for our topic.

In a nutshell, Kleinberg's algorithm (HITS) assigns two scores $h[v]$ (hub score) and $a[v]$ (authority score) with each node v. It then repeats the following iterations on the edge set E a suitable number of times: $a[v] \leftarrow \sum_{(u,v)\in E} h[u]$ and $h[u] \leftarrow \sum_{(u,v)\in E} a[v]$, interspersed with scaling $h[\,]$ and $a[\,]$ to sum to one. This iteration embodies the circular definition that important hubs point to important authorities and vice versa.

Note that beyond the selection of the initial graph, there is no role for relevance in the iterations, and the best hubs may not be very relevant to the focused crawl. (Even in the keyword search context, this leads to a phenomenon called *topic drift*: the best hubs and authorities found are on a topic more general and intensely cross-cited than the query intended [3, 7].)

For prioritizing crawls, we need a certain asymmetry in how we treat hubs and authorities.

1. Based on the recall-precision trade-off of the application and the availability of disk and network resources, set a suitable threshold ρ on relevance. We included between 10 and 20% of the most relevant nodes; our results were not sensitive to the precise choice in this range.

2. Of the pages fetched in the current crawl, consider only those exceeding the threshold as candidate *authorities*. Any page pointing to at least one authority is a candidate hub. Hubs are not thresholded.

3. Construct the edge set E, using only those links that are between pages on different sites. Unlike in HITS our edges have weights; edge (u, v) has three associated numbers: the relevance score of u called $R[u]$; the number of distinct pages on the same server as u that point to v, called $h'[u, v]$; and the number of distinct pages on the same server as v that u points to, called $a'[u, v]$. This is in accordandce with Bharat and Henzinger's view [3] that (with the exception of ISP sites) a server, not a page, roughly corresponds to one unit of human voting.

4. Perform the iterations using edge weights $R[u]/h'[u, v]$ while computing $a[v]$ and $R[u]/a'[u, v]$ while computing $h[u]$. We change the authority assignment to only consider pages that pass the relevance bar: $R[x] > \rho$.

It is quite simple to implement the iterations in a few lines of SQL.

4 Crawling strategies

The relevance and popularity rating modules stamp various scores on visited URLs, but it is up to the crawler to use them for careful scheduling of page fetches. The complete system is shown as a block diagram in Figure 3.

Any large-scale crawler has to do careful *frontier management*, by which we mean the checking in and selection of suitable pages to crawl and refresh. This is already quite tricky with standard crawlers, given bottomless sites, spider traps, and artificial link-based popularity gimmicks. Frontier management gets more complicated for focused crawling, because the issue of maintaining relevance is added to the above performance issues. In this section we discuss the various policies which may be used to guide the crawler.

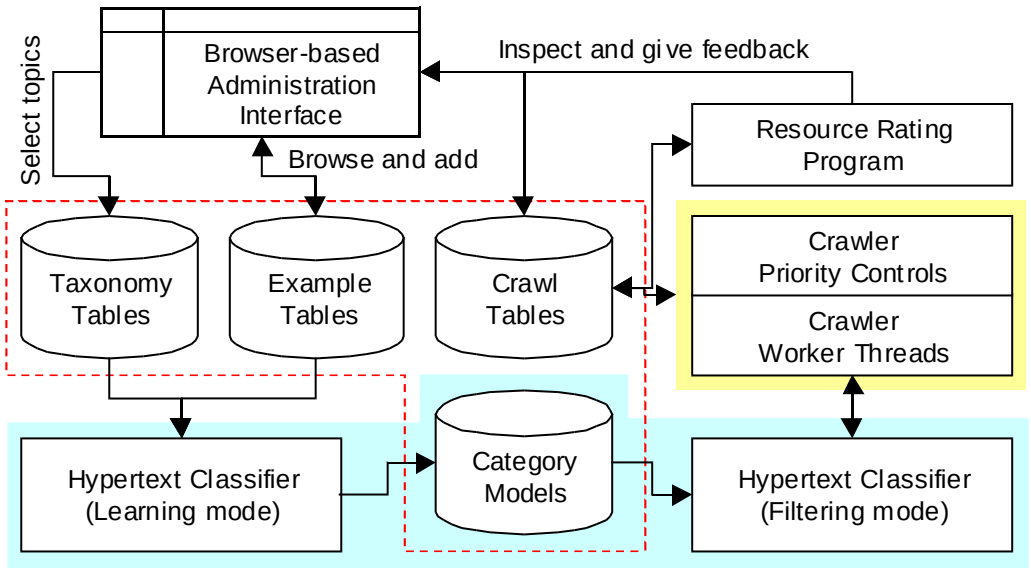

Figure 3: Block diagram of the focused crawler showing how the crawler interacts with the guidance mechanism.

4.1 Managing the frontier

Apart from usual metadata, the focused crawler stores the following fields that control the crawl priority of a page:

```
num_tries       smallint        not null,
priority        smallint        not null,
relevance       float           not null,
cid             smallint
```

`num_tries` is the number of times the crawler has attempted to retrieve the page. During early stages of a focused crawl, when there is ample storage *and* many relevant pages to fetch, the crawl must be first and foremost biased away from refreshing or retrying access to pages (this is the mode we will do most experiments in). When little progress can be made on a crawl because the disks are almost full or the topic is nearly entirely crawled, the priority must switch to favor revisiting pages retrieved earlier. `priority` and `relevance` are two fields that permit two kinds of crawler control that we call *hard* and *soft*. `relevance` is set by the classifier as stated earlier. The `priority` field will be discussed shortly. `cid` is the best matching category ID for the page.

Priorities using various combinations of the above fields will lead to different flavors of focused crawlers. In our system, a *watchdog* thread checks out work from the crawl database and allocates them to worker threads. It is important to avoid sorting the crawl table on priority fields. We use an multi-column index instead. This, however, makes database insertions slow. To recover speed, the watchdog reads the database in *uncommitted read* mode of isolation, in which dirty reads are possible.

One issue that has to be addressed in any crawler is that of being "polite" to servers. This involved obeying the `robots.txt` file if any, and not requesting service from the same server too often. The latter is also important for the initial stages of focused crawling: we'd rather collect new resources aggressively than delve deep into a few sites, even if they are very good. `priority` is a lazily updated field that roughly reflects the number of URLs crawled from the same site as the current URL. We could keep this number in a separate table, but then the watchdog would need a table join and a sort, which would be too expensive. There are many other database issues to resolve and optimize that are beyond the scope of this paper.

4.2 Hard and soft focusing

In hard focused crawling, the classifier is invoked on a newly crawled document in a standard manner. When it returns the best matching category path, the out-neighbors of the page are checked into the database if and only if some node on the best matching category path is marked as good. Crawling via hard focusing is conservative and is guaranteed to remain limited to a finite population of pages, according to a basic theorem of birth-death processes [26, Page 170].

In soft focused crawling, all out-neighbors of a visited page are checked into DB2, but their crawl priority is based on the relevance of the current page. This is reminiscent of web browsing and surfing models discussed by Huberman et al [15]: given the edge (x, y) where x is the currently visited page, they define the "value" of page y, called $V(y)$, as a random variable expressed as $V(y) = V(x) + \epsilon(x, y)$, where ϵ is a random variable. This model has been found to agree closely with user behavior. In our case, "value" is relevance; our model lets us define this precisely as a probability. When we visit x, we can estimate $V(x)$ using the classifier. We remain agnostic of ϵ and use only $V(x)$ to guess $V(y)$ and act accordingly. If and when we reach y later, we can correct our estimate of $V(y)$ to the actual value calculated by the classifier.

For the soft focused crawler, the crawler selects a batch of unvisited pages (typically, a few dozen per thread) at a time in lexicographic order of

```
(num_tries asc, relevance desc, priority asc, bytehash),
```

where `asc` means ascending and `desc` means descending. `bytehash` is a random number to resolve ties without loading any particular server. Each URL from the batch is downloaded and classified. This generally leads to a revision of its relevance score. The revised relevance score is also written into the new records created for unvisited out-links.

4.3 Crawl monitoring

In hard focused crawling there is a fear of *crawl stagnation*, i.e., failing to acquire additional relevant pages. Stagnation is usually never a problem for an ordinary crawler. In our case, depending on the narrowness of the topic and the degree of community formation on the topic, stagnation is possible. Stagnation may happen because the web graph is inadequately linked up, because the classifier makes errors in topic judgement, or (on rare occasions) because most of the relevant pages have indeed been retrieved by the crawler. In a world with incomplete knowledge, there can be no formal separation between the first and third possibilities.

In soft focused crawling, the fear is the opposite threat of *crawl diffusion*, i.e., acquiring less and less relevant pages. Diffusion takes place under the same circumstances that stagnation takes place for hard focus. Because the most relevant pages are prefered, less and less relevant pages will be retrieved if no relevant pages are available either because the web is not linked up adequately, or because there are no more relevant pages to be retrieved. If there still are relevant pages around, then at a certain moment a relevant page might be touched, and the crawl will start getting more relevant pages again. Again, in a world with incomplete knowledge, there can be no formal separation between these two cases.

Using simple SQL statements, it is possible to find out how the crawl is doing, and whether stagnation or diffusion are taking place. For example, stagnation can easily be found out by inspecting the result of the query

```
select num_tries,count(*) from arc.doc group by num_tries
```

To compare how many new pages have still to be visited (`num_tries`=0) and how many have been visited more than ones (`num_tries>1`). It is also fairly easy to find out how many documents there are for the different categories

```
with census(cid,cnt) as (select cid,count(oid) from arc.doc group by cid)
   select cid,cnt,kcname from census,arc.cidmap where cid=kcid order by cnt
```

Which prints out the number of pages that where classified as category `cid`.

5 Evaluation

There are at least four measures of the performance of a focused crawler: relevance (precision), coverage (recall), refresh rate, and quality of crawl. We will not deal with coverage or refresh rate in detail. It is extremely difficult to even define recall for a focused crawler, because we have a rather incomplete and subjective notion of what is "good coverage" on a topic. Whereas consensus has been forced in traditional IR benchmarks, such agreement would be very hard to arrive at in a reasonable manner in the case of the web. Likewise, unless we design artificial workloads, it is difficult to characterize the refresh rate. The focused crawling framework makes no commitment on the refresh policy. One can easily configure the crawler to do this at page level or site level, based on relevance score, quality score, or a combination of the two. We will study precision and quality in detail.

5.1 Experimental setup

The focused crawler is a C++ application that was run on a dual-processor 333 MHz Pentium-II PC with 256 MB of RAM and SCSI disk. IBM Universal Database v5 was used with 16 MB of buffer pool per crawl. The administration program is a JDBC-based Java applet that connects to the crawl database. Our test machines are connected through a half-duplex 10 MB/s Ethernet through the router to a SOCKS firewall machine. The firewall is connected to the ISP using full-duplex 10 MB/s SMDS over DS3. The ISP connects us to a 622 MB/s OC12 ATM backbone (UUNET High Performance Network[8]).

A full-scale crawler never operates through a firewall. Although we had access to machines outside the firewall, we decided to demonstrate the viability of focused crawling by running it inside the firewall and consuming negligible network resources. We ran the crawler with relatively few threads compared to what it can handle to avoid disrupting firewall performance for others.

We picked several topics that could be represented by one or few nodes in a master category list derived from Yahoo!in the summer of 1996. We name the topics informally; note that the topics are *defined* in terms of the categories, and are hence not syntactic objects. Here are the names of the topics we used, an informal gloss, and how they were mapped to the master categories:

Cycling: Recreational cycling, mountain biking, clubs, cycling equipment stores.

Gardening: Gardening supply stores, gardening tips, horticulture clubs.

HIV/AIDS: Research and government organizations, social issues.

Investing/Mutual funds: Investing, stocks and bonds, mutual funds.

The results were comparable for these and several other crawls. For concreteness we will present selected results from the above set. In the last case, we started the crawl looking for mutual funds alone, but the focused crawler quickly indicated that a crawl looking for mutual funds related pages that did not also accept pages on investment in general would stagnate rapidly because the topics are intimately mixed.

In the following sections we will make the following measurements.

- For a variety of topics, we study the absolute acquisition rate to see if it is high enough to warrant a focused crawl. We compared the distribution of relevance scores of soft focused, hard focused, and unfocused crawls.
- To judge the robustness of our system, we sampled disjoint fractions of the available set of "seed" URL's and started separate crawls. We compared the rate of acquisition of relevant pages between the two crawlers.
- As another test of robustness, we ran the quality rating program on the crawls that started from the samples, and then measured the extent of overlap between the top rated pages and servers (IP addresses) found by the two crawlers.

[8] http://www.uu.net/lang.en/network/usa.html

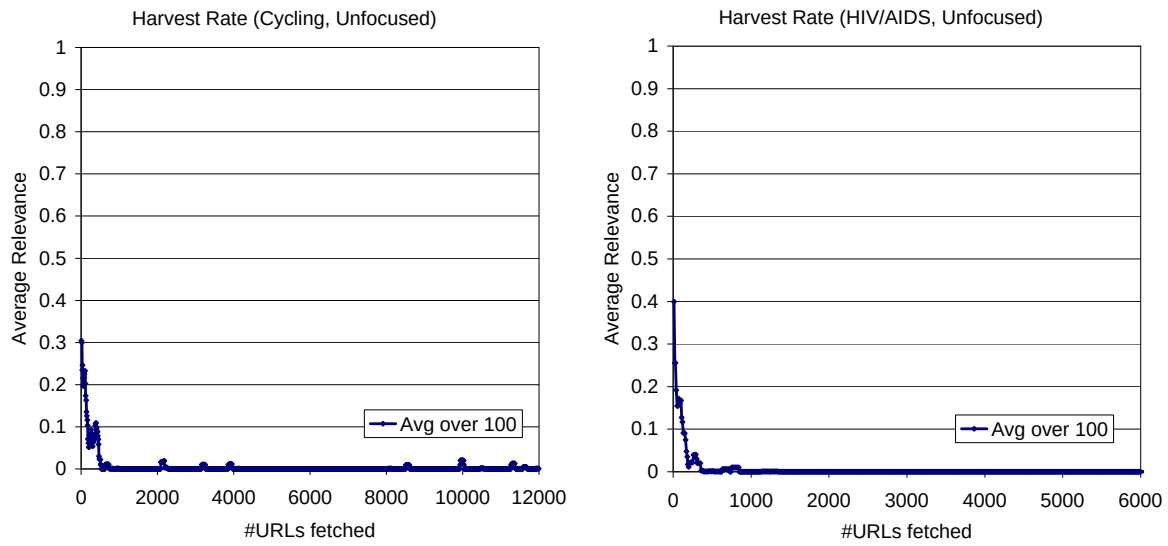

Figure 4: Rate of relevant page acquisition with a standard unfocused crawl on the topics of bicyling and HIV/AIDS.

- We present some of the top-rated lists as anecdotal evidence of the quality of resource discovery. We show examples where promoting unvisited neighbors of top-rated pages lead to further acquisition of relevant pages.

5.2 Relevant page acquisition rate

Perhaps the most crucial evaluation of focused crawling is to measure the rate at which relevant pages are acquired, and how effectively irrelevant pages are filtered off from the crawl. This *harvest ratio* must be high, otherwise the focused crawler would spend a lot of time merely eliminating irrelevant pages, and it may be better to use an ordinary crawler instead!

Human judgement, although subjective, would be best for measuring relevance. Clearly, even for an experimental crawler that acquires only ten thousand pages per hour, this is impossible. It may appear that using the same classifier to judge the relevance of crawled pages will lead to an overly optimistic view of the system, but this is not necessarily true. It is to be noted carefully that we are not, for instance, training and testing the classifier on the same set of documents, or checking the classifier's earlier work using the classifier itself. Just as human judgement is prone to variation and error [20], we have a statistical program that makes mistakes. Based on such imperfect recommendation, we choose to or not to expand pages. Later, when a page that was chosen is visited, we evaluate its relevance, and thus the value of that decision. Thus we are evaluating not the classifier but the validity and viability of the focused crawling architecture.

For each topic, three different crawls were done: *unfocused*, *soft focused* and *hard focused*. For each topic, the three crawls start out from the same set of URLs. These were collected by a search at Alta Vista and a search using HITS, followed by some screening by hand to eliminate irrelevant pages. These URLs were initially stored in the crawl database, all with the same (highest possible) priority. The crawler started from these URLs and then explored with different policies depending on the experiment, as explained earlier for the focused variants.

In the unfocused case, the crawler repeatedly fetches a batch of new URLs (typically, a few dozen per thread) in order of a hash value computed from the URL (effectively a pseudo-random order). The crawler downloads, parses and classifies each page. The relevance score is stored in the database. Also, all out-links in the current page are entered into the database for visitation. The crawler then repeats the process. The classification is necessary only to compare unfocused with the other two methods. This will slow the crawl down a little. For this reason, and also because network load fluctuates greatly from experiment to

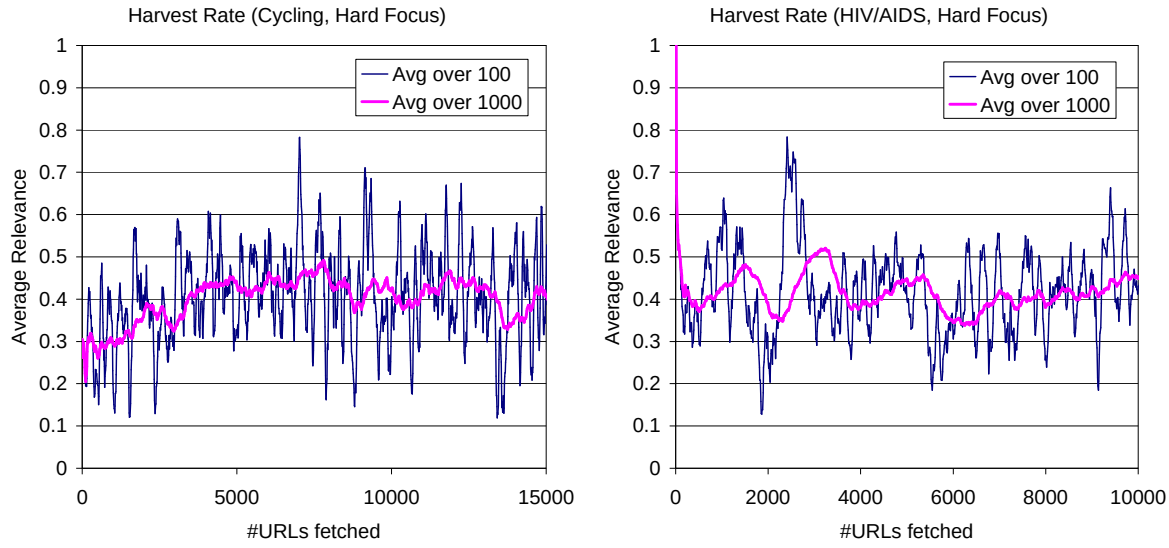

Figure 5: Rate of relevant page acquisition with a hard focused crawl.

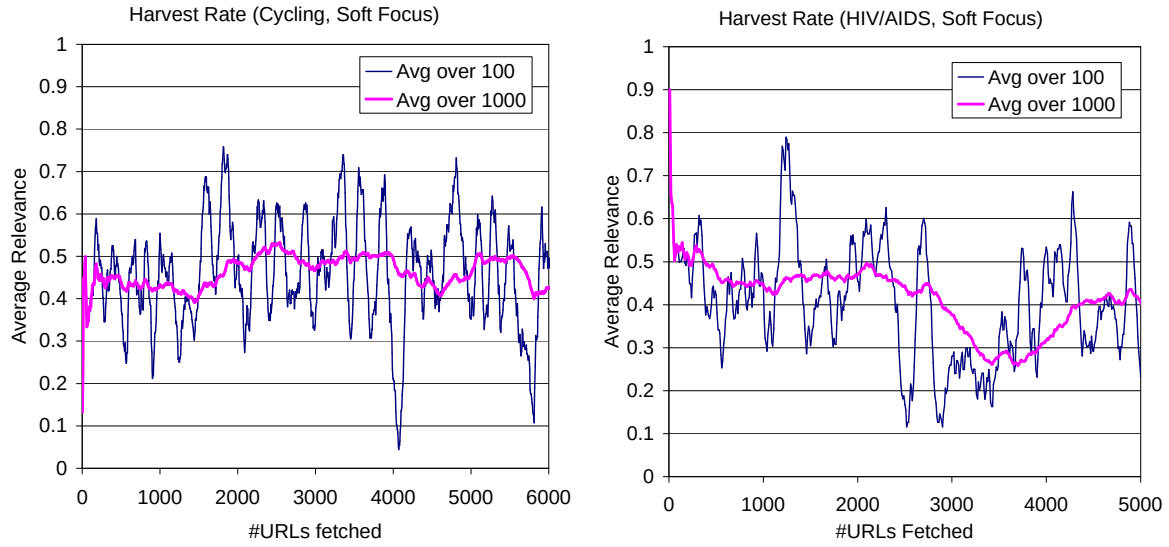

Figure 6: Rate of relevant page acquisition with a soft focused crawl.

experiment, in our results we present time not as wall-clock time, but the number of pages fetched so far.

Figure 4 shows the results of unfocused crawls for bicycling and HIV/AIDS, starting from seed sets of several dozen highly relevant URLs. The x-axis shows the number of pages acquired (as a representative of real time). The y-axis shows a moving average of $R(p)$, where p represents pages collected within the window. It is immediately evident that focused crawling does not happen by accident; it has to be done very deliberately. The unfocused crawler starts out from the same set of dozens of highly relevant links as the focused crawler, but is completely lost within the next hundred page fetches: the relevance goes quickly to zero. This happened even after we helped it in various ways, such as disabling such highly interconnected sites as `www.amazon.com`.

In contrast, it is heartening to see in Figure 5 that the hard-focused crawls keep up a healthy pace of acquiring relevant pages over thousands of pages, in spite of some short-range rate fluctuations, which is expected. On an average, between a third and half of all page fetches result in success over the first several thousand fetches, and there seems to be no sign of stagnation. This rate was in fact higher than what we had hoped for. Similar observations hold for the soft focused crawler, shown in Figure 6.

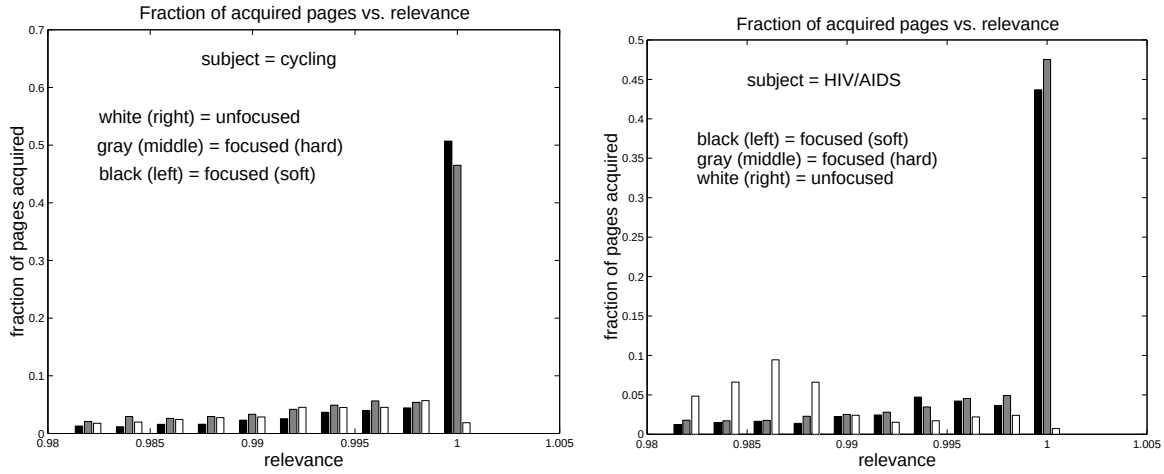

Figure 7: Distribution of relevance scores in the bicycling and HIV/AIDS crawls from the three crawlers.

Given that none of the crawls approached stagnation, it is difficult to compare between hard and soft focusing; they both do very well. For cycling, the hard crawler takes a little while to warm up because it loses some good opportunities to expand near-match pages. We believe the soft crawler is more robust, but needs more skill to monitor and guard against unwanted topic diffusion. The main technical problem in doing this is to distinguish between a noisy vs. systematic drop in relevance.

Figure 7 explain the earlier time-traces by showing the distribution of relevance of pages. Pages obtained by focused crawling show a very sharp peak at the highest possible relevance value, whereas the unfocused crawler shows essentially a flat distribution of relevance. It also appears (for cycling) that soft focusing can "tunnel through" mediocre pages to get slightly better pages than hard focusing.

5.3 Robustness of acquisition

Another important indicator of the robustness of a focused crawler is the ability to ramp up to and maintain a healthy acquisition rate without being too sensitive on the start set. To test this, we took the set of starting URL's and sampled subsets uniformly at random. We picked two disjoint random subsets each having about 30% of the starting URLs. For each subset, a different focused crawl was launched (at different times).

We will present two quantities. First we will measure the overlap of URLs crawled by the two crawlers. We will use bicycling and mutual funds as examples. The overlap is measured along time t, which is measured by counting the number of URLs fetched. (Direct comparison of wall-clock time is less meaningful owing to fluctuating network performance.) At any time t, the crawlers have collected URL sets $U_1(t)$ and $U_2(t)$. We plot $|U_1(t) \cap U_2(t)|/|U_1(t)|$ and $|U_1(t) \cap U_2(t)|/|U_2(t)|$ along time t (note that $|U_1(t)| = |U_2(t)|$ and therefore there is only one line in this case). Sample results, for bicycling and mutual funds, are shown in Figure 8.

We picked the two topics specifically because we wanted to study one co-operative community like bicycling and one competitive domain like investing and mutual funds. For cycling, the intersection between the set of URLs crawled grew rapidly to 90%. For mutual funds, it grew to over 60%. This confirmed our intuition about the two communities. The steady growth in overlap is heartening news, although it is a statement primarily about web behavior, not the focused crawler. It means that the choice of starting points is not critical for the success of focused crawling. We do have to double-check one thing, however. What if for reasons unknown, both crawlers started crawling pages out of one common site as soon as they reached there? This fear turns out to be ill-founded: a plot of the extent to which *IP-addresses* visited by the two cralers overlap against time shows generous visit to new IP-addresses as well as a healthy increase in the intersection of server IP-addresses. The intersections are plotted against time by first lining up the URL's fetched by each crawler side-by-side, then deriving the two sequences of IP-addresses visited, $S_1(t)$ and $S_2(t)$, and computing $|S_1(t) \cap S_2(t)|/|S_1(t)|$ and $|S_1(t) \cap S_2(t)|/|S_2(t)|$ for each t. In this case $|S_1(t)|$ is in general

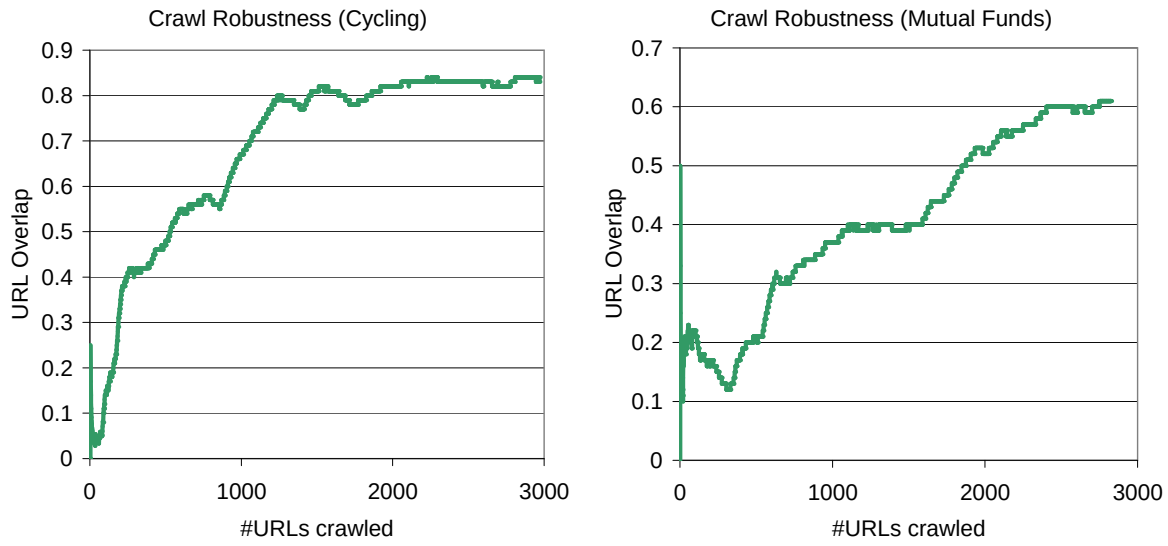

Figure 8: Overlap of URLs crawled by two soft focused crawlers starting from randomly sampled seed sets on bicycling and mutual funds.

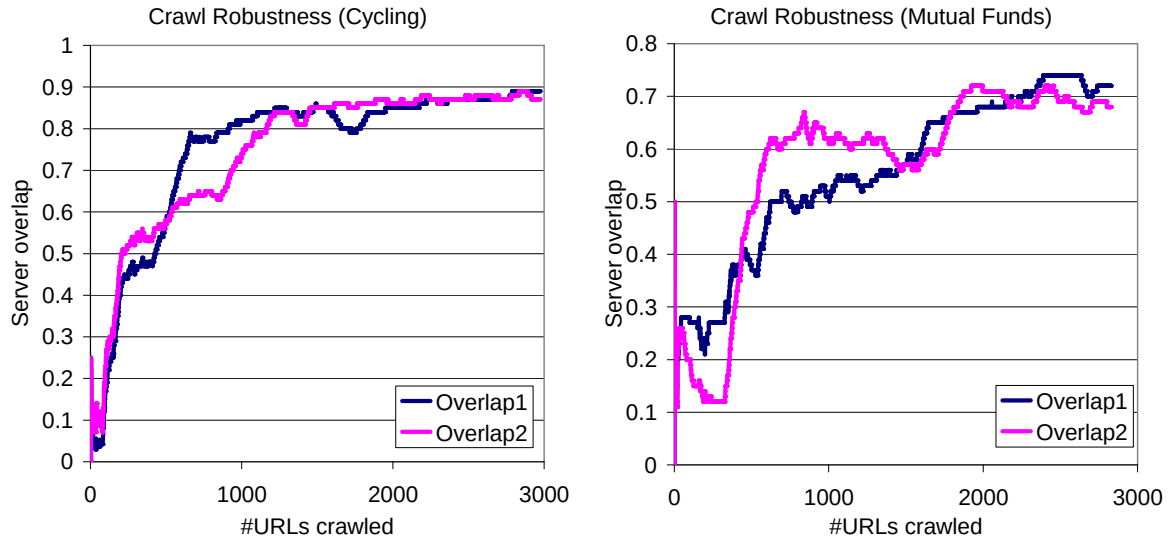

Figure 9: Overlap of servers crawled by two soft focused crawlers starting from randomly sampled seed sets on bicycling and mutual funds.

different from $|S_2(t)|$. The results are shown in Figures 9.

5.4 Robustness of resource discovery

Overlap in the set of servers and URLs crawled is a good indicator of inherent stability of the focused crawler. However, we wanted to also check that the topical subgraph of the web that is built by the focused crawler leads to robust estimations of popularity (estimated along the lines of recent topic distillation work). To do this we again used the two sets of crawlers that started from random samples of the available seed set. After acquiring 10,000 pages, we ran the popularity/quality rating algorithm with 50 iterations and produced a list of top "authorities" (as defined in HITS). Then we measured the intersection of server IP-addresses in the top 25. We picked addresses rather than URL's because many pages, especially in mutual funds and HIV/AIDS are heavily frames enabled and have slight variants in URL. The results are shown in Figure 10.

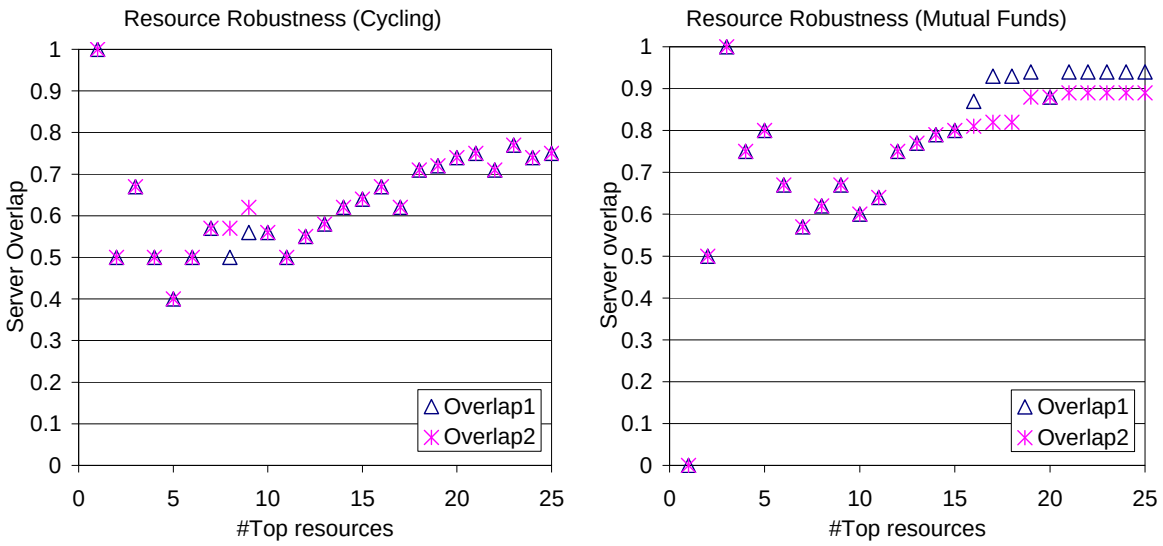

Figure 10: Overlap of the 100 best rated servers crawled by two soft focused crawlers starting from randomly sampled seed sets on cycling and mutual funds.

We see that in spite of slight local rank perturbations, the most popular sites are identified jointly by both runs of the focused crawler, although it started from different seed sets.

5.5 How remote are good resources?

Now we take a hard look at the following question: is the focused crawl doing any real exploration, or were the resources, specifically, the highly rated ones, within one or two links of the start set, or worse, *in* the start set? In Figure 11 we plot histograms of the number of servers in the 100 most popular ones that are a given radius away from the start set of URLs. We see a large number of servers at large radii, upto 10 and beyond. Millions of pages are within 10 links from almost any page on the web. Thus the focused crawler is doing non-trivial work in pursuing only certain paths and pruning others. (There is some probability that the distances we report are pessimistic, and much shorter paths to the best sites exist that were missed by the crawlers. Given we crawl best first on relevance, and that we tried multiple distinct seed sets, this should be rare.) Since our seed sets were collected using Alta Vista and HITS, this result also establishes the need to explore out aggressively from keyword-based and limited-radius search for resources.

5.6 Page rating: anecdotes

The primary output of the popularity rating algorithm is an ordered list of resource pages which link to a large number of pages relevant to the topic. These are similar to *authoritative sites* in Kleinberg's HITS system [17], with some differences. First, we have no clear distinction between hubs and authorities, indeed, the best pages are often both. Second, since we restrict the subgraph to highly relevant nodes only, our hubs tend to be topically very pure. Nothing short of human judgement is adequate for evaluating the rating algorithm; we strongly encourage the reader to visit the following URL's found by our system.

Cycling: `http://www.truesport.com/Bike/Links.htm`
`http://reality.sgi.com/employees/billh_hampton/jrvs/links.html`
`http://www.acs.ucalgary.ca/~bentley/mark_links.html`
`http://www.cascade.org/links.html`
`http://www.bikeride.com/links/road_racing.asp`
`http://www.htcomp.net/gladu/'drome/`
`http://www.tbra.org/links.shtml`

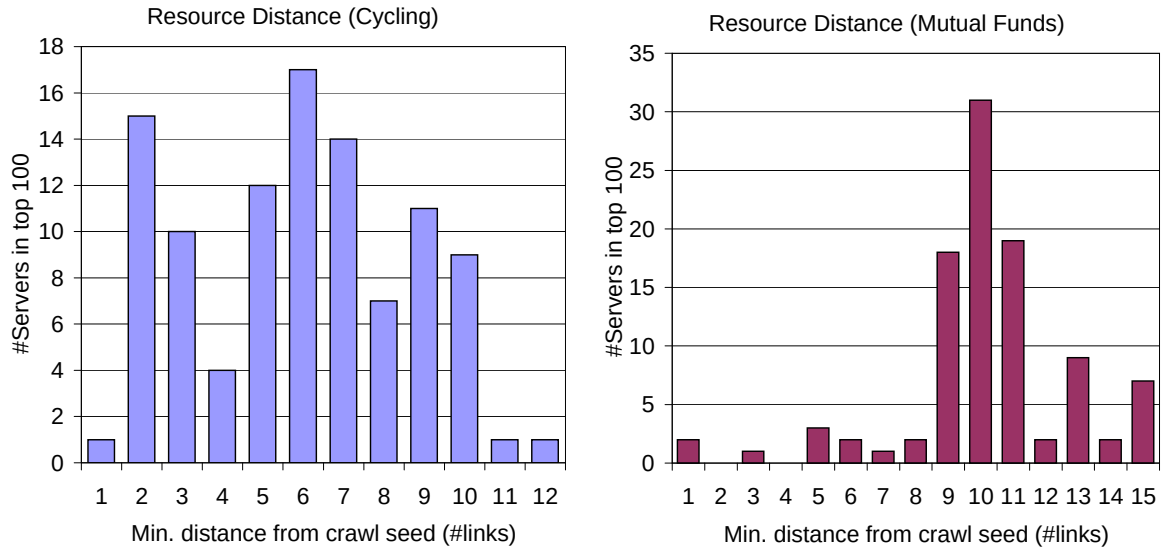

Figure 11: Distance in number of links from the seed set to the 100 most popular sites on cycling and mutual funds. The peak around 10 links for mutual funds is because great hubs were found around that distance.

http://www.islandnet.com/~ngs/SVCyclingLinks.html
http://www.novit.no/dahls/Cycling/hotlist.html
http://members.aol.com/velodromes/MajorTaylor/links.htm
http://www.nashville.com/~mbc/mbc.html
http://www.bikindex.com/bi/races.asp
http://www.louisvillebicycleclub.org/links.htm
http://world.std.com/~nebikclb/misc/netresources.html
http://crisny.org/not-for-profit/cbrc/links.htm
http://members.aol.com/velodromes/index.htm

HIV/AIDS: http://www.stopaids.org/Otherorgs.html
http://www-hsl.mcmaster.ca/tomflem/aids.html
http://www.ahandyguide.com/cat1/a/a66.htm
http://www.iohk.com/UserPages/mlau/aidsinfo.html
http://daphne.palomar.edu/library/subjects/aidslist.htm
http://www.us.unaids.org/highband/link.html
http://www.ashastd.org/links/hivlinks.html
http://www.hivresourcegroup.org/spd.htm
http://allpaths.com/rainbow/aids.htm
http://www.qrd.org/qrd/aids/
http://www.teleport.com/~celinec/aids.shtml
http://www.aids.wustl.edu/aids/inet.html
http://virology.science.org/aids.html
http://www.metrokc.gov/health/apu/links.htm
http://www.sfaf.org/links.html
http://www.aaas.org/science/aidslink.htm

We only presented a small number of our top-rated pages; the list continues into thousands of pages. Spot checking failed to reveal irrelevant pages up to the first few hundred links. We were impressed that we could find over three thousand relevant sites within only two hours of focused crawling per topic using a desktop PC and starting with a few dozen URLs. The system did not need to consult any additional large

keyword or link indices of the web, such as Alta Vista or Inktomi. Furthermore, almost half of our crawler's effort was useful from the point of view of the topics of interest.

5.7 Effect of page rating on crawling

For a focused crawler, the purpose of rating pages as valuable topical resources is not only for search and browsing, as in HITS, but also to further enhance the crawling process. It sometimes happens that a very relevant page is abandoned after mis-classification, for example, when the page has many image-maps and very little text, and/or the statistical classifier makes a mistake. After running the quality rating algorithm, it is quite easy to ask the following question:

> For the best URLs found so far, are there neighbors on a distinct site which have not yet been visited, and which are likely to be ignored because they have a poor priority?

For example, we can ask this of the top URL in the list above:

```
select url, relevance from arc.doc where oid in
  (select oid_dst from arc.link
   where oid_src = (select oid from arc.doc
                    where url = 'http://www.stopaids.org/Otherorgs.html')
   and sid_src <> sid_dst)
and num_tries = 0
```

This gives us a few good leads; collecting also the results from repeating the same query with the second URL in the list yields the following unvisited pages (that the reader is encouraged to visit):

```
http://www.planetout.com
http://gpawww.who.ch/gpahome.htm
http://www.actupny.org
http://www.gaypoz.com
http://www.users.dircon.co.uk/~eking/index.htm
http://aids.uspto.gov/AIDS/access/browse.html
http://www.aidsinfonyc.org
http://www.medibolics.com/nelson/index.htm
```

All of these turned out to be relevant and worth crawling. We can now update the visit priority of these neglected neighbors to, say, the maximum possible value and restart the crawl. This process can be automated to run interspersed with normal crawling activity.

5.8 Summary

We have presented evidence in this section that focused crawling is capable of steadily collecting relevant resources and identifying popular, high-content sites from the crawl, as well as regions of high relevance, to guide itself. It is robust to different starting conditions, and finds good resources that are quite far from its starting point. In comparison, standard crawlers get quickly lost in the noise, even when starting from the same URLs. We end this section with two obervations that come out of all these measurements:

- The web graph is *rapidly mixing* w.r.t. topics: random links lead to random topics within an extremely short radius.
- At the same time, there exist long paths and large subgraphs where topical coherence persists.

These observations are not necessarily contradictory, and this is exactly what makes focused crawling worth doing.

6 Related work

Our work is inspired by two main areas of previous work: information filtering agents and connectivity-based popularity estimation techniques. We review prior work in these areas and related areas of machine learning for text and hypertext.

6.1 Information filtering agents

The traditional use of machine learning and classification systems has been in the area of Information Filtering and Machine Learning. A great deal of early research exists in this area. WebWatcher, a learning agent developed at CMU [16] guides a user within a fixed collection by monitoring the user's interest profile. Pazzani et al experimented with a variety of classifiers to learn a two-way (hotlist and coldlist) classification problem [25]. Their experiments were limited to one user's judgement and only one website. Ackerman et al describe similar techniques [1]. In contrast to our technique, new pages are acquired in their system by extracting features that discriminate the hotlist from the coldlist and using these features for posing keyword queries to standard web crawlers such as Lycos[9]. In the context of query refinement, two-way interactive classifiers have been used for relevance feedback. None of these systems deal with filtering at the data acquisition level, and for a large taxonomy.

Early web crawlers simply followed every link acquiring more and more pages. Crawling techniques have grown more sophisticated, however, and more intelligence has been used in crawlers and other internet agents [13]. To our knowledge the earliest example of using relevance information to direct a web crawl is the *fish* [4] system. The Fish system starts from a *seed* set (one or a small number of pages relevant to a query) and collects more pages by crawling; the pages best matching the original key-word query are given priority in terms of following their outlinks. The original *WebCrawler* algorithm (described in [13, Chapter 4]) used a similar strategy, but starting with the result of a key-word search on a web-document index. Also the ranking of outlinks was done based on HTML *href* anchor text rather than document text, thus improving speed by eliminating the need to actually fetch pages in many cases. This kind of approach appears to have been refined in the work described in [12]. In addition to the web agents described in [13], there has been some interesting work at the University of Washington [14]. Of that work, the closest in spirit to ours appears to be the *Ahoy!* personal web page finder[27] (`http://www.cs.washington.edu/research/ahoy`) which uses the "Dynamic Reference Sifting" (DRS) architecture. Ahoy! is tailored for a specific kind of content: home pages. The focused crawler can be configured to crawl any topic given suitable examples.

SPHINX [22] is a Java toolkit and interactive development environment for Web crawlers. It is geared towards rapid prototyping of lightweight, client-level crawlers that are Web-site-specific, personally customized, and relocatable. A key concept in SPHINX is that of a generic "classifier" plug-in, which annotates pages and links with various types of metadata, depending on the application. Their notion of a "classifier" appears to be very broad, including topic classifiers as well as syntacting meta-data extractors. One example of a classifier they built was a plug-in that categorizes Alta Vista search responses into Yahoo! categories, similar in spirit to earlier work by Chekuri et al [11]. No mention is made of actually using the relevance or popularity ratings to actually direct the web crawl as we do. However, it should be possible, after adding support to SPHINX for statistical aggregate queries on a carefully organized crawl database, to implement a focused crawler on top of the SPHINX framework.

6.2 Connectivity-based popularity/authority estimation

Several techniques have been recently reported for finding popular web pages using connectivity: PageRank by Brin and Page [5], HITS (Hyperlink Induced Topic Search) by Kleinberg [17], CLEVER[7, 8, 9], and Topic Distillation by Bharat and Henzinger [3]. All these systems are built on top of a crawl and index of the web.

Page and Brin [24, 5] describes a technique that performs a query-independent quality ranking of all pages on the web. The system allows users to execute a boolean key-word query and documents matching the query

[9] http://www.lycos.com

are returned sorted by this rank, called their *Page Rank*. The ranking algorithm is described as simulating a random walk over the web taken by a web surfer. Assuming that each node (page) is equally likely as a starting point for the walk, the steady-state probability for the surfer being at any node is calculated based on a Markov-chain model and the pages are ranked by these probability values. Because this random walk uses the link structure, it is hoped that these steady-state probabilities capture the endorsement implicit in hyperlinks.

Rather than using a query-independent ranking, the HITS system [17] first obtains the result of a boolean key-word search applied to an index of the entire web. Then, starting with the first few hundred matching results, it performs a "graph expansion" acquiring every page one link distance away from any page in this initial set. This includes both pages pointed to by pages in the initial set as well as pages pointing to those initial-set pages. (The latter is possible because of the availability of a whole-web index.) Having obtained this graph, HITS then performs an iterative computation (described here in Section 3.2) on it to obtain two ranking scores for the pages. The pages are then returned in two rank-sorted lists corresponding to these two scores. HITS has a very clean graph model of the web. To deal with the far-from-ideal vagaries of the web, enhancements such as pruning the graph expansion, weighting the links using relevance, match between the query and anchor text, etc., have been proposed [3, 7, 8, 9]. These techniques are most effective for *broad* queries. Bharat and Henzinger recognized this explicitly by coining the term *topic distillation* for the process, and used document similarity in the vector space model to control undesirable topic generalization.

The focused crawler is different in a number of ways. It has no apriori radius cut-off for exploration, because it uses a general machine learning program to guide itself. It can also use earlier hub rating techniques for guidance. Thus the selection of relevant, high quality pages happens directly as a data acquisition step, not as post-processing or response to a query. From our experiments, it also appears that the focused crawler has a robust, high *harvest rate*: a large fraction of the data acquired is useful. It is therefore possible to build focused crawls with no permanent dependence on a regular crawler and search engine.

7 Conclusion

Generic crawlers and search engines are like public libraries: they try to cater to everyone, and do not specialize in specific areas. Web users are increasingly feeling a need to move away from public libraries into highly specialized and filtered university research libraries where they can explore their interest in depth [19, 28]. Unlike public libraries, web libraries have little excuse not to specialize, because it is just a matter of locating and linking to resources. Because a specialized library is relatively small, we would expect to locate and maintain these resources without crawling the web at large.

The focused crawler is a system that learns the specialization from examples, and then explores the web, guided by a relevance and popularity rating mechanism. It filters at the data-acquisition level, rather than as a post-processing step. Retrieving only relevant pages from the web is a non-trivial endeavor, because, although most good pages have links to other good pages, giving rise to good chains of links from one good page to the other, a large number of links in good pages point at irrelevant information: the whole of the web lies only a few links away from any page.

Our system selects work very carefully from the crawl frontier. A consequence of the resulting efficiency is that it is feasible to crawl to a greater depth (along link chains) than would otherwise be possible. This may result in the discovery of some high-quality information resources that might have otherwise been overlooked. As Marchiori [21] has noted, the quality of such resources may not always be strongly related to simple link-based popularity. The focused crawler has a high harvest rate: it acquires relevant pages a large fraction of the time. It has also proven to be insensitive to the crawl starting points in a number of test crawls. It is a large application built on top of a relational database, which is used for mining the collected graph, driving the crawl, and monitoring progress through ad-hoc SQL queries.

A number of questions about the web arise from our research. If we look at the acquisition rate charts, we see that the average relevance seems to fluctuate close to 40%. An interesting question is: how does this value change for different choices of subtrees for our topic? Obviously, choosing the top node makes all pages relevant and therefore the average relevance will be 100%. Is there a way of predicting this average as

a function of certain properties of the web and the structure of the taxonomy tree?

Another issue to research is the relationships between topics. While crawling topics described in this paper, we found a lot of anecdotal evidence that bicycle pages not only refer a lot to other bicycle pages, but also refer significantly more than one might expect to red-cross and first-aid pages. Similarly, HIV/AIDS pages often don't directly refer to HIV/AIDS pages of some hospital, but refer to the hospital home page (which would is the classified as *general health*) instead. Discovering these kinds of relationships will give interesting insights in the way the web is structured.

Acknowledgements: We thank Tom Mitchell, Dan Oblinger and Steve Gates for helpful discussions, Myron Flickner for generously contributing disks and computers, David Gibson for contributing code to the Java user interface, and Sunita Sarawagi, Amit Somani and Kiran Mehta for advice with DB2/UDB.

References

[1] M. Ackerman, D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. Kim, R. Klefstad, C. Lowe, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani, D. Semler, B. Starr, and P. Yap. Learning probabilistic user profiles: Applications to finding interesting web sites, notifying users of relevant changes to web pages, and locating grant opportunities. *AI Magazine*, 18(2):47–56, 1997. Online at `http://www.ics.uci.edu/~pazzani/Publications/AI-MAG.pdf`.

[2] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://www7.scu.edu.au/programme/fullpapers/1937/com1937.htm`; also see an update at `http://www.research.digital.com/SRC/whatsnew/sem.html`.

[3] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR Conference on Research and Development in Information Retrieval*, volume 21. ACM, 1998. Online at `ftp://ftp.digital.com/pub/DEC/SRC/publications/monika/sigir98.pdf`.

[4] P. D. Bra and R. Post. Information retrieval in the world-wide web: Making client-based searching feasible. In *Proceedings of the First International World Wide Web Conference*, Geneva, Switzerland, 1994.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW Conference*, volume 7, 1998. Online at `http://google.stanford.edu/~backrub/google.html`.

[6] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal*, Aug. 1998. Invited paper.

[7] 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 7th World-wide web conference (WWW7)*, 1998. Online at `http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html` and at `http://www.almaden.ibm.com/cs/people/pragh/www98/438.html`.

[8] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Topic distillation and spectral filtering. *AI Review*, to appear, 1998. Also IBM Research Report No. RJ-10127.

[9] S. Chakrabarti, B. Dom, D. Gibson, S. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *ACM SIGIR workshop on Hypertext Information Retrieval on the Web*, 1998.

[10] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*. ACM, 1998. Online at `http://www.cs.berkeley.edu/~soumen/sigmod98.ps`.

[11] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Sixth World Wide Web Conference*, San Jose, CA, 1996.

[12] H. Chen, Y.-M. Chung, M. Ramsey, and C. C. Yang. A smart itsy bitsy spider for the web. *J. Am. Soc. Inf. Sci.*, 49(7):604–618, 1998.

[13] F.-C. Cheong. *Internet Agents: Spiders, Wanderers, Brokers and Bots*. New Riders Publishing, Indianapolis, Indiana, 1996. ISBN: 1-56205-463-5.

[14] O. Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. In *Proceedings of AAAI-96*, 1996.

[15] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science*, 280:95–97, Apr. 1998.

[16] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the web. In *IJCAI*, Aug. 1997. Online at `http://www.cs.cmu.edu/~webwatcher/ijcai97.ps`.

[17] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1998. Also appears as IBM Research Report RJ 10076(91892), May 1997, and online at `http://www.cs.cornell.edu/home/kleinber/auth.ps`.

[18] R. Larson. Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace. In *Ann. Meeting of the American Soc. Info. Sci.*, 1996. Online at `http://sherlock.berkeley.edu/asis96/asis96.html`.

[19] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280:98–100, April 1998.

[20] S. Macskassy, A. Banerjee, B. Davidson, and H. Hirsh. Human performance on clustering web pages: A performance study. In *Knowledge Discovery and Data Mining*, volume 4, pages 264–268, 1998.

[21] M. Marchiori. The quest for correct information on the web: Hyper search engines. In *Sixth International World Wide Web Conference*, Santa Clara, Apr. 1997.

[22] R. Miller and K. Bharat. SPHINX: A framework for creating personal, site-specific web crawlers. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998.

[23] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 1999. Online at `http://www.cs.cmu.edu/~knigam/papers/emcat-mlj99.ps.gz`.

[24] L. Page. PageRank: Bringing order to the web. Stanford Digital Libraries working paper 1997-0072, Stanford University, 1997. Online at `http://www-pcd.stanford.edu/~page/papers/pagerank/index.htm`.

[25] M. Pazzani, L. Nguyen, and S. Mantik. Learning from hotlists and coldlists: Towards a www information filtering and seeking agent. In *Seventh International Conference on Tools with Artificial Intelligence*, 1995. Online at `http://www.ics.uci.edu/~pazzani/Publications/Coldlist.pdf`.

[26] S. Ross. *Introduction to probability models*. Academic Press, 1993.

[27] J. Shakes, M. Langheinrich, and O. Etzioni. Dynamic reference sifting: a case study in the homepage domain. In *Proceedings of the 6th World-Wide Web Conference (WWW6)*, 1997.

[28] J. White. Tricks of the agent trade: General Magic conjures PDA agents. *Internet World*, 7(5):67–76, 1996.