Abstract—The World Wide Web is growing rapidly and many search engines do not cover all the visible pages. Therefore, a more effective crawling method is required to collect more accurate data. In this paper, we introduce an effective focused web crawler containing smart methods. In text analysis, similarity measurement applies to different parts of the Web pages including title, 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, decay concept is used to determine the optimal order in which the targeted URLs are visited. In this measurement, two kinds of threshold are used to limit the crawler to the effective web pages. Finally, to provide sorting URLs, priority equation is used. Our method shows significant performance improvements in crawling efficiency over previous focused crawling.

Keywords—decay concept, focused web crawler, priority equation, similarity space model.

I. INTRODUCTION

The World Wide Web has grown from a few thousand pages in 1993 to more than eight billion pages at present [1]. According to recent statistics, 60 per cent of the user search for a special theme and often use popular and commercial search engines to obtain their results [2]. 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., NetAttache) [3].

In reality, many search engines do not cover all the visible pages [4]. 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 [5]. The following Fig.1 illustrates the difference between regular crawling and focused crawling.

![Comparison between focused and regular crawlers](image)

Focused crawlers rely on two types of algorithms to keep the crawling scope within the desired domain: (1) Web analysis algorithms are used to judge the relevance and quality of the Web pages pointed to by target URLs; and (2) Web search algorithms determine the optimal order in which the target URLs are visited [6]. In this paper, we use effective similarity measurements as Web analysis algorithm. In such a way, the crawler estimates the similarity of the current page and the subject. The similarity above a predefined threshold leads to fetching entire page’s links. Then, as Web search algorithm, decay concept is used to lessen the effect of inaccuracy.

The rest of the paper is organized as follows. Section 2 describes related works of the proposed system. Background of the theory, then, is presented in Section 3. And, the proposed framework of focused web 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. RELATED WORKS

One of the first generation of focused crawler was discussed in [7]. Their performance depends highly on the selection of good starting pages (seed pages). Typically users [8] provide a set of seed pages as input to a crawler or, alternatively, seed pages are selected among the best answers returned by a Web search engine [9], using the topic as query [10]. 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 [11]; others suggest using decision trees [12], First Order Logic [13], Neural Networks and Support Vector Machines [14]. In [15] 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.

Nandar Win Min is with University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar (phone: +95949309935; e-mail: nandarwinmin5.utycc@gmail.com).

Aye Nandar Hlaing is with University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar (e-mail: anandarhlaingucsy@gmail.com).

http://dx.doi.org/10.15242/IIE.E0314109
In the structural approach, link analysis of pages is vastly used building a relation graph. For example, the Page Rank algorithm can be used [16]. 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 [11].

The concept of “context graphs” was introduced in [9][17]; 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 [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, decay concept and priority equation are very useful in ranking results of user queries. Thus, our approach can be more efficient in accuracy and ranking scheme.

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 focused 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 analyzed 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. 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 tj and each document di can be represented as follows: di=(wij ,w1i ..., wit), where wij represents the weight assigned to term tj in document di.

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

\[
\text{cosine}(d_1, d_2) = \frac{\sum_{i=1}^{t} w_{1i}w_{2i}}{\sqrt{\sum_{i=1}^{t} w_{1i}^2} \sqrt{\sum_{i=1}^{t} w_{2i}^2}}
\]

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. Proposed Method

In each focused crawler, calculating the similarity of content between the desired subject and the page is the essential part. In this case, one drawback is how to determine the best threshold. Considering low threshold may lead to entering huge numbers of irrelevant pages. Such ineffective pages results in less efficiency. On the other hand, considering such threshold as high to comply with user’s criteria may result in loosing so many effective pages.

To overcome the drawbacks of thresholds, we propose a decay concept. For each page, we set a variable among 0 and 1. This variable shows the decay concept. Each page with value near 1 shows better similarity. Each child pages inherit the decay of the parent with a percentage of reduction. We set such a value to half of its parent decay [18].

The decays values of child pages always decrease which put among 1, 1/2, 1/4, 1/8, .... The decay concept is applied when the decay value reduces less than a threshold Td. In such cases, the crawler stops traversing new pages. Such assumption is not proper since we may face a page completely relevant to the
desired search query. Actually, we use two threshold values. When the decay value is less than a threshold $T_d$, if the similarity is greater than a threshold $T_s$, we reset the decay value to 1.

### B. Learning Classifier

A kNN 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**: 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)

After classification, the priority queue is sorted by Priority(page) equation as follows:

$$
\text{Priority}(page) = k_1(b_1+b_2) + k_2(b_3) + k_3(b_4)
$$

where $k_1$, $k_2$ and $k_3$ are constants.

### C. Proposed Crawling Algorithm

Our proposed crawling algorithm is shown in Fig. 2.

- **Step 1**: enqueue(url_queue, starting_url);
- **Step 2**: while (not empty(url_queue)) do
  - url = dequeue(url_queue);
  - page = crawl_page(url);
  - url_list = extract_urls(page);
  - enqueue(crawled_pages, (url, 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 body;
      - Calculate similarity of topic and keywords of page in title;
      - 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 KNN 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 priority equation.

### D. 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 Fig. 3.

<table>
<thead>
<tr>
<th>Table I</th>
<th><strong>SAMPLE TRAINING DATA SET</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Sim</td>
<td>URL Token Sim</td>
</tr>
<tr>
<td>0.9-1</td>
<td>0.8-0.89</td>
</tr>
<tr>
<td>0.8-0.89</td>
<td>0.8-0.89</td>
</tr>
<tr>
<td>0.7-0.79</td>
<td>0.8-0.89</td>
</tr>
<tr>
<td>0.5-0.79</td>
<td>0.5-0.59</td>
</tr>
<tr>
<td>0.3-0.39</td>
<td>0.5-0.59</td>
</tr>
<tr>
<td>0.1-0.19</td>
<td>0.3-0.39</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>0.6</td>
<td>0.69</td>
</tr>
<tr>
<td>0.4</td>
<td>0.49</td>
</tr>
<tr>
<td>0.3</td>
<td>0.39</td>
</tr>
<tr>
<td>0.2</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Step 1: Initialization: Users provide a set of seed pages as input to a crawler.

Step 2: Classification: As a test set, we performed a focused crawler for “Linux” section. A data collection obtained from DMOZ will be used as the training and validation collections. In this step, kNN 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: Assigning priority value: After classification, the priority value is assigned to each URL by using the priority equation (6). According to this value, URLs are sorted in priority queue.

Step 4: Crawling: The Breadth-first search crawling will be used to fetch new Web pages. The outgoing links of the surviving relevant Web pages will be collected and put into the crawling priority queue. The reason to choose breadth-first search is that it is not a local search algorithm although it may increase the crawling time. During crawling, our decay concept is applied to each page. Each page which does not comply with predefined threshold would cause to stoppage of the crawling.

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

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:

\[
\text{Target - Recall} = \frac{|P_c \cap P_t|}{|P_t|} \quad (7)
\]

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;

\[
F = \frac{2 \times \text{Average Precision} \times \text{Target Recall}}{\text{Average Precision} + \text{Target Recall}} \quad (8)
\]

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 simple PageRank crawlers. For 10%, we set 7% for validation.

We have shown that \(F\) of both 30% and 10% datasets in Table II. In our crawler, for two thresholds, we set \(T_d=0.25\) and \(T_s=0.5\). And for priority value, we set constants \(k_1=0.3\), \(k_2=0.2\) and \(k_3=0.15\).

<table>
<thead>
<tr>
<th>Class</th>
<th>Proposed Crawler</th>
<th>Simple SVM Crawler</th>
<th>Simple kNN Crawler</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>71%</td>
<td>56%</td>
<td>63%</td>
</tr>
<tr>
<td>10%</td>
<td>65%</td>
<td>54%</td>
<td>59%</td>
</tr>
</tbody>
</table>

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 ordering 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

http://dx.doi.org/10.15242/IIE.E0314109

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 University 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.