Topical Web Crawling for Domain-Specific Resource Discovery Enhanced by Selectively using Link-Context

Lu Liu, Tao Peng and Wanli Zuo
College of Computer Science and Technology, Jilin University, China

Abstract: *To enable topical web crawling, link-context is the critical contextual information of anchor text for retrieving domain-specific resources. While some link-contexts may misguide topical web crawling and extract wrong web pages, because several relevant anchor texts become irrelevant or several irrelevant anchor texts become relevant after calculating the relevance between the link-contexts and the feature terms of the specific topic. In view of above, this paper presents a heuristic-based approach by selectively using link-context and implements DOM tree to locate the anchor text. Unlike previous crawling algorithms, which only zero in on link-context and ignore whether it is really needed or not. Our method cares both link-context and evaluating its necessity to correctly use link-context to guide topical crawling. Accordingly, our topical crawler can retrieve more relevant web pages. Experimental results indicate that this approach outperforms breadth-first, best-first, anchor text only, link-context both in harvest rate and target recall.*

Keywords: *Topical crawling, domain-specific resource retrieving, selectively using link context, DOM tree.*

Received November 17, 2012; accepted May 19, 2013; published online April 23, 2014

1. Introduction

The focused or topical crawlers, which attempt to download only those pages that are about a particular topic or theme, also carefully decide which URLs to scan and in what order to pursue based on previous downloaded pages information. Focused crawlers rely on the fact that pages about a topic tend to have links to other pages on the same topic. In reality, a host of pages for a specific topic are used as URL seeds.

Anchor text is a key component of the web page. It is a powerful navigation for people browsing the Internet and it also helps search engines understand the relationship among web pages. For example, the link http://www.amazon.com/ is most likely to contain the word "amazon" in the anchor text. Many queries are similar to anchor texts because they both short topical description of web pages. Anchor text is not informative enough, so we expand it to link-context. We adopt a heuristic-based approach to deal with anchor text and the link-context to make the topic of out-link page clearer and more accurate. Although, there are many advantages using link-context to crawl the web pages, it is not known whether every anchor text should combine with several bytes link-contexts in a text window. Sometimes, some web page designers do not summarize the destination web pages in the anchor text. Instead, they use words such as "click here", "here", "read more", "more", "next" and so on to describe the texts around them in anchor text shown in Figures 1-a and 1-b. However, if we calculate every similarity or relevance between link-context and feature term about a specific topic, we may also omit some web pages that are relevant indeed, or extract some web pages that are not relevant.

* Corresponding author.
E-mail addresses: tpeng@jlu.edu.cn; taopeng@illinois.edu.

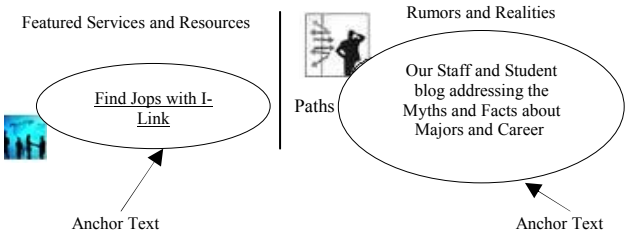
a. An example of an anchor text fitting for extracting link-context.

b. Another example of an anchor text fitting for extracting link-context.

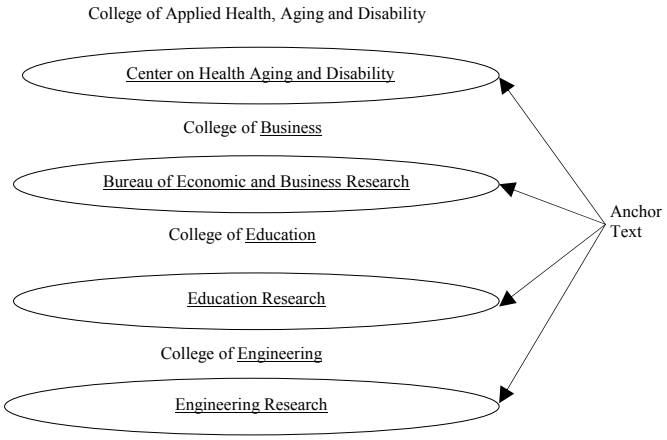
Figure 1. Instances of some anchor texts that fit for extracting link-context.

As Figure 2-a shown that, if we want to search "job" area and anchor text "Find Jobs with I-link" meets the requirement of relevance, but it becomes irrelevant after extracting link-contexts because the anchor text on the right of Figure 2-a does not belong to the "job" area. Then, we will miss the anchor text. Another kind of circumstance is shown in Figure 2-b. There are four anchor texts in Figure 2-b. Suppose we want to search "business" area, when we search the first anchor text, and we find out it does not belong to the "business" area. However, after combining its link-context, it belongs to "business" area, which leads to extract

wrong web pages. Motivated by these circumstances, selectively using link-context to enhance topical crawling for domain-specific resources is utilized to improve the efficiency and accuracy of topical crawlers.



a. An example of an anchor text unfitting for extracting link-context.



b. Another example of an anchor text unfitting for extracting link-context.

Figure 2. Instances that do not fit for extracting link-context.

Also, we crawl the web under the guidance of a classifier built by hundreds of the some specific topic web pages. We use DOM offset and HTML tag tree to expand the anchor text to make it more useful and apply it in two aspects in the process of focused crawling, the measurement and prediction.

The outline of this paper is as follows. We review the related work in section 2. Section 3 illustrates how to selectively use link-context to guide the topic crawling and the metrics of relevance. The whole crawling procedure is proposed in section 4. Some comprehensive experiments are performed to evaluate the effectiveness and the efficiency of our proposed method in section 5 in which our classifier is also implemented. Section 6 draws the conclusions.

2. Related Work

Many researchers delve into a search engine that focuses on a specific topic of information. A less expensive approach to get web pages for this kind of engine is focused or topical crawling. Mouton and Marteau [15] developed an approach that exploited link analysis to determine what constitutes a good content analysis metric. The routing information encoded into backlinks also improved topical crawling. De-Assis *et al.* [5] described an approach to focused crawling that exploited not only content-related information but also, genre information presented in web pages to guide the crawling process. Zhang and Lu [24] presented an online semi-supervised clustering approach for topical web crawlers. Topical crawlers selected the most topic-related URL based on the scores of the URLs in the unvisited list. Arya and Vadlamudi [2] designed an ontology-based topical crawling algorithm to access hidden web content. Liu and Milios [13] proposed a probabilistic model for topical web crawling. They captured sequential patterns along paths leading to the targets and modelled the process of crawling by a walk along an underlying chain of hidden states. Torkestani [22] designed a focused crawler which took advantage of learning automata and learned the most relevant URLs and the promising paths leading to the target on-topic documents. Ali [1] proposed an approach for focused crawling that integrated evidence from both focused crawling and intelligent multi-agent technology.

Due to the topic information and abstract that provided by anchor text, the hyperlink structures in web pages are utilized by many researchers to search the web [8]. McByan [14] built index in web crawler using anchor text. Duwairi and Al-Zubaidi [6] presented a classifier based on a modified version of the well known K-Nearest Neighbors classifier (K-NN). The method was required to be compared with category representatives when classifying a new document. Iwazume *et al.* [10] combined anchor text and ontology theory in order to guide web crawler. Brin and page, the founders of google search engine, also used anchor text to build index for URLs [3]. Jung [11] proposed the context-based focused crawler architecture to discover local knowledge from interlinked systems and a knowledge alignment process to integrate the discovered local knowledge. Eiron and McCurley [7] presented a statistical study of the nature of anchor text and real user queries on a large corpus of corporate intranet documents. Tateishi *et al.* [21] evaluated web retrieval methods using anchor text. Li *et al.* [12] proposed a focused crawler guided by anchor texts using a decision tree. Nath and Bal [16] presented a novel crawler system based on filtering off non-modified pages for reducing load on the network, which employed mobile agents to crawl the pages. These mobile agent based crawlers retrieved the pages, processed them, compared their data to filter out pages that were not modified after last crawl and then compressed them before sending them to the search engine for indexing.

SharkSearch applied not only anchor text but also its texts that appear in adjacent area to evaluate the benefit of crawling along the anchor text [9]. Chakrabarti *et al.* [4] obtained fixed bytes link-contexts around anchor text in their clever system. Through 5000 web pages test, they found out that "yahoo" was most likely to appear around the 50 byte range of anchor text corresponding to URL http://www.yahoo.com/. Pant [17] introduced a framework to study link-context derivation methods. The framework included a number of metrics to understand the utility of a given link-

context. Yuvarani *et al.* [23] proposed a focused crawler called LSCrawler which took into account the semantic similarity between the keywords in the link and the surroundings text of the link.

3. Selectively using Link-Context

3.1. Extracting Link-Context from HTML Tag Tree

Nowadays, links connecting pages and texts around links, which help people surf the internet, are key components of the web pages. Figure 3 shows a framework of a simple web consisting three web pages from the web crawler's perspective.

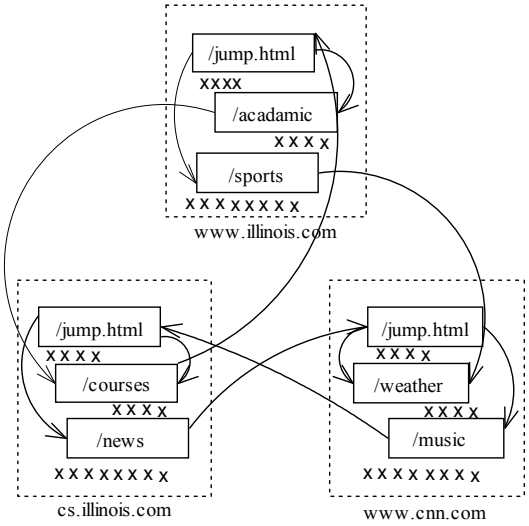


Figure 3. A simple web consisting three web pages. The arrows denote links between the pages.

In one web page, texts and links about the same topic are usually grouped into one content block. The size of the content block is varied. The big one may cover the whole web page, while the small one only takes 1/8 or 1/16 of the web page's total space and the smallest one is an anchor text. The anchor text is the visible, clickable text in a hyperlink [7]. Including information provided by the same content block where the link appears is an effective way to enrich the context of the anchor text. Gautam Pant presented two link-context derivation techniques in [17]. We use the method of deriving link-context from aggregation nodes, with some modifications. We tidy the HTML page into a well-formed one web page beforehand using HTML TIDY tool (http://www.w3.org/People/Raggett/tidy/) because many web pages are badly written. We insert missing tags and reorder tags in the "dirty" pages to make sure that we can map the context onto a tree structure with each node having a single parent and all text tokens that are enclosed between <text>…</text> tags appear at leave nodes on the tag tree. This pre-processing simplifies the analysis.

If we want to extract the link-context of an anchor text, then first, we locate the anchor. Next, we treat each node, which is on the path from the root of the tree to the anchor, as a potential aggregation node as shown in Figure 4. From these candidate nodes, we choose the parent node of anchor, which is the grandparent node of the anchor text as the aggregation node. Then, all of the texts in the sub-tree rooted at the node are retrieved as the contexts of the anchor showed in rectangle of Figure 4. If one anchor appears in many different blocks, combine the link-context in every block as its full link-context.

```
<html>
<head>
<title>CS@Illinois | Department of Computer Science at Illinois</title>
</head>
<body>
<h1 class="cs">Department of Computer Science</h1>
<a href="http://cs.illinois.edu/csillinois/overview">
<text>Overview</text></a>
<p>
<a href="/csillinois/history"><text>History</text></a>
<text>have a look at awards<a href="/csillinois/awards">
<text>Read more</text></a></text>
<h3>Connect with Us</h3>
</p>
</body>
</html>
```

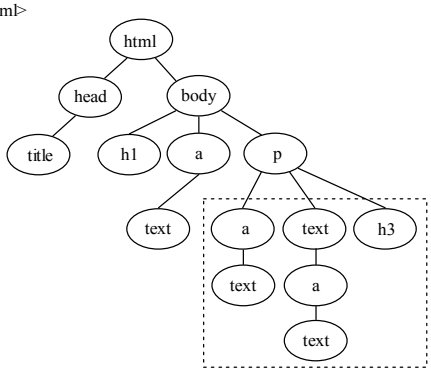


Figure 4. An HTML page and its corresponding tag tree.

Compared with [17], we fixed the aggregation node. In fact, for each page, we have an optimal aggregation node that provides the highest context similarity for the corresponding anchor with different aggregation node. It is very labor-some to tidy up web pages every time we analyze the web pages. Large size contexts may be too "noisy" and burdensome for some systems. Too small contexts, like single anchor texts, provide limited information about the topic. We set a trade off on quantity and quality.

Although, we set a trade off on quantity and quality, we might make many mistakes when extracting every link-context of its anchor. One kind of mistakes is that we may extract a host of web pages that are irrelevant to the specific domain, because the anchor text itself is irrelevant, however, it becomes relevant after combining with its link-context by mistake. Another kind of mistakes is that a host of relevant web pages are not extracted still caused by extracting link-context inappropriately. Our approach, which selectively makes use of link-context, can solve this problem quite well. Before extracting every anchor's link-context, we compute the relevance of anchor text first. If it meets the relevance requirement, then the corresponding URL is added into the waiting queue or frontier. There is no need to calculate the relevance of its link-context. As for the irrelevant anchor texts, we just extract their corresponding link-contexts to judge whether or not they indeed link to relevant pages. In addition, all the

anchors are picked up to compute their relevance, so another process is that our link-contexts do not contain anchor texts, instead, they are pure texts. The explanatory texts are more likely to be the pure texts rather than anchor texts. To some extent, it does not only improves the efficiency, but also reduces the risk of making mistakes. The pseudo-code for topical crawling enhanced by selectively using link-context is shown as follows.

Algorithm 1: Topical Crawling Enhanced by Selectively using Link-Context (TCESULC) (seed URLs)

```
url_queue=seedURLs //append element at the end of queue
for each (url in url_queue)
  {
    url=dequeue(url_queue) //remove the element at the
    beginning of queue and return it
    Page=crawl url and get the corresponding web page
    If (page is relevant)
      {
        put page into relevance_pageDB
      }
    temp_queue=extract all anchor texts
    for each (url in temp_queue)
      {
        judge each anchor text in temp_queue
        if (anchor text is relevant)
          {
            enqueue its url into url_queue
            dequeue url in temp_queue
          }
        else
          {
            extract link-context of the corresponding
            anchor  text
            if (link-context is relevant)
              {
                enqueue its url into url_queue
              }
            dequeue url in temp_queue
          }
      }
    reorder url_queue according to the priority weight
  }
```

3.2. Estimation Metrics of Relevance

In our focused crawler, we compute the weight of each term in the anchor text or link-context (contain its anchor text) based on tfc weighting scheme Equation 1 [20] after pre-processing which includes removing stop words and stemming.

$$x_{ki} = \frac{f_{ki} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \log\left(\frac{N}{n_r}\right)\right]^2}} \quad (1)$$

Where, f_{ki} is the frequency of word i in text unit k (anchor text or link-context), N is the number of text units in the collection, M is the number of all the features. n_i is the number of documents where word i occurs.

The relevance of anchor text, link-text and web pages is computed using our classifier [19]. Suppose we build a text classifier by applying the SVM algorithm on a training data set, including positive and negative data. The vector space model is used to represent the documents, which has the advantage of being a simple and intuitively appealing framework for implementing term weighting, ranking and relevance feedback. In this model, documents are assumed to be part of an m-dimensional vector space, where m is the number of index words (also called features or items) in the vocabulary. That is, m is the size of the vocabulary. So, a document d_i is represented by a vector of index terms as $d_i=(x_{i1}, x_{i2}, ..., x_{im})$ and x_{ij} represents the weight value for document d_i of the j^{th} feature. A document collection is represented as a matrix of feature weights, where each row corresponds to a feature vector.

$$collection = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & x_{ij} & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

Based on the above representation, an appropriate form of the term weighting must be determined in the vector space model. In fact, many different weighting schemes have been tried over the years, most of which are variations on TFIDF weighting, which is the most common weighting method used to describe documents in the vector space model and weights each feature vector component on the following basis.

$$x_{ki} = f_{ki} \times \log\left(\frac{N}{n_i}\right) \quad (2)$$

As we know, TFIDF can effectively reflect the importance of the term in the entire document set, in which all documents play the same roles (such as text clustering, IR). In contrast, if the documents play different roles (e.g., the text classifier positive and negative training set), TFIDF has certain defects.

For example, given a training set for text classifier contains 20 training documents (10 positive and 10 negative training documents), in which term t occurs in ten documents. According to TFIDF as shown in Equation 2, n_i=10, that is, all the Inverse Document Frequency (IDF) weights of term t is $log(20/10)$ in the collection. But, consider two cases, term t occurs in 5 positive and 5 negative examples and term t occurs in 9 positive and 1 negative examples. Obviously, term t reflects different importance in positive and negative examples in the two cases. Consequently, TFIDF does not take into account the difference of term IDF weighting, which in the positive or in the negative example sets. Motivated by this, we present an improved TFIDF term weighting method, Term Frequency Inverse Positive-Negative Document Frequency (TFIPNDF).

The weighting method named TFIPNDF, presented in this paper is based on statistical term frequency and IDF components from positive and negative training examples, respectively. Compared with TFIDF, term frequency component of TFIPNDF is the same with

TFIDF. However, when weighting IDF component, TFIPNDF method calculates the Inverse Positive Document Frequency (IPDF) and Inverse Negative Document Frequency (INDF) weight values in the positive and negative training examples, according to the distribution of the terms, respectively. In other words, IPNDF (or IPDF and INDF) reflects the importance of the term in the positive and negative training examples respectively. More comparison experiments are detailed in our research work on PU classification [18]. Therefore, TFIPNDF is composed of two parts:

$$TFIPDF = f_{ki} \times \frac{P_i}{S_P} \times \log\left(\frac{N}{n_i}\right) \quad (3)$$

$$TFINDF = f_{ki} \times \frac{N_i}{S_N} \times \log\left(\frac{N}{n_i}\right) \quad (4)$$

The effects of these two weights are combined by limiting the scope of the training set, i.e.,

$$TFIPNDF = \begin{cases} f_{ki} \times \frac{P_i}{S_P} \times \log\left(\frac{N}{n_i}\right) & (document_k \in P) \\ f_{ki} \times \frac{N_i}{S_N} \times \log\left(\frac{N}{n_i}\right) & (document_k \in N) \end{cases} \quad (5)$$

Where, f_{ki} is the frequency of word i in document k, N is the number of documents in the collection, n_i is the number of documents where, word i occurs in the collection, P_i is the number of positive documents where word i occurs, N_i is the number of negative documents where word i occurs, S_P and S_N are the numbers of positive and negative documents in the collection, respectively.

A document collection may contain documents of many different lengths. It is useful to use normalized weight assignments. A vector length normalization of TFIPNDF is defined as:

$$x_{ki} = \begin{cases} \frac{f_{ki} \times \frac{P_i}{S_P} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \frac{P_r}{S_P} \times \log\left(\frac{N}{n_r}\right)\right]^2}} & (document_k \in P) \\ \frac{f_{ki} \times \frac{N_i}{S_N} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \frac{N_r}{S_N} \times \log\left(\frac{N}{n_r}\right)\right]^2}} & (document_k \in N) \end{cases} \quad (6)$$

Where, M is the number of all the features.

4. Crawling Procedure

The web crawler has two jobs: Downloading pages and finding URLs. The crawler begins with a group of seed URLs, which are provided to the crawler as starting parameters. In the process of crawling, once a web page is downloaded, we parse the page's DOM tree after preprocessing (eliminating stopwords and stemming) and then the page will be classified by a conventional classifier with a high dimensional dictionary. The relevant pages are added into the relevant page set. Moreover, the page is parsed to pick up all the anchor texts. If the page linked by the anchor has been crawled or the anchor is in the queue of the crawling (frontier), then it is removed. The rest of anchor texts are computed to get the relevance, which is treated as priority. The selectivity of link-context is here: When the relevance of an anchor text meets the requirement, it is added into the url_queue or frontier. While once the relevance of the anchor text does not meet the requirement, its link-context needs to be extracted and calculate their relevance again. The unvisited URL that has highest priority will be first fetched to crawl. Whenever, a new batch of anchors is inserted into the waiting queue, the queue will be readjusted to create its new frontier. Figure 5 illustrates the architecture of our topical crawler.

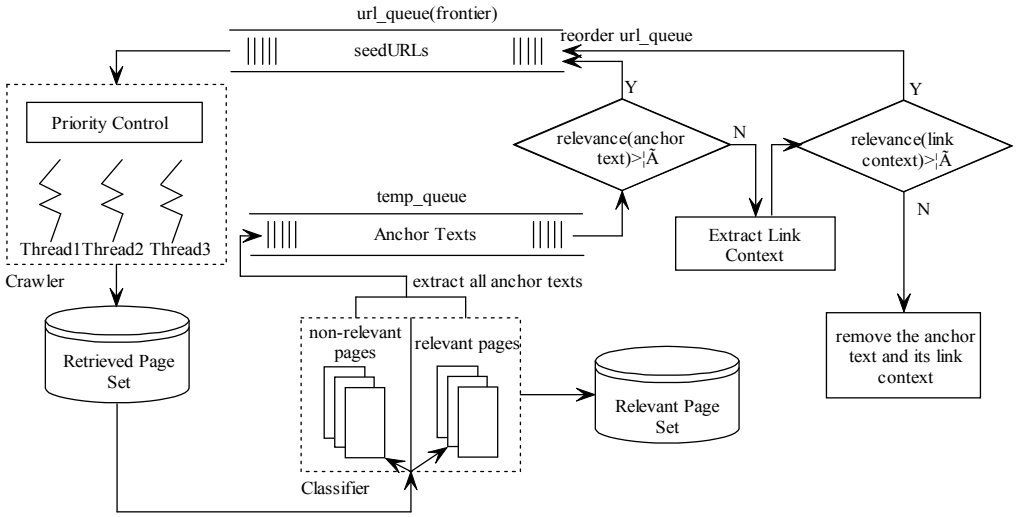


Figure 5. The architecture of topical crawler enhanced by selectively using link-context.

5. Experiments and Results

In this section, several tests have been used to verify whether a selectively using link-context technique holds for efficient topical crawling. In the experiment, we built crawlers as shown in Figure 6 that used different techniques to crawl the web and tested our method using multiple crawlers over 10 topics covering hundreds of thousands of pages. Under the guide of each method, crawler downloaded the pages according to the strategies of web page processing.

Figure 6. Crawling system intereface (crawling instance on top: science: computer science: Artificial intelligence topic in ODP, http://www.dmoz.org/).

5.1. Performance Metrics

The two most common effectiveness measures, *Harvest rate* and *Target recall*, were introduced to summarize and compare search results. Intuitively, *Harvest rate* is the fraction of pages crawled that are relevant to the topic, which measures how well it is doing at rejecting irrelevant web pages. We make this decision by using our classifier [19] instead of manual relevance judgment, which is costly. Target recall is the fraction of relevant pages crawled, which measures how well it is doing at finding all the relevant web pages. However, the relevant set for any given topic is unknown in the web, so the true *Target recall* is hardly to measure. In view of this situation, we delineate a specific network, which is regard as a virtual WWW in the experiment. Given a set of seed URLs and a certain depth, the range can be reached by a crawler using breadth-first crawling strategy is the virtual web. We assume that the target set T is the relevant set in the virtual web, $C(t)$ is the set of first t pages crawled. Therefore, we define *Harvest rate* and *Target recall* as follows:

$$Harvest\ rate = \frac{|C(t) \cap T|}{|C(t)|} \times 100\% \tag{7}$$

$$Target\ recall = \frac{|C(t) \cap T|}{|T|} \times 100\% \tag{8}$$

5.2. Data Sets

Our crawler is multi-threaded and implemented in java, which provides for reasonable speed-up. We run our algorithm using 20 threads of execution starting from 100 relevant URLs (Seed URLs) on each topic picked from Open Directory Project (ODP), http://dmoz.org/. 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 download 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 randomly select 10 categories and the associated ODP relevant sets. These selected categories 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 (contain 100 URLs), which are used to initialize the crawl. The second one serves to train our classifiers [19] used for evaluating the performance.

5.3. The Comparison of Different Techniques

In this experiment, we built focused crawlers using different techniques (Breadth-First, Best-First (full page), Anchor text only [21], Anchor text using decision tree [12], Link-context only [17] and Selectively using link-context), which are described in the following.

Breadth-First is a baseline crawler, the crawl frontier of which is a FIFO queue. Each thread of the crawler picks up the URL at the front of the queue and adds new unvisited URLs to the end of it. The crawler is multi-threaded as well as many pages are fetched simultaneously. The crawler adds unvisited URLs to the frontier only when the size of the frontier is less than the maximum allowed. Best-First crawler treats a web page as a set of words. It computes the similarity of the page to the given topic and uses it as a score of the unvisited URLs on the page. The URLs are then added to the frontier that is maintained as a priority queue using the scores. Each thread picks the best URL in the frontier to crawl and inserts the unvisited URLs at appropriate positions in the priority queue. Anchor text is the 'highlighted clickable text' in the web page, which appears within the bounds of an <A> tag in source code. Since, the anchor text tends to summarize information about the target page, it is a good provider of the context of the unvisited URLs. However, because of it is not informative enough, anchor text only crawler, which guides the crawling using anchor text, does not perform very well. Anchor text using decision tree is an approach which uses a decision tree on anchor texts of hyperlinks. Link-context is a popular technique using link-context for

determining the priorities, which derives link-contexts from HTML Tag Tree. The link context is a kind of extended anchor text. By applying it in prioritizing the unvisited URLs and guiding the crawling, the crawler's performance improves a lot. Selectively using link-context is just described in this paper.

Figures 7 and 8 show the average Harvest rate and average Target recall over ten topics in order to reflect the comprehensive of our method. Breadth-First without judging the unvisited URLs does not perform well. Therefore, Breadth-First fetched large numbers of irrelevant pages. It depends heavily on the localization of the relevant pages and web sites. Best-First predicts the relevance of the potential URLs by referring to the whole context of the visited web page. It only groups the unvisited URLs based on the page picked up from, and there is no difference within each group. So, it has low accuracy when there is a lot of noise in the page or the page has multiple topics. Only analyzing the anchor text may omit much useful textual information because it is short and not informative enough. However, extracting its link-contexts every time can not only reduce the efficiency but also cause multiple mistakes. Their weakness can be just overcome by our method. In summary, the topical crawler enhanced by selectively using link-context shows significant performance improvement over the crawlers mentioned above.

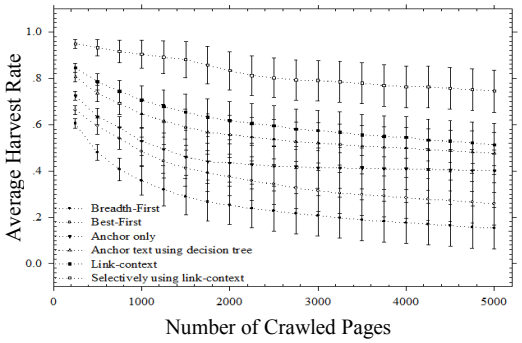


Figure 7. Dynamic plot of harvest rate versus number of crawled pages. Performance is average across topics and standard error bars are also shown. The error bars correspond to ± standard error.

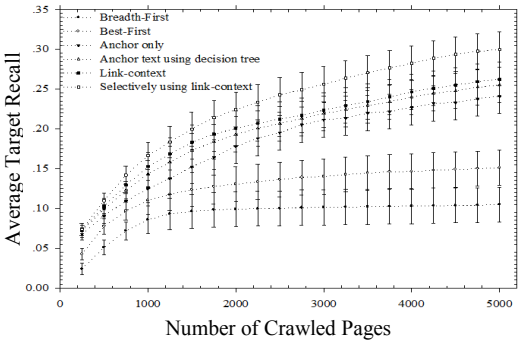


Figure 8. Dynamic plot of target recall versus number of crawled pages. Performance is average across topics and standard error bars are also shown. The error bars correspond to ± standard error.

In our experiment, we also set two counters to calculate the times of combining link-contexts and the times of using anchor text only, respectively. Besides, the number of crawled pages is 5000 for each topic. The comparison results as shown in Figure 9 show that there are indeed some web pages can be judged only using anchor text and harvest rate and target recall as shown in Figures 7 and 8 also indicate the improvement on the performance of topical crawlers.

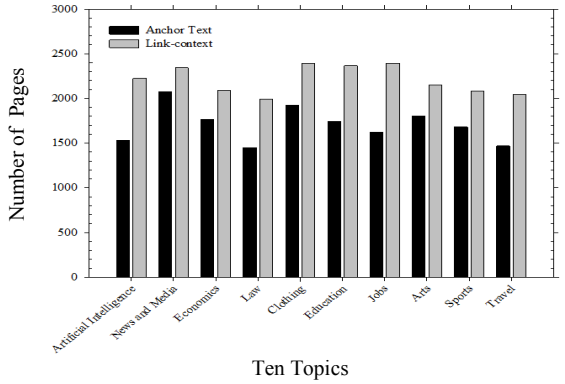


Figure 9. The comparison results of using anchor text and link-context in extracting web pages over ten topics.

6. Conclusions

With the flourish of WWW, people can have great opportunity to benefit from the abundant information in such an environment. Hence, it is an important task to extract domain-specific web pages. In this paper, a heuristic-based approach was presented focusing on selectively using link-context to enhance topical crawling. The approach this paper presented working with the document object model tree as opposed to raw HTML markup enables us to locate anchor text, extract the corresponding link-context instead of the whole page. tfc weighting scheme and our classifier are implemented to calculate the relevance of the anchor text or link-context. Treating anchor text and its link-context respectively can both improve the efficiency and bring the error rate down caused by the misguidance of crawler. The comparison between using anchor text and link-context also shows that there are indeed many web pages can be retrieved only using anchor text. The experimental using harvest rate and target recall as performance measurement verified that our approach significantly improved the focused web crawling performance when dealing with web pages in the complex web environment.

References

[1] Ali H., "Self Ranking and Evaluation Approach for Focused Crawler Based on Multi-Agent System," *the International Arab Journal of Information Technology*, vol. 5, no. 2, pp. 183-191, 2008.

[2] Arya V. and Vadlamudi R., "An Ontology-Based Topical Crawling Algorithm for Accessing Deep Web Content," *in Proceedings of the 3rd International Conference on Computer*

and Communication Technology, Pradesh, India, pp. 1-6, 2012.

[3] Brin S. and Page L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, no.1, pp. 107-117, 1998.

[4] Chakrabarti S., Dom B., Gibson D., Kleinberg J., Raghavan P., and Rajagopalan S., "Automatic Resource List Compilation by Analyzing Hyperlink Structure and Associated Text," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 65-74, 1998.

[5] De-Assis T., Laender F., Goncalves A., and Da Silva A., "A Genre-Aware Approach to Focused Crawling," *World Wide Web-interest and Web Information Systems*, vol. 12, no. 3, pp. 285-319, 2009.

[6] Duwairi R. and Al-Zubaidi R., "A Hierarchical K-NN Classifier for Textual Data," *the International Arab Journal of Information Technology*, vol. 8, no. 3, pp. 251-259, 2011.

[7] Eiron N. and McCurley S., "Analysis of Anchor Text for Web Search," *in Proceedings of the 26th ACM/SIGIR International Symposium on Information Retrieval*, Toronto, Canada, pp. 459-460, 2003.

[8] Glover J., Tsioutsiouliklis K., Lawrence S., Pennock M., and Flake W., "Using Web Structure for Classifying and Describing Web Pages," *in Proceedings of the 11th International Conference on World Wide Web*, Hawaii, USA, pp. 562-569, 2002.

[9] Hersovici M., Jacovi M., Maarek S., Pelleg D., Shtalhaim M., and Ur S., "The Shark-Search Algorithm: An Application: Tailored Web Site Mapping," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 317-326, 1998.

[10] Iwazume M., Shirakami K., Hatadani K., Takeda H., and Nishida T., "Iica: An Ontology-Based Internet Navigation System," *in Proceedings of the 13th National Conference on Artificial Intelligence Workshop on Internet Based Information Systems*, USA, Oregon, pp. 65-71, 1996.

[11] Jung J., "Towards Open Decision Support Systems Based on Semantic Focused Crawling," *Expert systems with applications*, vol. 36, no. 2, pp. 3914-3922, 2009.

[12] Li J., Furuse K., and Yamaguchi K., "Focused Crawling by Exploiting Anchor Text using Decision Tree," *in Proceedings of the 14th International Conference on World Wide Web*, Chiba, Japan, pp. 1190-1191, 2005.

[13] Liu Y. and Milios E., "Probabilistics for Focused Web Crawling," *Computational Intelligence*, vol. 28, no. 3, pp. 289-328, 2012.

[14] McBryan O., "GENVL and WWWW: Tools for Taming the Web," *in Proceedings of the 1st International Conference on the World Wide Web*, Geneva, Switzerland, pp. 79-90, 1994.

[15] Mouton A. and Marteau F., "Exploiting Routing Information Encoded into Backlinks to Improve Topical Crawling," *in Proceedings of International Conference of soft computing and pattern recognition*, Malacca, Malaysia, pp. 659-664, 2009.

[16] Nath R. and Bal S., "A Novel Mobile Crawler System Based on Filtering off Non-Modified Pages for Reducing Load on the Network," *the International Arab Journal of Information Technology*, vol. 8, no. 3, pp. 272-279, 2011.

[17] Pant G., "Deriving Link-Context from HTML Tag Tree," *in Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, CA, USA 2003.

[18] Peng T., Liu L., and Zuo W., "PU Text Classification Enhanced by Term Frequency-Inverse Document Frequency-Improved Weighting," *Concurrency and Computation: Practice and Experience*, vol. 26, pp. 728-741, 2014.

[19] Peng T., Zuo W., and He F., "SVM Based Adaptive Learning Method for Text Classification from Positive and Unlabeled Documents," *Knowledge and Information Systems*, *Springer*, vol. 16, no. 3, pp. 281-301, 2008.

[20] Salton G. and Buckley C., "Term Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.

[21] Tateishi K., Kawai H., Akamine S., Matsuda K., and Fukushima T., "Evaluation of Web Retrieval Method using Anchor Text," *in Proceedings of the 3rd NTCIR Workshop*, Tokyo, Japan, pp. 25-29, 2002.

[22] Torkestani A., "An Adaptive Focused Web Crawling Algorithm Based on Learning Automata," *Applied Intelligence*, vol. 37, no. 4, pp. 586-601, 2012.

[23] Yuvarani M., Iyengar N., and Kannan A., "LSCrawler: A Framework for an Enhanced Focused Web Crawler Based on Link Semantics," *in Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, Hong Kong, China, pp. 794-797, 2006.

[24] Zhang X. and Lu J., "SCTWC: An Online Semi-Supervised Clustering Approach to Topical Web Crawlers," *Applied Soft Computing*, vol. 10, no. 2, pp. 490-495, 2010.

Lu Liu received her BS in computer science from Jilin University in 2012. She is currently a PhD student at the College of Computer Science and Technology, Jilin University. Her research interests include Web mining, information retrieval, and machine learning. She was a visiting student at University of Illinois at Urbana-Champaign in the Department of Computer Science (2012-2013).

Tao Peng received his PhD and MSc in computer science from Jilin University in 2007 and 2004, respectively. He is currently an associate professor at the College of Computer Science and Technology, Jilin University. His research interests include Web mining, information retrieval, and machine learning. He was a postdoctoral researcher at University of Illinois at Urbana-Champaign in the Department of Computer Science (2012-2013).

Wanli Zuo is currently a professor at the College of Computer Science and Technology, and Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University. His research interests include database theory, data mining, Web mining, machine learning, and web search engine.