

Link Contexts in Classifier-Guided Topical Crawlers

Gautam Pant and Padmini Srinivasan

Abstract—Context of a hyperlink or *link context* is defined as the terms that appear in the text around a hyperlink within a Web page. Link contexts have been applied to a variety of Web information retrieval and categorization tasks. Topical or focused Web crawlers have a special reliance on link contexts. These crawlers automatically navigate the hyperlinked structure of the Web while using link contexts to predict the benefit of following the corresponding hyperlinks with respect to some initiating topic or theme. Using topical crawlers that are guided by a Support Vector Machine, we investigate the effects of various definitions of link contexts on the crawling performance. We find that a crawler that exploits words both in the immediate vicinity of a hyperlink as well as the entire parent page performs significantly better than a crawler that depends on just one of those cues. Also, we find that a crawler that uses the tag tree hierarchy within Web pages provides effective coverage. We analyze our results along various dimensions such as link context quality, topic difficulty, length of crawl, training data, and topic domain. The study was done using multiple crawls over 100 topics covering millions of pages allowing us to derive statistically strong results.

Index Terms—Web Search, Web mining, performance evaluation.

1 INTRODUCTION

LINK context is utilized in various Web-based information retrieval tasks. In most cases, anchor text or text within an arbitrary size window around a link is used to derive the context of a hyperlink. While the importance of link context has been demonstrated in areas such as Web page categorization [3] and search engines [5], topical crawlers (e.g., [17], [8], [25], [2]) have a special dependence on link context. Topical crawlers follow the hyperlinked structure of the Web using the *scent of information* [27] to direct themselves toward topically relevant pages. For deriving the appropriate scent, they mine the content of pages that are already fetched to prioritize the fetching of unvisited pages. Unlike search engines that use contextual information to complement content-based retrieval, topical crawlers depend primarily on contextual information. This is because topical crawlers need to predict the benefit of downloading unvisited pages based on the information derived from pages that have been downloaded.

Topical crawlers have been used in a variety of applications such as competitive intelligence [9], [30], search engines [24], and digital libraries [32], [34]. They allow a higher level application to gather from the Web, a focused collection rich in information about a topic or theme. Previous literature in topical crawling exhibits various definitions of link context. A large number of such crawlers use the entire Web page content as the context for the hyperlinks in that page [12], [8], [13], [32]. More granular

definitions of link context, such as selecting a few words around a hyperlink, have also been used [17], [7], [30]. Despite the many potential definitions of link context, no systematic study has been done to tease out their effect on topical crawling. The special importance of link context in topical crawling underlines the need for such a study. For instance, we do not know the optimal text window size around a link for topical crawling. It is also not known if one should combine the local information obtained from the chosen text window around a link with the more global information from the entire page. Additionally, there may be merit in using the HTML structure for obtaining link contexts. These and other related aspects in the setting of topical crawling motivate our research. In this paper, we present and analyze the performance of topical crawlers that are guided by a Support Vector Machine (SVM) and use various notions of link contexts. The study spans millions of crawled pages over 100 topics allowing us to derive statistically robust arguments.

In the next section, we describe related research. Section 3 details a flexible crawling infrastructure that supports our experiments and Section 4 describes the use of a classifier within the infrastructure. Topics for crawling and performance metrics are explained in Section 5 and Section 6, respectively. Section 7 clarifies our choice for the classifier. Section 8 describes the experiments, while Section 9 presents the corresponding results. We analyze the results in Section 10. Section 11 concludes the paper along with some directions for future research.

2 RELATED WORK

Since the early days of the Web, researchers have explored different link contexts in Web pages. Such link contexts have been used for page classification, topical crawling, and when building search engines.

- G. Pant is with the School of Accounting and Information Systems, University of Utah, 1645 E. Campus Center Dr., Salt Lake City, UT 84112. E-mail: gautam.pant@business.utah.edu.
- P. Srinivasan is with the School of Library and Information Science, University of Iowa, 3067 Main Library, Iowa City, IA 52242. E-mail: padmini-srinivasan@uiowa.edu.

Manuscript received 25 Aug. 2004; revised 1 Apr. 2005; accepted 11 July 2005; published online 18 Nov. 2005.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0296-0804.

2.1 Web Page Classification

Here, the goal is to assign a Web page to one or more predefined classes within a classification scheme. Given a Web page, Chakrabarti et al. [6] classify it by using the estimated classes of pages in its neighborhood graph. A technique that categorized Web pages based on the context of their URLs was proposed by Attardi et al. [3]. They exploited the HTML structure to derive a sequence of context strings that were then used for classification. Chakrabarti et al. [6] used a text window consisting of a number of bytes around the anchor to obtain the link context for automatic Web resource compilation. With an experiment using 5,000 Web pages, the authors found that the word "Yahoo" was most likely to occur within 50 bytes (approximately 6-8 words) of the anchor containing the URL <http://www.yahoo.com/>. We note that using bytes (or single characters) is, in general, less appealing than using words or phrases that offer more semantically meaningful units.

2.2 Search Engines

McBryan [23] used the anchor text to index URLs in the WWW Worm. Brin and Page [5] suggested the use of anchor text (in addition to page content) to index URLs in their Google search engine. Craswell et al. [10] have shown that using anchor text can provide more effective rankings for site finding tasks. In a recent thesis, Bradshaw [4] constructed a search engine that indexed pages solely based on link contexts.

Davison [11] tested hypotheses related to topical locality on the Web (i.e., most Web pages have links to other pages with similar content). The study was done using a large collection of Web pages called the DiscoWeb. In particular, Davison showed that a text window around a hyperlink in a *source* page has high similarity to the *destination* page as compared to a random page. The source page (also known as the *parent* page) is the page where the hyperlink appears and the destination page is the page that the hyperlink leads to. The author also suggested that enlarging the text window of a link by including more words should increase the chance of getting the important terms but at the cost of including more unimportant terms. This is an interesting trade-off between having precise information and having any information at all. We will discuss this issue further after presenting our own results.

2.3 Topical Crawling

As mentioned earlier, topical crawlers especially rely on link context. A simple notion of link context used in topical crawling is to treat the contents of the entire parent page as the context of the hyperlinks within it. Influential topical crawling works such as De Bra and Post [12] and Chakrabarti et al. [8] used this notion of link context. One of the early attempts at using more granular link contexts in topical crawlers was by Iwazume et al. [18]. They guided a crawler using the anchor text along with an ontology. FishSearch suggested by De Bra and Post [12] was later developed into SharkSearch [17]. SharkSearch used the anchor text and some of the text in its "vicinity" (or a *text window*) to estimate the benefit of following the corresponding link. It also linearly combined the information gained

from the smaller contexts with that from the entire parent page before estimating the net benefit of following a link. In recent work, Chakrabarti et al. [7] suggested the idea of using *DOM offset*, based on the distance of text tokens from an anchor on a *tag tree*, to score links for crawling. Tag tree is a tree representation of a Web page where the nodes are the HTML tags or the text within the page. The `<html>` tag forms the root of the tree. Fig. 1 shows an example of a Web page and its corresponding tag tree. The DOM offset of a text node was defined as a positive or negative number based on whether the text node appeared on the right or the left of the anchor using the tag tree representation. The magnitude of the offset was based on the number of leaf nodes of the tag tree between the anchor and the given text node. We note that this magnitude provides no information about the relative depths or hierarchical relationship (if any) between the anchor and a text node. That is, their work did not make use of the inherent hierarchy in an HTML page (and, hence, the tag tree) to derive link context. For example, anchor `<a>...` tags may appear within paragraph `<p>...</p>` tags. In that case, all the text within the paragraph tags may be considered as the context of the URL within the embedded anchor tag.

Different heuristics have been used to guide topical crawlers. Application of standard information retrieval techniques such as cosine similarity to on-topic examples, queries, or profiles can be found in several topical crawling studies [17], [26], [38]. User browsing patterns have also been used for personalized [22] as well as collaborative [1] crawling. However, with regard to the design of crawlers using classifiers, the related literature is quite sparse. Chakrabarti et al. [8] were the first to use a (Naive Bayesian) classifier to guide a topical crawler. While Naive Bayesian classifiers have been used in at least a couple of other topical crawlers [13], [7], more recently, Johnson et al. [20] suggested the use of an SVM with a linear (first degree polynomial) kernel for topical crawling. In a companion paper [31], we systematically compare crawlers built upon three different classifier schemes: Naive Bayes, Neural Network, and SVM. In it, we show that the SVM classifier with linear kernel is a better choice for topical crawling than Naive Bayes and is as good a choice as Neural Network. Therefore, this study examines link contexts using an SVM classifier.

3 CRAWLING INFRASTRUCTURE

We implement topical crawlers based on a multithreaded four-layer high-level model as shown in Fig. 2. Each thread follows a sequential *crawling loop* that includes the layers shown in the figure. The bottom layer, called the *networking layer*, is responsible for downloading pages from the Web, storing them in a repository, and maintaining a *history* of the sequence of URLs fetched. The *parsing and extraction layer* is involved in extracting information such as the terms and the hyperlink URLs from a downloaded page. This is where link contexts are identified. The extracted information is then represented in some mathematical form such as a TFIDF vector [36] in the *representation layer*. This representation is processed by the *intelligence layer* and the

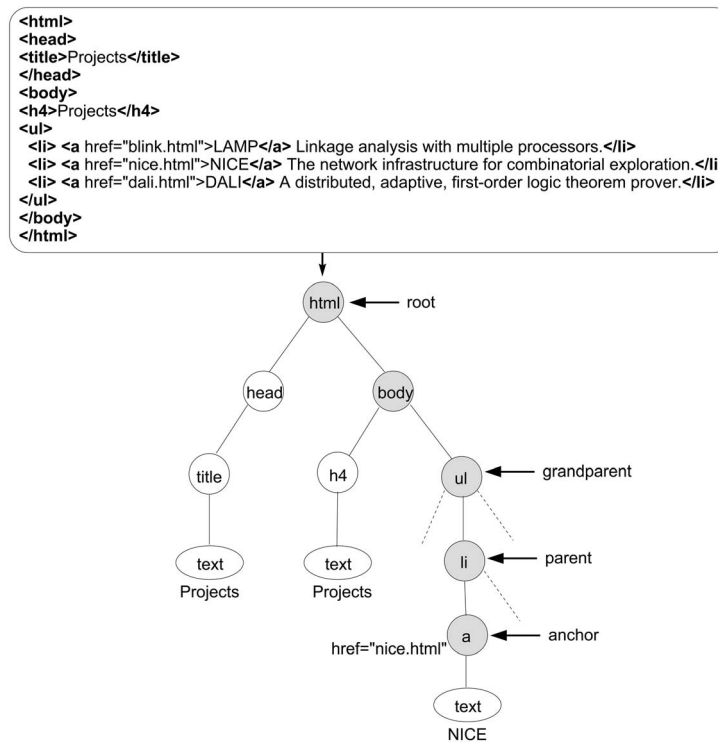


Fig. 1. HTML snippet and corresponding tag tree representation.

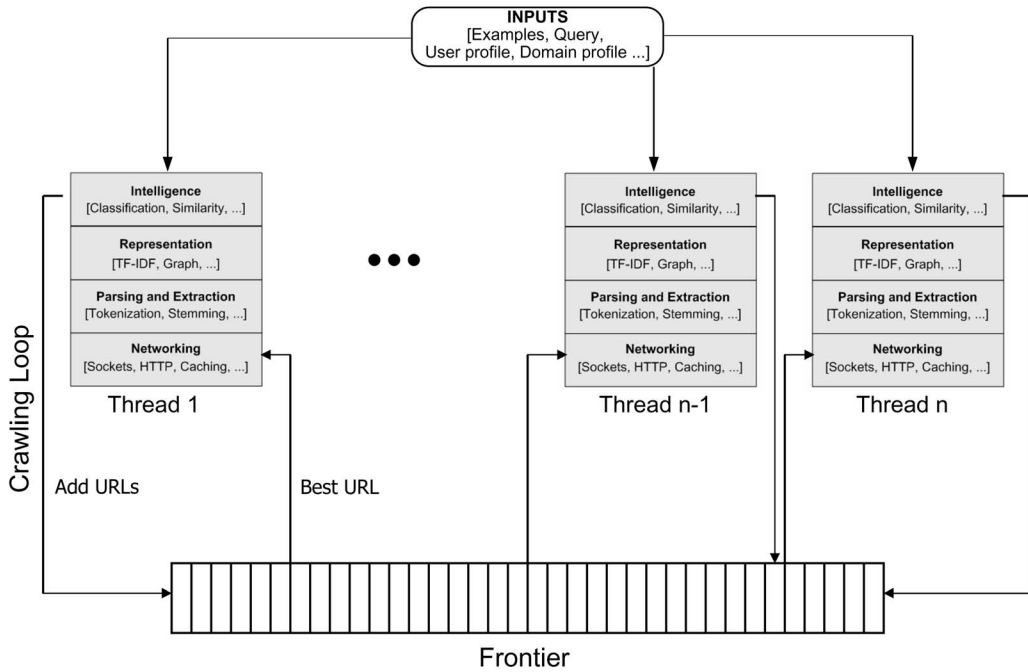


Fig. 2. High-level model of multithreaded topical crawlers.

layer assigns scores to the unvisited URLs extracted from the page. The unvisited URLs along with the assigned scores are added to the *frontier*, a list of unvisited URLs. The assigned scores represent the estimated benefit of following the corresponding URLs.

We implement this high-level layered design as multithreaded objects in Java. The threads follow the crawling loop where the “best” URL is picked from the frontier, the

page corresponding to the URL is fetched from the Web, the fetched page is processed through all of the infrastructure layers, and the unvisited URLs from the page with their scores are added to the frontier. All of the threads of a crawler share a single synchronized frontier.

Multithreading provides for reasonable speed-up when compared with a sequential crawler. We use 75 threads of execution while running a crawler and only the first 10 KB of each Web page is downloaded.

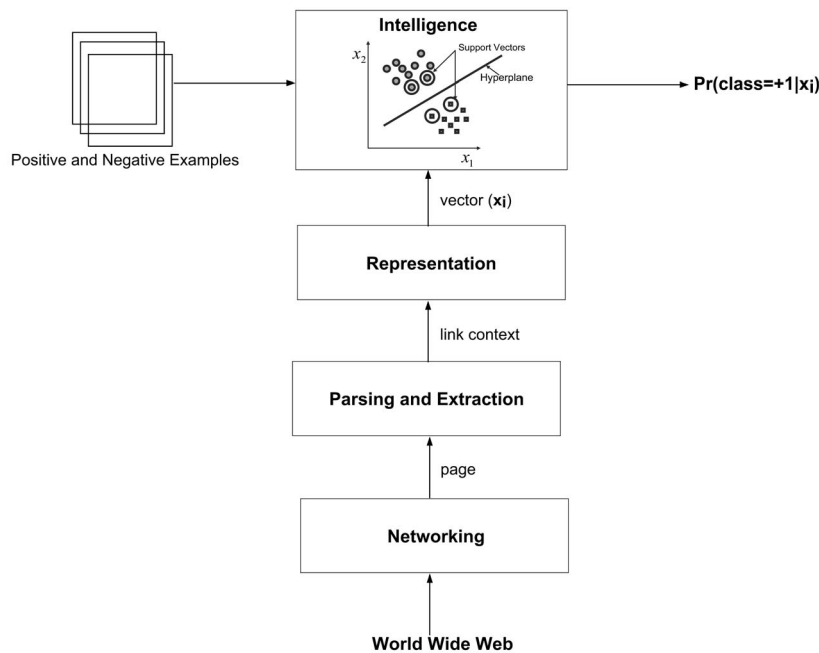


Fig. 3. Assigning scores to link contexts via an SVM classifier

4 CRAWLING CLASSIFIER

The intelligence layer contains a model of the topic or theme for which the crawling is being conducted. Using this model, the intelligence layer assigns scores to unvisited URLs based on the representation of their link contexts. One category of models uses classifiers [14]. A classifier can model the topic of interest using positive and negative examples (together, called a *training set*) which are *inputs* to the crawler (see Fig. 2). The positive examples are Web pages that are considered relevant to the topic while the negative examples are Web pages that are considered irrelevant. A classifier learns from these examples through a training process. Once trained, the classifier can be used for assignment of previously unseen objects (link contexts) to positive and negative classes.

The text from each HTML page in the training set is extracted and tokenized to identify the words within it. The common words or *stop-words* are removed and the remaining words are stemmed. These stemmed words or terms from all of the negative and positive examples form the vocabulary \mathcal{V} for the corresponding topic. This vocabulary will change with the topic at hand. The positive and the negative examples are represented as vectors where each element in the vectors corresponds to a term in \mathcal{V} . Thus, we represent a Web page p as a vector $\mathbf{x}_p = (w_{1p}, w_{2p}, \dots, w_{Np})$, where w_{sp} is the weight of the term s in page p and N is the size of \mathcal{V} . The term weights are TF-IDF values that are computed using a standard weighting scheme in the SMART system [35].

Once the positive and the negative examples are represented as vectors (or points) in a high-dimensional TF-IDF vector space, a classifier is “trained” to draw a decision surface in the space in an attempt to effectively separate the unseen positives from the unseen negatives. Given a real-world data set such as link contexts and their true classes, no decision surface will be perfect in classifying unseen data. However, we would like the classifier to

provide us with $\text{Pr}(\text{class} = +1|\mathbf{x}_i)$, the probability that the link context belongs to the relevant (positive) class given its representation. We can then associate this probability with the corresponding URL. This probability will serve as a score through which we can prioritize the frontier of URLs. Fig. 3 illustrates this process of assigning scores to link contexts.

We use the Weka¹ machine learning software for the implementation of the classifier within the intelligence layer. In particular, we chose the SVM [39] implementation available in Weka. We use an SVM classifier with a linear kernel. The choice of this classifier is further explained in Section 7. Weka utilizes J.C. Platt’s sequential minimal optimization (SMO) algorithm [33] for training an SVM.

5 TOPICS FOR CRAWLING

We obtained topics for our crawls from the Open Directory Project (ODP).² The ODP is a categorical directory of URLs that is manually edited and relatively unbiased by commercial motivations. The ODP provides the data contained in its directory in RDF format through its Web site. We first downloaded the RDF formatted *content file* from the ODP Web site. The content file contains a list of ODP categories and the external URLs or *ODP relevant set* corresponding to each category. We treat ODP categories as potential topics for our crawling experiments. The ODP relevant set consists of URLs that have been judged relevant to the category by human editors of ODP. We filter the content file to remove categories that contain words such as Regional, International, and Adults in its path (for example, the Regional:Asia:Education category is filtered out). This is needed to avoid the many semantically similar categories, sites involved in search engine spamming, as well as pages with non-English content. We also filter categories that list less than 40 external URLs so that

1. <http://www.cs.waikato.ac.nz/ml/weka/>.
2. <http://dmoz.org>.

we have topics with a critical mass of URLs for training and evaluation. From the remaining categories, we randomly select 100 categories and the associated ODP relevant sets. These selected categories will serve as topics for our crawling experiments. We further divide the ODP relevant set for a selected topic into two random disjoint subsets. The first set is the *seeds*. This set of URLs will be used to initialize the crawl. The crawlers move out from the seed pages to fetch other pages using the hyperlinked structure of the Web. The seeds also serve as positive examples for training the classifiers. The second set contains the *targets*. It is a holdout set that will not be used either to train or to seed the crawler. The seed set contains 20 URLs. The rest of the (20 or more) external URLs make up the targets. These targets will be used only for evaluation of crawl performance. The negative examples for a given topic are a random subset of positive examples from all of the selected topics excluding the given topic. The number of negative examples is kept at twice the number of positive examples.

6 PERFORMANCE METRICS

The output of a crawler is a temporal sequence of pages crawled. Any evaluation of crawler performance is hence based on this output. The key question for evaluation is: How do we judge the relevance of crawled pages? If the Web was a small controlled collection, we could use human judges to classify all pages as relevant or irrelevant to a given topic. With this knowledge, we could estimate the precision and recall of a crawler after crawling t pages. The precision would be the fraction of pages crawled that are relevant to the topic and recall would be the fraction of relevant pages crawled. However, the Web is neither controlled nor small. The relevant set for any given topic is unknown and, hence, the true recall is hard to measure. The precision could be measured using manual relevance judgement on the crawled pages. However, the manual relevance judgement for each of the crawled pages is extremely costly in terms of man-hours when we have millions of crawled pages spread over more than 100 topics. Even if we sample for manual evaluation, we will have to do so at various points during a crawl to capture the temporal nature of crawl performance. Hence, even such samples would be costly to evaluate over 100 topics. Hence, the two standard information retrieval (IR) measures, recall and precision, can only be estimated using surrogate metrics. Below, we describe *harvest rate* which we use as an estimate of precision and *target recall* which we use as an estimate of recall.

Harvest Rate. Harvest rate estimates the fraction of crawled pages that are relevant to a given topic. Since manual relevance judgment of Web pages is costly, we depend on multiple classifiers to make this decision. That is, we use a set of *evaluation classifiers* that act as a committee of automated judges to decide on the relevance of a crawled page based on a majority vote among them. We use three evaluation classifiers corresponding to three popular classification schemes—Naive Bayes, Support Vector Machine, and Neural Network. The evaluation classifiers are the “best” versions of each of these schemes based on a previous study on topical crawlers [31]. When compared with the crawling classifier, the evaluation classifiers are trained on a larger set (at least double in

number) of examples and, hence, are more “informed” about the topic. To explain, for each topic, we take one classifier at a time and train it using the pages corresponding to the entire ODP relevant set (instead of just the seeds) as the positive examples. The negative examples are obtained from the ODP relevant sets of the other topics. The negative examples are again twice as many as the positive examples. These trained classifiers are called evaluation classifiers to distinguish them from the crawling classifiers that are used to guide the crawlers. Harvest rate, $\mathcal{H}(t)$, after crawling the first t pages is then computed as:

$$\mathcal{H}(t) = \frac{1}{t} \sum_{i=1}^t r_i, \quad (1)$$

where r_i is the binary (0/1) relevance score for page i based on a majority vote among the evaluation classifiers. In order to obtain the trajectory of crawler performance over time, the harvest rate is computed at different points during the crawl [8], [26]. Here, time is estimated by t , the number of pages crawled. We have one such trajectory for each topic and each crawler. However, to provide an idea of the crawler’s performance in general, we average the harvest rate at various points over the 100 topics chosen for the experiment and draw this average trajectory over time for each crawler. We also compute the standard error around the average harvest rates. The performance trajectory can help an application designer decide on an appropriate crawler based on the length of crawl required by the application. For example, a crawler may perform well for crawls of a few hundred pages but poorly for longer crawls. We note that our harvest rate metric builds on that suggested by Chakrabarti et al. [8]. However, it differs in two critical aspects. Chakrabarti et al. [8] used a single classifier to make relevance judgments while we use three different classifiers. Secondly, we use classifiers that are trained on a larger set of examples compared to the classifier used for crawling. In contrast, they used exactly the same classifier (trained on the same examples) to judge the crawled pages and also to guide the crawler.

Target Recall. Target recall [30], [37] is an estimate of the fraction of relevant pages (on the Web) that are fetched by a crawler. As described earlier, true recall is hard to measure since we cannot identify the true relevant set for any topic over the Web. Hence, we treat the recall of the target set, i.e., target recall, as an estimate of true recall. If targets are a random sample of the relevant pages on the Web, then we can expect target recall to give us a good estimate of the actual recall. The target recall, $\mathcal{R}(t)$, after first crawling t pages for a given topic is computed as:

$$\mathcal{R}(t) = \frac{|C(t) \cap T|}{|T|}, \quad (2)$$

where $C(t)$ is the set of first t pages crawled, T is the set of targets, and $|T|$ is the number of targets. As with harvest rate, we compute the average target recall, the standard error, and represent the crawler’s performance as a trajectory over time.

7 CHOICE OF CLASSIFIER

We have previously studied several versions of three popular schemes of classification algorithms—Naive Bayes,

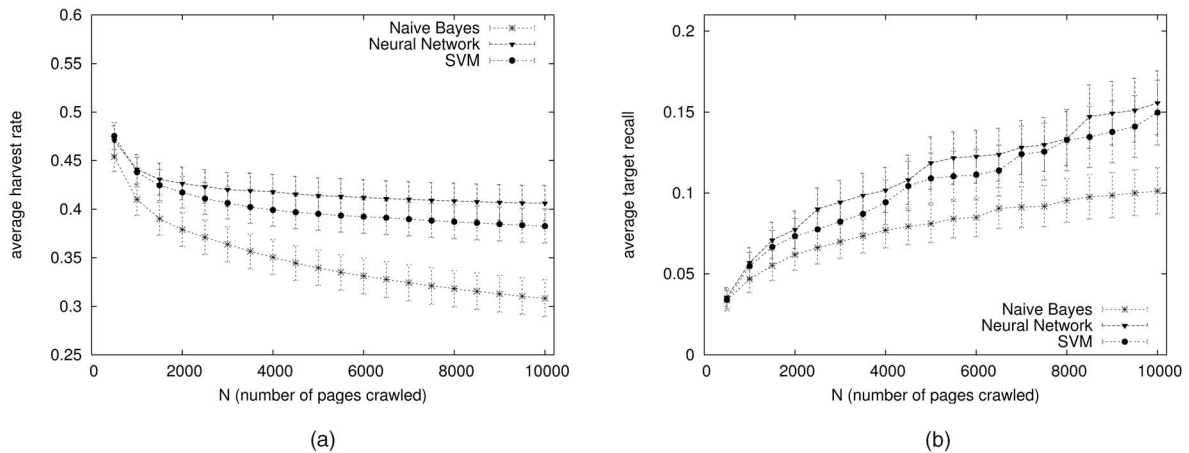


Fig. 4. Comparing classification schemes (from [31]). (a) Average harvest rate. (b) Average target recall.

Support Vector Machine and Neural Network—in their ability to guide a topical crawler. In a multistage study [31], we first explored the parameter space for each scheme and identified the best version (among those studied). In the second stage, we compared the schemes based on their best versions. We note that the link context derivation technique used by all of the crawlers in that study was to use the entire parent page of an unvisited URL as its context. As mentioned before, this strategy was used by other researchers [8], [13]. Figs. 4a and 4b show the performance of the best versions from each scheme. The points in these plots show the average harvest rate or average target recall over 100 topics obtained from the ODP. The error bars show ± 1 standard error. We find that, for most of the crawl (up until the end), the Naive Bayes crawler performs significantly more poorly as compared to SVM or Neural Network crawlers, both on average harvest rate as well as average target recall. The differences between the SVM and Neural Network crawlers are statistically insignificant. The significance of results is based on one-tailed t-tests using $\alpha = 0.05$. We also computed the average training times for each of the three classifiers. The Neural Network took on an average 56.637 ± 1.988 seconds to train, while the SVM took 0.518 ± 0.009 , and the Naive Bayes took 0.310 ± 0.005 . However, since training is an “offline” process, this difference in time may not be an issue for long crawls. Given a choice between Neural Network and SVM for this study, we opt for the latter given its popularity in current research involving both text classification [15], [19] and topical crawling [20].

8 EXPERIMENTS

We experiment with classifier-guided topical crawlers that use different techniques for obtaining link contexts. The classifier is fixed to be a Support Vector Machine with a linear kernel that is trained (before crawling) with positive and negative example Web pages.

Each of the context derivation techniques explored in this research yields a different crawler. Observe that, in each case, the classifier is trained identically, i.e., using the full Web pages of the same set of positive and negative examples provided for each topic. However, during the crawl, these classifiers will be applied on varying Web page

subsets, i.e., varying link contexts around the URLs being scored. This model of classifier-based crawler that we test is in itself novel. In contrast, we could have chosen an approach where the training is also done using different portions of Web pages. Such an approach would require an expert to provide us with the relevance data for these different portions of Web pages and, thereby, generate an appropriate training set. Instead of this level of complexity, which would be hard to implement, we train with full pages and vary only the link contexts to which the classifier is applied during crawling. (However, in Section 10.4, we offer a preliminary exploration of an alternative approach for training that uses smaller link contexts for training.) Since we train with the full pages, we can use the ODP data where humans have made relevance decisions on full pages. Moreover, the model we test is a simple and natural extension of classifier-guided crawlers [8], [13] that use the entire parent page as the link context. We set the number of seeds (positive examples) to 20. Also, we have twice as many negative examples as the positive examples. We will use the following context derivation techniques:

Full Page. The simplest context derivation technique is to use the entire parent page as the context of the URLs within it. In that case, all of the unvisited URLs extracted from the parent page are assigned the same score when added to the frontier. However, the frontier itself will have URLs with a variety of scores since they may come from different parent pages. Again, this simple context derivation technique has been used by Chakrabarti et al. [8] and Diligenti et al. [13], among others. We will use the crawler based on this technique as a base-line for performance analysis.

Text Window. This is a general technique of using arbitrary size windows of text around a hyperlink. It is a popular technique for deriving link contexts for a variety of applications such as topical crawling [18], [17], classification [16], indexing [23], [5], [4], and link analysis [6]. For each hyperlink URL, we will use a window of T words around its appearance on the parent page as the context. Whenever possible, the window will be kept symmetric with respect to the anchor text. That is, the text window will have $T/2$ words that appear before and another $T/2$ words that will appear after the anchor text. Hence, the size of the link context will be $T + |\text{anchortext}|$ words and will always include the entire anchor text. We will explore values of

$T = 10, 20, 40$. Each value of T corresponds to a separate context derivation technique.

Tag Tree. The tag tree representation of a Web page may be used to derive link contexts. Fig. 1 shows a tag tree corresponding to an HTML source. The `<html>` tag forms the root of the tree. Tidying an HTML page is an essential precursor to analyzing the HTML page as a tag tree [7], [28]. Many Web pages contain badly written HTML. For example, a start tag may not have an end tag, or the tags may not be properly nested. In many cases, the `<html>` tag or the `<body>` tag is all together missing from the HTML page. The process of converting a “dirty” HTML page into a well-formed one is called tidying an HTML page.³ It includes inserting missing tags and reordering tags in the “dirty” page. Further, the text tokens are enclosed between `<text>...</text>` tags which makes sure that all text tokens appear as leaves on the tag tree. While this step is not essential for mapping an HTML page to a tree structure, it does provide some simplifications for deriving contexts.

Once we have a tag tree corresponding to a crawled Web page, we use its hierarchical structure for deriving contexts of hyperlink URLs within it. We have previously studied various context derivation techniques for the purpose of describing the destination page [28] of a hyperlink. One of the main results of the study was that an adaptive context derivation technique that moved up the tag tree when the context had few words (≤ 2) provided better descriptions of the destination page than text window based techniques. We will explore the same technique here for topical crawling.

The following steps will be taken to derive the context of a URL from a given page. We begin by looking at the text within the anchor tag containing the URL. If the size of this text is less than or equal to MIN words, we move to the tag that contains the anchor tag (a level above in the tag tree). We then extract all the text that appears within this higher tag and check if the text size crosses the threshold of MIN words. If not, we move up yet another level. This process repeats until either we find a tag that contains within it a text of more than MIN words or we reach the top of the tree `<html>` tag). Once the text extraction process ends, we truncate (if necessary) the extracted text to MAX words. As explained earlier, we will set MIN (the minimum number of words in the context) to be 2. MAX will be decided based on the results from our text window experiments. Although the text windows do not produce the link context in a manner similar to tag tree, we rely on them to obtain what is hopefully a reasonable value for MAX . We do this in order to reduce the number of experiments. Hence, the derived context for a URL will be no longer than MAX words and no less than MIN words.

Based on the above mentioned context derivation techniques, we will perform the following experiments:

1. Text Window Experiments. We will compare three crawlers that use different text window sizes ($T = 10, 20, 40$) to derive link contexts. These comparisons will be made based on the 100 ODP topics and 10,000 pages will be crawled for each topic. We have chosen different text windows sizes (doubling each time with a minimum of 10 words) to find a

good text window size. Observe that Chakrabarti et al. [6] used a text window of size 50 (approximately 6-8 words) bytes around the anchor to obtain the link context for the purpose of weighted link analysis. We feel that text windows that respect word boundaries are more reasonable than those based on the number of bytes since words better approximate semantic units.

2. Text Window Combo Experiments. We combine the classifier-based score for different text windows ($T = 10, 20, 40$) with the classifier score for the entire page and use the combination as the score of the corresponding unvisited URL. We compute the combination *score* as:

$$score = \beta * page_score + (1 - \beta) * context_score, \quad (3)$$

where β is the relative weight assigned to the score for the entire page or *page_score*. We set $\beta = 0.25$ (as previously found to be effective [30]), which means that 75 percent weight is given to the score for the derived context or *context_score*. The crawlers defined by this score combining technique will be evaluated on the 100 topics with 10,000 pages crawled for each topic. It may be observed that our use of a classifier (trained on example Web pages) to score link contexts (that are subsets of a full page) is a previously unexplored strategy. The closest is Shark search [17] which combines the scores given to link context and full page. However, it does not use classifiers to derive the scores. Hence, this method of combining classifier scores for full page and smaller link context is new and untested.

3. Tag Tree Combo Experiments. We derive the link context using the tag tree hierarchy as outlined before. We then combine the score for the context with that for the entire page using the (3). The parameter MAX will be decided based on the results for different Text Window Combo crawlers. Again, this crawler is evaluated on the 100 topics with 10,000 pages crawled per topic.

The baseline crawler for each of these experiments is a Full Page SVM crawler that uses the entire parent page as the context for the hyperlinks in it.

9 RESULTS

We now present the results for each of the experiments described in Section 8. A thorough analysis that sheds light into various aspects of the results and provides possible explanations for them is deferred till Section 10.

9.1 Text Window Experiments

Figs. 5a and 5b show performances of our crawlers using text windows of different sizes as link contexts. We name the crawlers Text Window (T) crawler where T is the size (in number of words) of the text window selected around a hyperlink. We find that none of the Text Window ($T = 10, 20, 40$) crawlers show a significant performance improvement over using the Full Page crawler. On the contrary, Text Window (10) crawler shows significantly poorer performance when compared with the Full Page

3. <http://www.w3.org/People/Raggett/tidy/>.

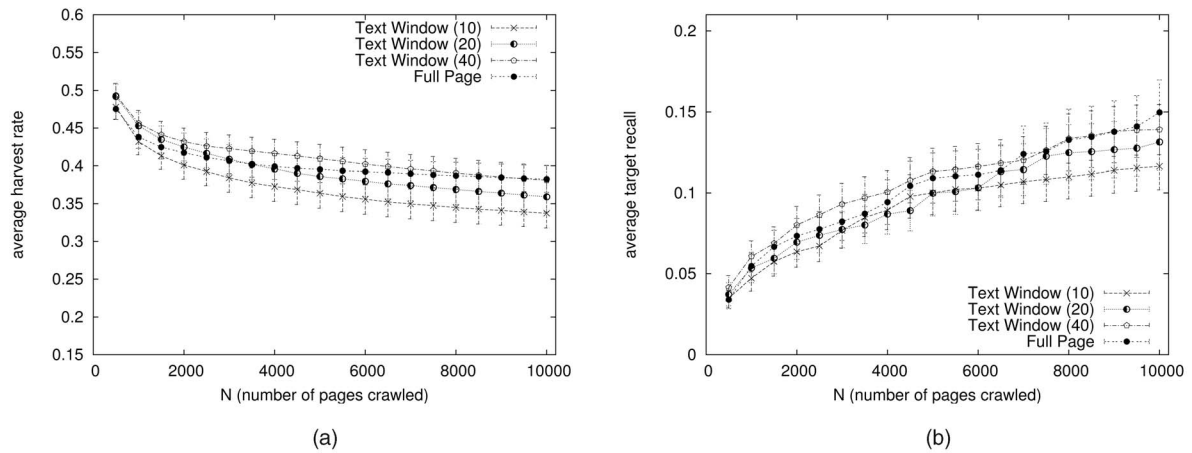


Fig. 5. Text Window. (a) Average harvest rate. (b) Average target recall.

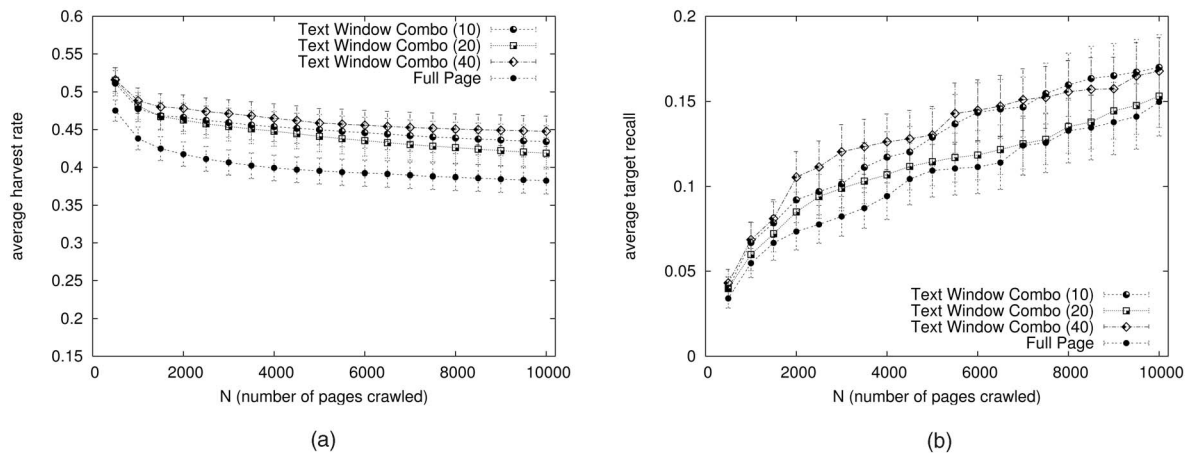


Fig. 6. Text Window Combo. (a) Average harvest rate. (b) Average target recall.

crawler. Based on one-tailed t-tests, this effect is significant both on average harvest rate (at $\alpha = 0.05$) and average target recall (at $\alpha = 0.1$).

We conclude from these results that there is no benefit from using text windows of these sizes over using the entire page as link context.

9.2 Text Window Combo Experiments

The performance of our SVM crawlers that use a combination of the (entire) page score and the scores given to text windows of different sizes is shown in Figs. 6a and 6b. Again, we name the crawlers Text Window Combo (T) crawler where T is the size (in number of words) of the text window selected around a hyperlink.

We find that all of the Text Window Combo ($T = 10, 20, 40$) crawlers significantly (at $\alpha = 0.1$) outperform the Full Page crawler on the average harvest rate (Fig. 6a). For temporal analysis, Fig. 7 is a summary display of the p-values of one tailed t-tests (on average harvest rate) at various points (N) during the crawl. The alternative hypotheses of these tests are that the Text Window Combo (T) crawlers, with different values of T , outperform the Full Page-based crawler. Fig. 6b shows that all three Text Window Combo crawlers perform the same as the Full Page crawler on the average target recall metric for most of the crawl. However, the Text Window Combo (40) crawler for a small part of the

crawl (2,000-3,500 pages) significantly ($\alpha = 0.05$) outperforms the Full Page crawler on average target recall.

Fig. 8 shows plots that compare the performances of the Text Window Combo ($T = 10, 20, 40$) crawlers with the corresponding Text Window ($T = 10, 20, 40$) crawlers on average harvest rate (Figs. 8a, 8c, and 8e) and average target recall (Figs. 8b, 8d, and 8f). We find that, for average harvest rate, all Text Window Combo crawlers significantly outperform their Text Window counterparts for most of the crawl. For average target recall, however, only the Text Window Combo (10) crawler significantly outperforms corresponding Text Window (10) crawler. Hence, Fig. 8 and Fig. 6 suggest that crawlers based on a combination of text window and full page outperform (at least on one of the performance metrics) both the crawlers based on just the full page and the crawlers based on just the text windows. Improvements on target recall relative to the full page crawl are harder to achieve.

9.3 Tag Tree Combo Experiments

The aim of this experiment is to evaluate the merit of a crawler that derives link contexts using the tag tree hierarchy in Web pages. Based on Fig. 6a and 6b, the Text Window Combo (40) crawler performs as good as or better than any of the other Text Window Combo crawlers evaluated. Hence, we set 40 words as the maximum link context size for the tag tree-based context derivation

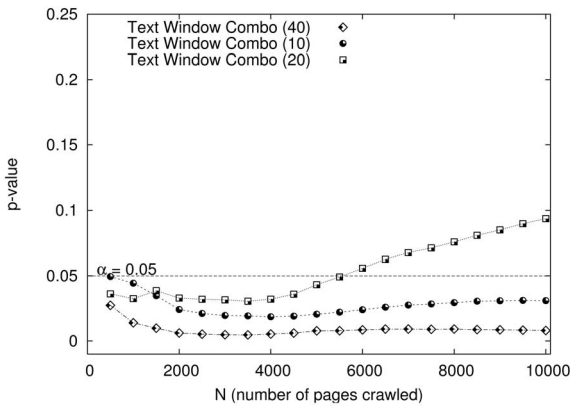


Fig. 7. p-values for one-tailed t-tests (based on average harvest rate) at various points during the crawls. The alternative hypotheses conjecture that Text Window Combo ($T = 10, 20, 40$) crawlers outperform the Full Page crawlers.

method (see Section 8). Hence, MAX is 40 and MIN is 2. The Tag Tree Combo crawler combines the derived link context with the entire page in a manner similar to Text Window Combo crawlers. Figs. 9a and 9b compare the performances of the Tag Tree Combo crawler and the Full Page crawler. On average harvest rate (Fig. 9a), the performance of the Tag Tree Combo crawler is insignificantly different from that of the Full Page crawler. However, on average target recall (Fig. 9b), the Tag Tree Combo crawler significantly ($\alpha = 0.05$) outperforms the Full Page crawler for a majority of the crawl. Although, toward the end of the crawl (last 500 pages), the significance reduces below $\alpha = 0.10$ level. Note that, among all the crawlers we have considered up until now, the Tag Tree Combo is the first crawler to significantly outperform the Full Page crawler on the average target recall metric for a majority timespan of the 10,000 page crawl.

Fig. 10 compares the Tag Tree Combo crawler with the Text Window Combo (40) crawler relative to the Full Page crawler on average target recall. The p-values indicate the significance of performance differences between the crawlers in the plot and the Full Page crawler. The plot shows that the Tag Tree Combo crawler significantly outperforms the Full Page crawler for most of the crawl both at $\alpha = 0.05$ and $\alpha = 0.10$. In contrast, the Text Window Combo (40) crawler shows only a couple of small bursts of significantly better performance than the Full Page crawler.

We note that all of the Text Window Combo ($T = 10, 20, 40$) crawlers significantly ($\alpha = 0.1$) outperformed the Full Page crawler on the average harvest rate (see Fig. 7). These results suggest that there may be a tradeoff. The Tag Tree Combo crawler helps us outperform Full Page on the average target recall while Text Window Combo ($T = 10, 20, 40$) crawlers outperform Full Page on average harvest rate.

10 DISCUSSION

We now discuss our results from different angles. Some of our analyses involve additional exploratory experiments.

10.1 Link Context Quality

An interesting conclusion from Figs. 5, 6, 7, and 8 was that neither short segments of link context nor the entire page alone provide as good a performance as a combination of

these two sources of evidence. We note, based on a sample of about 1 million crawled pages, that the average size of pages crawled is 247 words. Our largest text window of 40 words is therefore about 1/6 the average size of a crawled page. Hence, our text windows should be able to discriminate between URLs appearing in different portions of a Web page. However, it is possible that the very small text windows used alone (like that produced with $T = 10$) fail to capture important cues in other parts of a Web page that make the entire page relevant to the given topic.

In a previous study [28], we measured the “quality” of link context derived using different techniques for 10,549 *sample URLs*. Each sample URL corresponded to a random external link from a distinct ODP category/topic. The pages corresponding to the sample URLs were called *sample pages*. We found the back-links for each of the sample pages using the Google Web APIs.⁴ A random back-link was then chosen for each of the samples pages and the page corresponding to the back-link (or *back-link page*) was fetched. Different techniques for deriving link contexts were applied on the back-link page. The contexts were derived for the hyperlink leading to the original sample page from its selected back-link page. The quality of the derived contexts was measured through the following metrics:

Average context similarity. Given a context derivation technique \mathcal{M} , we measure the *context similarity* for each of the sample URLs. The context similarity is the cosine similarity of the derived context to the manual description of the sample URL. We obtain the manual description from ODP itself which provides such descriptions of each of the external URLs that are written by the human editors. The average context similarity for \mathcal{M} is then computed as:

$$avg_context_sim(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^n sim(c_i^{\mathcal{M}}, d_i),$$

where n is the sample size, $c_i^{\mathcal{M}}$ is the context derived using the technique \mathcal{M} for sample URL i , and d_i is the manual description of the sample URL i . The average context similarity is used as a measure of average quality of contexts obtained using a method.

Zero-similarity frequency. The zero-similarity frequency measures the frequency with which a method obtains a context that has no similarity to the manual description. This frequency is measured as a percentage of the sample size. While zero lexical similarity may not necessarily mean zero semantic similarity (due to variable word usage), it does mean that the method failed to provide many of the important words that were used by an expert editor to describe a given page. It is possible that a method provides us with high quality contexts for some URLs but provides us with little or no information about many others. The zero-similarity frequency is computed as:

$$zero_sim_freq(\mathcal{M}) = \frac{100}{n} \sum_{i=1}^n \delta(sim(c_i^{\mathcal{M}}, d_i)),$$

where the $\delta(x)$ is 1 if $x = 0$, and is 0 otherwise. Unlike average context similarity, we would like to have a low value of zero-similarity frequency.

Figs. 11a and 11b show the “quality” of different text window-based link context derivation techniques [28]. The

4. <http://www.google.com/apis/>.

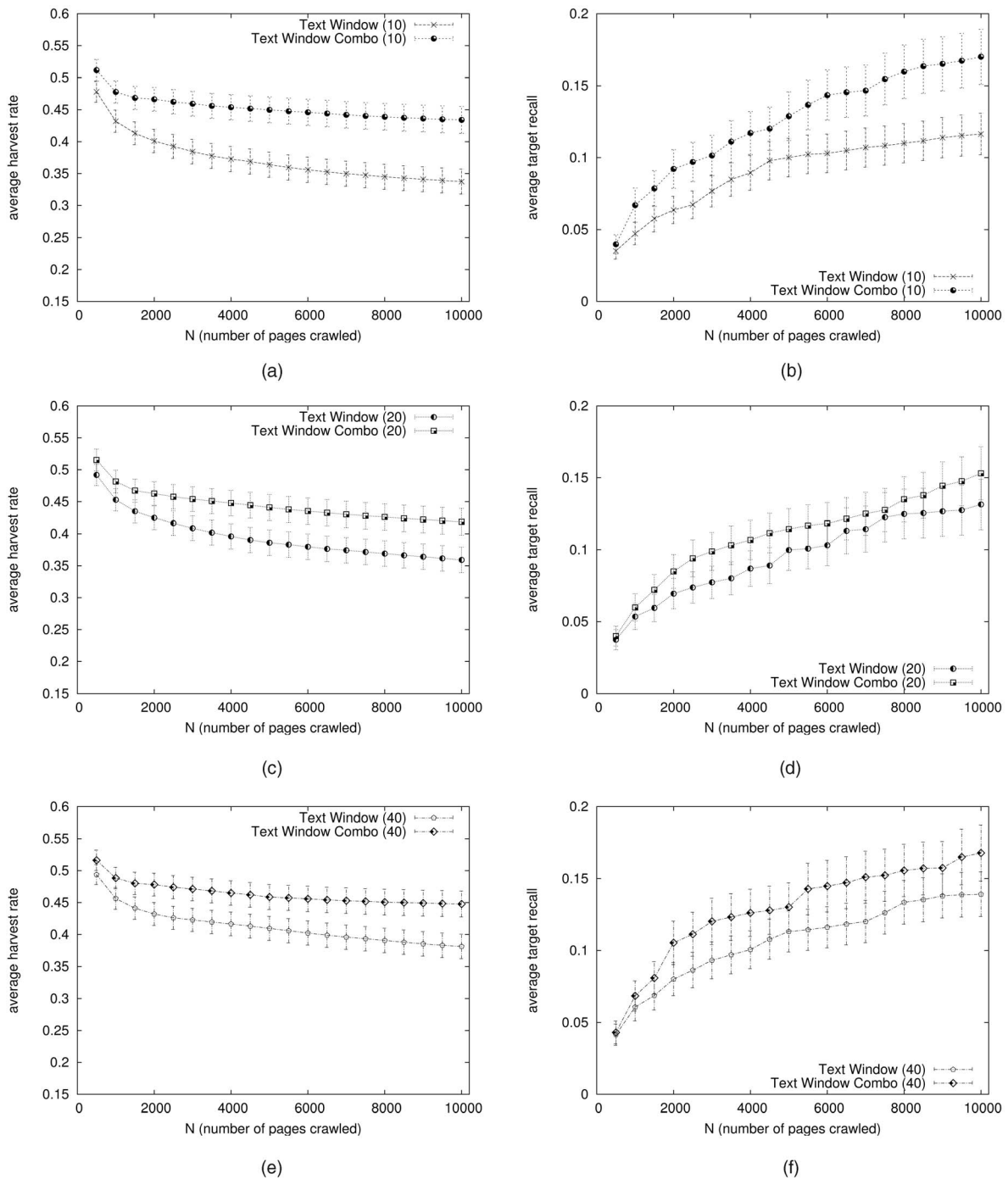


Fig. 8. Performance comparison between Text Window ($T = 10, 20, 40$) crawlers with corresponding Text Window Combo ($T = 10, 20, 40$) crawlers.

text windows used in [28] are a little different from the ones used in this study (described in Section 8). In particular, the text windows in [28] may or may not include the entire anchor text. Whereas, the text windows described in Section 8 always include the entire anchor text and their size is $T + |\text{anchortext}|$ words. However, this minor difference has little effect on the following analysis that applies the observations from previous work [28] to the current crawl data.

In Fig. 11a, we find that the smaller text windows ($T = 10, 20$) provide significantly higher average context similarity when compared with the larger text window of size 40 or the full page. Indeed, the words immediately around a hyperlink tend to provide the best description of the destination page.

Fig. 11b paints a different picture. We find that increasing the text window size from 10 to 40 decreases the zero-similarity frequency by half. If we use the entire page as the link context, the zero-similarity frequency falls to 2.75 percent. In other words, for only 2.75 percent of the sample pages, the back-link page has no similarity to the manual description of the sample page. Comparing the two figures (cf. Figs. 11a and 11b) from our previous study [28], we observe two opposing forces. On one hand, using the entire page as link context substantially decreases the possibility of completely missing important cues about the destination page of a hyperlink within it. On the other hand, using a smaller link context provides on an average more precise information about the

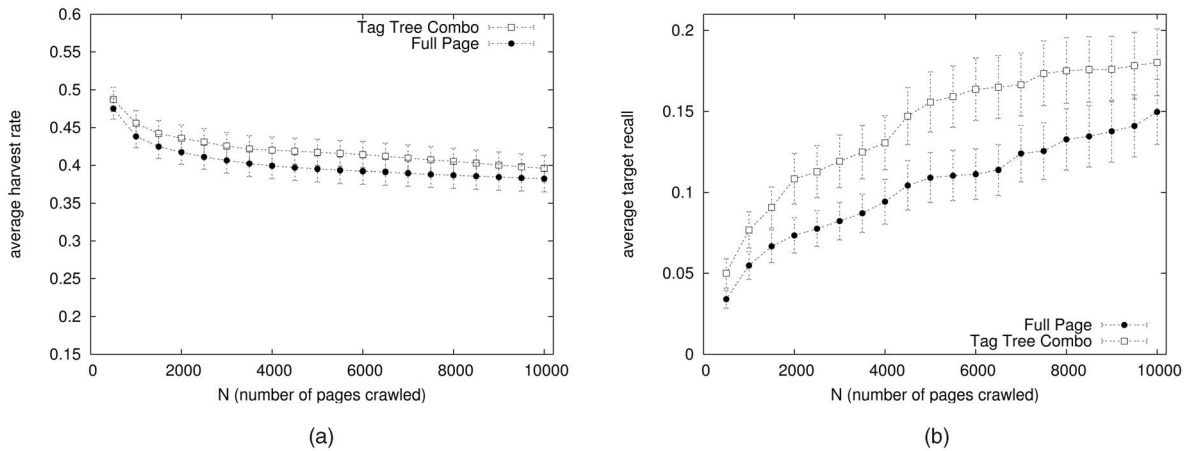


Fig. 9. Tag Tree Combo. (a) Average harvest rate. (b) Average target recall.

destination page. A balance between the two appears necessary for finding effective link contexts for topical crawling. This could explain the better performance of topical crawlers based on the combination of link context and full page when compared with crawlers based on link context or full page alone (Fig. 8 and Fig. 6).

10.2 Topic Difficulty

In Section 9, we found an interesting tradeoff. For a major time segment of the 10,000 page crawls, the Tag Tree Combo crawler outperforms the Full Page crawler on the average target recall. However, it is the Text Window Combo ($T = 10, 20, 40$) crawlers that outperform the Full Page crawler on average harvest rate. A good average recall (where the averaging is done over topics) can be obtained by a crawler that is very successful in harvesting relevant pages for topics that have fewer relevant pages on the Web. For such topics, simpler crawling techniques that use the entire parent page as the context of links within it may not work well and better “focus” in the crawlers may be needed to steer towards the few relevant Web pages. To explore this tradeoff, we rank the topics by the target recall obtained with the Full Page crawler after crawling 10,000 pages. This rank gives a measure of difficulty of the topic for the Full Page crawler with respect to finding relevant pages. That is,

the topic with rank 1 is the one that has the worst target recall and is thus the most difficult topic for the Full Page crawler. We then split the topics into four groups of 25 topics based on their ranks such that the first group of 25 topics (1 – 25) is the most challenging 25 topics based on the Full Page crawler’s target recall. Fig. 12 shows the average target recall of the Tag Tree Combo (40) crawler and the Text Window Combo (40) crawler on each group of topics. We find an interesting pattern. The Tag Tree Combo crawler significantly ($\alpha = 0.06$) outperforms the Text Window Combo (40) crawler on the first 25 topics where the Full Page crawler had the worst target recall performance. In other words, these are topics where quality of link contexts may be important for obtaining good coverage/recall. Moreover, our use of tag tree hierarchy provides a mechanism to get contexts of dynamic size (with *MAX* and *MIN* limits) that improve coverage when compared with a static text window. Through that mechanism, the Tag Tree Combo crawler gains the extra “focus” needed to get better coverage on these difficult topics. On the flip side, we note that, when the topics are easiest for the Full Page crawler (76 – 100), the Text Window Combo (40) crawler shows a better (though not significant) performance than the Tag Tree Combo. Apparently, for these topics, the extra work of moving along the hierarchy is producing little reward if any. Thus, the tradeoff between the two types of crawlers may depend upon the nature of the topic.

10.3 Long Crawls

For each experiment, given the multiple crawlers tested, we crawl up to a few million pages over the 100 topics. However, each individual crawl is limited to 10,000 pages. Hence, all of our analysis until now is limited to crawls of 10,000 pages per topic and per crawler. While crawls of 10,000 pages may be sufficient for some applications [29], [32], there are several applications in business intelligence, Web mining, and vertical search engines that might benefit from an order of magnitude more pages per topic. We note that most of the previous research in topical crawling has been limited to 1,000-25,000 page crawls and/or a small number of topics. Unfortunately, if a test bed comprises just a couple of topics, experiments, even with longer crawls, would lack statistical robustness. As a compromise between the number of topics and the number of pages crawled per topic, we pick 10 topics at random from the 100 topics used in the previous

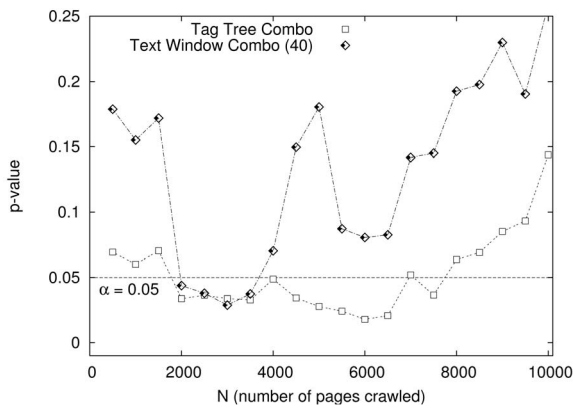


Fig. 10. p-values (based on average target recall) for one-tailed t-tests at various points during the crawls. The p-values provide a measure of significance of performance differences between the crawlers in the plot and the Full Page crawler.

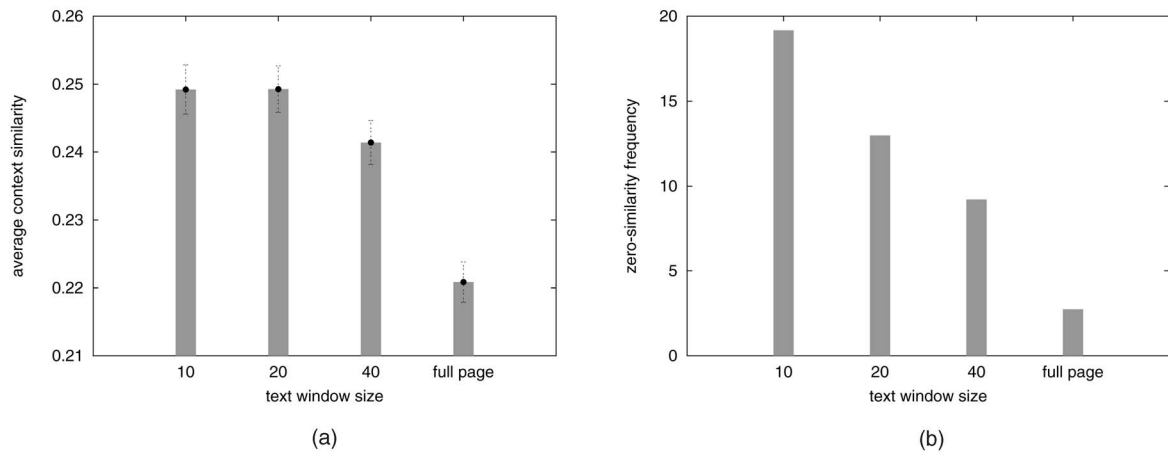


Fig. 11. Quality of text window-based link context derivation techniques (from [28]). (a) Average context similarity. (b) Zero-similarity frequency. Note that average context similarity can vary between 0 and 1, while zero-similarity frequency, being a percentage, can vary between 0 percent and 100 percent.

experiments and now conduct an experiment with crawls of 100,000 pages per topic. We use the Text Window Combo (40) and Tag Tree Combo (40) crawlers that performed well in previous experiments along with the Full Page crawler as a baseline. Figs. 13a and 13b show the performance of these three crawlers on the two metrics. Interestingly, the long-term relative performances of crawlers for large crawls turn out to be very similar to those seen in Fig. 6 and Fig. 9. In particular, with the long crawl of 100,000 pages, we find that the Text Window Combo (40) crawler significantly outperforms (at $\alpha = 0.05$) the Full Page crawler by the end of the crawl (see Fig. 13a). This is consistent with Fig. 6a. Also, in Fig. 13b, we find that the Tag Tree Combo (40) crawler almost consistently outperforms (sometimes significantly at $\alpha = 0.1$) the Full Page crawler. This is consistent with Fig. 9b. Hence, the longer 100,000 page crawls revalidate the main findings of the shorter 10,000 page crawls. The results for longer crawls further affirm the robustness of shorter crawl results obtained by averaging over a large number of topics.

10.4 Back-Links

Our experiments thus far have involved classifiers trained on collections of full pages and applied (during crawling) to link contexts of varying size (see Fig. 3 for this model). We now

explore the effect of changing this architecture. Instead of training on the full pages, we wish to determine the effect of training on smaller link contexts. This would make the training and testing phases of our crawlers, in a sense, mutually consistent. This architectural change is motivated by the fact that we obtained equal or worse performance using smaller text window based crawlers as compared to the Full Page crawler (see Fig. 5). Perhaps this result was caused by greater “incompatibility” in the former, i.e., between the full page trained classifiers and the smaller link contexts scored during the crawl. In other words, our original architecture may be biased in favor of crawlers scoring full pages. To investigate this issue, we pick the best text window size identified in Section 9.1 (40 words) and train our classifiers using chunks of texts of the same size. Moreover, we take these text chunks from Web pages that *point to* our relevant pages for a given topic, i.e., their back-link pages. Since the crawler uses link context to estimate the benefit of downloading the relevant page, we are, in essence, using the back-links to simulate relevance judgments. Thus, given a relevant page, we first gather its top 10 back-links using the Google Web APIs. Next, we filter out back-link URLs that contain the terms `directory.google.com` or `dmz to` to avoid mirrors of ODP. We then download the page corresponding to each retained back-link URL. Finally, we extract the text window around the hyperlink to the relevant page for which the back-link was obtained. We treat each text window thus extracted as a *potential positive link context* for classifier training. Unfortunately, since we obtain up to 10 back-links for each positive example, this process typically yields a much larger training set per topic than that used in the previous experiments. To keep the comparison balanced, we restrict our pool of positive link contexts to the same size as the pool of positive full page examples (i.e., 20) in the previous experiments. Hence, we randomly select 20 back-links from which we derive the positive link contexts. The *negative link contexts* for a topic are obtained randomly from the pool of positive link contexts for the other topics. As before, for a given topic, the number of negative link contexts are kept at twice the number of positive link contexts. Fig. 14 illustrates the idea of deriving positive links contexts starting from positive examples.

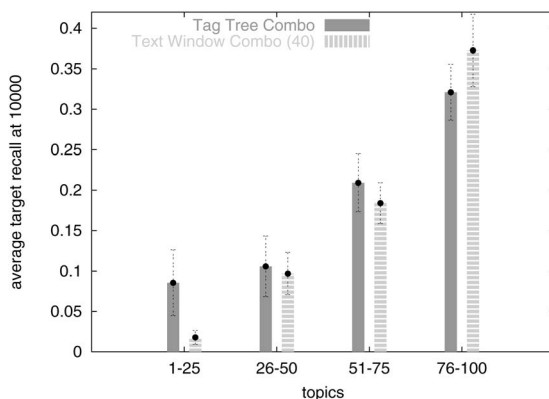


Fig. 12. Average target recall on four groups of 25 topics. The groups are ordered such that 1-25 are the 25 topics on which Full Page crawler's target recall is the worst.

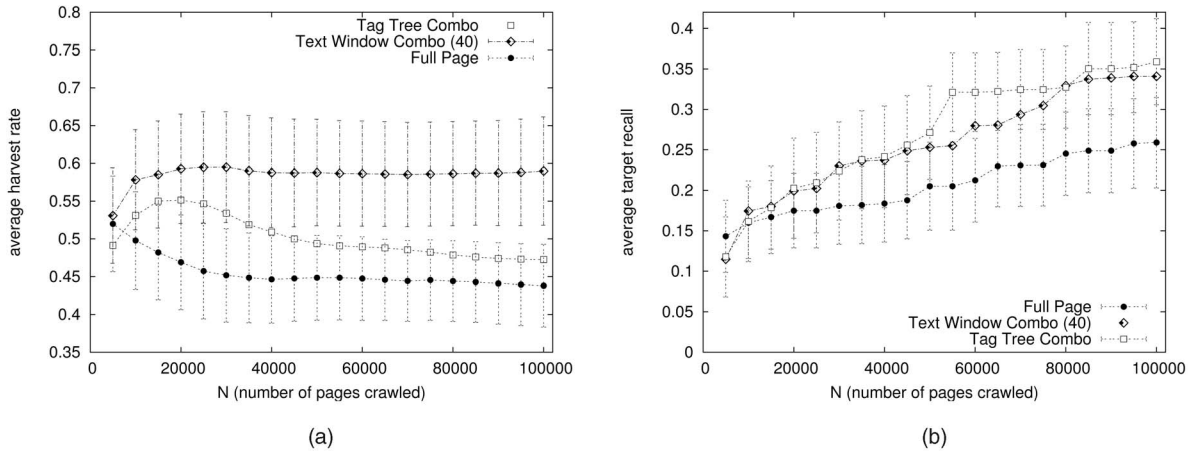


Fig. 13. Performance for crawls of 100,000 pages. (a) Average harvest rate. (b) Average target recall. To avoid clutter, we only show the error bars around the best and worst performing crawlers for each metric.

Once a classifier is trained for a topic, based on its positive and negative link contexts, it is used to guide a Text Window BackLink (40) crawler starting at the same seeds as before. The crawler uses the trained classifier to score link contexts based on text windows of size 40 words to prioritize the corresponding URLs. Hence, other than the training of the classifier, the Text Window BackLink (40) crawler works in exactly the same way as the Text Window (40) crawler. We run the Text Window BackLink (40) crawler and the Text Window (40) crawler for each topic. Given that we have changed the crawler classifier architecture significantly, we conduct this experiment with all 100 topics fetching 10,000 pages per crawl.

Figs. 15a and 15b show the performances of the two crawlers. Based on average harvest rate (Fig. 15a), we find that the Text Window (40) crawler performs significantly better (at $\alpha = 0.1$) than the Text Window BackLink (40) for almost all of the 10,000 page crawl. On average target recall (Fig. 15b), the difference between the two crawlers is statistically insignificant for the entire length of the crawls. Hence, the two crawlers perform the same for average target recall while Text Window (40) crawler, our original configuration, significantly outperforms the Text Window BackLink (40) crawler on average harvest rate. Thus, we conclude that, even when classifiers are used to score 40 word link contexts for prioritizing our frontier of URLs, training on the smaller contexts derived using the back-links of positive examples is less effective than training on the full page content of the same positive examples. In addition, the Text Window BackLink (40) crawler adds the cost of accessing a suitable search engine and is dependent on its availability. In hindsight, we should expect the positive link contexts extracted from back-link pages to be noisy. As noted in Section 8, because ODP pages are manually selected, we have relevance judgments for the content at the full page level. In contrast, there is no human relevance judgment available for the smaller link contexts. Our simulation through back-links does not add power to our crawlers. More generally, these results support our original model of using a classifier trained on full pages to later score smaller link contexts while guiding crawlers. This is a previously untested model for classifier-based crawlers that we suggest in the current study.

As an aside, we note that back-links may be useful for topical crawling, but in a different way. In particular, since back-links may be hubs [21], they may help us obtain better seeds for starting the crawler (ones more likely to reach relevant pages). In fact, we have previously shown [30] that adding potential hubs derived from back-links as seeds can significantly improve crawler performance (even if the crawler is not a topical one).

10.5 Topical Performance

The emphasis of this study is on the general performance of classifier-guided topical crawlers using different link context definitions over a wide-range of topics. However, crawler performance can be expected to vary due to differences in topic attributes. One significant factor differentiating topics is likely to be the underlying “domain.” As a first step in exploring the impact of this feature, we divide our 100 topics based on the broad domains such as “Business,” “Computers,” “Sports,” and “Home,” which correspond to top-level categories of ODP. For robust analysis, we filter out domains with less than 10 topics (among the 100 topics). We are left with the following domains: “Shopping,” “Business,” “Computers,” and “Society.” Figs. 16a and 16b show the average performance (at 10,000 pages) of the Full Page, Text Window Combo (40), and Tag Tree Combo (40) crawlers over topics belonging to different domains. Fig. 16a shows

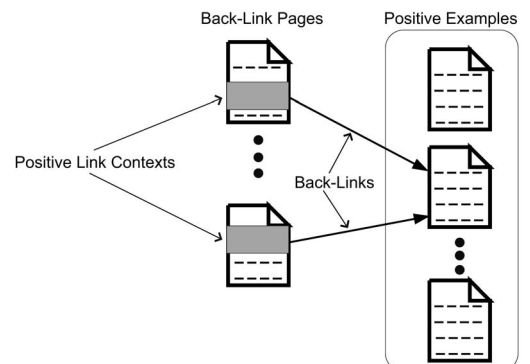


Fig. 14. Positive link contexts are obtained from pages corresponding to the back-links of positive examples.

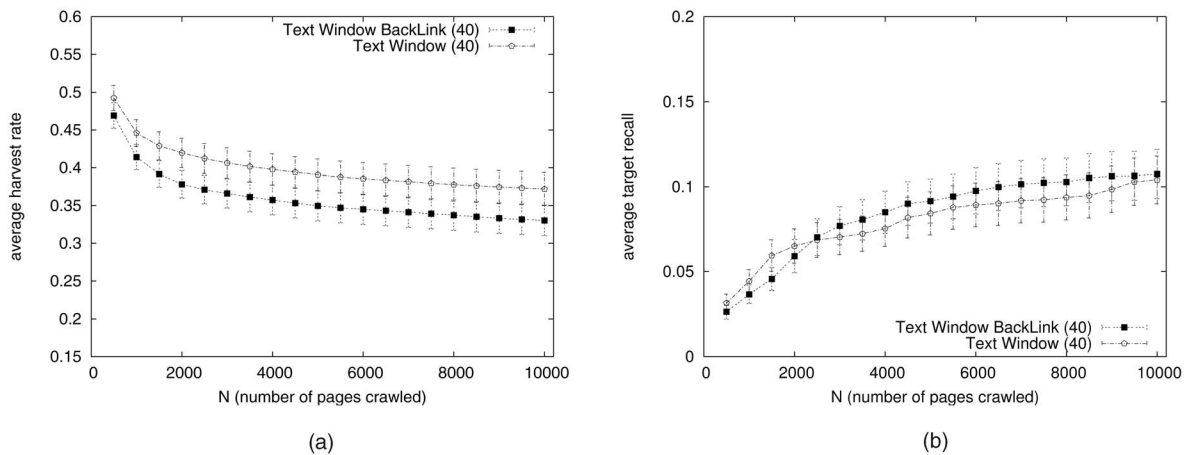


Fig. 15. Using back-links for training. (a) Average harvest rate. (b) Average target recall.

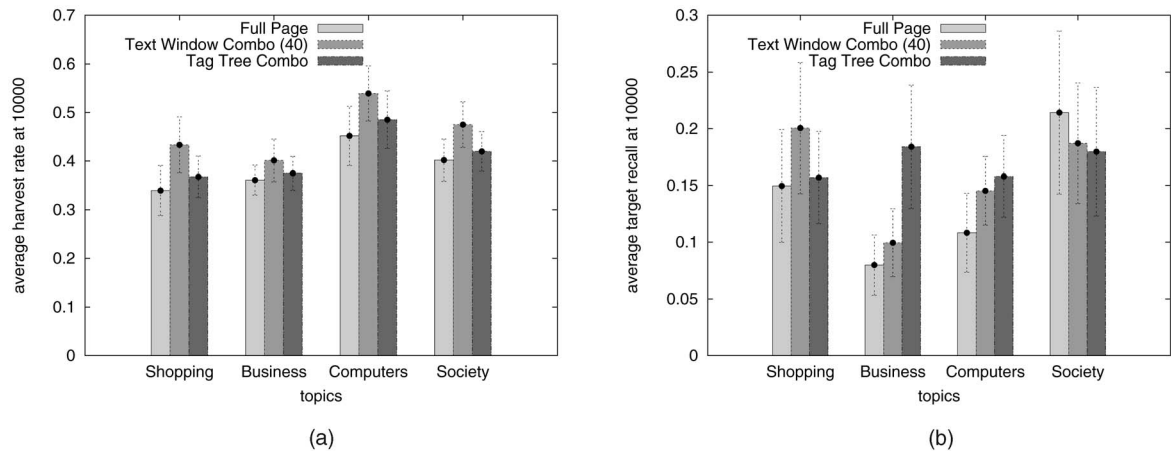


Fig. 16. Performance on different domains. (a) Average harvest rate. (b) Average target recall.

that, for each of the domains, there are no significant differences in average harvest rate achieved by the three crawlers. We further note that, in Fig. 16a, the order of performance of the crawlers remains the same across all domains with Text Window Combo (40) crawler performing the best and Full Page crawler performing the worst. This is consistent with Fig. 6a and Fig. 9a. We also find that all of the crawlers perform as good as or better on the “Computers” and the “Society” domains when compared to the “Business” and the “Shopping” domains. For example, the Text Window Combo (40) crawler performs significantly better (at $\alpha = 0.05$) in the “Computers” domain as compared to the “Business” domain.

The variations in crawler performances both within a domain and between domains is more evident for average target recall (see Fig. 16b). In particular, Full Page crawler and Text Window Combo (40) crawler perform significantly better (at $\alpha = 0.05$ and $\alpha = 0.1$, respectively) in the “Society” domain as compared to the “Business” domain. We speculate that this is due to the presence of more *collaborative* topics under “Society” and more *competitive* topics under “Business” [8]. A competitive topic is one where the relevant group of Web pages is expected to have fewer interconnections or links between them. For example, “pharmaceutical companies” as a topic is expected to be competitive. One does not expect, for example, Merck’s Web site to point to many pages from the Glaxo Smithkline

Web site. Collaborative topics represent the opposite notion. For example, Web pages relevant to a research area (e.g., Text Mining) are more likely to have interlinks. Hence, we may expect topical crawlers to achieve higher recall for collaborative topics. The opposite may be expected for competitive topics.

We also observe that the Tag Tree Combo (40) crawler is significantly the best when compared to the Full Page crawler (at $\alpha = 0.05$) and the Text Window Combo (40) crawler (at $\alpha = 0.1$) within the “Business” domain on average target recall (Fig. 16b). This is not observed in the other domains. A possible hypothesis to explain this could be that “Business” pages tend to more reliably reflect semantic information in their HTML structure and, hence, a crawler exploiting HTML DOM structure (such as Tag Tree Combo (40)) is able to perform better. This hypothesis is consistent with results from a previous study [30], where another tag tree-based crawler performed very well for business intelligence tasks. We plan to investigate this issue further in the future.

11 CONCLUSION

We have presented a systematic study involving various definitions of link context and their effect on the performance of SVM classifier-guided topical crawlers. This research is motivated by the fact that, while different

definitions of link contexts have appeared in the topical crawling literature, there is a large gap in terms of understanding their relative merits. Our flexible crawling infrastructure design allows for different link context derivation techniques to be easily plugged in within a crawler, hence facilitating various experiments.

We found that the use of a text window-based link context on its own provides no benefit over the use of the full page for scoring unvisited URLs. Also, our results strongly suggest that a crawler that uses both link context score and the (full) page score (Text Window Combo crawler) significantly outperforms a crawler that just uses link context score (Text Window crawler). Moreover, when compared with the Full Page crawler, our Text Window Combo crawlers are again significantly (at least at $\alpha = 0.1$) superior. We tested these ideas using three different text windows sizes ($T = 10, 20, 40$). These results could be explained through the two metrics of the “quality” of derived link contexts that we had previously suggested [28]—average context similarity and zero-similarity frequency. The two metrics reinforce the importance of using the local evidence of relevance available around a hyperlink with the more global (background) evidence of relevance within the page in which the hyperlink appears. Hence, a combination of scores based on the local and the global evidence from a Web page provides an effective way for scoring the unvisited URLs extracted from the page.

Based on these results, we may also suspect the merit of exponential text weighting schemes that put exponentially decaying weights on words within a Web page based on their distance from a hyperlink. We may expect such a scheme to have an effect similar to using simple text weights since above or below a point in the text the weights assigned to words will be negligible.

We suggested the Tag Tree Combo crawler that uses a new algorithm to derive link contexts. It was the only crawler that significantly outperformed the Full Page crawler on average target recall for the majority of the 10,000 page crawl. This good performance of the Tag Tree Combo crawler, based on our analysis, could be attributed to its success in finding relevant pages in the topics that were the “most difficult” for the Full Page crawler. For these difficult topics, even the best performing Text Window Combo crawler is significantly ($\alpha = 0.06$) outperformed by the Tag Tree Combo crawler.

Based on our results, if an application designer needs topical collections with high coverage, she should opt for the Tag Tree Combo crawler. On the other hand, if the need is for topical collections with high precision, she should consider the Text Window Combo (40) crawler. In our additional exploratory experiments, we found this result to be true even for crawls that are an order of magnitude longer (100,000 pages).

Topical analysis of crawl performances reveals that the “Business” topics form an especially difficult domain for both the Text Window Combo (40) and Full Page crawlers. On the other hand, the Tag Tree Combo (40) crawler is able to achieve significantly higher average target recall on these “Business” topics. We speculate that the lower performance of the Text Window Combo (40) and Full Page crawlers is due to the competitive nature of “Business” topics. In other words, relevant pages are not likely to be interconnected. We suggest that the Tag Tree Combo (40) crawler is able to offset the competitive nature of the business topics possibly

because the HTML structures (exploited by this crawler) in this domain succeed in reflecting meaningful semantic divisions in the Web pages. Some of our previous results in crawling for business intelligence tasks [30] support this hypothesis. However, a more concrete study to verify this is necessary and planned.

Our base set of experiments involves 100 topics and several different crawlers with each crawler fetching 10,000 pages per topic. To the best of our knowledge, this is the most extensive study on the influence of link context on the performance of topical crawlers. We further supplement these results with long crawls of 100,000 pages over 10 topics. The results for the longer crawls support the finding of the shorter crawls indicating the robustness achieved by using a large number of topics with shorter crawls. However, one would expect performances of different crawlers to converge as the number of pages fetched (length of the crawl) exceeds a threshold (perhaps hundreds of millions of pages). As an extreme case, if the different crawlers fetch all the pages on the Web, their precision and recall will be the same.

The model for topical crawling that we propose, where the guiding classifiers are trained on full pages and then used to score link contexts (full page combined with its subsets), is successful. In contrast, we find that a crawler that learns from smaller link contexts derived from the backlinks of training examples performs worse than a crawler based on our suggested model. In addition, unlike the backlink-based crawlers, crawlers based on our proposed model do not depend on the quality and availability of a search engine to initiate a crawl.

In related work [31], we looked at the sensitivity of crawlers, that use different classification algorithms, to the number of seeds. This was done by decreasing the number of seeds from 20 to just five seeds. We found that the order of performance between the crawlers remains the same although all of the crawlers degrade in performance as the number of seeds is reduced. In the future, we plan to repeat a similar analysis for our link context based crawlers.

Also, as future work, we would like to measure the affect of classifier stability on the performance of crawlers. We expect the stability of a classifier to be influenced by its algorithm, the quality of examples, and the characteristics of the topic. We plan to set up experiments that vary one of these dimensions at a time. We can then correlate the resulting volatility in classifier learning to variations in the performance of the classifier guided crawler. As a preliminary step, we performed a 5-fold cross-validation over all of the 100 topics and found that the SVM classifier had an average accuracy of 85.23 ± 0.78 percent. Hence, the classifier used in this study is fairly stable over the 100 topics.

It would be worthwhile to explore the use of phrases instead of words as lexical units extracted from the link context. Use of an ontology or at least of synonyms may be helpful for extending the context, especially when it is small in size. Such extensions could keep the link context precise (as compared to using the entire parent page) and yet informative. Other variations of Tag tree crawlers are also worth exploring. For example, the tag tree could be reduced by removing tags (nodes) such as ``, `<i>`, and `` that are less likely to enclose semantically cohesive information when compared with `<p>` or `<table>` tags [3].

This research also contributes to the general area of classifier-guided topical crawlers. In particular, we have

shown that a new model, where crawling classifiers are trained using full pages but applied to specific link contexts, can successfully find topical pages.

ACKNOWLEDGMENTS

The authors would like to thank Nick Street for providing the hardware for the experiments. Thanks go to Filippo Menczer, Nick Street, and Shannon Bradshaw for their helpful comments on this and other projects. They also thank the anonymous reviewers for their valuable suggestions.

REFERENCES

- [1] C.C. Aggarwal, "Collaborative Crawling: Mining User Experiences for Topical Resource Discovery," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 423-428, 2002.
- [2] C.C. Aggarwal, F. Al-Garawi, and P.S. Yu, "Intelligent Crawling on the World Wide Web with Arbitrary Predicates," *Proc. 10th Int'l World Wide Web Conf.*, May 2001.
- [3] G. Attardi, A. Gulli, and F. Sebastiani, "Automatic Web Page Categorization by Link and Context Analysis," *Proc. THAI-99, First European Symp. Telematics, Hypermedia, and Artificial Intelligence*, 1999.
- [4] S. Bradshaw, "Reference Directed Indexing: Redeeming Relevance for Subject Search in Citation Indexes," *Proc. Seventh European Conf. Research and Advanced Technology for Digital Libraries*, 2003.
- [5] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, nos. 1-7, pp. 107-117, 1998.
- [6] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan, "Automatic Resource List Compilation by Analyzing Hyperlink Structure and Associated Text," *Proc. Seventh Int'l World Wide Web Conf.*, 1998.
- [7] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated Focused Crawling through Online Relevance Feedback," *Proc. 11th Int'l World Wide Web Conf.*, May 2002.
- [8] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Proc. Eighth Int'l World Wide Web Conf.*, May 1999.
- [9] H. Chen, M. Chau, and D. Zeng, "Ci Spider: A Tool for Competitive Intelligence on the Web," *Decision Support Systems*, pp. 1-17, 2002.
- [10] N. Craswell, D. Hawking, and S.E. Robertson, "Effective Site Finding Using Link Anchor Information," *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 2001.
- [11] B.D. Davison, "Topical Locality in the Web," *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 2000.
- [12] P.M.E. De Bra and R.D.J. Post, "Information Retrieval in the World Wide Web: Making Client-Based Searching Feasible," *Proc. First Int'l World Wide Web Conf.*, 1994.
- [13] M. Diligenti, F. Coetzer, S. Lawrence, C.L. Giles, and M. Gori, "Focused Crawling Using Context Graphs," *Proc. 26th Int'l Conf. Very Large Data Bases*, pp. 527-534, 2000.
- [14] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, second ed. Wiley-Interscience, 2000.
- [15] S.T. Dumais, "Using SNMs for Text Categorization," *IEEE Intelligent Systems Magazine*, vol. 13, no. 4, 1998.
- [16] E.J. Glover, K. Tsioutsoulouklis, S. Lawrence, D.M. Pennock, and G.W. Flake, "Using Web Structure for Classifying and Describing Web Pages," *Proc. 11th Int'l World Wide Web Conf.*, ACM Press, 2002.
- [17] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalham, and S. Ur, "The Shark-Search Algorithm—An Application: Tailored Web Site Mapping," *Proc. Seventh Int'l World Wide Web Conf.*, 1998.
- [18] M. Iwazume, K. Shirakami, K. Hatadani, H. Takeda, and T. Nishida, "ICA: An Ontology-Based Internet Navigation System," *Proc. AAAI-96 Workshop Internet Based Information Systems*, 1996.
- [19] T. Joachims, "Learning to Classify Text Using Support Vector Machines," PhD thesis, Kluwer, 2002.
- [20] J. Johnson, K. Tsioutsoulouklis, and C.L. Giles, "Evolving Strategies for Focused Web Crawling," *Proc. Int'l Conf. Machine Learning*, 2003.
- [21] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM*, vol. 46, no. 5, pp. 604-632, 1999.
- [22] H. Lieberman, "Autonomous Interface Agents," *Proc. ACM Conf. Computers and Human Interface*, 1997.
- [23] O.A. McBryan, "Genvl and WWW: Tools for Taming the Web," *Proc. First Int'l World Wide Web Conf.*, 1994.
- [24] A.K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the Construction of Internet Portals with Machine Learning," *Information Retrieval*, vol. 3, no. 2, pp. 127-163, 2000.
- [25] F. Menczer and R.K. Belew, "Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web," *Machine Learning*, vol. 39, nos. 2-3, pp. 203-242, 2000.
- [26] F. Menczer, G. Pant, M. Ruiz, and P. Srinivasan, "Evaluating Topic-Driven Web Crawlers," *Proc. 24th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 2001.
- [27] C. Olston and E.H. Chi, "ScentTrails: Integrating Browsing and Searching on the Web," *ACM Trans. Computer-Human Interaction*, vol. 10, no. 3, pp. 177-197, Sept. 2003.
- [28] G. Pant, "Deriving Link-Context from HTML Tag Tree," *Proc. Eighth ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD03)*, 2003.
- [29] G. Pant and F. Menczer, "MySpiders: Evolve Your Own Intelligent Web Crawlers," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 2, pp. 221-229, 2002.
- [30] G. Pant and F. Menczer, "Topical Crawling for Business Intelligence," *Proc. Seventh European Conf. Research and Advanced Technology for Digital Libraries*, 2003.
- [31] G. Pant and P. Srinivasan, "Learning to Crawl: Comparing Classification Schemes," *ACM Trans. Information Systems*, vol. 23, no. 4, 2005.
- [32] G. Pant, K. Tsioutsoulouklis, J. Johnson, and C.L. Giles, "Panorama: Extending Digital Libraries with Topical Crawlers," *Proc. Fourth ACM/IEEE-CS Joint Conf. Digital Libraries*, pp. 142-150, 2004.
- [33] J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods: Support Vector Learning*, pp. 185-208, MIT Press, 1999.
- [34] J. Qin, Y. Zhou, and M. Chau, "Building Domain-Specific Web Collections for Scientific Digital Libraries: A Meta-Search Enhanced Focused Crawling Method," *Proc. Fourth ACM/IEEE-CS Joint Conf. Digital Libraries*, 2004.
- [35] G. Salton, *The SMART Retrieval System—Experiments in Automatic Document Processing*. Englewood Cliffs, N.J.: Prentice Hall Inc., 1971.
- [36] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [37] P. Srinivasan, F. Menczer, and G. Pant, "A General Evaluation Framework for Topical Crawlers," *Information Retrieval*, vol. 8, no. 3, pp. 417-447, 2005.
- [38] M. Subramanyam, G.V.R. Phanindra, M. Tiwari, and M. Jain, "Focused Crawling Using TFIDF Centroid," *Hypertext Retrieval and Mining*, Apr. 2001.
- [39] V.N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.



Gautam Pant received the PhD degree from the University of Iowa in 2004. He is currently an assistant professor in the School of Accounting and Information Systems at the University of Utah. His research interests include Web mining, machine learning, information retrieval, and knowledge discovery.



Padmini Srinivasan is a professor at the University of Iowa. Her research interests are in the design, implementation, and evaluation of Web crawlers, algorithms for mining hypotheses from text collections, and also building text-based applications in biomedicine. She holds a joint faculty position in the School of Library Science and the Department of Management Sciences with invited appointments in Computer Science and the College of Nursing.