Focused Crawls, Tunneling, and Digital Libraries

Donna Bergmark, Carl Lagoze, and Alex Sbityakov

Cornell Digital Library Research Group

Abstract. Crawling the Web to build collections of documents related to pre-specified topics became an active area of research during the late 1990's, crawler technology having been developed for use by search engines. Now, Web crawling is being seriously considered as an important strategy for building large scale digital libraries. This paper covers some of the crawl technologies that might be exploited for collection building. For example, to make such collection-building crawls more effective, focused crawling was developed, in which the goal was to make a "best-first" crawl of the Web. We are using powerful crawler software to implement a focused crawl but use tunneling to overcome some of the limitations of a pure best-first approach. Tunneling has been described by others as not only prioritizing links from pages according to the page's relevance score, but also estimating the value of each link and prioritizing them as well. We add to this mix by devising a tunneling focused crawling strategy which evaluates the current crawl direction on the fly to determine when to terminate a tunneling activity. Results indicate that a combination of focused crawling and tunneling could be an effective tool for building digital libraries.

1 Introduction

What are the tools and techniques to build truly large scale digital libraries? Answering this question is crucial to the success of the National Science Digital Library (NSDL) project [1, 2]. Our goal in NSDL is to build what will quite possibly be the largest and most diverse digital library to date. To accomplish this goal, we will not only include resources from partner NSF projects, but incorporate much of the wealth of scientific and mathematical information available on the open Web.

Building a digital library of this breadth and scale requires innovative techniques. While many of the tasks could undoubtedly be undertaken with ample and expert human labor, the scale of such labor is also prohibitively expensive. As noted by Arms [3] the key to moving digital libraries from dependency on expensive human labor lies in the combination relatively inexpensive raw computing power and smart algorithms.

This paper describes one aspect of our work in this area; techniques to automatically build online collections of topic specific resources. This task is critical to finding and organizing the many needles in the vast Web haystack. In an earlier paper [4] we described initial results from our attempts to use focused

Web crawling to build collections from an established topic hierarchy. In this paper we focus on techniques to make that process more efficient, which becomes increasingly important with the increasing size of the Web (while computers do get faster, the scale of the problems always seem to increase at a greater rate!). We describe a technique called *tunneling* as a means of determining what are the best links to follow from a source page during a selective crawl to build collections. Our hypothesis is that by examining the patterns of document-to-collection correlations along Web link paths we can devise more efficient selective crawling techniques.

The paper is structured as follows. Part 2 provides background on our work on automated collection building. Part 3 describes general issues in focused Web crawling. Part 4, the heart of the paper, describes tunneling as a technique to improve the efficiency of focused Web crawls. Part 5 provides results of our Web characterization work to develop better tunneling techniques. We close with Part 6 that describes how our results might be applied and Part 7 with some related work.

2 Automated Collection Building

We start with a set of *centroids* which are constructed from the subjects in the given topic hierarchy, by a process described below. Each centroid is a weighted term vector describing a topic. During the crawl, each down-loaded document is tentatively classed with the nearest subject vector, with the correlation being the degree to which the document is considered to be in that collection. If the correlation is sufficiently high, then links from that HTML page are also followed.[1] Once the collections have been built, each of the items in a collection is represented by its URL and a figure of merit indicating the degree to which the item belongs to this collection.

The *quality* of the collection depends on the initial input (the topic hierarchy and centroids) and on the metric used to assign downloaded documents to collections. The *efficiency* of collection building depends on being able to do a *focused crawl* [5]. Focused crawling is discussed in the next section.

We build a number of collections at once for a particular subfield of technology (e.g. mathematics, nano-physics). We start with a topic hierarchy or subject index, and then leverage Google to return a few good documents on each subject, and construct a centroid from those. (The hits from Google are used *only* to build centroids; crawl seeds are independently selected, and include such hubs as `yahoo.com`.) Specifically, for each topic query to Google, the centroid is constructed from the first k hits returned in the search results. (This is done automatically, by using a program to submit the query to Google and then scraping the resulting response page for the hit URLs. We are considering replacing this ad hoc approach with the more formal SOAP API, recently announced by Google.) We limit k to the range of 4 through 7. In other words,

[1] If the link is to a .pdf or .ps file, those are also added to the same collection, with their parental correlation values.

topics for which Google finds fewer than four documents are discarded; search result hits after the first 7 are ignored.

Then the concatenated k pages are turned into term vectors. The term weights are basically the frequency of the word in the set of k pages, divided by the frequency of the word in pages downloaded for all the queries. This is the TF-IDF weighting from standard information retrieval [6, 7]. (One advantage to building many collections simultaneously is that a term's document frequency can be estimated from term occurrence in the other query sets. However, the centroids can be weighted by term frequency alone if the centroids are being built in parallel [8].)

The general success of our implementation of focused crawling depends heavily on the initial subject descriptors. We found that using the index term list from the Math Forum[2] worked well as a source of math subject descriptors, but that a curriculum outline for 1st and 2nd grade science failed because it contained too many broad and general terms. Finding the proper term specificity and helpful subject hierarchies will be crucial for automatic NSDL collection synthesis.

3 About Focused Crawling

The selective harvesting of interesting URLs from the Web became an intense research topic in the late 1990's, although there was some early work on topic-directed web crawling at Xerox Parc by Pirolli and Pitkow [9]. Chakrabarti *et al.*'s seminal paper [5] introduced focused crawling for collecting topic-specific web pages, and described its advantages over search engines. Both produce collections of topic-specific pages, but search engines must be prepared to answer any query and therefore use Web crawlers which do not distinguish between which pages to analyze or which links to follow. Rather, any and all newly discovered links are placed onto the *crawl frontier* to be downloaded when their turn comes. Focused crawling, on the other hand, attempts to order the URLs that have been discovered to do a "best first" crawl, rather than the search engine's "breadth-first" crawl.

"Best" has been defined in a couple of ways. Kleinberg's hub and authorities [10] and Brin and Page's Page rank [11] define "best" in terms of being often linked to. The other definition – the one on which we concentrate – is relevance to a particular topic. These definitions can be combined of course; link analysis can be used to not only find pages on a specific topic, but also authorities on that topic.

Focused crawling grew out of mid-90's text categorization work, assigning documents to categories. Then the HITS algorithm did focused searching by staying within certain communities to extract their topics of interest[12].

Since then, a number of focused crawlers have been developed and used for Web experimentation. Chakrabarti *et al.* implemented a focused crawler using

[2] `www.mathforum.org`.

off-the-shelf database and storage managers [13]. Rennie and McCallum [14] designed a focused crawler that attempted to crawl pages only of a certain type by using feedback during the crawl. Menczer *et al.* [15, 16] have done work since 1999 on designing and evaluating focused crawlers. Mukherjee's WTMS [17] reports being able to build a topic-based collection with high precision. [18] is a good summarization of results in focused crawling as of the end of 1999. An interesting technique for focusing crawls by using Context Graphs was introduced by [19] in 2000.

We have continued this work by experimenting with a focused crawler based on the powerful Atrax/Mercator crawler software [20, 21]. This crawler is written in Java, uses a configuration file in which many options for a crawl can be specified, and has base classes which are extensible. The extensibility supports many different flavors of crawling, including focused crawling. Our extensions have led to a focused crawler which builds a number of collections simultaneously and automatically, starting only with a set of topic descriptions.

Several assumptions lie behind the idea of focused crawling. One is that by following links from a page which is relevant to the topic, one is more likely to get to another relevant page. That is, the assumption is made that if two pages are linked to each other, they are likely to be on the same topic. One study [22] actually found that the likelihood of linked pages having similar textual content was high, if one considered random pairs of pages on the Web.

Another assumption is that anchor text describes the content of the page being pointed to. The problem is that anchor text is often too brief to contain much content. (Common anchor texts include "here", "this", "top", etc.)

Another assumption is that it is possible through page analysis (say, using TF-IDF) to determine whether that page relates to the topic of the collection being assembled.

The final assumption is that the link itself, the URL, contains information that can help focus the crawl. Some studies [23] found that the text in the URL string can contain important information. For example, if one is collecting physics documents about lasers, a URL that contains "sailing" is less likely to be relevant than a URL that contains the word "optical". Stop lists can be applied to URLs as well, such as not following "contents.html" pages. This is supported directly via Mercator's configuration file, which includes an optional filter on URL strings. Similarly, if the crawl has encountered many low-scoring pages from a particular server, URLs to this server could be marked as less desirable. It should also be noted that there is a Semantic Web initiative to type links [24]. Such additional information attached to the links could help focused crawlers be more effective.

In summary, the trick to focused crawling has so far been to order the crawl frontier so that links to probably relevant pages are followed before links to probably off-topic pages. This requires estimating the relevance of the page pointed to by the link, based on some of the assumptions listed above. Reasonably enough, this estimation can be fraught with error.

Prioritizing the frontier is computationally intensive and is thus opposed to our goal of increased crawling efficiency. Although Mercator does support prioritized queues, we chose instead to focus our crawl by using two knobs, *threshold* and *cutoff*.

The threshold is a number between zero and one which determines the correlation level above which we consider a document to be on-topic with respect to some centroid. The correlation score for a downloaded normalized document d is

$$\text{score} \quad = \quad \underset{\{c\}}{\text{argmax}} \left\{ \sqrt{\frac{\sum_i d_i^2 \cdot c_i^2}{||c_i||}} \right\} \tag{1}$$

where c is a centroid. Argmax says choose the cluster whose centroid maximizes the score. Weights of term i are d_i, c_i. This is the standard *cosine correlation*.

The cutoff is how far to crawl from a page whose value is above the threshold. Since Mercator can keep track of the number of pages in a path, it is possible to keep incrementing a counter on a particular crawl path until it exceeds the cutoff, at which point we terminate the crawl going in this direction.

Setting the threshold high encourages high precision collections. Setting the cutoff low limits the amount of irrelevant material one has to crawl through. Our experience has been that good settings for a focused crawl are threshold=0.35 and cutoff=0.

While this gets you collections that have a better-than-average chance of being relevant [4], threshold and cutoff are rather crude ways of focusing the crawl. Essentially we are saying that the "best links" are those from an on-topic page and from pages not too far from that last on-topic page. While this does help minimize the size of the crawl frontier, it might be too limiting in scope. Tunneling is one way to extend the scope of a focused crawl.

4 Adding Tunneling to Focused Crawling

Focused crawling, while quite efficient and effective does have some drawbacks. One is that it is not necessarily optimal to simply follow a "best-first" search, because it is sometimes necessary to go through several off-topic pages to get to the next relevant one. With some probability, the crawl should be allowed to follow a series of bad pages in order to get to a good one.

It is important here to recall our objective: to build collections of 25–50 URLs of expository pages on given subjects. Thus precision is not defined in terms of number of crawled pages, but in terms of rank. In other words, downloading and inspecting what amounts to trash does not hurt precision or impede effectiveness; the only impact is on efficiency. The need is to obtain a high-precision result within a reasonable timeframe.

Another application for tunneling is right at the start of the crawl. One does not necessarily start with on-topic seeds. In our case where we build several dozen collections at a time, the starting seed will certainly not apply equally well to all collections. In this case, tunneling is useful for getting to desirable parts of the Web.

Clearly, tunneling can improve the effectiveness of focused crawling by expanding its reach, and its efficiency by pruning paths which look hopeless. So, the main challenge now becomes how to decide when to stop tunneling, i.e. terminate the direction in which the crawl is proceeding.

To be more precise about tunneling, we propose the following definitions. A *nugget* is a Web document whose cosine correlation with at least one of the collection centroids is higher than some given threshold. Thus the "nugget-ness" of a document is represented by its correlation score. A *dud*, on the other hand, is a document that does not match any of the centroids very highly. A *path* is

Table 1. Tunneling definitions and path notation

Notation	Definition
0	A nugget, i.e. correlation $\geq .5$
X	A dud
0-0	A path of length 2
0-X-X-X-X-0	A path of length 6
0-X-X-0-X-0	Two paths linked by a common relative, one path being length 4, the other path length 3
	Partial Paths
X-...-0	X is a seed but not a nugget; we treat the seed as a "pseudo" nugget
0-X...$X_{>c}$	Terminated path because final dud's distance from some nugget is greater than or equal to the cutoff
0-X...$X_{<c}$	Incomplete path due to crawl's time being up
0 / \	A very brief crawl, consisting of two paths, one incomplete
0 X	The crawl begins at the root of the tree and proceeds downward

the sequence of pages and links going from one nugget to the next. The path *length* is 2 minus the number of duds in the path. A *crawl* is the tree consisting of all the paths, linked together in the obvious way. Table 1 illustrates these definitions and introduces some notation.

Our experimental setup was a collection of 26 topics in the area of Mathematics. An example topic is `basic linear algebra graphing equations`. An 8 hour crawl was performed to collect the 50 best documents on each topic. But in order to collect experimental data about tunneling, we set the threshold to 0.50 and the cutoff to 20 (to maximize the tunneling). The data we saved for each downloaded URL included:

- the downloaded URL
- the correlation of that URL's page with the nearest centroid
- the URL of the parent (the document which contained a link to this URL; only the seeds of the crawl had no parent)
- the "distance" of the downloaded URL from its nearest nugget ancestor

The *distance* is an integer saying how far from the most immediate nugget ancestor this node is. It is thus a function of a page's cosine correlation value and the parent's distance. The simplest distance metric is related to path-length:

$$\text{distance} = \begin{cases} 0 & \text{if this node is a nugget} \\ 1 + \text{parent's distance} & \text{otherwise} \end{cases} \quad (2)$$

Related to distance is *level*, which is simply 1 plus the parent's distance.

To more precisely characterize the benefits of tunneling, a tunneling crawl can be performed and then the results can be statistically analyzed. With this data in hand, we can gain insight into the structure of the crawled portion of the Web. Our path data has many properties: path length, sequence of correlation values, trends, signal to noise ratios, etc. By statistically analyzing the collected data, we stand a chance of empirically being able to answer the following questions:

- How far might you have to go from one nugget to get to the next? (The diameter of the Web would be the maximum.)
- Given that a page is downloaded with correlation X, how well does that predict that one of its links is a nugget? Or does it instead depend on what led up to this dud (i.e. are we in a bad part of the Web and will never get to a nugget even though the correlation is as high as X)?
- How good is the correlation score for predicting whether it will lead to a nugget?
- What is the shape of the paths like? If they are purely white noise, then we cannot use information as we crawl along a path to determine whether we should abandon the path.
- What, if anything, can be inferred from path length? Intuitively, it seems that the longer a path becomes, the more likely it is that the crawl is in an irrelevant part of the Web.

The answers to these questions could lead to finding ways to make tunneling effective. In other words, the evidence could be used to estimate, dynamically, whether tunneling in a particular instance is going to pay off. The next section reports on some statistical results of this empirical study.

5 Experimental Results

With the experimental setup, almost 500,000 unique documents were downloaded and analyzed. While this is only a very small portion of the Web, it is a sufficiently large sample from which to make some statistical observations. No paths were terminated due to the cutoff (which was set very high), so the results approximate a maximally tunneling crawl. We hope to find some information in the statistics that would allow us to design a good tunneling strategy.

5.1 It Can be a Long Way from One Nugget to the Next

In this run there were 6620 completed paths; the most common length of a completed path was between 7 and 8. This corresponds rather well with the

figures in Table 2 which shows the distribution of distance from the closest ancestor nugget for the general population. Note also that at every length from an ancestor nugget, it is possible to find another nugget (column 4 has high values at all levels). These figures suggest that tunneling does have merit, although at some point it may not be worthwhile to go that far for a nugget.

Table 2. Correlation vs. path length. Column 1 is the distance from the closest ancestor nugget. Column 2 is the number of nodes at that level, including nuggets. Average correlation over all levels is 0.153. Seeds are not included

Level	Number	avg. corr.	max corr.
1	16422	0.1520	0.877
2	18106	0.2008	0.894
3	12699	0.1740	0.828
4	26356	0.1536	0.798
5	49018	0.1460	0.857
6	62728	0.1559	0.826
7	82287	0.1547	0.853
8	109297	0.1383	0.855
9	59370	0.1427	0.859
10	36672	0.1336	0.828
11	12390	0.1479	0.801
12	5981	0.0926	0.627
13	6604	0.1536	0.627
14	1485	0.1913	0.706

5.2 Better Parents have Better Children

In Table 3, we consider the distribution of correlations across the general population and compare it with distribution of the children of high-scoring nodes. The distribution is approximated by dividing the data into buckets of .05 correlation each. The second column indicates the likelihood of falling into any particular bracket; the third column shows the distribution of children of parents who scored in the .45 to .5 range (i.e. high-scoring, but not a nugget). We then look at the distribution of their children, and note that they are more likely to be nuggets than on average.

The opposite is not true. If you look at the children of nuggets (Table 4), you see that their correlation bracket predicts nothing (high standard deviation). This means that making a big effort to estimate the relevance of link targets is perhaps misguided.

5.3 Path History Matters

If the score of an individual node is a poor predictor, what about the history of a path? Should the crawl strategy be dynamically altered according to the shape

Table 3. Better parents have better children: here we consider parent nodes in the .45-.50 range and look at the distribution of their children. These parents have higher-scoring children than the general population

Correlation Bracket	General Population No. Nodes (%)	Nodes with corr (.45-.5] No. Nodes (%)
.05	149188 (21)	1102 (12)
.10	99479 (14)	422 (4.5)
.15	147185 (21)	892 (10)
.20	115576 (17)	897 (10)
.25	66530 (10)	1134 (12)
.30	42529 (6)	1022 (11)
.35	27710 (4)	944 (10)
.40	16549 (2)	740 (7.9)
.45	11787 (2)	768 (8.2)
.50	8126 (1)	614 (6.6)
.55	4613 (.6)	310 (3.3)
.60	3444 (.5)	224 (2.3)
.65	1971 (.3)	148 (1.6)
.70	1166 (.2)	78 (.83)
.75	656 (.1)	38 (.41)
.80	310 (.04)	11 (.12)
.85	96 (.01)	2 (.02)
.90	29 (.004)	1 (.02)
Total	696944 (100)	9347 (100)

Table 4. Average path length for children of nuggets according to their correlation score. Path length is 0 if the child contains a link to a nugget

Correlation Bracket	Number of Nodes	Average Path Length to Nugget	Standard Deviation
0.05	3380	4.64	2.19
0.10	3152	3.52	1.52
0.15	4702	3.64	2.07
0.20	3008	2.61	1.58
0.25	2345	2.28	1.14
0.30	5479	4.00	1.93
0.35	2973	3.08	1.83
0.40	2102	2.84	1.63
0.45	2727	4.03	2.88
0.50	1077	0.67	1.36

of the path? The question is should we treat paths *path1*: `0.45 0.25 0.30 0.35` and *path2:* `0.10 0.05 0.09 0.35` equivalently, or should we prioritize path1? Intuitively, path1 appears better because it is in a relevant area of the Web. Path2 may not be.

We chose to analyze this by comparing paths like ...X-X-0 (sequences ending in a nugget) with paths like ...X-X-X (sequences ending in a dud) to see if the penultimate two documents would predict the "nuggetness" of the final document. Figure 1 shows the results. This is the clearest indication that it might be good to pay attention to path history since the last nugget. The display is divided into two graphs - one for low-scoring dud parents (0 to .25) and one for high-scoring dud parents (scores between .25 and .5). Then we look at the distribution of the grandparents' scores in each case. For low-scoring parents, the score of the parent hardly matters. The probability of reaching a nugget matches that of the general population. On the other hand, for high-scoring parents, highly scoring grandparents do make a difference; the curve shifts distinctly to the right. A path of "almost-nuggets" is a significant predictor of future success.

5.4 Frontier Grows Rapidly

This has been discovered by many researchers, and again by us and is worth keeping in mind: after a few minutes of a breadth-first crawl, there will be a million URLs on the frontier. In this experiment, there were 4 million after 3 hours. A large frontier makes prioritizing computationally expensive.

Focused crawling is very helpful in weeding out URLs before they land on the frontier and slow down the run. Typically a full breadth-first crawl with Mercator will during the first 3 hours drop from an initial speed of 400 pages per second to about 50 pages per second, as the data structures grow larger and larger. If the threshold is dropped from .5 to .4, the crawl will drop even further, to 40 documents/second largely because the crawl runs into an increasing amount of the same material. Focusing the crawl is crucial to efficiency, provided it does not impact precision. Allowing tunneling, but knowing when to stop will help achieve both.

6 Putting the Findings to Practical Use

We have been focusing our crawls via threshold and cutoff values. But the statistical path data indicate that path history should also be taken into account in a way more complex than giving each node a fixed number of edges to go before the crawl is cut off. We take the path into account by replacing distance metric (2) with the following adaptive one:

$$\text{distance } = \begin{cases} 0 & \text{if this is a nugget} \\ min(1,\ (1-c)e^{2d_p/C}) & \text{otherwise} \end{cases} \tag{3}$$

where c is the correlation score for the current node, d_p is the parent's distance, and C is the cutoff, which remains constant for the crawl. Thus a node's distance

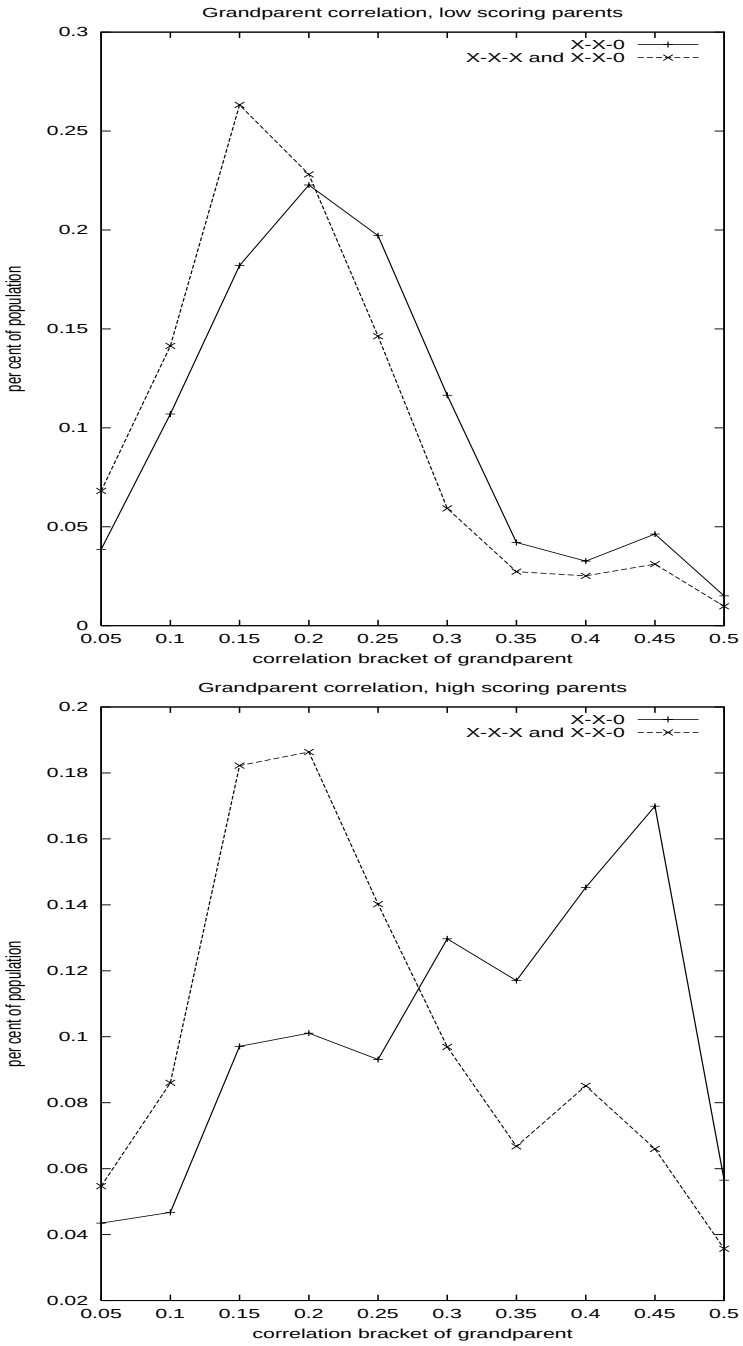

Fig. 1. Path shape does matter. Here we look at the two ancestors of a nugget vs. the population at large. The combination of high-scoring parents and high-scoring grandparents is likely to lead to a nugget.

metric grows exponentially as the parent's distance approaches the cutoff, and the lower the correlation score of a node, the greater its distance.

We used this adaptive cutoff in a 3.3 hour crawl with threshold 0.5 and cutoff 20. The results are shown in Figure 2 and Table 5. In Figure 2 the adaptive crawl shows a smooth decline in node correlation with distance from the nugget, which implies that the dynamic distance measure is accurately reflecting the goodness of the current node with respect to all the correlations along the path so far.

Also the graph indicates that suddenly 20 seems a very reasonable cutoff, whereas without the adaptive metric 20 was equivalent to infinity.

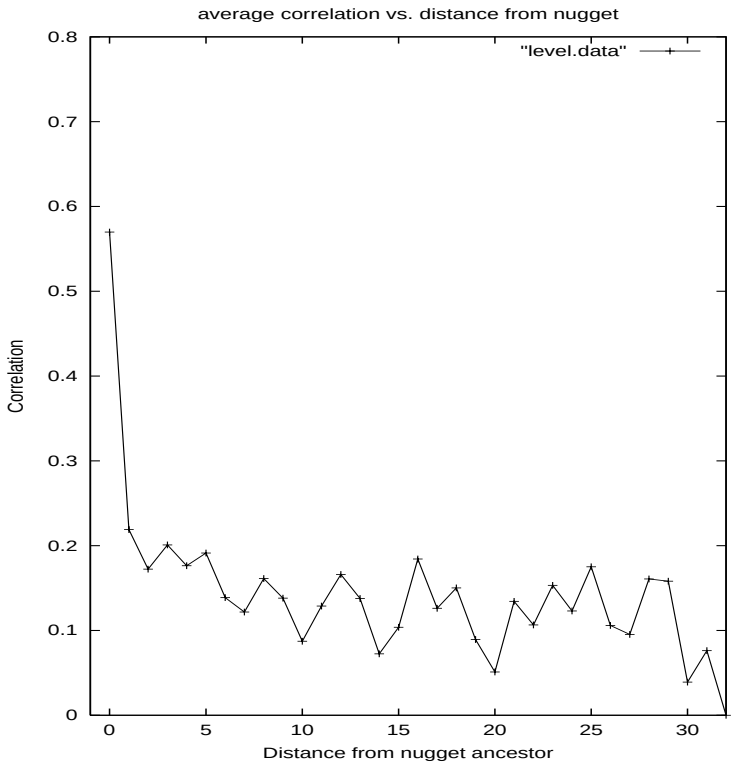

Fig. 2. Average correlation of all downloaded papers by distance from nearest nugget ancestor.

One of the best results of the focused crawl with tunneling based on an adaptive cutoff is the frontier had only 1.6 million URLs on it after 3.3 hours. Other comparative figures are in Table 5.

The focused crawl with adaptive cutoff after 200 minutes had 365,050 complete paths, vastly more than than the fixed cutoff crawl. This simply means that many paths were terminated because the distance from the nearest ances-

Table 5. Comparison of 100 minutes of regular focused crawl vs. 200 minutes of focused crawl with adaptive cutoff. In the second part of the table, percentages are of the number of documents ≥ 0.25

	Regular Crawl	Adaptive Cutoff
Threshold	.5	.5
Cutoff	20	20
Length	100 min	200 min
Final docs/sec	15.1	43.8
Urls on frontier	1M	1.6M
unique docs	91,859	541,205
≥ 0.25	13,265	48,196
[.5–.6)	1195 (1.94%)	4497 (4.22 %)
[.6–.7)	461 (.75 %)	1094 (2.27%)
[.7–.8)	125 (.20%)	285 (.59%)
[.8–.9)	27 (.04%)	39 (.08%)
[.9–1.)	0	0

tor nugget exceeded the cutoff. The results in the table show the benefits that accrued. In the first part of the table, we see various statistics related to the crawl. The most surprising is the difference in the terminal documents/second being downloaded. Even though the adaptive cutoff has a larger frontier than the first 100 minutes of the regular crawl, and thus a larger data structure, the download rate is much faster. Why?

The crawls were performed at comparable times during the day, so network congestion is unlikely to be the answer. More likely is that without the cutoff going into effect, the first crawl was staying much longer in already well-crawled territory and spending time figuring out that URLs and/or documents had already been seen. This is reflected in the unique documents downloaded figure as well. The adaptive focus has clearly gained us a lot of efficiency.

The second part of the table talks about the quality of the downloaded documents. First, anything with a correlation of < 0.25 is just ignored. Of the documents above that, what percentage are higher ranking? Again, the adaptive cutoff wins hands down. A much larger proportion of the downloaded documents are actually nuggets.

7 Related Research

Our work is similar to other work in focused crawling, e.g. Kluev [25]. Kluev's work is conceptually similar to ours but executes differently. He starts with a hand-picked set of documents, runs only one thread, and builds only one collection at a time. He reports on two collections, both broad in scope. We on the other hand exploit parallelism and automatic processing insofar as possible. Han and Karypis [26] use exactly the same vector space model as we do, but like

Kluev and others, starts with a given set of documents, from which the centroid is constructed. We require no such hand seeding.

Topic distillation [27, 13] is a technique that locates topic-related pages on the Web, but lets the topics emerge from what is intrinsically there. We start with a given taxonomy. But the question is, how do we know that the taxonomy is current? In future, topic distillation work may well become crucial to automating digital libraries, by determining the current topic structure of the Web.

Another aspect of collection building is its relationship to clustering in the field of artificial intelligence. Many recommend the application of AI clustering techniques such as Support Vector Machines to text classification, which is close in spirit to collection building. [26], for example, used a set of 23 existing document collections and constructed a centroid for each collection by making a term vector out of 80% of the collection. Then the remaining 20% of the documents were classified with the nearest centroid. The classification was good if that document was classified into the collection from which it originated. What is really interesting about this work is that centroid-based classification was compared with classification algorithms more typical in artificial intelligence: naive Bayesian, C4.5 Decision Trees, and k-nearest neighbor algorithms. The experiment concluded that the centroid-based document classification was the most accurate. If these results can be extended to Web documents, then we feel justified in using the centroid-based approach.

8 Conclusion

In this paper we described several technologies for using a Web crawler to help build collections for large scale digital libraries. We build on technologies introduced by other researchers: from information retrieval we take centroids, term vector space, and the cosine correlation; from the Web community we take crawling, focused crawling, and tunneling. We introduce the terms nugget and dud, and the concept of paths from one nugget to the next. We then performed a large crawl to gain statistical insight into the nature of the paths. This indicated that path history was relevant. Using this information we designed an adaptive cutoff that reflects path history, and expanded on the tunneling concept to achieve highly efficient and effective focused crawling. We expect to apply this strategy to future work in topic-specific digital collection synthesis.

Acknowledgments

This work was funded in part by the NSF grant on Project Prism, IIS 9817416. Thanks are due to Bill Arms for suggesting this particular research focus on automated collection building. We also acknowledge the considerable technical help received from the Systems Research Center at Compaq, especially from Raymie Stata, Marc Najork, and Richard Schedler. Chris Wilper implemented the Mercator extensions to capture path data. Thanks to Vicky Weissman for suggesting the term, “nugget”. Sbityakov came up with the term “dud”.

References

1. Lagoze (ed.), C., Arms, W., Gan, S., Hillmann, D., Ingram, C., Krafft, D., Marisa, R., Phipps, J., Saylor, J., Terrizzi, C.: Core services in the architecture of the National Digital Library for science education NSDL). In: Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries, Portland, OR (2002)
2. Zia, L.L.: The NSF national science, technology, engineering, and mathematics education digital library (NSDL) program: New projects and a project report. D-Lib Magazine: The Magazine of Digital Library Research **7** (2001)
3. Arms, W.: Automated digital libraries: How effectively can computers be used for the skill tasks of professional librarianship. D-Lib Magazine: The Magazine of Digital Library Research (2000) <http://www.dlib.org/dlib/july00/arms/07arms.html>.
4. Bergmark, D.: Collection synthesis. In: Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries, Portland OR (2002) Available: <http://mercator.comm.nsdlib.org/CollectionBuilding/bergmark-paper.pdf>.
5. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. In: Proceedings of the Eighth International World-Wide Web Conference., Toronto, Canada (1999) 545–562 Available: <http://www8.org/w8-papers/5a-search-query/crawling/index.html> and <http://www.cs.berkeley.edu/~soumen/doc/www99focus/> Current as of August 2001.
6. Belew, R.K.: Finding Out About. Cambridge Press (2001)
7. Salton, G.: Automatic Information Organization and Retrieval. McGraw-Hill, New York (1968)
8. Bergmark, D.: Using high performance systems to build collections for a digital library. In: Proceedings of the 2002 International Conference on Parallel Processing Workshops (ICPP 2002 Workshops), Vancouver, Canada (2002) Preprint available at <http://mercator.comm.nsdlib.org/CollectionBuilding/DCADL_bergmark.ps>.
9. Pirolli, P., Pitkow, J., Rao, R.: Silk from a sow's ear: Extracting usable structures from the Web. (1996) Available: <http://www.acm.org/pubs/articles/proceedings/chi/238286/p118-pirolli/p118-pirolli.html>.
10. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46** (1999) 604–632
11. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. In: Proceedings of the 7th International World Wide Web Conference (WWW7), Brisbane, Australia (1998) Available online at <http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>, (current as of 28 Feb. 2001).
12. Gibson, D., Kleinberg, J., Raghavan, P.: Inferring Web communities from link topology. In: Proceedings of the 9th ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space – Structure in Hypermedia Systems (HYPERTEXT'98, Pittsburg, PA). (1998) 225–234
13. Chakrabarti, S., van den Berg, M., Dom, B.: Distributed hypertext resource discovery through examples. In: Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, Morgan-Kaufman (1999) 375–386
14. Rennie, J., McCallum, A.: Using reinforcement learning to spider the Web efficiently. In: Proceedings of the International Conference on Machine Learning (ICML). (1999)
15. Menczer, F., Belew, R.K. In: Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web. (1999) 1–45 Republished in Machine Learning, 39(2/3) pp. 203–242, 2000.

16. Menczer, F., Pant, G., Srinivasan, P.: Evaluating topic-driven Web crawlers. In: SIGIR '01, September 9–12, New Orleans, La. USA (2001)
17. Mukherjea, S.: WTMS: A system for collecting and analyzing topic-specific Web information. In: Proceedings of the 9th International World Wide Web Conference: The Web: The Next Generation, Amsterdam, Elsevier (2000) Available: <http://www9.org/w9cdrom/293/293.html> (current as of August 2001).
18. Chakrabarti, S.: Recent results in automatic Web resource discovery. ACM Computing Surveys (1999) Available: <http://www.acm.org/pubs/articles/journals/surveys/1999-31-43es/a17-chakrabarti/a17-chakrabarti.pdf>.
19. Diligenti, M., Coetzee, F., Lawrence, S., Giles, C., Gori, M.: Focused crawling using context graphs. In: Proceedings of the 26th International Conference on Very Large Databases. (2000)
20. Heydon, A., Najork, M.: Mercator: A scalable, extensible Web crawler. World Wide Web **2** (1999)
21. Najork, M., Heydon, A.: High-performance Web crawling. Technical Report Research Report 173, Compaq SRC (2001) Available at <http://gatekeeper.research.compaq.com/pub/DEC/SRC/research-reports/abstracts/src-rr-173.html>.
22. Davison, B.D.: Topical locality in the Web. In: Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000), Athens, Greece, ACM (2000)
23. Joachimes, T.: A support vector method for learning ranking functions in information retrieval (2002) Cornell University Colloqium.
24. Parsia, B.: A simple, prima facie argument in favor of the semantic web. MonkyFist (2002) Available: <http://monkeyfist.com/articles/815>.
25. Kluev, V.: Compiling document collections from the Internet. SIGIR Forum **34** (2000) Available at <http://www.acm.org/sigir/forum/F2000/Kluev00.pdf>.
26. Han, E.H.S., Karypis, G.: Centroid-based document classification: Analysis & experimental results. Technical Report 00-017, Computer Science, University of Minnesota (2000)
27. Katz, V., Li, W.S.: Topic distillation on hierarchically categorized Web documents. In: Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange, IEEE (1999)