Intelligent Focused Crawler: Learning which Links to Crawl

Duygu Taylan, Mitat Poyraz, Selim Akyokuş and Murat Can Ganiz
Department of Computer Engineering
Doğuş University
Istanbul, Turkey
{dtaylan, mpoyraz, sakyokus, mcganiz }@dogus.edu.tr

Abstract— **A web crawler is defined as an automated program that methodically scans through Internet pages and downloads any page that can be reached via links. With the exponential growth of the Web, fetching information about a special-topic is gaining importance. A focused crawler is a web crawler that attempts to download only web pages that are relevant to a pre-defined topic or set of topics. In order to determine a web page is about a particular topic, focused crawlers use classification techniques. In this study we focus on the classification of links instead of downloaded web pages to determine relevancy. We combine a Naïve Bayes classifier for classification of URLs with a simple URL scoring optimization to improve the system performance. Our results demonstrate that proposed approach performs better.**

Keywords- focused crawler; link classification; Turkish web pages; URL optimization; naive bayes; machine learning

I. Introduction

A web crawler is defined as an automated program that methodically scans through internet pages and downloads any page that can be reached via links [1]. With the exponential growth of the web, fetching information about a special topic is gaining importance. A focused web crawler aims to download only web pages that are relevant to a pre-defined topic. In order to determine a web page is about a particular topic, focused crawlers use *page scoring* and *link scoring* techniques. Page scoring determines whether a page is relevant or irrelevant. Link scoring determines the links to be crawled, and prioritize the order of the links to be crawled. Classification algorithms are generally used in page scoring. In general, they first download a page and then use the classifier to decide whether the page is about the topic [2].

In this study, we add several improvements to an existing study done in [3] to increase the performance of the system both in terms of accuracy and in terms of speed. First, we use a simple URL optimization method. As a result, we crawl few amount of but more accurate links. Consequently system performance increases. The URL optimization method analyses link context, and tokenizes the terms that appears in both anchor text and in URL. The information obtained from link context is used for link scoring. For link scoring, we use several different methods. These methods determine links to be crawled. Our results demonstrate that the link scoring method based on the Naïve Bayes (NB) classifier increases accuracy of the system considerably. Based on these results, we modified NB link classifier to incrementally update its training set during crawling to improve the system performance further. We also apply a simple stemming algorithm that is developed for Turkish language [4] and Turkish stop word list to improve efficiency of our system.

The rest of the paper is organized as follows. The related work and background are covered in section II. We then present our approach for focused web crawling based on URL optimization in section III. We follow this by summarizing our experimental results in section IV. Section V includes a conclusion and discusses the future work.

II. Related Work and Background

A variety of methods and algorithms is proposed for building focused crawlers. A comparison of learning schemas that is employed by focused crawlers can be found in [5]. There are several works, which employs link-ranking mechanism [6,7,3] to help the classification of web pages. In [8] they described a focused crawler, which searches the web and finds relevant pages on a given topic. They used a classifier to determine the relevancy of a page and a distiller to evaluate page links. The classifier uses Bayesian classifier to determine the relevancy of a page to a pre-defined topic. Fish search [9] is one of the early works in focused crawling. In this algorithm, each URL simply behaves like a fish and its chance to survive depends on that page's relevancy. Shark-search algorithm [10] improves fish search algorithm by making contributions in calculating relevancy of a page and the topic. In most of these systems, vector space model is used to represent information. Recently the researches are focused on the area of the relationship between pages and the outgoing links in those pages. The link structure analysis [3], the page rank algorithm [11] and the hits algorithm [12] are among the algorithms that use this kind of relationships. Several machine learning algorithms [13,14] are used in focused crawlers. Naïve Bayes (NB) is one of the most popular classification algorithm that is used to decide a page's relevancy to the given topic [15].

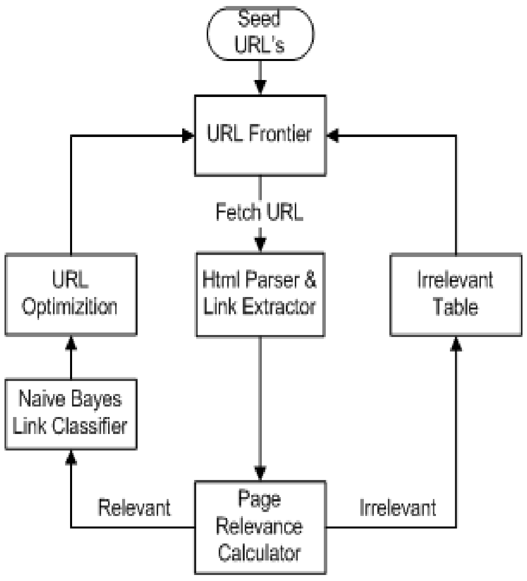

Figure 1. System architecture

III. Approach

Figure 1 shows the general architecture of the focused crawler system. The focused crawler starts with a set of seed URLs and topic words. The topic words are recorded in topic words weight table. First, a query is submitted to a well-known web search engine such as Google in order to find related first n pages. Words that appear in these pages are included in the topic words weight table. The weights are calculated by using the frequency of occurrence of terms in pages. Furthermore, the URLs of these first n pages also form the initial set of seed URLs that is added to URL frontier.

The URL frontier component is a queue that stores links waiting to be crawled. The system works until URL frontier queue is empty. The URL frontier retrieves a page if a visited site allows visiting that is specified in robot.txt file. If a site includes a sitemap.xml file that lists the links for that site, the crawler retrieves this sitemap file in addition to the requested page. Links specified in sitemap file are parsed using an XML parser and sent to the frontier's URL queue. Each retrieved page is parsed with an HTML parser and its content is tokenized into terms. We apply preprocessing methods including stemming and stop word filtering. The page is represented in the vector space model using term frequencies.

The page relevance calculator calculates a page score by computing the similarity between the retrieved page and the topic words weight table. The pages with scores above a specific threshold are accepted as similar. If the retrieved page is similar to the query terms given in the topic words weight table, then it is called relevant, otherwise irrelevant.

After the page relevance calculation, the links included in each page are analyzed. Then, the crawler determines the links to be added to the URL frontier queue for further crawling. At this phase, we use two different link-scoring methods depending on page relevancy. If the page irrelevant, its link scores are calculated as explained in section C. If the page is relevant, we use URL optimization for link scoring, as it is explained in section D.

The following sections explain the construction of topic words weight table, relevance calculations, link scoring for irrelevant pages, Link scoring with URL optimization and the crawling process.

A. Topic Words Weight Table

Topic words weight table keeps the weight of topic words. To create table, the topic word or sentence sent as a query to a search engine (e.g. Google) and first n results are retrieved before crawling [16]. Turkish stop words such as "ama", "ve"," ile"," o", are eliminated from the page text before further processing [4]. We also apply a very simple stemming algorithm called fixed prefix stemmer (FPS), which is reported to be an effective stemmer [4, 17]. According to this algorithm, the first m character of a word is taken as root. If length of a word is less than m, the word forms the root by itself.

Following this, we use standard tf x idf weighting method [18] to calculate the weight of each term. In this method, tf is the number of occurrences of word w in a document and idf varies inversely with the number of documents in the collection that w occurs in. Following calculation, words are ordered by their weight and first n words are selected as topic keywords. Then, we normalize these weights using equation (1)

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

B. Relevance Calculations

We calculate the weight of words in a page using the topic words table. We assume that the weight of a word can change depending on the location of a word in a page. For example, the title text or a meta text is more important and descriptive than the body text, thus meta text should have a higher weight. The weight of topic words in different positions calculated as follows [7]:

$$wk = \begin{cases} C_t \; fk & \textit{Title-Meta Text} \\ C_b \; fk & \textit{Body Text} \end{cases}$$

In the equation above, C_t and C_b are meta and body text coefficients respectively. We assume C_t=2 and C_b= 1. Wk is the weight of a topic word k and fk is the frequency of word k in the related text. This weight variable is used to calculate the page relevance with cosine similarity measure as follows:

$$\text{Similarity (t,p)} = \frac{\sum_{i=1}^{n} W_{t_i} * W_{p_i}}{\sqrt{\sum_{i=1}^{n} (W_{t_i})^2 + \sum_{k=1}^{n} (W_{p_i})^2}} \qquad (3)$$

In the equation (3) above t is the topic words weight table, p is the web page fetched. W_{t_i} is the weight of words in the topic words weight table, whereas W_{p_i} is the weight words in the page. The relevance score obtained from the equation is between 0 and 1.

C. Link Scoring for Irrelevant Pages

A typical focused crawler only crawl links from a page if it is relevant; otherwise, links from that page are skipped. Therefore, we cannot reach a relevant page if it can only be accessed through following irrelevant parent pages. To avoid this we employ link scoring algorithm that is proposed in [3]. In this method, the crawler also checks the links included in irrelevant pages and it keeps on to crawl through them until a maximum tree level specified by user. Then, the crawler calculates link scores using the equation (4), and decides whether to fetch pages specified by links or not. Thus, crawler spends less time on irrelevant pages and does not miss relevant pages, which are child of an irrelevant page. The link score is calculated as follows[6]:

$$\begin{aligned}\mathrm{LinkScore}\ (\mathrm{u}) = {} & \mathrm{URLTextScore}\ (\mathrm{u}) + \mathrm{AnchorTextScore}\ (\mathrm{u}) \\ & + \mathrm{NumberOfParentPages}\ (\mathrm{u}) \\ & + [\mathrm{Relevance}\ (\mathrm{p}_1) + \mathrm{Relevance}\ (\mathrm{p}_2) + \cdots \\ & + \mathrm{Relevance}\ (\mathrm{p}_\mathrm{n})] \qquad (4)\end{aligned}$$

In the equation (4), LinkScore (u) is the score of link u, URLTextScore(u) is the relevance between the HREF information of u and the topic words, and AnchorTextScore(u) is the relevance between the anchor text of u and the topic words. NumberOfParentPages (u) is the number of links from relevant crawled pages to link u, p_i is the i 'th parent page of link u. The parent page is the page from which a link was extracted.

D. Link Scoring for Relevant Pages

In a crawling process, the effectiveness of the focused crawler does not just only rely on the maximum amount of relevant pages to be fetched but it also depend on the speed of the crawling process. The speed of a crawler depend the number of relevant URLs inserted into the Frontier's URL queue. Therefore, a mechanism, that we call URL optimization, is required for frontier to select links; those are more likely to be relevant. Most of the crawlers as in [3] directly insert the links into the Frontier's URL queue without URL optimization. The URL optimization enables the removal of certain links whose link scores are below a predefined threshold point.

For URL optimization, we used two different link score evaluation methods for relevant pages. The first method simply uses the link score calculation equation given in (4). The second method uses a Naïve Bayes (NB) classifier to compute link scores.

Naïve Bayes (NB) is one of the most widespread algorithms in text classification. There are two commonly used event models used with NB in text classification: Multi-variate Bernoulli and multinomial models [19]. In the first model, a document is described as a vector of 1's and 0's representing existence of a word. On the other hand, in multinomial model, the vector consists of word frequencies.

A document described by words and their frequencies. N_{an} is the frequency of word W_n in document d_a. Then the multinomial distribution would be [19]:

$$\mathrm{P}(d_a|\mathrm{c}_b;\theta) = \mathrm{P}(|d_a|)|d_a|!\ \prod_{n=1}^{|V|}\frac{\mathrm{P}(W_n|\mathrm{c}_b;\theta)^{N_{an}}}{N_{an}!} \qquad (5)$$

Hence the approximation of the possibility of word W_n in class $\mathrm{c_b}$would be:

$$\theta_{W_a|c_b} = \mathrm{P}(W_a|\mathrm{c}_b;\theta_b) = \frac{1+\sum_{n=1}^{|D|}N_{na}\,\mathrm{P}(c_b|\mathrm{d}_n)}{|V|+\sum_{k=1}^{|V|}\sum_{n=1}^{|D|}N_{nk}\mathrm{P}(c_b|\mathrm{d}_n)} \qquad (6)$$

To classify a new document the following formula is used:

$$\mathrm{P}(c_a|d_b;\theta) = \frac{\mathrm{P}(\mathrm{c}_a|\theta)\,\mathrm{P}(d_b|\mathrm{c}_a;\theta_a)}{\mathrm{P}(\mathrm{d}_b|\theta)} \qquad (7)$$

The NB classifier uses terms (words) that appear in both anchor text and in URL as feature vectors. Anchor texts of links and links themselves are tokenized using regular expressions and those tokenized words are used as features in NB. For instance, characters such as "_,.,-,/,+,?,=" are eliminated and the link is spitted into words.

To train Naïve Bayes, we use two different training methods. In the first method, NB uses a fixed data set that includes pre-labeled relevant and irrelevant links obtained from the pages with high pages scores initially. The second method applies incremental learning. During the crawling period, the training set is updated with new appropriate relevant and irrelevant links which are over some probability rate. The learned model is updated at certain periods (after every 50 fetched pages). . As a consequence, the more the crawler traverses the pages, the more our NB approach learns.

E. Crawling Process

A summary of crawling algorithm is given below. A set of seed URLs is given as input to the crawler.

```
1. while the list of URLs to search is not empty
2. {
3.   get the first URL in the list.
4.   move the URL to the list of URLs already searched.
5.   check the URL to make sure its protocol is http
       // if not, break out of the loop, back to 1
6.   see whether there is a robots.txt file at this site
       that includes a "disallow" statement.
       // if so, break out of the loop, back to 1
7.   try to fetch the URL
8.   if it is not an html file, break out of the loop,
                  back to 1
9.  while the html text contains another link,
10. {
11.   validate the link's URL
        // just as in the outer loop
12.   if it is an html file,
13.   {
14.     if the URL is not present
15.     {
16.       add it to the to-search list.
17.     }
18.     else if the type of the file is valid
19.     {
20.      add it to the list of files found
21.     }
22.   }
23. }
24.}
```

IV. EXPERIMENT SETUP

The focused crawler system is implemented. In order to test the system, we used the following query (topic words): "Bilgisayar Mühendisliği İş İlanları", that means "computer engineering job postings" in English, to search over Turkish web sites. The crawling system sends this query to a well-known web search engine (Google) in order to find related first 10 pages. The words that appear in these pages are included in the topic words weight table. Table I shows the topic words where weights are normalized.

TABLE I. TOPIC WORDS WEIGHT TABLE

Keyword	Weight
Bilgisayar,kariyer,mühendis	0,9
Uzman,deneyimli,üniversitelerin,sektör,başvuru	0,4
Mezun,iş	0,7
Yazılım,bilişim	0,8
c#,java	0,3
İlan	0,6

We used 10 seed URLs and crawled 1000 pages at every experiment. Our NB training set has 64 instance links at the beginning of the crawling process. In experiments, five different methods are applied and their performances are analyzed. These methods are given below:

- LSFC (Link Structure Based Focused Crawler): It uses page relevance and link scoring for irrelevant pages [3].
- Optimized LSFC: It uses link scoring method given in equation (4) for relevant pages.
- NB (Naïve Bayes): For URL optimization, it uses NB classification. The training set is given at the beginning and it is not updated.
- NB with Threshold: For URL optimization, it uses NB classification. The training set is given at the beginning and it is not updated. In addition, the links are added into URL frontier queue if their probability exceeds a predetermined threshold limit.
- NB Updateable: For URL optimization, it uses NB classification. It uses incremental training. The learned model is updated with new URLs obtained during crawling process at certain periods (after every 50 fetched pages).

We calculated the harvest rate in order to compare the results of different crawling methods. The harvest rate is the fraction of relevant pages fetched [16]. Let N be the number of pages crawled that have been crawled by any given time t, then the harvest rate at time t is defined as the fraction of crawled pages that are relevant. More precisely;

$$\text{Harvest Rate} = \frac{1}{N}\sum_{i=1}^{N} I(p_i > 0) \qquad (8)$$

In the equation (8), p_i is the relevance of the i th fetched page and I takes the value 1 if pi is greater than 0. However, in our study, the relevancy of a page lies between 0 and 1, thus harvest rate also lies between 0 and 1 as well. So strictly speaking harvest rate is defined as:

$$H.R = \frac{\text{number of relevant pages}}{\text{number of crawled pages}} \qquad (9)$$

In our experiments the pages, whose score is greater than 0.8, are considered as relevant pages.

Figure 2 compares the harvest rates of LSFC (link structure based focused crawler), optimized LSFC and updateable NB methods. As it seen from the figure, Updateable NB provides the best performance.

Figure 3 compares the harvest rates of NB, NB with threshold, NB updatable and LSFC. NB with threshold provides the best performance.

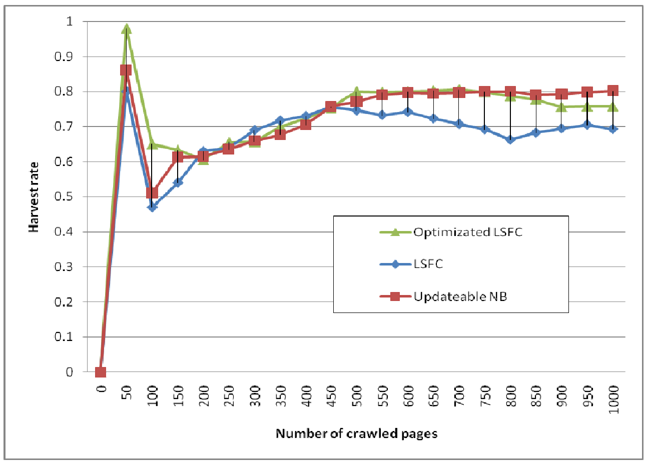

Figure 2. Example

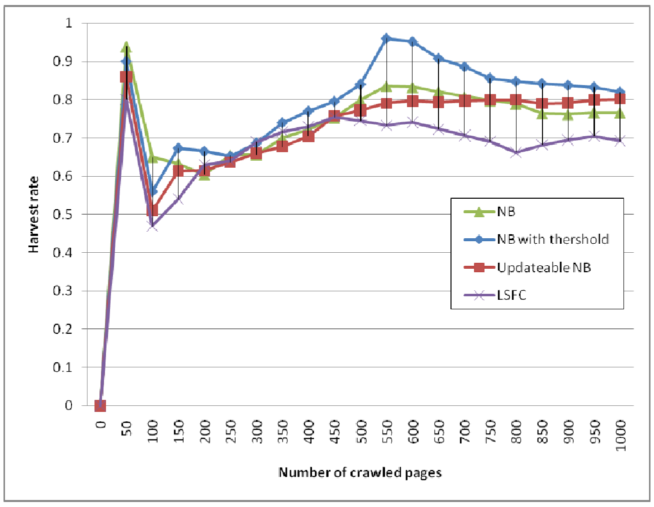

Figure 3. Example

V. Conclusion & Future work

A focused crawler is a web crawler that aims to download only web pages that are relevant to a pre-defined topic or set of topics. In order to determine a web page is about a particular topic, focused crawlers use *page scoring* and *link scoring* techniques. Page scoring determines whether a page is relevant or irrelevant. Similarity measures or classification algorithms are usually used in page scoring. Link scoring determines the links to be crawled, and prioritize the order of the links to be crawled. The right selection of the relevant links provides performance improvement in both execution time and page relevance.

In this paper, a focused crawler is implemented by applying different link scoring methods. In this study, three different types of Naïve Bayes (NB) classifiers are used for link scoring. The application of these methods provides considerable performance improvements. We show that the NB classifier with threshold provides the best performance among the others. As a future work, we plan to apply different page scoring and link scoring methods and semantic approaches for focused web crawling.

References

[1] E. Gatial, Z. Balogh, M. Laclavik, M. Ciglan, L. Hluchy, "Focused Web Crawling Mechanism based on Page Relevance." NAZOU project. Bratislava,Slovakia,2008.

[2] W. ,B. Croft, D. Metzler, T. Strohman, Search Engines: Information retrieval in practice, New Jersey:Pearson Education, 2009.

[3] A. Pal, D.S Tomarfea, S.C Shrivastava, "Effective Focused Crawling Based on Content and Link Structure Analysis". (IJCSIS) International Journal of Computer Science and Information Security, vol. 2 , 2009.

[4] F. Can, S. Kocberber, E. Balcik, C. Kaynak, H.C. Ocalan, O. M. Vursavas, "Information retrieval on Turkish texts." Journal of the American Society for Information Science and Technology, vol. 59, pp. 401-421, 2008.

[5] G. Pant, P. Srinivasan, "Learning to crawl: Comparing classification schemes". ACM Transactions on Information Systems, vol. 23, pp. 430-462, October 2005.

[6] F. Yuan, C. Yin and Y. Zhang "An application of Improved PageRank in focused Crawler" , Fourth International Conference on Fuzzy Systems and Knowledge Discovery. China,vol. 2, pp. 331-335, August 2007.

[7] Q. Cheng, W. Beizhan, W. Pianpian "Efficient focused crawling strategy using combination of link structure and content similarity", IT in Medicine & Education, IEEE International Symposium, pp. 1045-1048, December 2008.

[8] S. Chakrabarti, M. van den Berg, B. Dom, "Focused crawling: a new approach to topic-specific Web resource discovery," in 8th International WWWConference, May 1999.

[9] P. De Bra, G.-J. Houben, Y. Kornatzky, and R. Post, "Information retrieval in distributed hypertexts", in: Proceedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management, New York, NY, 1994.

[10] M. Hersovici, A. Heydon, M. Mitzenmacher, D.pelleg, "The Shark search Algorithm-An application: Tailored Web Site Mapping". Proc of World Wide Conference", Brisbane. Australia, pp. 317-326, 1998.

[11] S. Bri, L. Page, "The anatomy of large-scale hypertext Web search engine", Proc of World-Wide Web Conference, Brisbane, Australia, pp. 107-117, 1998

[12] Jon M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", Journal of the ACM, vol. 46, pp. 604-632, 1999.

[13] T. T. Tang, D.Hawking, N.Craswell ,K.Griffiths," Focused Crawling for both Topical Relevance and Quality of Medical Information", CIKM'05, New York:ACM, pp. 147-154, 2005.

[14] W. Huang, L. Zhang, "Semantic Focused Crawling for Retrieving E-Commerce Information". Journal of Software, vol. 4, pp. 436-443, 2009.

[15] W. Wang, X. Chen, Y. Zou," A Focused Crawler Based on Naive Bayes Classifier", Third International Symp. on Intelligent Information Technology and Security Informatics, China, pp. 517-521, 2010.

[16] A.Ghozia, H.Sorour, A.Aboshosha "Improved focused crawling using Bayesian object based approach" 25th national radio science conference(NRSC 2008).

[17] F.C. Ekmekcioğlu, P.Willett, "Effectiveness of stemming for Turkish text retrieval". Program, vol. 34, pp. 195-200, 2000.

[18] G. Salton and C. Buckley. "Term weighting approaches in automatic text retrieval".Inf. Process. Manage., vol. 24, pp. 513–523, 1988.

[19] A. Mccallum, K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification". AAAI-98 Workshop on 'Learning for Text Categorization', 1998.