Improving Relevance Prediction for Focused Web Crawlers

Mejdl S. Safran[1,2], Abdullah Althagafi[1] and Dunren Che[1]
Department of Computer Science
[1]Southern Illinois University Carbondale
Illinois 62901, USA
[2]King Saud University
{mejdl.safran, althagafi.02, dche}@siu.edu

Abstract - *A key issue in designing a focused Web crawler is how to determine whether an unvisited URL is relevant to the search topic. Effective relevance prediction can help avoid downloading and visiting many irrelevant pages. In this paper, we propose a new learning-based approach to improve relevance prediction in focused Web crawlers. For this study, we chose Naïve Bayesian as the base prediction model, which however can be easily switched to a different prediction model. Experimental result shows that our approach is valid and more efficient than related approaches.*

Keywords: *Focused crawler, web mining, relevance prediction.*

I. INTRODUCTION

Typing “New York stock exchange” as keywords into Google search engine would lead to around 25 million results with quotation marks and 237 million results without quotation marks. With the same keywords, Yahoo search engine leads to around 8 million results with quotation marks and 139 million results without quotation marks, while MSN search engine leads to around 8 million results with quotation marks and around 137 million results without quotation marks. These huge numbers of results are brought to the user, but most of them are barely relevant or uninteresting to the users.

This scenario indicates a key issue – the relevance issue of a webpage to a specific topic. Behind the scene, these popular search engines depend on their indexing databases that rely on running various web crawlers collecting information. The key point with a focused crawler is how to predict the relevancy of a new, unvisited URL to the predefined search topic without downloading the target page’s content. We are thus motivated to apply data mining techniques to improve the relevance prediction and thus the performance of focused crawlers.

The Internet is the most popular and the largest information source over the world. It is estimated that there are more than 206,741,990 web sites, and 23.2 billion unique web pages [1]. Given the current scale of the Web, it is practically prohibitive for a general web crawler to go over all the web pages as that will consume too much time, space, and network bandwidth. Focused crawling therefore has gained significant attention as a research topic of Web mining [2]. A focused crawler selectively seeks out and downloads only the web pages that are relevant to a specific topic. In order to find web pages relevant to a specific topic, a focused crawler identifies and follows links that lead to relevant pages, and ignore links that lead to irrelevant pages.

In this paper, we use “Stock Market” as a sample topic, and extend the learning-based relevance prediction model proposed in [3] from two relevance attributes to four relevance attributes. In [3], for an unvisited URL, only two relevance attributes, i.e., the URL words and the anchor text, are used to build a learning-based focused crawler. We extend to four relevance attributes for each URL and get able to increase the accuracy of relevance prediction for unvisited URLs. The two new relevance attributes we added are regarding the parent pages and the surrounding text of an URL. Meanwhile, we found that using only the URL words and the words of the anchor text are not enough to get an accurate prediction; so we used *WordNet* (a free online lexical database) to find and add new related keywords to further improve prediction accuracy. We implemented our approach in a prototype crawler and compared the performance of our crawler with two other related crawlers. The results show that our approach is valid and superior in terms of performance due to improved prediction accuracy on unvisited URLs.

The rest of the paper is organized as follows. Section II reviews related work. Section III describes our approach to relevance prediction that is the key to the performance of a focused crawler. Section IV shows our experimental result, and finally Section V concludes our work.

II. RELATED WORK

The Fish-Search [4] is an example of early crawlers that prioritizes unvisited URLs on a queue for a specific search goal. The Fish-Search approach assigns priority values (1 or 0) to candidate pages using simple keyword matching. One of the disadvantages of Fish-Search is that all relevant pages are assigned the same priority value 1 based on keyword matching. The Shark-Search [5] is a modified version of Fish-Search, in which, Vector Space Model (VSM) is used, and the priority values (more than just 1 and 0) are computed based on the priority values of parent pages, page content, and anchor text.

InfoSpiders and Best-First are additional examples of focused crawling methods [6]. The difference between them is that InfoSpiders uses Neural Networks, while Best-First method applies VSM to compute the relevance between candidate pages and the search topic. Shark-Search crawlers may be considered as a type of Best-First crawlers, but the former has a more complicated function for computing the priority values. In [6], Best-First was shown most successful due to its simplicity and efficiency. *N*-Best-First is generalized

978-0-7695-4694-0/12 $26.00 © 2012 IEEE
DOI 10.1109/ICIS.2012.61

from Best-First, in which *N* best pages are chosen instead of one.

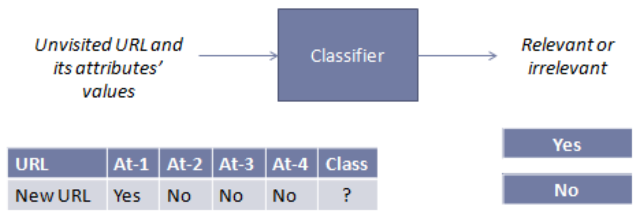
A relatively more recent type of focused crawlers adopts learning-based approaches to relevance prediction. This approach first trains a classifier using a training dataset and then applies the classifier to unvisited URLs. Our crawler is a learning-based one with an enhanced relevance prediction model.

Our work is most related to [7] and [3]. In [7], a classical focused crawler was proposed. Two years later, several improvements in terms of speed and prediction accuracy were made in [3] and resulted in a learning-based crawler. In [7], out-links carry relevance scores from the originating/parent pages. However, irrelevant pages are not ignored immediately, but exploited for discovering new relevant pages led from these irrelevant pages. This mechanism will be discussed and used in our implementation. In [3], an optimization method was introduced that uses two attributes – URL words and anchor text of visited URLs – to support a learning-based focused crawler for predicting the relevance of unvisited URLs. Naïve Bayesian classifier had been used as the learning model [3].

In our work, we added two more relevance attributes, i.e., parent pages and surrounding texts. In addition, we extended the keywords set of a search topic by using *WordNet* (a free online lexical database at http://wordnet.princeton.edu/).

III. Proposed Approach

In this section, we address two major issues. First, we discuss the training set preparation, which contains values of the aforementioned four relevance attributes: URL words relevancy, anchor text relevancy, parent pages relevancy, and surrounding text relevancy, besides the class label for each seed page (relevant=Yes or irrelevant=No). Second, we train the Naïve Bayesian classifier (shortly as NB) using the training set, and then apply the trained classifier to predict the relevancy of unvisited URLs with regard to the crawling topic – "Stock Market" in this paper. Figure 1 shows the main components in our trained classifier.



URL	At-1	At-2	At-3	At-4	Class
New URL	Yes	No	No	No	?

Figure 1: Classifier's inputs and outputs

1) *Training Set Preparation*

A. *Relevant Seed URLs Generation*

Figure 2 shows the process of generating relevant seed URLs. First, the topic words, Stock Market, are sent as query to Google, Yahoo, and MSN search engines. Then, the top *k*-URLs resulted from each search engine are selected as candidate seed URLs. URLs that appear in the result list of at least two of the three search engines are then considered as *relevant* seed URLs (e.g., the first six URLs in Table 2 are *relevant* seed URLs).

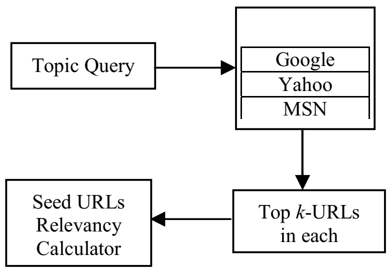


Figure 2: Relevant Seed URLs generator

B. *Irrelevant Seed URLs Generation*

To train the Naïve Bayesian classifier, the training set shall contain irrelevant seed URLs as well. We use the same process to generate relevant seed URLs. This time, "-Stock Market" is sent as the query (note the minus sign in front of "Stock" is potentially added). The last two URLs in Table 2 are examples of *irrelevant* seed URLs.

C. *Creation of Topic Keywords Weights Table*

We need a set of weighted keywords to more adequately represent our search topic (the words extracted from the query string is not enough). This set of keywords is called topic keywords, and will be stored in a table named Topic Keywords Weights Table as in [3]. To find the topic keywords and their weights, the following steps are applied:

TABLE 1: Topic Keywords Weights Table

No.	Keyword	Weight
1	Stock	1
2	Market	0.780120
3	Quote	0.427710
4	Finance	0.355421
5	Trade	0.322289
6	Invest	0.268072
8	Exchange	0.243975
9	Price	0.225903
10	Business	0.135542
11	Bank	0.132530
12	Chart	0.132530
13	Economy	0.129518
14	Money	0.126506

1) All the seed pages that are stored in the training set are downloaded and parsed.

2) After eliminating all stop words such as "the" and "is", Porter's stemming algorithm [9] is called to get all the remaining terms.

3) The tf×idf weighting method proposed by Salton and Buckley [10] is then used to give each term a weight. In this method, tf measures the frequency of term "t" in a page, and idf varies inversely with the number of pages that contain "t". Matematically, idf is defined by the following equation: idf = log (N/n) where N is the total number of documents and n is the number of documents that contain "t".

4) The top 14 keywords that have the highest weights are selected as the topic keywords (more keywords may be selected, but we found 14 is enough in our experiments).

5) The 14 weighted keywords are normalized using the following equation (1), where W_i is the weight of keyword i, and W_{max} is the highest weight among all the keywords.

$$Weight = W_i \,/\, W_{max} \qquad (1)$$

6) Finally, the 14 selected weighted keywords are stored in the Topic Keywords Weights Table as shown in Table 1.

D. *Calculation of Seed Pages' Relevance Attributes*

This subsection describes how to calculate values for the 4 relevance attributes in our training dataset.

1) *URL Words Relevancy*

URL words are the words that comprise the link. To find the relevancy of the URL words to the search topic, the following steps are applied:

a) Each URL is tokenized using '/', '&', '?' and other appropriate delimiters.

b) The obtained word set is then extended using WordNet.

c) The URL Words Weights Table is created in 2 steps:

i. Copy all topic keywords into the table.

ii. Then for each word, if it is also an URL word, assign it the same weight as that in the Topic Keywords Weights Table; otherwise assign 0. Figure 3 illustrates the procedure.

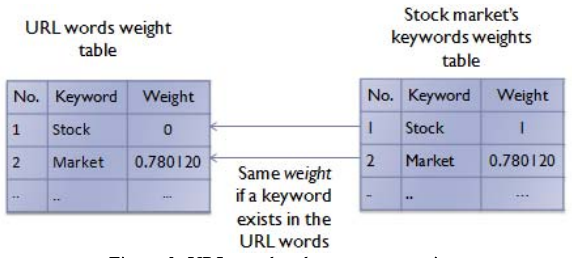
d) The Topic Keywords Weights Table and the URL Words Weights Table are then sent as inputs to the cosine similarity function, as in (2), to compute the relevance measure of the URL words relevance attribute, and the computed result will be filled into the corresponding column in Table 2.

The summation in Equation (2) ranges over 14 words (as a parameter, 14 can be changed), where $wk_{topic\_table_i}$ is the weight of the i^{th} keyword in the Topic Keywords Weights Table and $wk_{attribute\_table_i}$ is the weight of the same keyword in the URL Words Weights Table.

$$Similarity\,(topic_{table}, attribute_{table}) = \qquad (2)$$

$$\frac{\sum_{i=1}^{14}(wk_{topic\_table_i} \times wk_{attribute\_table_i})}{\sqrt{\sum_{i=1}^{14}(wk_{topic\_table_i})^2} \times \sqrt{\sum_{i=1}^{14}(wk_{attribute\_table_i})^2}}$$

In our experiments, we applied a relatively small threshold, 0.30, to the URL words relevancy, above that, the attribute is interpreted as relevant, otherwise irrelevant. We found that URL words often do not provide enough information about our search topic, and 0.30 has been identified as reasonable threshold value for our sample search topic – "Stock Market".



URL words weight table
Stock market's keywords weights table
Same weight if a keyword exists in the URL words

No.	Keyword	Weight
1	Stock	0
2	Market	0.780120
..	..	...

No.	Keyword	Weight
1	Stock	1
2	Market	0.780120
-	..	...

Figure 3: URL words relevancy computing

2) *Anchor Text Relevancy*

The anchor text of an URL is the visible and clickable text of hyperlink. From the link itself, we cannot know the anchor text, so we need to find the parent pages that hold references to the target URL in order to get its anchor text. We used the facility at http://www.backlinkwatch.com that takes an URL as input and produces a set of back-links pointing to the parent pages. To compute the Anchor text relevancy, the following steps are applied:

a) The anchor texts of each seed URL are extracted from the first 5 parent pages, from which the keywords are further extracted.

b) WordNet is used to extend the extracted words set.

d) An Anchor Text Keywords Weights Table for each URL is created using basically the same approach as for the URL Words Weights Tables.

c) The Anchor Text Keywords Weights Table and the Topic Keywords Weights Table are then sent to the cosine similarity function, as in (2), for computing the anchor text relevancy of the seed URL.

The third column (Anchor text relevancy) in Table 2 shows the results computed as the anchor text relevancy for each selected seed URL in our sample training dataset. With the anchor relevance attribute, we applied a relatively large threshold, 0.75, above which, the attribute is interpreted as relevant, otherwise irrelevant. We found that anchor texts usually provide adequate information about the search topic; through experiments, we found 0.75 as a reasonable threshold with regard to the relevancy of anchor texts to a search topic.

3) *Parent Pages Relevancy*

We now discuss the computing of the relevancy measures of the parent pages of the seed URLs in the training set. The way for computing this relevancy attribute is slightly different from that for the URL words relevancy and the anchor text relevancy. The following procedure is applied for this purpose:

a) The contents of the first 5 parent pages of each seed URL are extracted.

b) The top 14 weighted keywords in the parent pages are extracted in the same way as for the Topic Keywords Weights Table, resulting in a table called Parent Pages Keywords Weights Table.

c) An adjusted version of this table, named the Adjusted Parent Pages Keywords Weights Table, is created using the Topic Keywords Weights Table and the Parent Pages Keywords Weights Table as in the following: first, copy all keywords from the Topic Keywords Table; second, for the words that also appear in the Parent Pages Keywords Weights Table, copy their weights from there; lastly, for the remaining words, assign 0 as their weights. Figure 4 illustrates this step.

d) Send both the Adjusted Parent Pages Keywords Weights Table and the Topic Keywords Weights Table as inputs to the cosine similarity function, as in (2), to decide the parent pages relevancy.

With the parent pages relevance attribute, we applied a threshold 0.60, which is experimentally decided. Above this value, the attribute is interpreted as Yes, otherwise No.

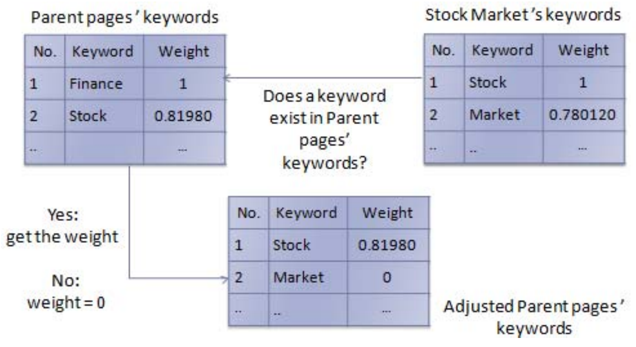


Figure 4: Deciding the Parent Pages Relevancy

4) *Surrounding Text Relevancy*

The surrounding text of an URL is the text before and after the URL in the parent pages. In our implementation, we simply take the 20 non-stop words before and after the anchor text. To compute the surrounding text relevancy for each seed URL, the following procedure is applied (which is similar to the above procedure that computes the parent pages relevance values):

a) The surrounding text of each seed URL is extracted from its first 5 parent pages.

b) The top 14 weighted keywords in the surrounding text are selected in the same way as for selecting the top 14 topic keywords of the search topic, resulting in a table called the Surrounding Text Keywords Weights Table.

c) A new version of the table, called the Adjusted Surrounding Text Keywords Weights Table, is then produced from the Topic Keywords Table and the Surrounding Text Keywords Weights Table in the same way as producing the Adjusted Parent Pages Keywords Weights Table.

d) Send the Adjusted Surrounding Text Keywords Weights Table and the Topic Keywords Weights Table as inputs to the cosine similarity function, as in (2), to compute the surrounding text relevancy.

With the surrounding text relevance attribute, we applied the same threshold, 0.60, as for the parent pages relevance attribute. Above that value, the attribute is interpreted as Yes, otherwise No.

Our training set has now been fully built as shown in Table 2. Our Naïve Bayesian classifier is then trained on this training set and used to predict the relevancy of unvisited URLs. This task is elaborated next.

TABLE 2: A sample of the final training set

Seed URLs	URL words relevancy	Anchor text relevancy	Parent pages relevancy	Surround text relevancy	Class
http://www.nasdaq.com	0.25	0.97	0.64	0.76	Yes
http://www.nyse.com	0.66	0.67	0.69	0.65	Yes
http://finance.yahoo.com	0.32	0.79	0.60	0.32	Yes
http://www.marketwatch.com	0.0	0.82	0.67	0.46	Yes
http://money.cnn.com/data/markets	0.54	0.89	0.67	0.29	Yes
http://abcnews.go.com/business	0.12	0.85	0.29	0.69	Yes
http://marketbarchicago.com	0.0	0.50	0.20	0.60	No
http://www.worldmarket.com	0.51	0.50	0.16	0.18	No

2) *Training and Prediction*

This section discusses how Naïve Bayesian classifier is trained to predict the relevancy of unvisited URLs and proposes a new focused crawler algorithm.

A. Naïve Bayesian Classifier

Our prediction approach is based on the Naïve Bayesian classification method described in [8]. The main method that we followed to calculate the probability of a given unvisited URL as relevant or irrelevant is reflected in Inequality (3) (see below). The left side of the inequality can be considered as the probability of X being relevant, and the right side as the probability of X being irrelevant, where X is the unvisited URL. Therefore, if the inequality holds, X is relevant, otherwise X is irrelevant. The way of computing the terms in (3) is explained by equations (4) and (5).

$$P(X \mid Relevant = Yes) * P(Relevant = Yes) > P(X \mid Relevant = No) * P(Relevant = No) \quad (3)$$

$$P(Relevant = Yes) = \#\, of\, relevant/total \quad (4)$$

$$P(X \mid Relevant = Yes) = P(URL\, words\, relevancy \mid Relevant = Yes) *$$

$$P\ (Anchor\ text\ relevancy \mid Relevant\ = Yes) *$$
$$P\ (Parent\ pages\ relevancy \mid Relevant\ = Yes) *$$
$$P\ (Surrounding\ text\ relevancy \mid Relevant\ = Yes)$$
$$P\ (Relevant\ =\ No)\ =\ \#\ of\ irrelevant/total \qquad (5)$$
$$P(X \mid Relevant\ =\ No)\ =$$
$$P\ (URL\ words\ relevancy \mid Relevant\ = No) *$$
$$P\ (Anchor\ text\ relevancy \mid Relevant\ = No) *$$
$$P\ (Parent\ pages\ relevancy \mid Relevant\ = No) *$$
$$P\ (Surrounding\ text\ relevancy \mid Relevant\ = No)$$

We train our classifier in two different ways. First, we use a fixed training set – i.e., the training set never change once built. Second, we use an updatable training set that is constantly fed with new relevant and irrelevant URLs after built and during crawling process.

B. *Proposed Focused Crawler Algorithm*

Our crawling algorithm (predicting the relevance of unvisited URLs) is shown in Figure 5. In our implementation, relevant URLs and irrelevant URLs are maintained in separate tables: the *Relevant Table* keeps the identified relevant URLs and the *Irrelevant Table* keeps the identified irrelevant URLs. The first step in our algorithm puts all out-links from seed pages in a *Queue*. Each URL taken from the *Queue* is sent to a function that computes its relevance attribute values (i.e., URL words relevancy, anchor text relevancy, parent pages relevancy, and surrounding text relevancy). Then, NB classifier takes the URL with its attributes values as inputs and makes prediction of its relevancy to the search topic. Based on the predication made, the algorithm differentiates the following two cases accordingly.

```
Algorithm: Predicting the relevancy of unvisited URLs
1: Insert relevant seeds to ReleventTable (RT) and irrelevant seeds to
IrrelevantTable (IRT)
2: Download seed pages and insert all out-links from seed pages in the Queue
3: While (Queue is not empty) do
4:        url = dequeue ()
5:        If (url in RT or IR) go to 3
6:        RelevancyResults = computeRelevancyOfAttribute (url)
7:        prediction = NB (RelevancyResults)
8:        If (prediction is Relevant)
9:        then      url.levelcounter = 0
10:                 insert-RT (url)
11:                 download (url)
12:                 enqueue (out-links from url)
13:       else
14:                 url.levelcounter = url.levelcounter + 1
15:                 insert-IRT (url)
16:                 If (url.levelcounter < Level_Limit)
17:                 then      download (url)
18:                           enqueue (out-links from url)
19: Endwhile
```

Figure 5: Focused Crawler Algorithm

The first case: the URL is predicted as relevant. In this case, the URL is inserted into the *Relevant Table*, and the page is downloaded and all out-links from it are added to *Queue* that keeps a list of URLs waiting to be crawled. At the same time, the *level counter* is reset to 0 in this case. The level counter variable is increased by one if the crawler moves from an irrelevant page to another irrelevant page; and it is reset to 0 upon entering a relevant page.

The second case: the URL is predicted as irrelevant. In this case, the URL is inserted into the *Irrelevant Table*, and the page is downloaded (and all out-links are extracted into *Queue*) if the level counter is currently less than the *level limit* (another control variable to be explained shortly), otherwise the URL is ignored.

Figure 6 illustrates the process embodied in the algorithm for crawling and prediction.

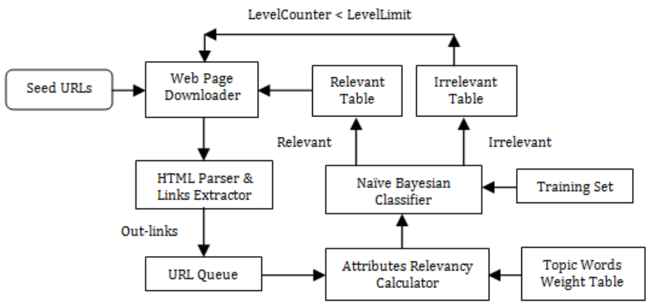


Figure 6: The process of crawling and prediction

C. *Discussion of A Special Case of Crawling*

A special case that may occur during crawling regards the processing of irrelevant pages. The following questions need be answered when encountering an irrelevant page: Should we go further to check the children of an irrelevant page as the children might be relevant? If we go further to the children of an irrelevant page, when should we stop? Our answers to these questions form our policy and are implemented via the use of the two crawling control variables: *level counter* and *level limit* (see Figure 5). Similar discussing was first made in [7]. In our algorithm, the *level limit* is set to 3, which means that any path in the crawling tree that has three consecutive irrelevant pages must be given up.

IV. Experimental Results

We implemented two variants of our learning-based focused crawler: one using a fixed training set and another allowing dynamic update of its training set. For comparison purpose, we also implemented two other crawlers based on related work: a general crawler using Breadth First Search (BFS) and a focused crawler based on [3] using only two relevance attributes (URL words relevancy and anchor text relevancy). All four crawlers are implemented in Java, and summarized as follows:

a) A Breadth First Search crawler (named **BFS**) that simply downloads every page (not learning-based).

b) A focused crawler (named **NB2**) based on [3] that inspects only two relevance attributes (i.e., the URL words relevancy and the anchor text relevancy).

c) The first variant of our proposed focused crawler using a fixed training set (named **NB4**).

d) The second variant of our proposed focused crawler using a dynamically updatable training set (named **NB4 Updatable**). Its updatable training set allows dynamic update/replacement of URLs and their associated attributes during the crawling process.

Our experiment compares the relevance of the visited URLs to the sample search topic among these four crawlers. We adopted the precision metric defined by Equation 6 (see below) with regard to relevance prediction (notice that BFS does not make prediction at all).

$$Precision\ rate = \frac{\#\ of\ relevant\ pages}{\#\ of\ downloaded\ pages} \quad (6)$$

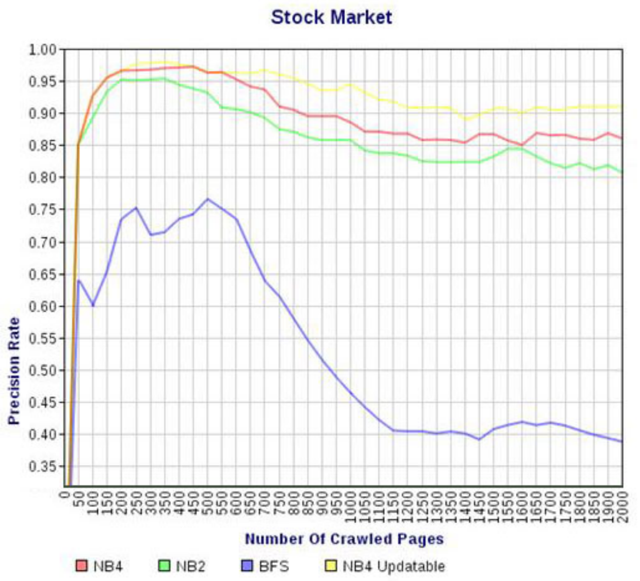


Figure 7: Experimental results

Figure 7 shows our experimental results. All four crawlers are given a set of 2000 potential web pages to crawl, starting with the same set of relevant seed pages. After 50 pages were crawled, the precision rate for the four crawlers is 0.64 or higher. As more pages being crawled, our NB4 and NB4 Updatable reached a precision rate of 0.85 or higher. BFS, after crawling 500 pages, showed a precision rate of about 0.76, and as the number of crawled pages further increases, the precision rate of BFS further decreased and eventually dropped to around 0.4. We see our NB4 and NB4 Updatable consistently outperform NB2, and NB4 Updatable consistently outperforms NB4. The reason is well understood – NB4 Updatable dynamically updates its training set that can better characterize the search topic.

V. Conclusion and Future Work

A focused crawler selectively seeks out and downloads web pages that are relevant to the search topic given to the crawler. A learning-based focused crawler has learning ability to adapt to its search topic and to improve the accuracy of its prediction of relevancy of unvisited URLs. In this paper, we presented a new learning-based focused crawling approach that uses four relevance attributes to predict the relevance of unvisited URLs. The four attributes are the URL words, its anchor text, the parent pages, and the surrounding text. Our approach adopted Naïve Bayesian classification model, which can be extended to other more sophisticated models. Performed experiment shows that our approach is valid, and the accuracy of relevance prediction is obviously better than related crawling methods, and if additionally allowing dynamic update of the training dataset, the prediction accuracy is further boosted.

As future work, we plan to do more extensive tests with larger volumes of web pages, to incorporate and test with other classification models such as Support Vector Machine (SVM) and Neural Network.

References

[1] G. V. Joshi, "20 Years of World Wide Web." Legal Era Magazine, pp. 55- 57, 2011.

[2] S. Chakrabarti, M. Berg, and B. Dom, "Focused Crawling: A New Approach for Topic Specific Resource Discovery", In Journal of Computer and Information Science, vol. 31, no. 11-16, pp. 1623-1640, 1999.

[3] D. Taylan, M. Poyraz, S. Akyokus, and M. C. Ganiz, "Intelligent Focused Crawler: Learning Which Links to Crawl", In Proc. Intelligent Systems and Applications Conference (INISTA) , pp. 504-508, 2011.

[4] P. De Bra, G-J Houben, Y. Kornatzky, and R. Post, "Information Retrieval in Distributed Hypertexts", In Proc. of RIAO-94, Intelligent Multimedia, Information Retrieval Systems and Management, pp. 481-492, 1994.

[5] H. Michael, J. Michal, M. Yoelle, P. Dan, S. Menachem, and U. Sigalit, "The Shark-Search Algorithm - An Application: Tailored Web SiteMapping", In Computer Networks and ISDN Systems, vol. 30, no 1-7, pp. 317-326, 1998.

[6] F. Menczer, G. Pant, and P. Srinivasan, "Topical Web Crawlers: Evaluating Adaptive Algorithms", ACM Transactions on Internet Technology (TOIT), vol. 4, no. 4, pp. 378–419, 2004.

[7] A. Pal, D. S. Tomar, and S.C. Shrivastava, "Effective Focused Crawling Based on Content and Link Structure Analysis", In International Journal of Computer Science and Information Security (IJCSIS), vol. 2, no. 1, 2009.

[8] H. Jiawei, K. Micheline, and P. Jian, "Data Mining: Concepts and Techniques", 2nd ed. San Francisco: Morgan Kaufman, 2006.

[9] M. F. Porter, "An Algorithm for Suffix Stripping", Program: Electronic Library and Information Systems, vol. 40, no. 3, pp. 211-218, 2006.

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