Design and Implementation of Contextual Information Portals

Jay Chen
New York University
jchen@cs.nyu.edu

Russell Power
New York University
power@cs.nyu.edu

Lakshminarayanan Subramanian
New York University
lakshmi@cs.nyu.edu

Jonathan Ledlie
Nokia Research Center
jonathan.ledlie@nokia.com

ABSTRACT

This paper presents a system for enabling offline web use to satisfy the information needs of disconnected communities. We describe the design, implementation, evaluation, and pilot deployment of an automated mechanism to construct *Contextual Information Portals* (CIPs). CIPs are large searchable information repositories of web pages tailored to the information needs of a target population. We combine an efficient classifier with a focused crawler to gather the web pages for the portal for any given topic. Given a set of topics of interest, our system constructs a CIP containing the most relevant pages from the web across these topics. Using several secondary school course syllabi, we demonstrate the effectiveness of our system for constructing CIPs for use as an education resource. We evaluate our system across several metrics: classification accuracy, crawl scalability, crawl accuracy and harvest rate. We describe the utility and usability of our system based on a preliminary deployment study at an after-school program in India, and also outline our ongoing larger-scale pilot deployment at five schools in Kenya.

Categories and Subject Descriptors

H.5.4 [**Hypertext/Hypermedia**]; K.3.1 [**Computers and Education**]

General Terms

Experimentation, Human Factors, Design, Algorithms

1. INTRODUCTION

The Internet has transformed the way people access and interact with information. In the developed world, the continuous availability and comprehensive content of the web makes it a compelling resource for communication and research. In contrast, many people in the developing world lack access to this resource; Internet access tends to be expensive, unreliable and largely confined to urban areas. In rural areas the Internet is unusable for a variety of reasons that result in poor and unreliable connectivity [12]. Providing reliable high bandwidth connectivity to these regions through conventional means has been shown to be economically not viable [43]. Systems that rely on wireless networks and mechanical backhauls have been explored for disconnected locations [48, 55], but these require significant expertise and training to deploy and maintain.

Poor Internet access in the rural developing world does not represent a "corner case." Beyond the digital infrastructure frontier, slow, unavailable, or expensive connections are consistently the status quo. Even in areas where reasonably good bandwidth is available (such as a 1 Mbps leased line in universities and small companies), the network connection is often shared across many users ($50 - 1000$ users in universities) resulting in an extremely poor user experience [18]. Only tens of kilometers outside Nairobi, Kenya, our researchers found several secondary schools where the local Internet Service Provider (ISP) was simply unwilling to lay down the "last hop" of fiber. Some of these schools desired, and were willing to pay for digital content and course materials; in fact, a small cottage industry exists to create these "digital libraries" [1]. This content is often in the form of "turnkey" solutions: pre-packaged, possibly including hardware, and ready for immediate use by the institution. Unfortunately, these digital libraries are relatively expensive and often of meager educational merit. One school we visited spent approximately 1 million Ksh (12,400 USD) on a digital library that a private company manually constructed specifically for their course syllabus. Teachers we spoke to described the content as both "too shallow" and of "low quality".

This paper describes the design and implementation of a system to automatically construct a *Contextual Information Portal (CIP)*. A CIP provides an offline searchable, browseable repository composed of content from the web about specific topics. In previous work [16], we proposed the basic framework for CIPs and outlined some of the research challenges involved in building and populating them. Building a CIP for any given task is a challenging problem and requires: (a) a document classification engine which can determine with high accuracy whether a page is relevant to a topic or not with minimal training; (b) a focused web crawler which can efficiently crawl the web starting from a few authoritative pages to determine the set of relevant pages on a given topic; (c) a presentation layer that enables users to efficiently search and browse a CIP. We had previously explored the document classification problem in the frame-

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

work of CIPs [51]. In this paper, we describe the design and implementation of a complete system to automatically generate CIPs for any topic. We demonstrate how our system efficiently constructs a CIP with relatively low crawl overhead and high classification accuracy. We have constructed two educational portals based on high school syllabi from India and Kenya on mathematics and computer studies topics. We report our initial experiences deploying these portals at schools in India and Kenya.

The rest of the paper is organized as follows. Section 2 motivates the concept of CIPs and Section 3 outlines the design of our system. Section 4 details the implementation of the various components of our system. Section 5 presents the evaluation of our system using microbenchmarks, lessons learned from a short preliminary study, and our deployments at five secondary schools near Nairobi, Kenya. Section 7 discusses related work, and Section 8 concludes with future work.

2. MOTIVATION

Contextual Information Portals represent a powerful and new paradigm for information access in regions without sufficient information sources and networking infastructure. CIPs present a fundamentally different model of information access since they can be stored on any large storage media (e.g., hard disks, DVDs, USB Keys, or SD Cards), and shipped as a self-contained web cache to any geographic location. CIPs are complementary to existing mechanical modes of bulk digital information dissemination which makes them easily deployable [24, 29, 50, 54, 55]. CIPs may also be used to address existing connection or content scarcity by bootstrapping local web caches behind slow connections, as a stand-alone portal within a kiosk service, or as an information source for low cost digital information distribution using TV and DVDs [24]. We also plan to miniaturize CIPs for use with cheap mobile devices. Since CIPs can be easily tailored to any topics of interest on any browser-capable device with sufficient storage, we envision even small grassroots organizations or NGOs with comparatively few computing resources building their own CIPs for field use on smartphones (Third-parties could supply customized CIPs as a business).

In the broader scope of using ICTs to serve the underserved, the focus of much effort is on the "bottom of the pyramid". Many recent ICTD projects have focused on constructing information portals or repositories for the illiterate and focus on challenges resulting from using voice as the communication and publication medium [35, 36, 47]. Typically, after such a system is constructed, locally generated content is the main source of information available. While this information is highly context appropriate to the local population, it does not leverage the huge amount of digital content already on the Internet. In contrast, our target populations are assumed to be literate, but disconnected. While populations with computing resources and a lack of connectivity are a minority, this scenario is challenging and worth solving as a first step before considering additional conflating constraints.

There are several specific areas where CIPs could be useful for different knowledge domains and scenarios:

Agriculture: eChoupal [2] is a large initiative by ITC in India that has established nearly 7,000 kiosks with VSAT Internet connectivity in villages throughout India to directly link with farmers for procurement of agricultural produce, especially soybeans and wheat. While the current usage model of these kiosks is limited, an India-specific information portal on soybeans and wheat could be an important value-added service that eChoupal could offer its rural farmers at each kiosk. We have begun work with agricultural universities in India to develop such portals.

Medicine: Bluetrunk Libraries [6] is a massive project by WHO to ship mobile libraries of 150 healthcare books into remote regions of Africa as an educational guide for healthworkers on the field. St. Johns Medical College, one of the largest hospitals in India has expressed interest in a similar resource for medicine. The same idea can be extended to disease specific portals for important diseases such as HIV, TB, malaria, diabetes or for portals for specific specializations such as opthalmology or surgery [5].

Education: Given the volume of educational material available online, educational portals for specific topics may be automatically constructed separately or as supplementary materials to well-structured portals such as MIT OpenCourseWare [4] or educational video repositories such as Khan Academy [3].

Locality-specific web portals: Collaborative caching over delay tolerant networks [28] has been observed to improve cache hit rates dramatically due to similar interests across nearby villages. The prefetching component in these systems could be improved by learning which location specific topics to prefetch and when time varying topics should be updated (e.g. weather and news should be updated frequently).

3. SYSTEM DESIGN

The basic premise behind Contextual Information Portals is that the context informs the specific information needs of communities in developing regions. More specifically, a context refers to a set of *topics* that represent the primary interests of the target users. For example, the educational syllabus of a school could represent the list of topics of interest for students and teachers within a school. We divide designing a CIP into three high-level design constraints. First, the information stored in the cache should be context appropriate. Second, the solution should be efficient in terms of relevant pages downloaded; both crawl bandwidth and disk space are limited resources. Third, the final set of pages in the CIP should not only be searchable but also *browseable* in some fashion.

A simple for construcing a CIP is to index an existing web proxy cache from a similar school nearby. However, such a cache was not available to us. If a compatible cache did exist, it would most likely be significantly smaller and still require crawling to achieve sufficient per-topic coverage. Moreover, the pages in such a cache would not be organized across topics, and would thus require topic classification before being useful to a crawler.

Another natural way to construct CIPs would be to issue several queries about a topic to a search engine and download the search results to construct a CIP on a topic. While the simplicity of such a design is appealing, it turns out to have several flaws. First, ranking algorithms used by search engines (such as PageRank) are global optimization functions across all pages on the web. As a result, given several different topics as queries, the same pages with high PageRank keep reappearing (such as the Wikipedia pages) as the top results regardless of their rank relative to the target topic. Second, the list of pages returned by a search

engine are not well-connected by hyperlinks to enable users to easily browse the contents of the portal.

The approach we take is to use a focused web crawler to crawl the web for pages that are relevant to the specified topic. The crawl is initially bootstrapped using the search page results corresponding to a fixed set of queries about the specified topic to a search engine. Blindly crawling web pages even when given a relevant starting point is problematic: pages a few links away are often irrelevant. Building a focused crawler requires us to use a good document classifier that accurately classifies pages relevant to a topic. Next, we describe our approach, classifier and focused crawler components.

3.1 Our Approach

Our system involves four steps:

Step 1: Defining topics. Topics may be defined many ways. One simple way of gathering sub-topics is to use manually-generated ontologies such as Wordnet [23] or machine generated ontologies such as the suggested queries returned by large Internet search engines. In this paper, we assume that a set of desired topics is provided, and in our evaluation we show how a set of topics may be defined quickly. In scenarios, where user information needs do not fall into specific topics, the notion of CIPs may not be highly appropriate.

Step 2: Training a classifier. The purpose of a document classifier is to determine whether a page is relevant to a specific topic. While there has been much work on document classification in general [21, 25], personalized or context-aware classification is still an area of active research [26, 59]. In earlier work [51], we designed and implemented a custom feature extractor that is compatible with several existing classifiers and provides high classification accuracy with minimal training.

Step 3: Focused Crawling. To perform the actual crawling process, we implemented a focused crawler based on the Shark crawler [27]. The goal of a *focused crawler* is to crawl only the relevant portion of the web that relates to the topic while minimizing the waste of downloading unrelated pages [7, 46, 59]. Our crawler uses the input of the classifier and various page and link level features to reduce the crawling of irrelevant pages. We make several modifications to the Shark crawler to improve the crawl efficiency.

Step 4: Presentation. While the crawler simply outputs a large number of web pages, it is essential to organize these pages into a searchable and browseable repository. We use a combination of existing proxy cache implementations and search indexing implementations to construct a presentation layer which presents a simple search and URL interface usable from any standard browser.

Although our system is largely automated, the human is included in the loop to guide the automation. The topics are generated by people, sites that are irrelevant are blacklisted, and presentation blacklisting useless sites, and although we do not discuss it in detail, we use a simple bookmarking tool to assist instructors in building lesson plans from the pages.

3.2 Document classification overview

Document classification plays a central role in our system. It is used by the crawler both to determine whether a particular page should be added to the CIP and also to decide on the direction for further crawling. The task of our document classifier is: given an arbitrary topic and a web page, determine whether a web page is related to that topic. Document classification is a well studied problem in information retrieval [31, 37, 40], and specifically in relation to the content on the web [52, 56, 58]. Even with this previous work, classification of web pages continues to be challenging, and existing approaches either do not provide high levels of accuracy or require extensive training.

There are two main factors which make document classification difficult: (a) noisy features and (b) topic ambiguity. First, in any document classification algorithm, extracting the right set of features plays a critical role in determining the accuracy of classification. In our case, the set of training documents often contains empty or irrelevant pages – a naive classifier will attempt to use these features as part of its model, resulting in decreased overall effectiveness. Second, many broad topics are often ambiguous, making classification of documents for these topics a hard problem. For ambiguous or broad topics, the topic may have different meanings and the topic and its related terms may reappear in various contexts. To overcome these two problems, we use a feature extraction algorithm that is compatible with statistical classifiers to improve classification accuracy.

3.3 Feature extraction

Our feature extraction algorithm builds upon on our previous work [51] where the main idea is to use a combination to two different and potentially opposing metrics to extract textual features for a given topic: (a) *popularity* and (b) *rarity*.

Popularity of words related to a given word is commonly used across existing classifiers [52] to weigh closely related terms within a document. Given a training set of documents related to the topic, the popularity metric determines a list of popular terms that are closely related to the topic.

Rarity is a metric that identifies the list of rare terms that are closely related to the topic. To measure rarity of any given term (which need not be a single word but an n-gram), we leverage the "web 1T 5-gram Version 1" dataset from the Linguistic Data Consortium (LDC) [38] to learn the frequency of occurrence of any n-gram on the web.

Either metric by itself may not be sufficient to achieve good classification accuracy. Also, filtering the feature set using these metrics reduces the noise in the classification model when compared to using the entire text of a document for classification. An additional advantage of our approach is the feature extraction process is fast and does not require extensive training sets. The smaller set of features also significantly reduces the time to train a model. Finally, our feature extraction algorithm may be used in conjunction with many statistical classifiers such as Bayes or SVM.

Given a particular topic, we consider the top N pages from a search engine such as Google as our candidate classification training set. To detect the relative frequency of a term t, we used the Linguistic Data Consortium dataset, which provides the web frequency of $n-$grams for $n \leq 5$. We denote this as $LDC(t)$. We use the LDC dataset to discount terms from the feature set relative to their popularity. For every term t, we compute the TF-IDF [32] value of that term as:

$$tfidf(t) = tf(t) \times log(N/N(t))$$

where we approximate $N(t)$ using $LDC(t)$ and N using the maximum value of the LDC across all $n-$grams. The LDC

is a commonly used dataset for estimating the inverse document frequency of a term when no good corpus exists for the specific application under consideration.

To extract popular features, we define a threshold P_{max} which represents the LDC max value below which we extract features of interest. We define a term to be *popular* relative to the topic if the following constraint is met:

$$tfidf(t) > T_{th}, LDC(t) < P_{max}$$

where T_{th} is a lower bound on the TF-IDF value for a term to be considered. P_{max} is the upper bound on the LDC count to remove extremely popular terms from consideration. Based on manual inspection across different topics, we set P_{max} roughly equal to $100,000,000$ for the LDC dataset. (In practice, we will set $P_{max} = 2^k R_{max}$ where k, R_{max} are two parameters described below)[1] The value of T_{th} was also computed as a common base value across different topics. For $N = 100$, across $15 - 20$ focused topics spanning different areas, we found $T_{th} = 4$ to be a good separation point across topics.

We use a graded measure to compute *rare terms*. The basic definition of a *rare* term is based on the following constraint:

$$tfidf(t) > R_{th}, LDC(t) < R_{max}$$

Here, R_{th} is a lower bound on the TF-IDF value and is typically much smaller than T_{th} for popular terms. For a threshold of R_{th}, we set R_{max} to be the upper bound on the LDC count to restrict this set to only consider rare terms.

Using a base threshold of R_{th} and R_{max}, we set up a graded threshold for measuring rarity. We divide the LDC frequency range in a logarithmic scale between R_{max} and P_{max} and derive an appropriate threshold for every range. For every scaling of the LDC threshold by a factor 2, we scale the TF-IDF by a scaling factor β. This introduces the following secondary condition - for $1 \leq l \leq k$:

$$2^{l-1} R_{max} \leq LDC(t) < 2^l R_{max}$$

and

$$tfidf(t) > \beta^l R_{th}$$

This represents a graded way of measuring rare terms across each range within the LDC frequency spectrum. In other words, we defined a base low value of $R_{max} = 1000$ as the smallest value under consideration and divided the LDC scale based on a logarithmic scale; we considered exponentially scaled up versions of R_{max} such as $2R_{max}$, $4R_{max}, \ldots$ $2^k R_{max}$. To extend this as a continuous measure between rarity and popularity and combine the two spectrums together, we set $P_{max} = 2^k R_{max}$. Similarly, $T_{th} = \beta^k R_{th}$. A specific choice of values appropriate for the 2004 LDC data set are: $R_{max} = 1000$, $k = 17$ and $P_{max} = 131,072,000$. Hence, our algorithm sets up a basic lower bound threshold on rarity based on R_{max} and a upper bound threshold based on P_{max} and uses a graded measure to extract features based on the terms with an LDC frequency in the range of $[R_{max}..P_{max}]$. Very rare terms with a frequency lower than R_{max} are selected as topic specific features if their TF-IDF score is greater than R_{th}.

[1]Note that the LDC dataset that is currently publicly available is circa 2004 and may not be reflective of the currently web frequency. No later versions were released by Google.

3.4 Classifier

We tested our feature extraction algorithm in conjunction with two standard classifiers for our system, Support Vector Machines (SVM) and Naive Bayes. Both of these have been shown to be effective for text classification [30,37]. For our system, we used the bow classification toolkit [57], which provides both Naive Bayes and SVM implementations. Classification was evaluated on the reduced feature set produced by the rare/popular method described above. In our previous work [51], we found evidence that our feature extraction algorithm improved the classification accuracy of both the classifiers. Since we found SVM to yield slightly better results than Naive Bayes, we use the SVM classifier in our implementation.

3.5 Focused Crawler

Focused crawling is a fairly well explored topic. Our focused crawler is based on Shark [27], a heuristic crawler that orders the nodes to be visited by sorting them according to their similarity to their ancestors in a priority queue. The key difference between a focused crawler and a standard crawler is that a focused crawler uses a ranking function to determine which outgoing link to traverse next with the goal of only crawling relevant pages corresponding to a topic.

Designing a good ranking function to decide the set of outgoing links to follow to direct the focused crawl is a challenging task and many heuristics have been proposed. The general methodology for designing a ranking function is to estimate a probability value for every outgoing link, that it would be relevant for a topic or not. We experimented with a combination of different parameters many of which are used by the Shark crawler: (a) number of crawled pages pointing to an out-going link; (b) the relevance of the parent page(s) to the topic as output by the classifier; (c) cumulative relevance of the parent page as output by the estimation function; (d) fraction of relevant pages crawled from the parent page; (e) relevance of anchor text around the link.

We made several optimizations to the Shark crawler to enhance the performance of the focused crawler for constructing CIPs. First, the basic Shark crawler uses a Naive Bayes classifier to estimate relevance of pages and the ranking function; our feature extraction algorithm coupled with the SVM classifier is what we use instead. Second, Shark can be bootstrapped with an initial set of *A authoritative documents or authorities*. In our case, the initial set was chosen based on the search results from a large search engine; since we leverage Google based results which leverages the underlying web graph, we believe our initial set to have good hubs and authorities for a topic.

Third, we added a heuristic technique called *tunneling* [10] that allows a number of irrelevant pages to be traversed to reach a relevant page. This is important because, pages relevant to a topic are not necessarily connected in the web graph. In our implementation, we set the tunneling depth to $D = 3$.

Fourth, we also made several changes to the Shark ranking function. We run the classifier on the initial set of authoritative pages to calculate a relevance probability p for each page and insert the page in a priority queue with probability p. This is particularly important for Google search results since we found that a non-trivial fraction of search result pages on a focused topic may not even contain the keyword. We introduced a scoring function where the inherited score

of a child node, c, of a relevant current node depends on, the probability p assigned by c as the relevance of the current node.

Finally, we use the anchor text around the hyper-link by extracting text from a fixed number (currently set to 2 words) of surrounding markup elements. We passed the anchor text through the classifier and used the classification probability in the ranking function.

3.6 Presentation Layer

The use of a CIP is similar to that of a normal web browsing session except that a CIP is completely offline. Similar to a search engine, CIP uses a simple presentation layer of a search interface and a list of high-level topics covered by the portal. Once we obtain the list of web pages from a focused crawler for a given topic, we use Lucene [39] to index the documents for a given topic. We also provide a simple mechanism for a user to search the list of topics within the portal. We also save the HTTP response headers, URL, and page title in the index for search and presentation.

Apart from a search interface, a user accesses a CIP using standard URLs. From a web browsing perspective, a CIP resembles a simple proxy cache in that if a page referred by a URL is present in the CIP, the page is directly accessible by its URL.

4. IMPLEMENTATION

We implemented our crawler in approximately 5000 lines of multi-threaded Python code. Our crawler is not a parallel crawler because at each iteration the main crawling algorithm downloads a page and computes the most promising next page to download. Threads are only used to parallelize tasks within this serial processing framework. We allocate a thread pool of up to 50 worker threads per process. Each thread may be given one of three tasks: *seedTask*, *embeddedTask*, *frontierTask*. The main thread performs the crawl and assigns tasks to the worker threads as necessary, and waits for the threads to complete or time out before continuing. The classifier runs in its own thread and called to classify documents by each worker thread. We use Lucene [39] to index the documents.

During execution, the main thread begins by requesting the URLs of A authorities and assigns each URL to a seedTask thread. Each seedTask thread downloads an authority page and adds it to the queue of "nodes". The main thread then begins crawling until either the queue is empty or the quota of pages to download is reached. For each node in the queue, the main thread classifies and caches the page if it is relevant. If the page is relevant, the main thread finds any embedded object references and assigns embeddedTasks to download the objects. We have implemented parsers for detecting embedded objects referenced by HTML 2.0 to 5.0 and CSS object references. Since our crawler does not execute Javascript, dynamic objects are not fetched. Finally, the main thread gathers the outlinks on the page if they exist and ranks them using the classifier again based on the anchor text and surrounding text. After the quota is reached, the main thread assigns frontierTasks to download the text-only pages in the frontier.

To demonstrate the portability of CIPs we integrated our repository with several user interfaces. First, we used Carrot2 [13], an off-the-shelf open source document clustering and interaction interface. We found that while the UI presented by Carrot2 was visually appealing, the automatic clustering algorithm did not always compose topics that facilitate the process of finding information targets. It was definitely useful for getting a sense of the general ideas in the CIP. We also integrated our repository and index with RuralCafe [17] user interface. The benefit of this is that if there is some network available, RuralCafe could dynamically update its contents and slowly improve the CIP on the fly. Carrot2 uses its own indexing mechanism which required slight modification to the crawler. Other UIs and presentation formats may require modification to the document and indexing mechanism, but these two adaptations required the addition of only a few lines of code.

Subject	# of Topics	Example Topics
accounting	408	ledger, business transactions, budget
agriculture	489	farm layout, livestock production, potassium
arabic	39	adjectives, particles, inflexion declension
art and design	433	insects, finishing, cubism picasso braque
biological science	647	fish, life cycles, cell physiology
chemistry	778	scientific objectives, salts, forces
christian religious education	475	apostleship, african religious heritage, luke
commerce	444	supermarkets, supply, promissory notes
computer studies	658	printing, internet software, fibre-optic cables

Table 1: Example subjects with topic counts and example topics.

5. MICROBENCHMARKS

To evaluate our system we constructed several CIPs based on the syllabus of a Kenyan secondary school. For each subject (Table 1), the syllabus contained requirements and course descriptions. We extracted the topics from the syllabus by unbinding and scanning the relevant pages of the hardcopy, OCRing it into a digital copy, cleaning up the topics automatically using several noise filters to remove garbage symbols and a simple stopword filter.[2] For the purposes of evaluating our system we did not further clean or manually rewrite any of the topics. If a topic was poorly defined, it is easily identified during the feature extraction process and may be rewritten manually. A sample of the topics we extracted is shown in Table 1. Using this set of topics, we evaluated our system first using microbenchmarks of the classifier and crawler. We then summarize our experience with a CIP generated for a preliminary study, our improvements following that study, and demonstrate the ease of deployability of our CIPs by setting up a larger-scale pilot at 5 schools in Nairobi.

[2]These steps could be eliminated if we had a softcopy which we did not.

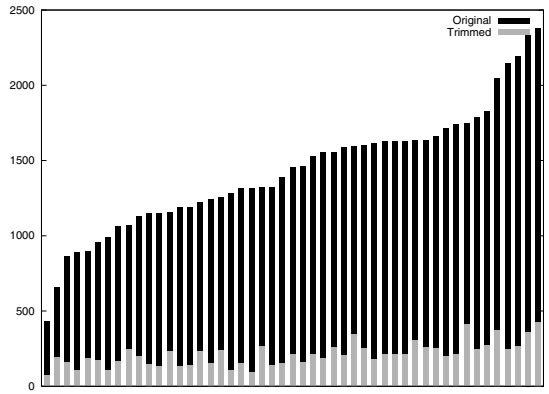

Figure 1: Total and Filtered Terms, per topic

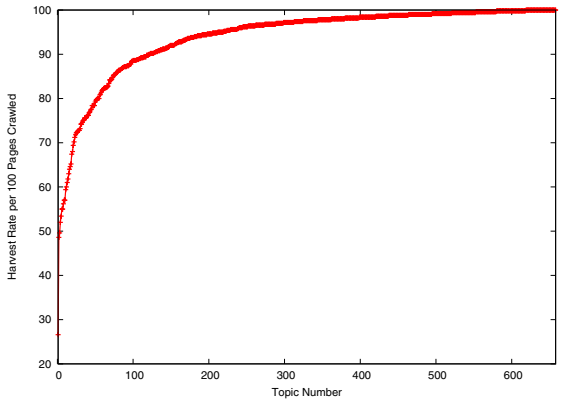

Figure 2: Harvest Rate per 100 Pages Crawled.

5.1 Classification

First we perform several benchmarks on our classifier using these authentic topics to complement and compare against our previous results [51] which used artificial topics.

5.1.1 Feature Extraction

To evaluate our feature extraction algorithm for a given topic, we generate a training set of candidate documents based on search engine queries for the topic and extraction of the text from the resulting pages. Filtering is applied to this initial set of documents to remove uninformative documents. The remaining documents are randomly partitioned to generate training and test data sets. In addition a negative test set is generated by extracting 10000 random documents from the English wikipedia corpus.

We verify that the number of features kept by the rare and popular filters is very small relative to the original feature counts. The number of features kept by the filtering process is shown in Figure 1.

5.1.2 Classification Results

We tested two classifiers on our corpus of topics, a Naive Bayes classifier and an SVM learner; we used the bow [41] toolkit to construct and test our classifiers. The models used for the classifiers were generated using the text of the documents as a unigram bag-of-words model. The classifiers were separately trained and tested using a set of words corresponding to the union of the rarity and popularity filters.

Our classifiers were run with their default settings. For libbow, this generates a learner based on a unigram word model. The Naive Bayes learner is smoothed by assuming a Dirichlet prior for zero valued features. The SVM learner uses a linear kernel with linear weighting on term frequencies.

The results of evaluating our topics were surprising - both the SVM and Bayesian classifier had very high precision the full range of subjects.

For this particular classification task, we found the effectiveness of the Naive Bayes classifiers to be significantly better then our SVM learner when training against the full feature set; the reverse occurs when training against the restricted word set. The effect of filtering terms from the feature set dramatically altered the behavior for our classifiers, in differing manners.

When working on the filtered set, our Naive Bayes classifier lost a small amount of recall, and showed better precision when rejecting documents of the negative test set. The classifier exhibited perfect precision in rejecting random documents on 3 of the topics. The most dramatic change was within our math topic set. We suspect the reason for the loss of recall to be related to the distribution of words within the math topic: the number of related but uncommon words for that topic is significantly larger then for the other sets.

When run against the filtered set, the SVM learner showed a uniform improvement in recall (acceptance related documents) for all categories and trivial losses in precision (rejecting unrelated documents). The SVM learner operating against the filtered feature sets outperformed all of our other classification attempts by a significant margin.

Finally, we observed qualitatively that the time to train the SVM was drastically reduced when running on the filtered features.

5.2 Crawling

Until now we have only been measuring accuracy of our classifier using our training set as the basis for comparison. To get a sense of how well our crawler and classifier worked for realistic topics we ran our crawler across each topic in the syllabus. We ran our crawler on cluster of 12 Duo and Quad core machines each with between 2 and 8 GB of RAM on the 658 computer studies subject topics. Across our cluster the crawlers requested 12,318,752 files of which there were 2,284,722 unique files from 53,685 domains.

5.2.1 Harvest rate

The harvest rate for topics in our corpus in Figure 2. We found that the harvest rate for 82% of topics in computer studies remained above 90% efficiency. This value is not directly comparable to other focused crawlers in the literature, but it does give a sense of the absolute efficiency of our system. The effects of varying the crawling algorithm, and parameters may be found in our previous benchmarks [16].

5.2.2 Crawling speed and scalability

Our crawler was not optimized for speed. Using a 1.8Ghz Duo Core Intel PC with 2 GB of RAM and a 2 Mbps broadband connection the crawl rate was on average approximately 1 hour per 500 page topic including time to download the frontier which is on the order of 20 - 40 thou-

sand pages. With our small cluster of 12 machines we were able to crawl 658 computer studies topics totaling 103 GB in approximately 2.5 days. The execution profile from running our crawler was dominated by server latency and connection setup overhead. This is to some extent inevitable due to the nature of the Internet, but could be optimized by simply reducing the connection timeout and parallelizing further. The crawler speed was sufficient for our purpose.

5.2.3 Crawling accuracy

While the classifier and crawler both report high accuracy and harvest rates, the final metric for the crawled results is the usefulness of the pages as judged by a human. We went through the results of each subject manually and sampled search results for a few of the topics within each subject. We found that the quality of the pages in the CIPs generated for each topic varied dramatically between subjects despite the high precision for the classifier microbenchmarks. In particular, for topics within the some subjects, e.g. music, christian religious education, german, and arabic the results were poor because the syllabi themselves lacked clear and concise topic descriptions. To address this problem, the syllabi themselves would need to be corrected such that the topics would at least result in relevant results after being queried to a search engine. We also observed that for some topics of subjects such as metalwork and drawing and design, the terminology was too vague to capture the intended meaning (e.g. "riveting", "oil cans" are ambiguous by themselves as they could have a large number of results irrelevant to metalwork). Including the subject in the query along with the topic may fix this issue when training and seeding the crawler. We found that for the topics of subjects relating to math, science, and engineering our crawler performed well; while we expect to address these crawler issues in the future, for our main studies we used the topics for mathematics and computer studies.

6. DEPLOYMENT

Prior to our main deployment we conducted a preliminary study with an initial prototype. We constructed a CIP for mathematics for 8th standard from an after school program in Kalpakkam, India. We summarize the findings from that study, describe several system and presentation level changes we made based on that experience, and then discuss our pilot at 5 secondary schools near Nairobi, Kenya.

6.1 Preliminary Study

For the after school program in Kalpakkam, India we worked with one teacher and 9 of her students. The CIP was constructed using the titles of 17 chapters of grade 8 Mathematics curriculum of the Central Board of Secondary Education of India (e.g. "square roots", "volume and surface area", "introduction to statistics", etc.). The portal was configured to download 500 complete web pages for each of the topics, and the set of pages downloaded was approximately 5GB in total (compressed). The total time for gathering the appropriate pages was approximately 8 hours on a single desktop machine in the U.S. with a 2Mbps connection.

We setup the portal on a computer that was already in the classroom. We then demonstrated the use of both the portal and a simple bookmarking tool for constructing lesson plans to the teacher. Software setup and training took roughly 30 minutes and 1 hour respectively. The lesson plans, experiments, and usage of the portal were designed, executed, and decided by the teacher herself without any direction from us. The teacher was given complete freedom on how to use the portal. She used the portal to teach the 8th standard material to her students who were in 6th and 7th standard. We observed her use of the portal over the course of one week, and at the end of the week we interviewed the teacher and gave the students questionnaires relating to their experiences.

All of our participants reported positive overall experience with the portal. The students felt that the portal allowed them to quickly access information and that there was more information available than usual. The teacher was also able to easily find information in the offline portal appropriate to her course topics and quickly construct a lesson plan. The portal itself was constructed quickly, and cost almost nothing to deploy in our setting. Finally, in terms of sustainability and ease of deployment, the installation and training were also quick and painless.

6.2 System Design Iteration

Despite both the high accuracy performance and overall positive feedback from the participants in our preliminary study, we observed several areas that could be improved. We discuss these changes at a high level as they are relatively simple.

Paywalled Sites: First, sites behind paywalls and sites selling educational products would frequently contain pages that were accepted by our classifier. This problem occurred across roughly half of the topics. While the classifier itself performed well after training on the training set, the authority pages returned by google for training the classifier were not always useful for our purposes. As a result, the pages eventually included in the CIP would sometimes contain pages with many relevant keywords, but no actual educational content. Thousands of pages from sites such as "www.tutorvista.com" and "www.sciencedirect.com" were retrieved and unsable. It was somewhat surprising that pages containing advertisements were not nearly as prevalent. While our bookmarking tool was useful for mitigating this problem during a teaching session, we would ideally remove such pages completely from the CIP.

Dynamic Pages: Second, pages that were highly dependent on dynamic content were displayed poorly or incorrectly. The resulting page presented to the user would be essentially unusable. This problem is similar to previous issue with paywalled sites. While the bookmarking tool helped, dynamic pages that failed to render properly were also a waste of resources and time. For both the paywalled sites and dynamic pages we implemented a simple blacklist filter in both the crawler and the presentation software. At the crawler end, the blacklist prevented crawl time wasted on unwanted sites. At the school itself, the blacklist was left exposed to the administrator (i.e. teacher) so he/she could block useless or unwanted pages on the fly. The blacklists could easily be merged into the global blacklist at the crawler for content updates.

Dead Links: Third, due to the offline nature of the CIP and the number of pages included, it is inevitable that beyond one or two pages browsed by a user links leading to pages that are not in the CIP. In our initial prototype we indexed all pages downloaded by the crawler. We found that the text-only pages downloaded for frontier pages usu-

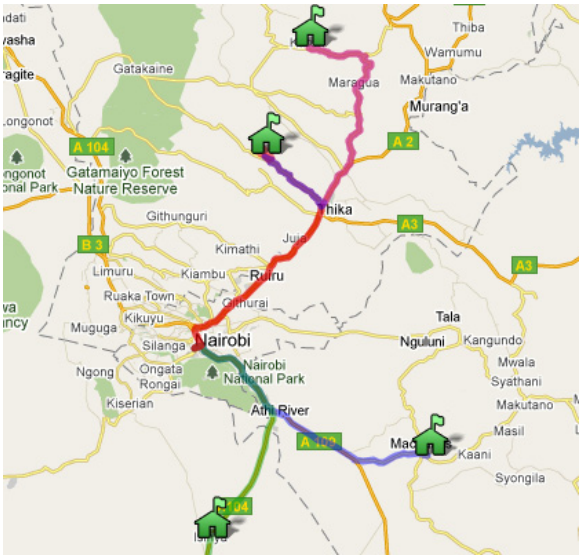

Figure 3: Map of the 5 participating schools of our large-scale pilot.

ally contained mostly dead links. We removed these pages from the index for our larger pilot. In addition, we implemented a Firefox browser extension to highlight links that are found in the cache so users could avoid clicking on links.

6.3 Large-scale Pilot

For our large-scale pilot we worked with a university in Nairobi that was already involved in weekend teaching activities at 5 schools near the city (Figure 3). The schools were all within 75 km of Nairobi, though the distance is slightly deceptive due to the poor quality of some roads. We were requested by the teachers at the university to construct a CIP for "computer studies" which is why much of our evaluation thus far has focused on that subject.

Each school has varying computer lab resources and some required local area networking hardware and setup. Table 2 summarizes the school hardware setup and any additional resources we purchased to complete the CIP setup. As the table indicates, the hardware itself is typically sufficient for a full-fledged computer lab.[3] Otherwise, the schools machines were either donated or purchased recently by the Ministry of Education in Kenya. Also, the actual maintenance of all labs was not always perfect.[4] The overall additional cost for physical resources was marginal, ranging between between 50 and 150 US dollars. We visited each of the schools twice, once to assess the hardware situation and a second time to setup the required hardware and CIP. Setup time at the schools took between 30 minutes and 3 hours depending on the level of maintenance and people working to setup the network.

The university program involved 4−5 undergraduate computer science students traveling to each school nearly every Saturdays during the school year and teaching students subjects in computer studies. We decided early on to leverage this existing program to bootstrap the use of the CIPs. The students and their program manager have agreed to use the CIP for the next several months as the primary teaching resource. The students themselves are qualified to teach, maintain, and upgrade the software for the CIP and resulting portals. The initial student and teacher responses at both the university and the schools have been extremely positive.[5] The software is being open sourced, and the students are encouraged to contribute by coming up with projects based on or around the CIP code base.

School #	Existing Hardware	Additional Hardware
School 1	20 computers, 1 switch, 1 server	One external hard disk
School 2	23 computers, 1 switch	One external hard disk, additional network cables
School 3	20 computers, 1 switch, 1 server	One external hard disk
School 4	6 computers	One external hard disk, one 24-port switch, network cables
School 5	28 computers, 1 switch, 1 server	One external hard disk

Table 2: Existing school hardware and additional requirements for pilot.

7. RELATED WORK

The application of focused crawlers to particular topics in an effort to build digital libraries and web caches has been considered before [10, 22, 53], but the application of these automatic techniques to the context of development has not been evaluated. Vertical search portals use similar focused crawling techniques to index a specific domain on the web, and the underlying crawling and classification technology is well studied.

A wide variety of document classifiers have been developed: Decision Tree [25], Support Vector Machines [21], taxonomic classifiers [15], and many others [59]. The two techniques appearing most often for web page classification are naive Bayes [37] and SVM learners [31]. In recent years, other techniques have been proposed and used, with some success [45].

Prior work for web page classification largely focuses on feature extraction and selection. In addition to the page text, additional components of pages and their relationships have been integrated to improve classification including HTML tags [58] and the geometry of the rendered page [56].

A broad class of features for documents comes from anchors. Anchor-text from links pointing into a page can be associated with the page itself [14]. In addition, URLs and text from other pages may be accumulated into the candi-

[3]School 4 had apparently lost several machines due to corruption by the previous principal.

[4]School 2 in particular was poorly maintained, 6 brand new machines in the middle of the room were plugged in, but not networked. Several other machines were not in working order due to minor issues such as broken or missing network cables/mice/monitors.

[5] Deployment at School 3 has been delayed due to internal conflict between the instructor and the principal which is still in the process of being resolved.

date feature set. Some attempts to assign additional feature information eschew the traditional association of relatedness via anchors, and instead attempt to group together pages on the basis of their relative positions in a site tree or web graph (sibling-relationships). Still other approaches view the labeling of pages as a optimization problem on the graph of pages formed by anchors, and attempt to optimize labeling on the graph [8].

Other work in the area has been on URL-only classifiers; these ask the question of whether is it feasible to classify a page knowing only the URL (and possibly some link structure) [33]. This is of particular interest for systems such as web-crawlers and focused crawlers, where there is a desire to classify a page as desirable *before* retrieving it. Even with this restriction on available information, classification precision above 60% has been achieved on the WebKB dataset [9].

There is extensive of literature on web crawling algorithms [11, 34, 44]. Focused crawling was defined and formalized by [15], which introduced taxonomic classification or text analysis to determine document relevance and distillation or link analysis to identify authoritative sources of relevant documents. Shark is one of the earlier focused crawlers, and more sophisticated variations exist [46,49]. Also, extensive follow up work has compared focused crawling against a variety of other crawling techniques across a variety of domains [19,20,42]. It has been shown in recent work [7] that uses a latent semantic indexing (LSI) classifier, which combines link analysis with text content, that a variant of the simple Shark-search can be surprisingly efficient when compared to more sophisticated and expensive techniques such as LSI and PageRank (PR).

8. CONCLUSIONS AND FUTURE WORK

In this work we designed and implemented a complete system for automatically constructing Contextual Information Portals. We used a combination of information retrieval techniques to collect web pages for a CIP using relatively limited computing resources. We incorporated this into a complete solution and demonstrated its effectiveness for real world subject domains. We also integrated our CIP with an existing user interface to provide a portable offline digital library for communities without libraries or access to other information sources. After conducting a preliminary deployment and incorporating participant feedback and improvements into our overall system, we deployed a large-scale pilot in 5 schools near Nairobi, Kenya. Our system requires only the addition of a hard disk, and as a result both the "capital expenditure" and on-going "operating costs" are low. In the future we plan to thoroughly assess the impact of CIPs on education in our pilot, and we hope to extend CIPs to more schools and other environments.

9. ACKNOWLEDGEMENTS

The authors would like to thank Mangala Kanthamani for her help in our preliminary study. We would also like to thank Joseph Sevilla and Emmanuel Kweyu for facilitating our large scale pilot.

10. REFERENCES

[1] Cyber school technology solutions. http://www.cyberschooltech.com.

[2] echoupal. http://www.itcportal.com/rural-development/echoupal.htm.

[3] Khan academy. http://www.khanacademy.org.

[4] Mit opencourseware. http://ocw.mit.edu.

[5] Web initiative for surgical education. http://wise-md.med.nyu.edu.

[6] World health organization - blue trunk libraries. http://www.who.int/ghl/mobile_libraries/bluetrunk/en.

[7] G. Almpanidis, C. Kotropoulos, and I. Pitas. Combining text and link analysis for focused crawling-an application for vertical search engines. *Information Systems*, 2007.

[8] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. *ACM SIGIR*, Jan 2006.

[9] E. Baykan, M. Henzinger, L. Marian, and I. Weber. Purely url-based topic classification. *18th International Conference on World Wide Web*, Jan 2009.

[10] D. Bergmark, C. Lagoze, and A. Sbityakov. Focused crawls, tunneling, and digital libraries. *Lecture notes in computer science*, 2002.

[11] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st annual international ACM SIGIR conference*, 1998.

[12] E. Brewer, M. Demmer, M. Ho, R. Honicky, J. Pal, M. Plauche, and S. Surana. The challenges of technology research for developing regions. *Pervasive Computing, IEEE*, 2006.

[13] Carrot2 - Open Source Search Results Clustering Engine. http://project.carrot2.org/.

[14] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Jun 1998.

[15] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networking*, pages 1623–1640, 1999.

[16] J. Chen, T. Karthik, and L. Subramanian. Contextual Information Portals. In *Proceedings of AAAI Spring Symposium*, 2010.

[17] J. Chen, L. Subramanian, and J. Li. RuralCafe: web search in the rural developing world. In *Proceedings of the 18th international conference on World wide web*, 2009.

[18] J. Chen, L. Subramanian, and K. Toyama. Web search and browsing behavior under poor connectivity. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3473–3478. ACM New York, NY, USA, 2009.

[19] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. In *Proceedings of the seventh international conference on World Wide Web*, Amsterdam, The Netherlands, The Netherlands, 1998.

[20] B. D. Davison. Topical locality in the web. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000.

[21] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proceedings of the 23rd annual international ACM SIGIR*, page 263, 2000.

[22] M. Ehrig and A. Maedche. Ontology-focused crawling of Web documents. In *Proceedings of the 2003 ACM symposium on Applied computing*, 2003.

[23] C. Fellbaum et al. *WordNet: An electronic lexical database*. MIT press Cambridge, MA, 1998.

[24] K. Gaikwad, G. Paruthi, and W. Thies. Interactive DVDs as a Platform for Education. In *IEEE/ACM International Conference on Information and Communication Technologies and Development*, 2010.

[25] S. Gao, W. Wu, C. Lee, and T. Chua. A maximal figure-of-merit learning approach to text categorization. In *Proceedings of the 26th annual international ACM SIGIR*, 2003.

[26] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid

classifier for text categorization. In *Proceedings of the 18th international conference on World wide web*, 2009.

[27] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm. an application: tailored web site mapping. In *Proceedings of the seventh international conference on World Wide Web*, 1998.

[28] S. Isaacman and M. Martonosi. The C-LINK System for Collaborative Web Usage: A Real-World Deployment in Rural Nicaragua. *Workshop on Networked Systems for Developing Regions*, 2009.

[29] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. *ACM SIGCOMM 2004*, Jan 2004.

[30] T. Joachims. Making large scale svm learning practical. *Advances in Kernel Methods, Support Vector Learning*, Jan 1998.

[31] T. Joachims, C. Nedellec, and C. Rouveirol. Text categorization with support vector machines: learning with many relevant. *10th European Conference on Machine Learning*, Jan 1998.

[32] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[33] M. Kan and H. Thi. Fast webpage classification using url features. *ACM CIKM*, Jan 2005.

[34] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, pages 604–632, 1999.

[35] P. Kotkar, W. Thies, and S. Amarasinghe. An audio wiki for publishing user-generated content in the developing world. In *HCI for Community and International Development (Workshop at CHI 2008), Florence, Italy*. Citeseer, 2008.

[36] A. Kumar, S. Agarwal, and P. Manwani. The spoken web application framework: user generated content and service creation through low-end mobiles. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, pages 1–10. ACM, 2010.

[37] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science*, Jan 1998.

[38] Linguistic Data Consortium. http://www.ldc.upenn.edu.

[39] Lucene. http://lucene.apache.org.

[40] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[41] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[42] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz. Evaluating topic-driven web crawlers. In *Proceedings of the 24th annual international ACM SIGIR conference*, 2001.

[43] S. Mubaraq, J. Hwang, D. Filippini, R. Moazzami, L. Subramanian, and T. Du. Economic analysis of networking technologies for rural developing regions. *Workshop on Internet Economics*, 2005.

[44] M. Najork and J. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th international conference on World Wide Web*, pages 114–118. ACM New York, NY, USA, 2001.

[45] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. Jan 1999.

[46] S. Pandey and C. Olston. Crawl ordering by search impact. In *Proceedings of the international conference on Web search and web data mining*, pages 3–14. ACM, 2008.

[47] N. Patel, D. Chittamuru, A. Jain, P. Dave, and T. Parikh. Avaaj Otalo.A Field Study of an Interactive Voice Forum for Small Farmers in Rural India. In *Proceedings of the Proceedings of the 28th international conference on Human factors in computing systems (Atlanta, GA, USA, 2010). ACM*.

[48] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks. *NSDI*, 2007.

[49] T. Peng, C. Zhang, and W. Zuo. Tunneling enhanced by web page content block partition for focused crawling: Research Articles. *Concurrency and Computation: Practice & Experience*, 2008.

[50] A. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, Jan 2004.

[51] R. Power, J. Chen, T. Karthik, and L. Subramanian. Document Classification for Focused Topics. In *Proceedings of AAAI Spring Symposium*, 2010.

[52] X. Qi and B. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)*, 2009.

[53] J. Qin, Y. Zhou, and M. Chau. Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, 2004.

[54] A. Seth, D. Kroeker, M. Zaharia, and S. Guo. Low-cost communication for rural internet kiosks using mechanical backhaul. *Mobicom 2006*, Jan 2006.

[55] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. *Mobicom*, pages 334–345, 2006.

[56] L. Shih and D. Karger. Using urls and table layout for web classification tasks. *Proceedings of the 13th international conference on World Wide Web*, Jan 2004.

[57] The Bow Toolkit. http://www.cs.cmu.edu/ mccallum/bow/.

[58] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, Jan 2002.

[59] H. Zheng, B. Kong, and H. Kim. Learnable focused crawling based on ontology. *Lecture Notes in Computer Science*, 2008.