Focused crawling: a new approach to
topic-specific Web resource discovery

Soumen Chakrabarti a, Martin van den Berg b, Byron Dom c

a Computer Science and Engineering, Indian Institute of Technology, Bombay, 400076, India
b FX Palo Alto Laboratory, 3400 Hillview Ave, Bldg 4, Palo Alto, CA 94304, USA
c IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120, USA

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 resource discovery system called a Focused Crawler. The goal of a focused crawler is to selectively seek out pages that are relevant to a predefined set of topics. The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible Web 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, and 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.

To achieve such goal-directed crawling, we designed two hypertext mining programs that guide our crawler: a classifier that evaluates the relevance of a hypertext document with respect to the focus topics, and a distiller that identifies hypertext nodes that are great access points to many relevant pages within a few links. 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.

Keywords: Web resource discovery; Classification; Categorization; Topic distillation

1. Introduction

The World-Wide Web, having over 350 million pages, continues to grow rapidly at a million pages per day [2]. About 600 GB of text changes every month [19]. Such growth and flux poses basic limits of scale for today’s generic crawlers and search engines. At the time of writing, Alta Vista’s crawler called the Scooter, runs on a 1.5 GB memory, 30 GB RAID disk, 4 × 533 MHz 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, 2× 533 MHz AlphaServer 4100-5/300. (The query engine is even more impressive, but is not relevant to our discussion.) Other giant Web crawlers use similar fire-power, although in somewhat different forms, e.g., Inktomi uses a cluster of hundreds of Sun Sparc workstations with 75 GB of RAM and over 1 TB of spinning disk, and it crawls over 10 million pages a day.

In spite of these heroic efforts with high-end multiprocessors and exquisitely crafted crawling software, the largest crawls cover only 30–40% of the Web, and refreshes take weeks to a month [2,22]. The overwhelming engineering challenges are in part due to the one-size-fits-all philosophy: Alta Vista and Inktomi try to cater to every possible query that might be made on the Web. Although such services are invaluable for their broad coverage, the resulting diversity of content often snare 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. The growth of the Web matters little to a physicist if at most a few dozen pages dealing with quantum electrodynamics are added or updated per week. Seasoned users also rarely roam aimlessly; they have bookmarked sites important to them, and their primary need is to expand and maintain a community around these examples while preserving the quality.

We argue that a giant, all-purpose crawl is neither necessary nor sufficient for this purpose. In our experience (Section 2), keyword queries cannot naturally locate resources relevant to specific topics. It is also unreasonable to have to first crawl and index 350 million pages in order to distill fifty good resources related to quantum electrodynamics! 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 and further explore relevant regions of the Web. It might be argued that a central crawl amortizes work across multiple topics. But our results (Section 4) suggest that topical Web exploration is efficient enough for distributed deployment.

**Our contributions:** In this paper, we describe a Focused Crawler which seeks, acquires, indexes, and maintains pages on a specific set of topics that represent a relatively narrow segment of the Web. It 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. The focused crawler is guided by a classifier which learns to recognize relevance from examples embedded in a topic taxonomy, and a distiller which identifies topical vantage points on the Web. We describe the architecture in Section 3 and our experiences in Section 4.

Eventually, our goal is to impose sufficient topical structure on the Web so that powerful semi-structured query, analysis, and discovery are enabled. Here are some compelling examples:

**Discovering linkage sociology:** Is there a hyperlink between the Web page of a speed trap (traffic radar) maker and an auto insurance company? Apart from other bicycling pages, what topics are prominent in the neighborhood of bicycling pages? (First aid is one answer found by our system.)

**Locating specialty sites:** Getting isolated pages, rather than comprehensive sites, is a common problem with Web search. Now we can order sites according to the density of relevant pages found there. E.g., we can find the top five sites specializing in mountain biking.

**Semi-supervised learning:** Human-supervised topic learning yields very high-quality filtering, but needs labor-intensive training. Finding specialty sites can quickly generate large amounts of additional training data with little effort.

**Detecting community culture:** Simple statistics about the link graph reveal important informa-
tion about the community of the focused topic, e.g., whether it is competitive or collaborative (Section 4), the typical time taken by a good resource to become popular, etc.

**Estimating community timescales:** 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 the Web ontologists at Yahoo! or The Mining Company.

There is much awareness that for serious Web users, focused *portholes* are more useful than generic *portals*：“The most interesting trend is the growing sense of natural limits, a recognition that covering a single galaxy can be more practical — and useful — than trying to cover the entire universe” [16]. A focused crawler is an example-driven automatic porthole-generator. In a companion paper [8] we have proposed new HTTP infrastructure to support bidirectional hyperlinks to facilitate exploration of fine-grained communities. We feel that the ability to focus on a topical subgraph of the Web, as in this paper, together with the ability to browse communities within that subgraph, will lead to significantly improved Web resource discovery.

2. Focused crawler administration

Central to a focused crawler is a canonical topic taxonomy with examples. To run a specific instance, initial human input has to be provided in two forms. The user has to select and/or refine specific topic nodes in the taxonomy, and may also need to provide additional example URLs which serve as starting points for the crawl. In this section we give a user’s view of the system.

2.1. Operation synopsis

**Canonical taxonomy creation:** When the system is built, the classifier is pre-trained with a canonical taxonomy (such as Yahoo!, The Open Directory Project, The Virtual Library or The Mining Co.) and a corresponding set of examples. The canonical (coarse-grained) classification tree is part of the initial system.

**Example collection:** The user collects URLs that are examples of her interest. These are submitted to the system, e.g., by importing her bookmarks file.

**Taxonomy selection and refinement:** The system proposes the most common classes where the examples fit best. The user can choose and mark some of these classes as *good*. Sometimes, the user may find the taxonomy too coarse, and refine some categories and move documents from one category to another.

**Interactive exploration:** The system also proposes additional URLs in a small neighborhood of the examples, that appear to be similar to the examples. (This can be regarded as a slow-speed, interactive startup phase.) The user may inspect and include some of these as examples. The steps thus far are illustrated in Fig. 1a.

**Training:** The classifier integrates the refinements made by the user into its statistical class models.

**Resource discovery:** At this stage the system is ready to perform resource discovery as described in the rest of the paper.

**Distillation:** Intermittently and/or concurrently, the system runs a topic distillation algorithm to identify pages containing large numbers of relevant resource links, called *hubs*. The (re)visit priorities of these pages and immediate neighbors are raised.

**Feedback:** Typically, the user inspects the system regularly. The system reports the most popular sites and resource lists, and the user can give feedback by marking them as useful or not. This feedback goes back to the classifier and distiller.

The user collects examples by browsing. The applet shown in Fig. 1a monitors the page being rendered by the browser. Using the *Classify* menu, the user can make the classifier route the current page to the few best matching nodes in the category tree, marking all nodes on the way. After sufficient browsing, 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*. These are shown highlighted in the tree view. E.g., for the topic of “recreational bicycling,” two subtrees were found to be good choices. One (/Recreation/Sports/Cycling) is shown.
The other was /Business/Companies/Sports/Cycling. 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 [5,8]. Any pages thus found useful can also be added to the examples by dragging and dropping.

Sometimes the user may 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 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 [5]. 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.

At this stage, the focused crawler can be started. It is a complex system 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 maintains a plot of page relevance against time. In Fig. 1b, 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.

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.

2.2. Justification and discussion

A different design is conceivable in which keyword search is used to locate an initial set of pages (using a giant crawl and index), expand this graph to a limited radius and then look for popular sites in the expanded graph using weighted degree measures [25,31,4,21,6,3]. This approach was tried as a semi-automatic means to build a taxonomy like Yahoo!. For 966 topics picked from Yahoo!, keyword queries were constructed manually. E.g., the query for the topic /Business/Companies/Electronics/PowerSupply was +"power supply" +"switch mode" smps -multiprocessor +" uninterrupted power supply" ups -parcel. Typically, several query refinements were needed to match the quality of Yahoo! in blind user tests. The resulting queries were complex (as above) compared to the average Alta Vista query [29]. The above experiment used an average of 7.03 query terms and 4.34 operators (+-*); an average Alta Vista query has only 2.35 terms and 0.41 operators. Query construction is not a one-time investment, because as pages on the topic are discovered, their additional vocabulary must be folded in manually into the query for continued discovery.

Yet another design is possible in which the focused crawler only uses the examples found by the user, but does not use a taxonomy with the pre-packaged examples. E.g., we can set up a simple two-class learning problem (relevant/irrelevant) for each focus topic. However, we have a few reasons to believe that our approach is more promising.

Better modeling of the negative class: In the two-class learning of text, characterization of the negative class (e.g. “a page 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 merely a mental artifice, but it also affects the accuracy of learning algorithms significantly [26], because commonly used statistical models have large estimation errors on the diverse negative class.

Reuse of classifier training effort: Learning to
recognize text documents is made difficult by the large dimensionality and consequent sparsity. It may be burdensome for every user to prepare enough sample documents to have an adequate number of positive and negative examples for learning her interest profile. The work of mapping the user’s interest onto a predefined set of categories, refining them when needed, will usually be significantly less than finding an adequate number of positive and negative examples. One may even envisage that a standards organization will design many “backbone taxonomies” for smaller groups to fill in and refine. A similar procedure is espoused for maintaining many Web directories.

**Discovery of 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 commingle. It is able to do this because it can classify the Web pages it encounters 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 the mutual funds example, it is better to broaden the set of categories to those that provide 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.

**3. System architecture**

The focused crawler has three main components: a classifier which makes relevance judgments on pages crawled to decide on link expansion, a distiller which determines a measure of centrality of crawled pages to determine visit priorities, and a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller. A block diagram is shown in Fig. 2. Here we briefly outline the basic processes. In subsequent work we have redesigned the modules on top of a relational database and efficiently integrated them [9].

Based on the discussion so far, we can summarize the role of the focused crawler in the following terms. We are given a directed hypertext graph $G$ whose nodes are physically distributed. In this paper, $G$ is the Web. There is a cost for visiting any vertex (Web page) of $G$. There is also a tree-shaped hierarchical topic directory $C$ such as Yahoo!. Each topic node $c \in C$ refers to some pages in $G$ as examples. We denote the examples associated with topic $c$ as $D(c)$. These pages can be preprocessed as desired by the system. The user’s interest is characterized by a subset of topics $C^* \subseteq C$ that is marked good. No good topic is an ancestor of another good topic. Ancestors of good topics are called path topics. Given a Web page $q$, a measure of relevance $R_c(q)$ of $q$ with respect to $C^*$, together with a method for computing it, must be specified to the system. $C^*$ will be omitted if clear from the context. In this paper, we will use a probability measure $0 \leq R(q) \leq 1$. By definition, $R_{root}(q) = 1 \forall q$. If \{ $c_i$ \} are children of $c_0$, then $\sum_{c_i} R_{c_i}(q) = R_{c_0}(q)$. The system starts by visiting all pages in $D(C^*)$. In each step, the system can inspect its current set $V$ of visited pages and then choose to visit an unvisited page from the crawl frontier, corresponding to a hyperlink on one or more visited pages. Informally, the goal is to visit as many relevant pages and as few irrelevant pages as possible, i.e., to maximize average relevance. Therefore we seek to find $V \supseteq D(C^*)$ where $V$ is reachable from $D(C^*)$ such that $\sum_{V} R(v)/|V|$ is maximized.

Our formulation would pose a hopeless problem if pages of all topics were finely dispersed all over the Web. However, this is not likely to be the case. Citations signify deliberate judgment by the page author. Although some fraction of citations are noisy, most
citations are to semantically related material. Thus the relevance of a page is a reasonable indicator of the relevance of its neighbors, although the reliability of this rule falls off rapidly with increasing radius on an average. This explains our use of the classifier. Secondly, multiple citations from a single document are likely to cite semantically related documents as well. This is why the distiller is used to identify pages with large numbers of links to relevant pages.

3.1. Classification

Relevance is enforced on the focused crawler using a hypertext classifier [5,7]. We assume that the category taxonomy induces a hierarchical partition on Web documents. (In real life, documents are often judged to belong to multiple categories. We plan to extend our model in future work.) 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 root category is 1. In general Pr[c|d] = Pr[parent(c)|d]Pr[c|d, parent(c)]; this can be recursed using chain rule. Using Bayes rule we can write:

Pr[c|d, parent(c)] = Pr[c|parent(c)]Pr[d|c] / \sum_{c'\text{parent}(c')=parent(c)} Pr[c'|d]

where the sum ranges over all siblings c' of c.

To find Pr[d|c] we need a model for document generation. Pr[c|parent(c)] define the prior distribution of documents. In our generation model, the page generator first decides the topic on which to write the document d by using these probabilities to pick a leaf node c*. Each class, in particular c*, has a die with as many faces as the number of unique words (terms, tokens) in the universe. Face t turns up with probability \( \frac{t}{n} \). The generator picks an arbitrary length \( n(d) \) for the document. Then it repeatedly flips the die for c*, and writes out the term corresponding to the face that turns up. A document is thus seen as a bag of words, without order information or inter-term correlation. If term t occurs \( n(t) \) times, then: Pr[d|c] = \( \prod_{t} \theta(c, t)^{n(t)} \). In spite of its simplicity, this model has been very successful. During crawling, the task is the reverse of generation: given a document, we seek to find the best leaf class c*. Two modes of focusing are possible with the classifier.

**Hard focus rule:** While fetching a document d, the above formulation is used to find the leaf node c* with the highest probability. If some ancestor of c* has been marked good we allow future visitation of URLs found on d, otherwise the crawl is pruned at d.

**Soft focus rule:** The probability that a page is relevant to the focused crawl is \( R(d) = \sum_{\text{good}(c)} Pr[c|d] \), because a good node is never the ancestor of another. We do not eliminate any page a priori, but guess that the priority of visiting each ancestor of the current page d is the relevance of d. In case

![Fig. 2. Block diagram of the focused crawler showing how the crawler, classifier and distiller are integrated.](image-url)
of multiple paths leading to a page we take the maximum of their relevance. When a neighbor is actually visited its score is updated.

3.2. Distillation

Relevance is not the only attribute used to evaluate a page while crawling. A long essay very relevant to the topic but without links is only a finishing point in the crawl. A good strategy for the crawler is to identify hubs: pages that are almost exclusively a collection of links to authoritative resources that are relevant to the topic.

Social network analysis [31] is concerned with the properties of graphs formed between entities such as people, organizations, papers, etc., through coauthoring, citations, mentoring, paying, telephoning, infecting, etc. Prestige is an important attribute of nodes in a social network, especially in the context of academic papers and Web documents. The number of citations to paper \( u \) is a reasonable but crude measure of the prestige \( p(u) \). A better measure is weighted citations, or the total prestige of papers that cite a paper. This notion is circular but can be resolved by an iterative eigen computation to find the fixpoint of \( p = Ep \), where \( E \) is the directed adjacency matrix, as described by Katz [20] in 1953 and adapted to the Web by Page et al. [4].

Mizruchi et al. [25] recognized that centrality in a social network can be disaggregated into derived and reflected centrality. They found two types of nodes: bridges which have high derived centrality, and hubs which link with good authorities and thereby have high reflected centrality. Kleinberg later exploited the same phenomenon on the Web to find hubs and authorities (bridges) [21]. Each node \( v \) has two corresponding scores, \( h(v) \) and \( a(v) \). Then the following iterations are repeated 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 the vectors \( h \) and \( a \) to unit length. This iteration embodies the circular definition that important hubs point to important authorities and vice versa.

For focused crawling, two important enhancements are needed: the edge weights must be carefully controlled and there should be a certain asymmetry in how we treat hubs and authorities. To appreciate the model that we will propose, observe that pages relevant to our interest refer to irrelevant pages and vice versa with appreciable frequency, owing to the diversity of Web authorship. Pages of all topics point to Netscape and Free Speech Online. Conversely, many hubs are multi-topic in nature, e.g., a published bookmark file pointing to sports car sites as well as photography sites.

We will not only have non-unit edge weight but differentiate the forward and backward edge weights into two different matrices \( E_F \) and \( E_B \). We propose that the weight \( E_F[u, v] \) of edge \( (u, v) \) be the probability that \( u \) linked to \( v \) because \( v \) was relevant to the topic, i.e., \( R(v) \). This has the effect of preventing leakage of prestige from relevant hubs to irrelevant authorities. Similarly, we propose that \( E_B[u, v] \) be set to \( R(u) \), to prevent a relevant authority from reflecting prestige to an irrelevant hub. Finally, we will use a relevance threshold \( \rho \) to include potential authorities into the graph, although for hubs we have no such requirement. We include between 10 and 20% of the most relevant nodes; our results were not sensitive to the precise choice in this range. The remaining steps follow:

1. Construct the edge set \( E \) using only those links that are between pages on different sites, with forward and backward edge weights as above.
2. Perform the iterations using the weighted edges. Always restrict the authority set using the relevance threshold.

3.3. Integration with the crawler

The crawler has one watchdog thread and many worker threads. The watchdog is in charge of checking out new work from the crawl frontier, which is stored on disk. New work is passed to workers using shared memory buffers. Workers save details of newly explored pages in private per-worker disk structures. In bulk-synchronous fashion, workers are stopped, and their results are collected and integrated into the central pool of work.

The classifier is invoked by each thread as it encounters a new page. The \( R \) value computed is part of the page result mentioned above. The central work pool is a priority queue implemented using the Berkeley DB B-tree storage manager.
soft crawling, candidate URLs are ordered using a
lexicographic combination

(numtries ascending, R descending),

where numtries is the number of times the crawler
has already tried to fetch the page, with or without
success. For hard crawling, the URLs that survive
are picked in increasing order of numtries; for
the same value of numtries the remaining order is
arbitrary.

The crawler also populates the link graph kept
on disk. Currently this consists of a forward and
backward edge list, stored using the hash access
method of Berkeley DB. Periodically, the crawler is
stopped and the distiller is executed. This generates
a number of top hubs to revisit. We also prepare to
visit unvisited pages cited by the top hubs.

In ongoing work [9] we have reimplemented the
system using a relational database to store the crawl
frontier and facilitate dynamically changing prioriti-
zation strategies, and concurrent activity between the
crawler, distiller and classifier. The integration also
facilitates crawl monitoring and diagnostics using
ad-hoc SQL queries.

4. Evaluation

In this section we will present our experiences
with focused crawling. There are many indicators
of the performance of a focused crawler. Relevance
(precision), coverage (recall) and quality of resource
discovery are some of them. We will measure preci-
sion and provide anecdotes on the quality of resource
discovery. It is extremely difficult to measure or 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 rea-
sonable manner in the case of the Web. We will
provide indirect evidence of robust coverage.

4.1. Experimental setup

The focused crawler is a C++ application run-
ning on a dual-processor 333 MHz Pentium-II PC
with 256 MB of RAM and SCSI disk. Our test ma-

URLs and started separate crawls. We compare the rate of acquisition of relevant pages between the two crawlers. This is an indirect indicator of coverage.

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

4.2. Rate of harvesting relevant pages

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!

It would be good to judge the relevance of the crawl by human inspection, even though it is subjective and inconsistent [23]. But this is not possible for the hundreds of thousands of pages our system crawled. Therefore we have to take recourse to running an automatic classifier over the collected pages. Specifically, we can use our classifier. It may appear that using the same classifier to guide the crawler and judge the relevance of crawled pages is flawed methodology, but it is not so. 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. We are evaluating not the classifier but the basic crawling heuristic that neighbors of highly relevant pages tend to be relevant.

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 a few dozen relevant URLs. These were collected by a keyword query at Alta Vista followed by traditional topic distillation and some screening by hand to eliminate irrelevant pages. In the unfocused case, the crawler fetches new URLs in pseudo-random order, and all out-links are registered for exploration. The pages are classified to find $R$, but no use is made of it except measurement. This will slow the crawl down a little. For this reason, and also because network load fluctuates greatly from experiment to experiment, in our results we present time not as wall-clock time, but as the number of URLs fetched so far.

The first column in Figs. 3 and 4 shows the results of unfocused crawls for bicycling and HIV/AIDS. The $x$-axis shows the number of pages acquired (as a representative of real time). The $y$-axis shows a moving average of $R(u)$, where $u$ 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.

![Fig. 3. Rate of relevant page acquisition with a standard unfocused crawl, a hard focused crawl, and a soft focused crawl on the topic of bicycling.](image-url)
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 tends to happen even if we help it in various ways, such as disabling such highly interconnected sites as Amazon.com.

In contrast, it is heartening to see in the second column of Figs. 3 and 4 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 the third column.

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.

Fig. 5 explains 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.

4.3. 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 URLs 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 are shown in Fig. 6.

We picked the two topics specifically because we wanted to study one co-operative community like bicycling and one competitive domain like invest-
Fig. 5. Distribution of relevance scores in the bicycling and HIV/AIDS crawls from the three crawlers.

Fig. 6. Overlap of URLs crawled by two soft focused crawlers starting from randomly sampled seed sets on bicycling 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 crawlers overlap against time shows generous visits 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 URLs fetched by each crawler side-by-side, then deriving the two sequences of IP-addresses visited, $S_1(t)$ and $S_2(t)$. 
Results were similar with other topics. Whereas these results do not imply perfect coverage, they do indicate that core topical communities are fairly coherent, and emerge naturally from the crawl independent of the starting set. It would be interesting to stress the robustness by starting from smaller and smaller URL samples.

4.4. 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 [21]). Then we measured the intersection of server IP-addresses in the top 25. We picked addresses rather than URLs 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 Fig. 7. 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.

4.5. 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 Fig. 9 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 distances from the start set, up to 10 links 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 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 [21], this result also establishes...
Fig. 8. Overlap of the 100 best rated servers crawled by two soft focused crawlers starting from randomly sampled seed sets on cycling and mutual funds.

Fig. 9. 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.

the need to explore out aggressively from keyword-based and limited-radius search for resources.

A glance at the two histograms exposes the cooperative and competitive nature of the two communities. Cycling organizations are inclusive and social. Hence good hubs (consequently, authorities) are found at a variety of link distances. In contrast quite some exploration was needed for mutual funds and investment until good hubs were found, at radius 8–9. The focused crawler is good at pointing out these features.

4.6. Distillation anecdotes

Many post-processing operations on a focused crawl may be useful, such as clustering, indexing,
etc. One more piece of evidence that a focused crawl is qualitatively better at resource discovery can be obtained by presenting the results of the distiller. Since we restrict the authority subgraph to only highly relevant nodes, 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 URLs found by our system and shown in Fig. 10 (verified to be accessible as of March 1, 1999).

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 an hour 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.

4.7. Effect of distillation on crawling

The purpose of distillation in a focused crawler is not only an end-goal, but also a further enhancement to the crawling process. It sometimes happens that a very relevant page is abandoned after misclassification, for example, when the page has many image-maps and very little text, and/or the statistical classifier makes a mistake. After running the distiller, it is quite easy to look for unvisited citations from the top hubs. E.g., performing this step with the HIV/AIDS hubs gives us the following unvisited URLs (that the reader is encouraged to visit):

http://www.planetout.com
http://www.aidslink.org
http://www.under.org
http://www.aids.gov
http://www.medibolics.com

Many 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.

4.8. 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 observations that come out of all these measurements:

- The Web graph is \emph{rapidly mixing} with respect to 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.

5. Related work

Traditionally, machine learning techniques have been used to design filtering agents. WebWatcher [18] and HotList and ColdList [27] are examples of such filtering programs. Ackerman et al describe similar techniques [1]. In contrast to our technique, new pages are acquired in some of these systems by first extracting features that discriminate the hotlist from the coldlist and then using these features for posing keyword queries to standard Web search engines. 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. Crawlers and agents have grown more sophisticated [11]. To our knowledge the earliest example of using a query to direct a limited Web crawl is the \emph{Fish Search} system [14]. Similar results are reported for the WebCrawler [11, chapter 4], Shark Search [17], and by Chen et al. [10]. The focused crawler is different in using a topic taxonomy, learning from example, and using graph distillation to track topical hubs.

Ordinary search engines and directories are called \emph{portals} or entry points into the Web. There is growing consensus that \emph{portholes} — sites that specialize in specific topics — are often more useful than portals. A few systems that gather specialized content have been very successful. Cho et al compare several crawl ordering schemes based on link degree, perceived prestige, and keyword matches on the Stanford University Web [12]. Terveen and Hill use similar techniques to discover related “clans” of Web pages [30]. \emph{Ahoy!} [15,28] is a homepage search service based on a crawler specially tuned to locate homepages. \emph{Cora} is a search engine for computer science research papers, based on a crawler trained to extract such papers from a given list of starting points at suitable department and universities. These are special cases of the general example- and topic-driven automatic Web exploration that we undertake.

Social networks have been analyzed for decades to find nodes with high prestige and reflected prestige [20,25,31]. Similar to PageRank [4], HITS [21], CLEVER [6], topic distillation [3] and link-based similarity search [13], we use social network analysis as a subroutine in our system. However, there are several important distinctions. Our distiller integrates topical content into the link graph model. PageRank has no notion of page content, and HITS and CLEVER explore the Web to a preset radius (typically, 1) from the keyword query response. All involve pre-crawling and indexing the Web. The focused crawler has no a priori radius cut-off for exploration, because it can use the classifier and distiller to guide itself. Thus the selection of relevant, high quality pages happens directly as a goal-directed data acquisition step, not as post-processing or response to a query.

6. Conclusion

Generic crawlers and search engines are like public libraries: they try to cater to everyone, and do not specialize in specific areas. Serious Web users are increasingly feeling a need for highly specialized and filtered 'university research libraries' where they can explore their interest in depth [16,22]. Unlike public libraries, Web libraries have little excuse not to specialize, because it is just a matter of locating and linking to resources.

We have demonstrated that goal-directed crawling is a powerful means for topical resource discovery. 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. 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 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 [24] has noted, the quality of such resources may not always be strongly related to simple link-based popularity.

A number of questions arise from our research. The harvest rate at the root is by definition 100%, and we have been seeing harvest rates of 30–40%. How does this depend on the specificity of the topic? At what specificity can focused crawls be sustained? Another issue to research is the sociology of citations between topics. While crawling topics described in this paper, we found a lot of anecdotal evidence that bicycle pages not only refer 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 other HIV/AIDS pages, but refer to hospital home pages, which are more general. Discovering these kinds of relationships will give interesting insights in the way the Web evolves.

Acknowledgements

We thank Tom Mitchell, Dan Oblinger and Steve Gates for helpful discussions; Myron Flickner for generously contributing disks and computers; David Gibson for helping with the Java user interface; and Sunita Sarawagi, Amit Somani and Kiran Mehta for advice with disk data structures.

References


