Journal of Web Engineering, Vol. 12, No.3&4 (2013) 203-214
© Rinton Press

TOPICAL CRAWLING ON THE WEB THROUGH LOCAL SITE-SEARCHES

YALING LIU and ARVIN AGAH

Department of Electrical Engineering and Computer Science, University of Kansas

Received April 2, 2012
Revised November 19, 2012

In this paper, we investigate the feasibility of discovering topical resources by combining Web searches and local site-searches. Existing techniques of topical resource discovery consist of crawling the Web and searching the Web. The former typically analyses linkage among Web pages to estimate the relevance of an unseen document to a topic. The latter exploits the indices of generic search engines to discover documents relevant to a topic. Although the local site-search has been a simple and convenient feature of a Web site for human users to quickly locate desired information within the site that hosts tremendous number of documents, this feature has been ignored by the techniques of automatic topical resource discovery. A typical local site-search returns a list of titles, hyperlinks, and snippets of relevant documents that can be used to estimate the relevance of the documents to the topic before actually fetching the documents. We propose an operational model to make use of this simple feature, and address how this model can be realized. Experiments have shown that this simple but efficient approach can provide much more precise estimations than a sophisticated intelligent topical crawler.

Key words: Web searching, topical resource discovery, topical crawling, local site-search
Communicated by: D. Schwabe & E.-P. Lim

1 Introduction

A typical crawler for a generic search engine traverses through hyperlinks in fetched Web pages to explore the undiscovered portion of the Web and fetch more Web pages recursively [1]. As the Web has grown to the size of billions of pages, it has become a challenge for a generic crawler to ensure the coverage of fetching all pages for all topics. To solve this problem, topical crawling techniques have been intensely investigated recently, to only fetch documents relevant to a specific topic or topics.

The basic idea behind topical crawling is to estimate the relevance of undiscovered documents using the relevance of fetched documents that directly or indirectly link to the undiscovered ones [2] [3] [4] [5]. Topical crawlers maintain a priority queue. The most potentially relevant documents have the highest priority to be fetched under the constraints of computing resources. However, hyperlinks have proven to be very noisy evidences as deciding the relationships of conveying relevance between linking and linked pages. A topic-relevant page may link to non-relevant pages (e.g., sponsor links or ads), or all links to a relevant page may be non-relevant pages (e.g., navigation pages). Due to these uncertainties, an intelligent crawler realized by sophisticated statistic techniques can only achieve an average harvest rate of about 40% [4].

Instead of using these heuristic techniques to guess the relevance of an unseen document to a topic, in this paper, we propose an alternative approach, which uses local site-searches to obtain important information of documents in order to better estimate their relevance to a topic before fetching them.

Local site-searches have been provided for human users by most medium and large-scale sites to easily locate desired pages. This simple feature has not been adopted in topical crawling. Existing topical crawling algorithms are based on hyperlink analysis that was brought out by the crawlers designed for generic search engines. However, we believe that local site-searches are much more efficient and effective for estimating the relevance of unseen documents to a query. Also, this feature greatly simplifies the implementation of topical resource discovery.

In contrast to topical crawlers, using statistic models to estimate the relevance of unseen documents, locale site-searches can easily obtain more precise estimations from the titles and snippets of the unseen documents in the response of locate site-searches. Because there are commonly accepted regularities for local site-search forms and responses, it is rather simple for a topical crawler to find the form, learn how to submit the form, and parse results from the responses.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 shows the estimated percentage of the sites that host the local site-search among all sites. Section 4 proposes the operational model of topical crawling through Web searches and local site-searches. Section 5 describes a realization of the operational model. Section 6 shows the experimental results, followed by a conclusion.

2 Related work

Different from generic crawlers, topical crawlers traverse the topic-relevant portion of the Web and fetch only potentially topic-relevant pages. Hyperlinks among Web pages have been used as the main evidence for estimating the relevance of a document to a topic before actually fetching the document. As the linkage can be either a strong or weak evidence for topic-relevance at different portions of the Web, a statistic model has been proposed to realize an intelligent topical crawler, that can learn how well the crawler should trust the linkage evidence and others on its way to traverse and crawl the Web [4].

Focused crawling is a similar concept as topical crawling [2]. It gathers relevant documents to a set of topics. Different from the topical crawling described previously, exemplary documents are used in focused crawling to represent the target topics, instead of keywords, or queries. Both topical crawling and focused crawling use hyperlinks to estimate the relevance of an unseen document to a topic

An alternative solution to topical resource discovery is to search the Web by exploiting generic search engines [6]. Supervised machine learning techniques have been developed to learn a set of queries from manually labelled positive examples. Later, these queries are sent to generic Web search engines to search topic-relevant documents.

These two main approaches (crawling or searching) have been compared in terms of execution time and output completeness [7]. Making the choice between them is task-specific. Besides these two main approaches, there have been attempts from several different perspectives to solve the topical resource discovery problem. For instance, user access logs stored on proxy servers have been investigated to enhance topical crawling. This approach is referred to as collaborative crawling [8]. A context focused crawler [9] queries generic search engines to construct context graphs representing the distances (in terms of linkage) between the current documents and the target documents. Hidden Markov Models have also been used in studying the linkage context [10]. Meta-search approaches can help explore broader portions and prevent topical crawlers from being trapped in a few topical locals [11]. The Document Object Model (DOM) features of HTML pages are another type of valuable evidence to enhance the estimation of relevance in topical crawling [12]. Sophisticated classification algorithms (for instance, Naïve Bayes, Support Vector Machine, and Neural Network) have been studied in guiding topical crawling [13].

So far, we have listed the two main solutions and a few alternatives to topical resource discovery. In this paper, we make use of the local site-searches to effectively gather topic-relevant documents. This feature has not been investigated in solving this problem. However, a similar setting (crawling the hidden Web) should be compared with the approach we propose in this paper. Hidden web crawling targets the data that can only be accessed through submitting on-line forms [14]. In contrast to the hidden Web problem, in this paper, we investigate the feasibility of using the local site-search forms to efficiently access the topic-relevant documents that are on the surface of the Web, and can also be accessed through hyperlinks. The HTML forms of local site-searches are ignored by hidden Web crawlers because local site-searches target the surface of the Web not the hidden Web [15].

3 Local Site-Search

We have estimated the percentages of the Web sites that host a local site-search form among all the sites in terms of categories. We collected URLs of Web sites from "Index Of the Web.com" [16]. We have also considered Yahoo! as the source of URLs. However, many hyperlinks on Yahoo! link to inner pages, not homepages of the Web sites. "Index Of the Web.com", on the other hand, indexes only Web sites, that is convenient for conducting an automatic survey on local site-searches.

A total of 838 Web sites in eight different categories have been retrieved. The survey program, that we developed, attempted to find a local site-search form from each of the retrieved sites. The survey program invokes components of the proposed system, thus, the forms found by the survey program can be processed by the proposed system. As shown in Table 1, although different categories exhibited different percentage of sites that host local site-search, most of them are around 40%. The percentage of all categories is 43%. These results show that local site-search is common on the Web.

Table 1. Percentages of the sites that host a local site-search form.

Category	Total number of the evaluated sites	Number of sites having local site-search	% of sites having local site-search
Automobile manufacturers	62	25	40
Business	196	81	41
Colleges and universities	127	93	73
Finance	78	21	27
Government departments	183	73	40
Health and medicine	93	35	38
Sports	59	23	39
Travel	40	8	20
Total	838	359	43

The source, “Index Of the Web.com”, contains only a small portion of the sites on the Web, This portion is relatively high-quality compared to the general Web. As the goal of a topical crawler is to gather high-quality relevant Web pages, not necessarily all the relevant pages on the Web, the results of the survey based on “Index Of the Web.com” is sufficient for supporting the system proposed in this paper. For the 57% sites that do not have a local site-search form or the program has failed to find one, traditional techniques (random crawling, focus crawling, or intelligent topical crawling) may be used.

4 Operational model

Figure 1 illustrates the proposed operational model of topical crawling through Web searches and local site-searches. The target topic is initially represented by a broad query (a couple of keywords) describing a category, that can be submitted to a generic Web search engine. From the response of the Web search, those results which are homepages of Web sites are extracted. For each of the homepages, the six steps shown in Figure 1 are performed.

In the initial Web searches, only results which are homepages are considered. This policy can be justified by the following reasons: (1) Homepage of a Web site genuinely describes the categories of the contents published on the site. If a site has a large number of relevant information with regard to the topic, the homepage of the site will very likely be found by a generic search engine. (2) If the homepage of a site is not relevant to the topic, but an inner page of the site is relevant to it, then the inner page is ignored, because it is less likely that a large number of relevant documents can be obtained from this portion of the Web.

```
function: topical_crawling_by_searching
input: an initial query representing the topic
output: documents relevant to the topic
begin
    Web search: Use a generic search engine to search sites, denoted by $s_i$, $i = 1, 2, \ldots, n$, whose homepages are
    relevant to the initial query.
    for each $s_i$, perform local site-search
        (1) Fetch the home page of $s_i$.
        (2) Find the local site-search form.
        (3) Learn a query to submit the form by probing.
        (4) Submit the search form using the learned query.
        (5) Parse the response page and fetch all result links.
        (6) If next link exists in the response page, then fetch the next page and goto step 5.
    end-for
    return the pages fetched
end
```

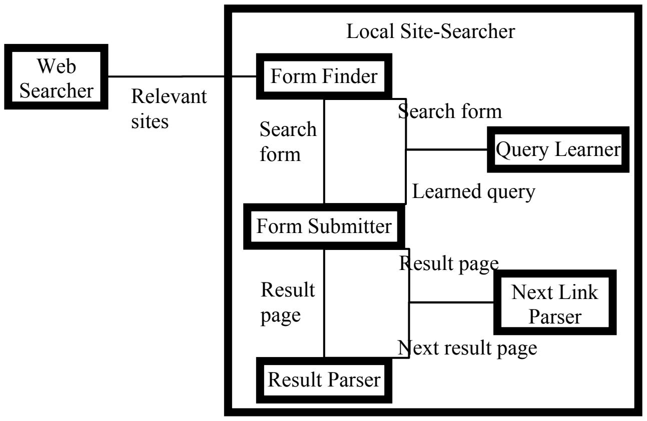
Figure 1. Topical crawling through Web searches and local site-searches.

For each of the homepages returned by the Web searcher, the six steps shown in the for loop in Figure 1 are performed orderly. First, the homepage is fetched. Second, a heuristic approach is used to identify the HTML form of local site-search from the home page. If the search form does not exist, then it continues to the next homepage. Third, a query is learned by probing, starting from submitting an initial query (a couple of keywords) to the local site-search. More related keywords are extracted from the top relevant documents returned from the probing. These related keywords are appended into the initial query to form a final query. Fourth, the final query is submitted to the local site-search. Fifth, result links are parsed from the response of the local site-search. The anchor text and the snippet of a link are analyzed to estimate if the document is relevant to the query. If the link passes the relevance test, then it is fetched. Sixth and finally, the system looks for a link to the next page. If the next page is found, then the system goes back one step, otherwise the process for this site ends.

The system architecture of a realization of this operational model is shown in Figure 2, where, the rectangles represent components, and the arrows represent the directions of data flow. The system consists of two main components: the Web searcher and the local site-searcher. The local site-searcher further contains five sub-components to realize the functionalities of finding the search form, submitting the form, and parsing relevant documents from the response of the search, respectively. The following section explains the implementation of each component in detail.

5 Implementation

The operational model is realized as a two-level querying system. At the first level, the system conducts a broad topic search on the Web. The query used for this level usually describes a category, for instance, "automotive". The second level search is the local site-search on the filtered sites provided by the first level Web search. The query for the local site-search describes the desired topic in more detail, for instance, "Toyota SUV".



Figture 2. System architecture.

5.1 Web Searcher

The Web searcher component filters sites that are relevant to a query. A generic Web search API (Google SOAP search API [17]) is invoked in the current implementation. The adopted API does not have a parameter for only searching sites. Therefore, it is used to search all relevant Web documents, and then the results, that are the default pages of sites, are filtered. Table 2 illustrates the example of searching Web sites which host "how to" articles. Our next attempt will be using meta-searches (combining other publicly available search APIs, e.g., Yahoo! and Microsoft Live search) to explore the possibility of improving the coverage of the Web search.

5.2 Local Site-Searcher

The sites filtered by the Web searcher are sent to the local site-searcher that makes use of the keyword search service provided by the sites to obtain the documents relevant to the query. The following sections introduce the implementation of the sub-components that constitute the local site-searcher.

Table 2. Sites filtered by the Web searcher component querying "how to".

URL list returned by Google search API	Sites filtered by the Web searcher component
http://www.ehow.com/	http://www.ehow.com
http://www.wikihow.com/	http://www.wikihow.com
http://www.instructables.com/	http://www.instructables.com
http://www.wonderhowto.com/	http://www.wonderhowto.com
http://www.howtodothings.com/	http://www.howtodothings.com
http://www.expertvillage.com/	http://www.expertvillage.com
http://www.howcast.com/	http://www.howcast.com
http://openvpn.net/howto.html	http://www.diynetwork.com
http://tldp.org/HOWTO/	…
http://www.diynetwork.com/	
…	

5.3 Locate Site-Search Form

For each site filtered by the Web searcher, the system attempts to locate a site-search form from the homepage of the site. Figure 3 illustrates a typical local site-search form and its HTML source, where the information that we intend to retrieve from the form is indicated in bold face. We observed that the common features of these search forms are as follows:
(1) There are less than three input elements in the form.
(2) There is only one text input element.
(3) There may be the term of "search" occurring in the form.

The first two features are used to locate the form. If there are more than one such forms found, then the third feature is used to discriminate the site-search form from the others.

After a search form is located, the value of the "action" attribute and the name of the text input element are extracted and stored for submitting the form later. In addition, all hidden inputs in the form are parsed and later appended to the HTTP request while submitting the form because the submission may fail otherwise. It needs to be noticed that forms implemented by Javascripts cannot be handled in the current implementation that is also a problem in crawling the hidden Web [13].

5.4 Parse Result Page

A typical local site-search result is illustrated in Figure 4. We are interested in three types of information on the page, that are indicated by rounded rectangles in Figure 4: (1) the number of results displayed per-page and the total number of results, (2) the hyperlink to the next result page, and (3) the list of URLs, titles, and snippets of the displayed results. Three parsers are created to extract the information respectively.

From the response page, the total number parser seeks the pattern as follows:

results [no more than 40 arbitrary characters] 1 - $number_perpage [less than 10 arbitrary characters] $total_number

The pattern starts from the term “results”, and then after no more than 40 characters, “1 -” occurs, followed by the first target: the result number per-page. After the first target is retrieved, there may be less than 10 characters before the occurrence of the second target: the total number. The next link parser seeks a link whose anchor text contains the word “next”, and the length of the anchor text is less than ten. The result parser decides if a hyperlink links to a result document by seeking the query in the anchor text of the link and 40 words following the link (the snippet).

```
<form action="http://www.ku.edu/~http/cgi-bin/search/search.php" class="ku_search" method="post">
  <label for="searchtype" style="visibility:hidden; display:none;">Search Type</label>
  <select id="searchtype" name="site">
    <option value="kuweb" selected="selected">Search KU Web</option>
        <option value="kupeople">Search KU People</option>
        <option value="kuevents">Search KU Events</option>
        <option value="kuinfo">Search KU Info</option>
    </select>
  <label for="searchtext" style="visibility:hidden; display:none;">Search Text</label>
  <input    id="searchtext"      class="ku_searchtext"    name="keyword"    type="text"    value="keyword/name"
onclick="if(this.value == 'keyword/name') this.value=";" size="20" />
  <input  class="button"  name="Search"  type="image"  src="http://webmedia.ku.edu/images/template2009/search.gif"
alt="Search" />
</form>
```

Figure 3. The local site-search form on http://www.eecs.ku.edu.

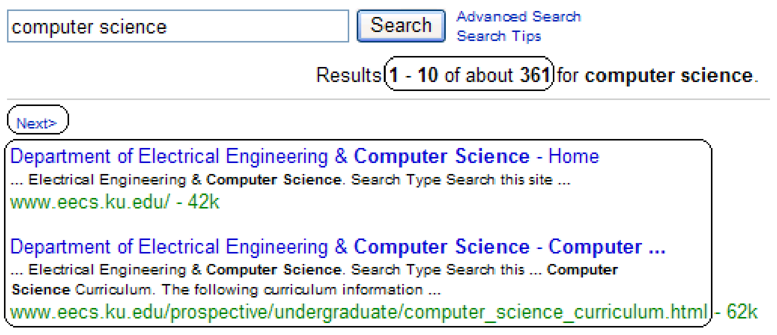
It is known that the total number of relevant results returned by the search engines may not truthfully reflect the actual number of available relevant results [18] [19]. Therefore, in addition to the total number parser, we propose an alternative approach to better estimate the total number. This approach fetches the next link repeatedly until a next link cannot be found, in order to count the actual number of response pages, and then the estimated total number is the product of the number of results per-page and the number of response pages.

5.5 Learn query

There are often more than one set of terminology to describe a topic. The query learner component seeks a suitable query for a certain site by probing, starting from submitting an initial query to the local site-search. A query is good if it can obtain the most relevant results to the topic. However, it is not efficient or meaningful to fetch all the results to decide the goodness of a query, denoted as G. Therefore we use the product of the precision of the first response page and the total number of the returned results to estimate G:

G = precision of the first response page × total number of results

The results on the first response page are fetched and sent to a classifier to decide if they are relevant to the topic. The precision of the first page equals the number of the obtained relevant results divided by the number of results per-page.



computer science

Search

Results 1 - 10 of about 361 for **computer science**.

Department of Electrical Engineering & **Computer Science** - Home
... Electrical Engineering & **Computer Science**. Search Type Search this site ...
www.eecs.ku.edu/ - 42k

Department of Electrical Engineering & **Computer Science** - **Computer ...**
... Electrical Engineering & **Computer Science**. Search Type Search this ... **Computer Science** Curriculum. The following curriculum information ...
www.eecs.ku.edu/prospective/undergraduate/computer_science_curriculum.html - 62k

Figure 4. Local site-search results of http://www.eecs.ku.edu.

The query learning function is illustrated in Figure 5. The probing process starts from an initial query. The ten most common terms (stop words are eliminated) are extracted from the relevant documents on the first response page. These terms are candidates for being appended to the initial query to form the final learned query. Two through five (the range is adjustable) terms are randomly picked from the ten candidates to repeat the probing iteration. The query with the highest goodness value G is returned when the program reaches a certain threshold for termination.

6 Experimental Results

There is no universal metric to measure and compare the efficiency of all topical crawlers. However, for specific tasks, we are able to evaluate the trade-off between efficiency and completeness of the topical crawling [20] [21]. We aimed to compare the performance of the crawler proposed in this paper with the intelligent crawler [4]. We used the same or similar queries. Thus the results we obtained can be compared to those in the related work [4].

We used harvest rate (percentage of the fetched documents that are relevant to the topic [4]) to measure the performance of the crawler. A human user reviewed all retrieved documents in order to determine the harvest rate. As described previously, the crawler seeks the query in the title or the snippet of a link to decide whether the document is relevant and should be fetched. To calculate the harvest rate, we need to know the number of the fetched documents that are actually relevant to the topic. We used a human user to identify the relevant ones from the fetched documents. To make this possible, we limited the scale of the experiments to thousands of documents from hundreds of Web sites.

In the experiments, the maximum number of the results returned from the Web searches was set to 1,000. From the up to 1,000 results, the homepages of Web sites were filtered, and the inner pages were eliminated. The maximum number of the results fetched from one certain site was set to 100, to ensure the generality of the experiment. Four pairs of queries were used in the experiments, as shown in Table 3. The first two pairs ("Sports, baseball" and "Travel, Paris") had been used to evaluate the intelligent crawler [4], because only the first two pairs were available for all the systems to compare. We believe that eventually all intranet sites will have local site-search. Right now it is still a developing process. The education category has a much high percentage because educational institutes have the advantages of many free resources, both financial support and intelligent resources, and access to most recent technologies.

Compared to the 40% average harvest rate of the intelligent crawler [14], the system proposed in this paper achieved much higher accuracy (an average harvest rate of 88%). The comparison of the proposed system to the intelligent crawler and the random crawler evaluated in related work [4] in terms of harvest rate is shown in Figure 6. The crawling by the proposed two-level querying (Web search + local site-search) has shown to be very efficient despite its simplicity.

```
function: query_learning
input: an initial query and a local site-search form
output: learned query
begin
    initialization
        $Q_{best}$ = the initial query
        Submit $Q_{best}$ to the local site-search
        Fetch the result documents on the first response page
        Identify the relevant ones from the fetched documents (assuming a classifier is available)
        Extract top 10 common terms from the relevant documents, denoted as $K$
        $probing\_number = 20$
        Calculate $G_{best}$ for $Q_{best}$ using (1)
    while $probing\_number > 0$
        Randomly pick 2-5 terms from $K$, denoted as $K'$
        $q$ = the initial query + $K'$
        Submit $q$ to the local site-search
        Calculate $G_q$ using (1)
        if $G_q > G_{best}$ then
            $G_{best} = G_q$
            $Q_{best} = q$
        $probing\_number = probing\_number - 1$
        end-if
    end-while
    return $Q_{best}$
end
```

Figure 5. Learning the query for local site-searches.

The 10-20% non-relevant documents fetched by the system fell into the following two categories:

(1) Non-result links on the response page wrongly classified by the system as relevant links.

(2) Non-relevant results wrongly returned by the local site-search.

Although the title and the snippet of a document provide a strong evidence of whether the document is relevant to a topic, there are noisy links on the response page (e.g., ads) that do not truthfully describe themselves. Also, a small portion of the local site-searches are still using simple ranking algorithms (e.g., the Boolean model), that have exhibited low precisions.

We have applied the proposed two-level querying crawler to fetch articles regarding “how to do things”, in order to build a process base for a previously proposed search engine [22] [23]. Compared to other crawling techniques, the proposed system was able to gather a large number of how-to articles without wasting computing resources on fetching many non-relevant documents.

7 Conclusion

In this paper, we have proposed an alternative topical crawler, which gathers topical resources through Web searches and local site-searches. This approach has good trade-off between efficiency and completeness and has achieved 88% average harvest rate in the experiments. Another advantage of this approach is that it is easy to implement. Therefore, we were able to present the feasibility of using local site-searches in topical resource discovery. The proposed approach can be combined with existing solutions to implement more generally applicable and reliable topical crawlers.

Table 3. Harvest rates of different queries.

Query for Web search	Query for local site-search	Sites	Fetched documents	Fetched relevant documents	Harvest rate (%)
Sports	baseball	139	873	716	82
Travel	paris	138	483	435	90
Education	computer science	556	1,907	1,839	96
Finance	stock	130	808	586	73
Total		963	4,071	3,576	88

Besides topical crawling, the local site-searches can also be used in generic crawlers. Although nowadays many local site-searches use Google Custom Search [24], most dot-com sites host their own local site-search engines. Taking advantage of the local site-searches can help the generic crawlers reduce the size of the crawling frontier and simply the implementation and the operation. Another benefit a generic crawler can get from local site-searches is to find isolated pages that have no in-linking pages. Such pages cannot possibly be discovered by a traditional crawler, but they may be found by local site-searches.

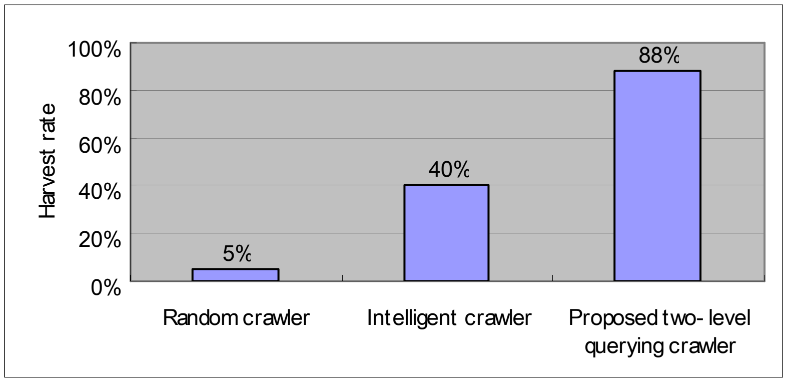


Figure 6. Comparison of the harvest rates of the three crawling techniques.

References

[1] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., and Raghavan, S. (2001). Searching the Web. *Transactions on Internet Technology* 1, 1 (Aug. 2001), 2-43.

[2] Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. In *Proceedings of the Eighth international Conference on World Wide Web* (Toronto, Canada). 1623-1640.

[3] Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Distributed Hypertext Resource Discovery Through Examples. In *Proceedings of the 25th international Conference on Very Large Data Bases* (September 07 - 10, 1999). 375-386.

[4] Aggarwal, C. C., Al-Garawi, F., and Yu, P.S. (2001). Intelligent crawling on the World Wide Web with arbitrary predicates. In *Proceedings of the 10th international Conference on World Wide Web* (Hong Kong, Hong Kong, May 01 - 05, 2001). 96-105.

[5] Aggarwal, C.C., Al-Garawi, F., and Yu, P.S. (2001). On the design of a learning crawler for topical resource discovery. *ACM Transactions on Information Systems* 19, 3 (Jul. 2001), 286-309.

[6] Cohen, W.W. and Singer, Y. (1996). Learning to query the web. In *AAAI Workshop on Internet-Based Information Systems,* 1996.

[7] Ipeirotis, P.G., Agichtein, E., Jain, P., and Gravano, L. (2006). To search or to crawl?: towards a query optimizer for text-centric tasks. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (Chicago, IL, USA, June 27 - 29, 2006). 265-276.

[8] Aggarwal, C.C. (2004). On Leveraging User Access Patterns for Topic Specific Crawling. *Data Mining and Knowledge* Discovery 9, 2 (Sep. 2004), 123-145.

[9] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused Crawling Using Context Graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases* (September 10 - 14, 2000). 527-534.

[10] Liu, H., Milios, E., and Janssen, J. (2004). Probabilistic models for focused web crawling. In *Proceedings of the 6th Annual ACM International Workshop on Web information and Data Management* (Washington DC, USA, November 12 - 13, 2004). 16-22.

[11] Qin, J., Zhou, Y., and Chau, M. (2004). Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries* (Tuscon, AZ, USA, June 07 - 11, 2004). 135-141.

[12] Chakrabarti, S., Punera, K., and Subramanyam, M. (2002). Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th International Conference on World Wide Web* (Honolulu, Hawaii, USA, May 07 - 11, 2002). 148-159.

[13] Pant, G. and Srinivasan, P. (2005). Learning to crawl: Comparing classification schemes. *ACM Transactions on Information Systems* 23, 4 (October 2005), 430-462.

[14] Raghavan, S. and Garcia-Molina, H. (2001). Crawling the Hidden Web. In *Proceedings of the 27th International Conference on Very Large Data Bases* (September 11 - 14, 2001). 129-138.

[15] Madhavan, J., Ko, D., Kot, Ł., Ganapathy, V., Rasmussen, A., and Halevy, A. (2008). Google's Deep Web crawl. In *Proceedings of VLDB Endowment* 1, 2 (Aug. 2008), 1241-1252.

[16] Index of the Web.com. http://www.indexoftheweb.com/.

[17] Google SOAP Search API. http://code.google.com/apis/soapsearch/.

[18] Anagnostopoulos, A., Broder, A. Z., and Carmel, D. (2005). Sampling search-engine results. In *Proceedings of the 14th International Conference on World Wide Web* (Chiba, Japan, May 10 - 14, 2005). 245-256.

[19] Anagnostopoulos, A., Broder, A. Z., and Carmel, D. (2006). Sampling Search-Engine Results. *World Wide Web* 9, 4 (Dec. 2006), 397-429.

[20] Menczer, F., Pant, G., Srinivasan, P., and Ruiz, M.E. (2001). Evaluating topic-driven web crawlers. In *Proceedings of the 24th Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (New Orleans, Louisiana, United States). 241-249.

[21] Menczer, F., Pant, G., and Srinivasan, P. (2004). Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology* 4, 4 (Nov. 2004), 378-419.

[22] Liu, Y. and Agah, A. (2009). Crawling and extracting process data from the Web. In *Proceedings of the 5th International Conference on Advanced Data Mining and Applications* (Beijing, China, Aug. 17-19, 2009) 545-552.

[23] Liu, Y. and Agah, A. (2009). A Prototype Process-Based Search Engine. In *Proceedings of the 3rd IEEE International Conference on Semantic Computing.* Berkeley, CA, September 14-16, 2009, pp. 481-486.

[24] Google Custom Search. http://www.google.com/coop/cse/.