ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 11, November 2013

An Effective Focused Web Crawler for Web Resource Discovery

Nandar Win Min, Aye Nandar Hlaing

Abstract— **In the given volume of the Web and its speed of change, the coverage of modern search engines is relatively small. Web crawling is the process used by search engines to collect pages from the Web. Therefore, collecting domain-specific information from the Web is a special theme of research in many papers. In this paper, we introduce a new effective focused web crawler. It uses smart methods to speed up crawling of relevant pages and then follow the promising links first in order to find the most important, topically related pages. The first result from a meta-search engine is used as the seed pages to increase the search coverage of the web. In analysing text, similarity measurement applies to different parts of the web pages including the title, the body, anchor text and URL tokens. It can increase the relevance and quality of the Web pages pointed to by target URLs. To enhance the accuracy of crawling, Naive Bayes Classifier is used to determine the optimal order in which the target URLs are visited. The result shows that a crawler with a good crawling method leads to the good accuracy.**

***Index Terms*—focused web crawler, seed pages, meta-search, Naive Bayes classifier, similarity space model.**

I. Introduction

The Web, containing a large amount of useful information and resources, is expanding rapidly. Collecting domain-specific documents/information from the Web is very important for the scientific community. A crawler is a program that retrieves Web pages, commonly for use by a search engine or a Web cache.

Crawlers are widely used today. Crawlers for the major search engines (e.g., Alta Vista, InfoSeek, Excite, and Lycos) attempt to visit most text Web pages, in order to build content indexes. Other crawlers may also visit many pages, but may look only for certain types of information (e.g., email addresses). At the other end of the spectrum, we have personal crawlers that scan for pages of interest to a particular user, in order to build a fast access cache (e.g, NetAttche) [1].

In reality, many search engines do not cover all the visible pages [2]. Therefore, there is a need for a more effective crawling method to collect more accurate data. One of the most common approaches is to limit the crawler to a few specified subjects. In this way, the crawler is able to retrieve the most common pages. This method called Focused crawling [3]. The following Fig.1 illustrates the difference between regular crawling and focused crawling.

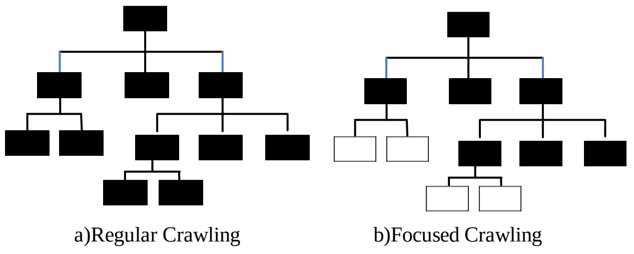

Figure 1. Comparison between focused and regular crawlers

Focused crawlers can selectively retrieve Web documents relevant to a specific domain to build collections for domain-specific search engines. Traditional focused crawlers normally adopting the simple Vector Space Model and local Web search algorithms typically only find relevant pages with low precision. Recall is also low, since they explore a limited sub-graph of the Web that surrounds the starting URL set, and will ignore relevant pages outside this sub-graph. In this paper, we investigated how to apply an inductive machine learning algorithm and meta-search technique, to the traditional focused crawling process, to overcome the above mentioned problems and to improve performance.

The rest of the paper is organized as follows. Section 2 describes literature reviews of the proposed system. Background of the theory, then, is presented in Section 3. And, the proposed framework of the focused crawler is discussed in Section 4. The performance evaluation is explained in Section 5. Finally, conclusion of the system is stated in Section 6.

II. Literature Reviews

One of the first generation of focused crawler was discussed in [4]. Their performance depends highly on the selection of good starting pages (seed pages). Typically users provide a set of seed pages as input to a crawler in [5] or, alternatively, seed pages are selected among the best answers returned by a Web search engine using the topic as query in[6, 7]. Early approaches to learning crawlers use a Naïve Bayesian classifier (trained on web taxonomies such as Yahoo) for distinguishing between relevant and not relevant pages [8]; others suggest using decision trees [9], First Order Logic [10], Neural Networks and Support Vector Machines

Manuscript received Nov, 2013.

***Nandar Win Min**, Faculty of ICT, University of Technology (Yatanarpon Cyber City, Pyin Oo Lwin, Myanmar.*
***Aye Nandar Hlaing**, Faculty of ICT, University of Technology (Yatanarpon Cyber City, Pyin Oo Lwin, Myanmar. (e.*

[11]. In [12] Support Vector Machines are applied to both page content and link context, and their combination is shown to outperform methods using page content or link context alone.

The concept of "context graphs" was introduced in [6,13]; first back links to relevant pages are followed to recover pages leading to relevant pages. These pages along with their path information form the context graph. The original context graph method builds classifiers for sets of pages at distance 1, 2,… from relevant pages in the context graph. The focused crawler uses these classifiers to establish priorities of visited pages, priorities assigned to links extracted from these pages. An extension to the context graph method is the Hidden Markov Model (HMM) crawler [5].

In the structural approach, link analysis of pages is vastly used building a relation graph. The only Page Rank algorithm was used in [1]. The Page Rank method computes the score of the URL not fetched yet as well as the URL with best scores firstly retrieved. The Page Rank algorithm ranks them according to the frequency of repetitions and the importance of these in other pages. Under this approach, the crawling would not stop upon reaching an irrelevant page [8].

Unlike the above research, in our approach, similarity measurement is applied to different parts of the web pages including the title, the body, the anchor text and URL token. Furthermore, Naive Bayes Classifier is very useful in deciding results of user queries. Thus, our approach can be more efficient in accuracy and performance.

III. Theory Background

Crawlers used by general purpose search engines retrieve massive numbers of web pages regardless of their topic. Focused crawlers work by combining both the content of the retrieved Web pages and the link structure of the Web for assigning higher visiting priority to pages with higher probability of being relevant to a given topic. Focused crawlers can be categorized as follows:

1. Classic focused crawler
2. Semantic focused crawler
3. Learning focused crawler

In this paper, our focused crawler is considered with particular emphasis on learning crawler.

A. *Learning Focused Crawler*

Typically, a learning crawler is supplied with a training set consisting of relevant and not relevant Web pages which is used to train the learning crawler. The crawler learns user preferences on the topic from a set of example pages (training set). Specifically the user provides a set of pages and specifies which of them are relevant to the topic of interest. During crawling, each downloaded page is classified as relevant or not relevant and is assigned a priority. Higher visit priority is assigned to links extracted from web pages classified as relevant to the topic.

Learning focused crawler design is discussed as follow:

a) **Input**: Crawlers take as input a number of starting (seed) URLs and a training set.
b) **Page downloading**: The links in downloaded pages are extracted and placed in a queue. A focused crawler reorders queue entries by applying content relevance or importance criteria.
c) **Content processing**: Downloaded pages are lexically analysed and reduced into term vectors (all terms are reduced to their morphological roots by applying a stemming algorithm and stop words are removed).
d) **Priority assignment**: Extracted URLs from downloaded pages are placed in a priority queue where priorities are determined based on the type of crawler and user preferences.
e) **Expansion**: URLs are selected for further expansion and steps (b) **-** (e) are repeated until some criteria (e.g. the desired number of pages have been downloaded) are satisfied or system resources are exhausted.

B. *Meta-Search Engine*

Meta-search engine is a search system that does not have its own database of Web pages. It answers the user query by combining the results of some other search engines which normally have their databases of Web pages.

After receiving a query from the user through the search interface, the meta-search engine submits the query to the underlying search engines (called its component search engines). The returned results from all these search engines are then combined (fused or merged) and sent to the user.

It increases the search coverage of the Web. It may also improve the search effectiveness. Each component search engine has its ranking algorithm to rank relevant pages, which is often biased. By combining the results from multiple search engines, their biases can be reduced and thus the search precision can be improved [14].

C. *Similarity Measurement*

Text analysis methods include similarity scoring approaches and machine learning algorithms. In this paper, cosine similarity measure is used. This content-based similarity measure has been applied to the content of Web. Moreover, it has been widely used in scientific research activities especially in the text classification field. To compute this, suppose we have a collection with t distinct index terms t_j, a document d_i can be represented as follows: $d_i=(w_{i1} ,w_{i2} ,...,w_{it})$, where w_{ij} represents the weight assigned to term t_j in document d_i.

For the cosine measure, the similarity between two documents d1 and d2 can be calculated as:

$$cosine(d_1,d_2) = \frac{\sum_{i=1}^{t} w_{1i} * w_{2i}}{\sqrt{(\sum_{i=1}^{t} w_{1i}^2) * (\sum_{i=1}^{t} w_{2i}^2)}} \tag{1}$$

This similarity measure is used in different parts of Web pages including the body, the title, anchor text and URL tokens.

IV. Focused Crawling Framework

A. *Learning Classifier*

A Naive Bayes classifier is built for making class decision to each Web page. During classification, four features representation of the pages are used.

1. Title Text feature
2. Body Text feature
3. Anchor Text feature
4. URL Tokens feature

According to these features, training data set table is created. Table 1 is the sample training data set table. The notation for feature k and observed web page o is $b_k(O)$. We propose four features (k = 1..4).

1. Title Text Feature: Maximal similarity value between the title of the content of a given candidate page and the set of targets.

$b_1(O)$ = similarity of topic and keywords of page O in title

(2)

2. Body Text Feature: Maximal similarity value between the body of the content of a given candidate page and the set of targets.

$b_2(O)$ = similarity of topic and keywords of page O in body

(3)

3. Anchor Text Feature: The anchor text around the link pointing to an observed page O often is closely related to the topic of the page. Human's skills and knowledge of discriminating between links when they browse mostly rely on the anchor texts.

$b_3(O)$ = similarity of topic and keywords of page O in anchor text (4)

4. URL Tokens Feature: The tokens in the URL of an observed page may contain valuable information about predicting whether or not a page is a target page or potentially leading to a target. For example, a URL containing "linux" is more likely to be a web page about linux related information, and a URL which contains the word "operating system" or "OS" indicates that with high probability, it may lead to a Linux page. We first parse the URL into tokens, then compute the similarity between tokens and topic keywords.

$b_4(O)$ = similarity of topic and keywords of page O in URL tokens (5)

TABLE I. Sample Training Data Set

Title Text Sim	Body Text Sim	Anchor Text Sim	URL Tokens Sim	Class
0.9-1	0.8-0.89	0.9-1	0.8-0.89	Yes
0.8-0.89	0.8-0.89	0.7-0.79	0.7-0.79	Yes
0.7-0.79	0.8-0.89	0.6-0.69	0.7-0.79	Yes
0.7-0.79	0.6-0.69	0.7-0.79	0.5-0.59	Yes
0.4-0.49	0.3-0.39	0.5-0.59	0.4-0.49	No
0.6-0.69	0.5-0.59	0.6-0.69	0.7-0.79	Yes
0.6-0.69	0.7-0.79	0.5-0.59	0.5-0.59	Yes
0.3-0.39	0.2-0.29	0.4-0.49	0.3-0.39	No
0.5-0.59	0.6-0.69	0.5-0.59	0.4-0.49	Yes
0.5-0.59	0.5-0.59	0.5-0.59	0.4-0.49	Yes
0.3-0.39	0.2-0.29	0.3-0.39	0.1-0.19	No

B. *Proposed Focused Crawling Framework*

In this section, we present the high level steps of our focused crawler and the complete system flow diagram is shown in figure 2.

Steps of our proposed focused crawler are as follows:

Step 1: Initialization. As a test set, we performed a focused crawler for "Linux" section. A query is given through the search interface. Top 10 results from some search engines are combined. These results are used as seed pages. It can obtain diverse relevant URLs globally from the whole search space.

Step 2: Classification. A data collection obtained from DMOZ will be used as the training and validation collections. In this step, Naive Bayes Classifier with four features representation is used to decide if the Web page is a linux-related Web page. If yes, this Web page will survive and be put into queue, then be given the class "Yes". Otherwise, this Web page will be given the class "No".

Step 3: Sorting priority queue. After classification, the priority queue is sorted by the average similarity value of each page. This value can be obtained by averaging the four similarity measurements.

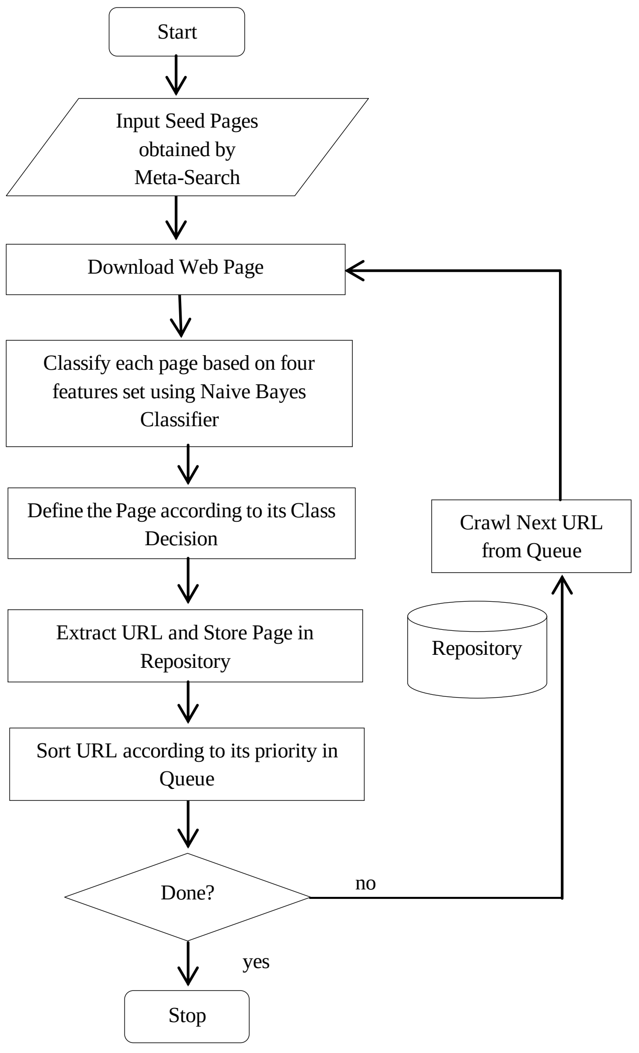
Start
Input Seed Pages obtained by Meta-Search
Download Web Page
Classify each page based on four features set using Naive Bayes Classifier
Define the Page according to its Class Decision
Extract URL and Store Page in Repository
Sort URL according to its priority in Queue
Done?
no
yes
Stop
Crawl Next URL from Queue
Repository

Figure 2. System flow diagram of the proposed focused crawler

Step 4: Crawling. The outgoing links of the surviving relevant Web pages will be collected and put into the crawling priority queue. During crawling, it starts with the head page of the priority queue.

Step 5: Termination. Steps 2 to 4 are repeated until the number of Web pages in a local collection repository reaches.

C. Proposed Crawling Algorithm

Our proposed crawling algorithm is shown in figure 3.

```
Step 1: enqueue(url_queue, starting_url obtained from
            Meta- search);
Step 2: while (not empty(url_queue) ) do
    url = dequeue( url_queue);
    page = crawl_page(url);
    enqueue(crawled_pages, (url, page));
    url_list = extract_urls(page);
    for each u in url_list do
      if [u not in url_queue] and [(u,-) not in
      crawled_pages] then
        Calculate similarity of topic and
        keywords of page in title;
        Calculate similarity of topic and
        keywords of page in body;
        Calculate similarity of topic and
        keywords of anchor text;
        Calculate similarity of topic and
        keywords of url token;
        Classify whether u is related page or
        not using Naive Bayes Classifier;
        if [classification result is yes] then
        enqueue(url_queue, u);
        end if
      end if
    end for
  reorder_queue(url_queue);
end while
```

Function Description

enqueue(queue, element) : append element at the end of queue.

dequeue(queue) : remove the element at the beginning of queue and return it.

reorder_queue(queue) : reorder queue using average similarity value.

Figure 3. Proposed Crawling Algorithm

V. Performance Evaluation

For the evaluation, Average Precision and Target Recall are computed. The Precision shows the accuracy of the algorithm while the recall represents the integrity of search algorithm.

Average Precision: It is a more general form of harvest rate. In our proposed crawler, the scores will be provided through priority value. Then, these scores will be averaged over the crawled pages. Such averages will be computed over the progress of the crawl (first 10 pages, first 20 pages and so on).

Target Recall: A set of known relevant URLs will split into two disjoint sets-*targets* and *seeds*. The crawler will be started from the seeds pages and the recall of the targets will be measured. The target recall will be computed as:

$$\text{Target} - \text{Recall} = \frac{|P_t \cap P_c|}{|P_t|} \quad \textbf{(6)}$$

where P_t is the set of target pages, and P_c is the set of crawled pages.

Finally, final performance evaluation F is calculated as follows;

$$F = \frac{2 * \text{Average Precision} * \text{Target Recall}}{\text{Average Precision} + \text{Target Recall}} \quad (7)$$

To show the strength of our method, we used 30% and 10% of DMOZ resources. In 30% dataset, we considered 25% for validation and the rest for testing. We compare our method with simple SVM and only Naive Bayes Classifier crawlers. For 10%, we set 7% for validation. F of both 30% and 10% datasets is shown in Table 2.

TABLE II. Comparison Evaluation

Class	Meta Search + Naive Bayes Classifier	Naive Bayes Classifier	Simple SVM
30%	72%	65%	56%
10%	67%	61%	54%

VI. Conclusion

In this paper, particular emphasis is given to learning focused crawlers capable of learning not only the content of target pages but also paths leading to target pages. In fact, learning crawlers perform a very difficult task: they attempt to learn web crawling patterns leading to relevant pages possibly through other not relevant pages thus increasing the probability of failure. But, with a good crawling strategy, it seems to be possible to build crawlers that can rather quickly obtain a significant portion of the hot pages.

Normally, a document relevant to a specific topic frequently contains explicitly a set of topic-specific keywords. For example, a TCP/IP document often contains keywords "tcp, ip, header, protocol", etc. Therefore, the lexical keywords are a significant factor. In this paper, the user needs to specify these keywords before his/her topic-specific browsing.

References

[1] Junghoo Cho, Hector Garcia-Molina and Lawrence Page, "Efficient Crawling Through URL Ordering", In Proceedings of the seventh international conference on World Wide Web 7, p.161-172, The Netherlands, 1998.

[2] A. Gulli, A. Sigmorini, "The Indexable web is more than 11.5 billion pages", In Proceedings of the 14th international conference on World Wide Web, pp. 902-903, ACM Press, 2005.

[3] R. Baeza- Yates and B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.

[4] Bra, P.D., Houben, G., Kornatzky, Y., and Post, R., "Information Retrieval in Distributed Hypertexts".in Proceedings of the 4th RIAO Conference. p. 481-491. 1994. New York.

[5] Hongyu Liu, EvangelosMilios, Jeannette Janssen. "Probabilistic Models for Focused Web Crawling".6th ACM International Worksho on Web Information and Data Management, November 2004, Washington, DC, USA.

[6] M. Diligenti, F. Coetzee, S. Lawrence, C. Giles, and M. Gori."Focused Crawling Using Context Graphs".In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, September 2000.

[7] Pant, G., Tsjoutsiouliklis, K., Johnson, J., and Giles, C. L."Panorama: Extending digital libraries with topical crawlers".Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries.2004b, p. 142–150.

[8] SoumenChakrabarti, Martin van den Berg, Byron Dom. "Focused Crawling: a new approach to topic-specific Web resource discovery".In Proceedings of the 8th International World Wide Web Conference (WWW8), 1999.

[9] Li, Jun, Furuse, K. and Yamaguchi, K."Focused Crawling by Exploiting Anchor Text Using Decision Tree". Proceedings of the 14th International World Wide Web Conference. 2005, pp. 1190-1191.

[10] Xu, Qingyang and Zuo, Wanli. "First-order Focused Crawling". Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, Pages: 1159 – 1160, 2007.

[11] Pant, G. and Srinivasan, P."Learning to Crawl: Comparing Classification Schemes".ACM Transactions on Information Systems (TOIS). 23(4), 430-462. 2005.

[12] Pant, G. and Srinivasan, P."Link contexts in classifier-guided topical crawlers".IEEE Transactions on Knowledge and Data Engineering.vol.18 (01), p. 107-122, 2006.

[13] Zhaoqiong GAO, Yajun DU, Liangzhong YI, Yuekui YANG, Qiangqiang PENG. "Focused Web Crawling Based on Incremental Learning". In Journal of Computational Information Systems6:1(2010) 9-16, China.

[14] Bing Liu, "Web Data Mining", ISBN-10 3-540-37881-2 Springer Berlin Heidelberg New York, 2007.

Nandar Win Min is from Myanmar. She was born on 5th May, 1984. She finished her bachelor degree in computer science at Computer University (Taunggyi), Myanmar in 2006. Her Master Degree was finished at Univesity of Computer Studies (Mandalay), Myanmar in 2010. Now, she is currently studying Ph.D of Information and Communication Technology at University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar.