Third International Symposium on Intelligent Information Technology and Security Informatics

A Focused Crawler Based on Naive Bayes Classifier

Wenxian Wang, Xingshu Chen*, Yongbin Zou
Network and Trusted Computing Institute, College of Computer Science
Sichuan University
Chengdu, China
catean@163.com, chenxsh@scu.edu.cn, zouyongbin@gmail.com

Haizhou Wang, Zongkun Dai
Institute of Information Security
Sichuan University
Chengdu, China
whzh.nc@163.com

***Abstract*—The exponential growth of information on the World Wide Web makes it increasingly difficult to discover relevant data about a specific topic. In this case, growing interest is emerging in focused crawler, a program that traverses the Internet by choosing relevant pages to a predefined topic and neglecting those out of concern. A new focused crawler based on Naive Bayes classifier was proposed here, which used an improved TF-IDF algorithm to extract the characteristics of page content and adopted Bayes classifier to compute the page rank. Then the crawler developed was compared with a BFS crawler and a PageRank crawler, and the results show that our crawler has better performance than the PageRank crawler and BFS crawler in harvest ratio.**

Keywords- Focused Crawler; Naive Bayes; Classifier; TF-IDF

I. INTRODUCTION

There are over 1.6 billion Websites in the World Wide Web (WWW) [1], and the indexed Websites contains at least 21.7 billion pages [2]. Search engine is widely used to retrieve information. According to CNNIC's "23nd Survey for Chinese internet development", 68.0% Chinese use search engines [3], such as Yahoo, Google and Baidu, to find info their concerned.

With the exponential growth of information on the World Wide Web, it gets harder for search engines to retrieve the information relevant to a specific topic. A crawler is widely used to retrieves Web pages for a search engine nowadays. Due to the enormous growth of the Web information, exhaustive crawling will consume vast storage and bandwidth resources. Because of limited computing resources and limited time, focused crawler has been developed to retrieve Web pages that are relevant to a topic. Focused crawler carefully decides which URLs to browse and in what order to pursue based on information of previously downloaded pages. Therefore, it is important to evaluate topic relation of pages.

II. RELATED WORKS

Early work on the topic of focused collection of data from the Web was done by De Bra et al. [4]. They proposed the fish-search algorithm for collecting topic-specific pages. Hersovici et al. [5] used the shark-search algorithm by improving fish-search algorithm. Cho et al. [6] proposed calculating the PageRank [7] score on the graph induced by pages downloaded before and then using the score as a priority of URLs for the next crawl. Chackrabarti et al. [8] used an existing document taxonomy and pre-defined documents to build a model for classifying retrieved pages into categories. The use of a taxonomy helps modeling of the irrelevant pages. Ganesh et al. [9] introduced a metric which valuated the semantic content of the URL based on the domain dependent ontology. The Link-Structure-Based method analyzed the reference information among the pages to estimate the value of every page. These kinds of famous algorithms included the Page Rank algorithm and the HITS algorithm [10].

A major problem of above focused crawlers is that it is difficult to learn that some sets of off-topic documents link to highly relevant documents, which causes problems in traversing the hierarchical web sites layouts. To solve this problem, a focused crawler was proposed in this paper. A reinforcement learning was used to train the crawler on specified example web sites containing precise taxonomy. In addition, a new page valuating algorithm based on Naive Bayes classifier is proposed which will be more effective than old ones to gather as many relevant web pages as possible.

III. SYSTEM ARCHITECTURE

Figure 1 shows the architecture of focused crawler. It has three main components: a **classifier** which makes relevance judgments on pages crawled, a **URL Queue** which determines the priority of URL obtained from crawled pages, and a **Page Downloader** which download pages from WWW. Here the basic processes are briefly outline.

A. Classifier

The classifier includes three modules: page analysis, characteristic extraction and relevance analysis.

1) Page analysis

When a page is downloaded, it is necessary to analyze its content and to extract information in order to decide on a particular link to follow. This work is done by page analysis

* Corresponding author.

module. And the processes mainly include three aspects: checking up HTML content to analyze HTML label tree, deleting useless words and extracting remainder, and transforming the URL into canonical form.

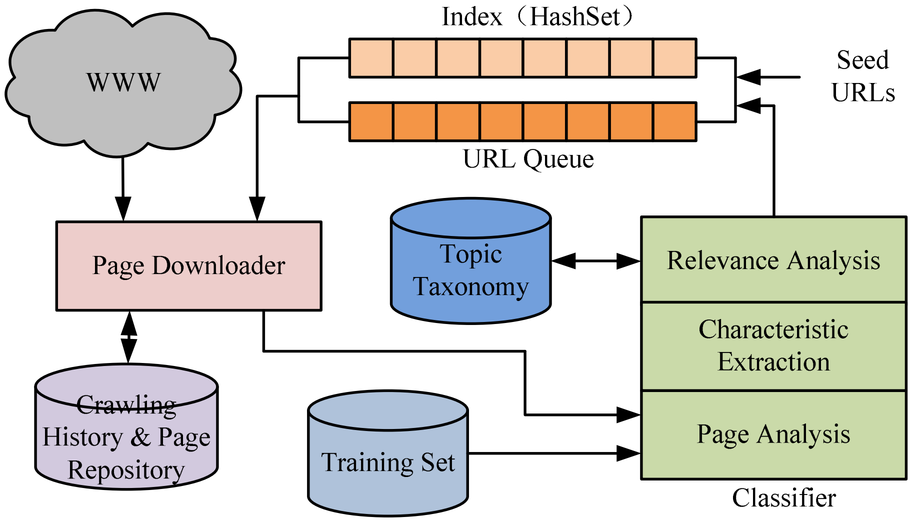

Figure 1. The Architecture of Focused Crawler

2) Characteristic extraction

Characteristic extraction module is responsible for describing the extracted content of pages with abstract mathematics. These mathematics descriptors are used to calculate the priority of unvisited URL by classifier. The contents of a link context can be represented as an n-dimensional vector, which corresponds to each element of a vocabulary word. Link context is the content adjacent to a link in the page and hints the theme of the target page. As an extreme case, the entire page can be viewed as link context of all hyperlinks in it.

Each element of vector, representing a part of Link context, is a weight that denotes the importance of corresponding word. The most common used vectorization algorithm of documents is TF-IDF (Term Frequency - Inverse Document Frequency) [11]. Here TF is the frequency of a word in a given link context, while the IDF decreases the weight of a word if it makes a lot of occurrences on the page. TF-IDF algorithm is used to compute the weight of each word in a link context. If a word doesn't exist in context, its weight is zero.

For each topic, all words, from the positive and negative samples phrases, are put into the vocabulary T. Each of positive and negative samples is expressed by a vector, and each element of a vector corresponds to a word in vocabulary T. Therefore, a page p is represented as a vector $X_p = \{w_{1p}, w_{2p}, \ldots, w_{np}\}$. Here w_{1p} denotes the weight of the first word in page p, and n is the size of the vocabulary T. In order to express conveniently, w_{sp} is used to denote the weight of word s in page p. In this paper, a modified TF-IDF algorithm was proposed to calculate the weight of each word, and the formula is as follows:

$$w_{sp} = \underbrace{\left(0.5 + \frac{0.5 \cdot tf_{sp}}{max_{s' \in Tp}\, tf_{s'p}}\right)}_{TF} \cdot \underbrace{ln\left(\frac{|E|}{df_s}\right)}_{IDF} \qquad (1)$$

Here tf_{sp} (word frequency) is the frequency of word s in page p, T_p is the set of all words in page p; E is the collection of positive and negative samples, and df_s (document frequency) is the number of documents containing the words.

3) Relevance analysis

The Bayes Classifier [12], based on the so-called Bayesian theorem, was used to carry out relevance analysis of pages. It assumes that the data x obey a certain probability distribution in a given category. For example, if the type is positive, then the appearance probability of x is $P(x|class=+1)$. Therefore, using of Bayesian formula, the posterior probability $Pr(class=+1|x)$ can be calculated by the following formula:

$$Pr(class=+1\,|\,x) = \frac{P(x\,|\,class=+1)\cdot Pr(class=+1)}{P(x)} \qquad (2)$$

Where $Pr(class=+1)$ is prior probability of related categories. So for a second-class problem, there is a following formula:

$$P(x) = P(x\,|\,class=+1) + P(x\,|\,class=-1) \qquad (3)$$

Another assumption of Naive Bayesian classifier is that the property of data *x* is independent of its distribution. Data x is in term of $\{x_1, x_2, \ldots, x_n\}$. So formula (2) can be written as:

$$Pr(class=+1 \mid x) = \frac{\prod_{i=1}^{n} P(x_i \mid class=+1)\, Pr(class=+1)}{P(x)} \quad (4)$$

The prior probability *Pr(class=+1)* and *Pr(class=-1)* are calculated according to the frequency of a category in the training data. For a given *x*, *P(x)* is a constant. The Naive Bayes algorithm is used to estimate element distribution of a given category. There are two ways to estimate these distributions.

- Normal Density: It is assumed that the element values of a given category are normally distributed. Hence, $P(x_i|class=+1)$ and $P(x_i|class=-1)$ are normal distributions with some mean and variance. Then the learning problem is estimating the parameters--mean and variance. Since examples from both the positive and the negative category have been trained, the training results can be used to estimate the needed parameters. The training data are used to derive the Maximum Likelihood Estimates (MLE) of the mean and the variance of the distributions. The estimates will be the sample mean and variance of element values for each category where the sample is the training examples.
- Kernel Density: when the normality assumption is removed, a more powerful model is obtained. It relies on non-parametric statistical methods for estimating the distribution for $P(x_i|class=+1)$ and $P(x_i|class=-1)$ [13]. $P(x_i|class=+1)$ is computed as:

$$P(x_i \mid class=+1) = \frac{1}{n} \cdot \sum_{j=1}^{n} K(x_i, \mu_j, \sigma_j) \quad (5)$$

Where *n* is the number of training examples that belong to the relevant category, x_{ij} is the *i* attribute of vector x_j (representation of the *j*th positive example), and *K* is a normal density function with mean $\mu_j = x_{ij}$ and standard deviation $\sigma_j = 1/\sqrt{n}$. A terse explanation for the (5) is that it takes an element values x_i from a vector representation of an object *x* (link context), and measures its average distance from the corresponding element values in each of the positive training examples and uses that as the $P(x_i|class=+1)$. $P(x_i|class=-1)$ is measured in a similar way. As to the proposed focused crawler in this paper, the distance is calculated using a Gaussian Kernel function [14].

And *Pr(class=+1|x)* is computed using (4). *Pr(class=−1|x)* is computed in a similar manner. Once *Pr(class=+1|x)* and *Pr(class=−1|x)* have been calculated, the assignment of category is simple. If *Pr(class=+1|x)* > *Pr(class =−1|x)*, the category of *x* is positive (or relevant), else it is negative (or irrelevant).

B. URL Queue

URL queue contains a list of unvisited URLs and is initialized with seed URLs. The classifier analyzes whether the content of parsed pages is related to topic or not. If the page is relevant, the URLs extracted from it will be added to the URL queue, otherwise will be discarded.

URL queue uses a dynamic array to store unvisited URLs. Elements of the array are of order by the estimated priority of URLs. If two URLs' priority is same, they are sorted by discovered time. Therefore, if all of the URLs have the same priorities, URL queue is a FIFO queue. The URL with the highest priority will be firstly taken out from the queue, and the corresponding page is downloaded. In the downloaded page, links will be extracted and added to the URL queue according to their priorities which are computed with the heuristic algorithm. In order to prevent duplicate links in URL queue, a Hash table has been introduced.

If there are a large number of URLs pointing to the same page, the crawler is likely to be trapped [15]. A way to alleviate this problem is to limit the number of pages that a crawler can get from a domain name. In this paper, the solution is that URL queue is guaranteed for crawling k (for example, k =100) pages from k different domain names. Advantage of this approach is that the load of crawler to web servers has been substantially reduced and that the contents of crawled pages are diverse.

C. Page Downloader

The page downloader fetches URLs from URL queue and downloads corresponding pages from the internet. In order to download a page, the page downloader contains a HTTP client used to send the HTTP request and read the response. The client needs to set the timeout to ensure that it will not take much time to read large files or wait for response of web servers. In the actual implementation, the HTTP client is limited to only download the first 10KB of a page.

Another important consideration of a crawler is the Robot Exclusion Protocol [16]. The protocol provides a mechanism that a web server administrator can define file access policy used to indicate which files cannot be accessed by crawlers. The method used to exclude robots from a server is to create a file on the server, and this file must be accessible via HTTP on the local URL "/robots.txt". Before a crawler download pages from a server, it must first obtain the appropriate robots.txt file and check whether the downloading is allowed or not. The files are cached to improve efficiency, which can avoid re-downloading the file when downloading pages from the same server. Our focused crawler follows the Robot Exclusion Protocol and caches 1,000 robots.txt files recent visited.

Crawling history is a downloaded URLs list which includes the relevance of a topic, downloaded time, crawling path of a page to another page and so on. When a page was downloaded, its URL will be added to the crawling history later. Crawling history is saved on a database and will be used for data mining after crawling. Crawling history provides a fast query to check whether a page has been crawled before or whether a page is updated if its download time exceeds a predefined threshold. Once a page is downloaded, it will be cached and stored on a page repository for further processing. In the page repository, each page is mapped to a unique file name with a low-collision Hash function. In this paper, MD5 algorithm was used to generate a 128bit Hash code for URL of a page, then the 128bit Hash value is converted into 32 hexadecimal characters as the file name of the page. For example, contents of http://www.scu.edu.cn/ are stored in a file named 355bd40a1cffbb871d1fce0048a19972.

IV. Experimental Results

The focused crawler is a C# application running on a Dual Core Processor 1600 MHz Pentium-D PC with 1024 MB of RAM and SATA disk. Our test machines are connected through a half-duplex 100Mbps Ethernet through the router to a SOCKS firewall machine. We ran the crawler with 20 threads. Chinese Web Page Classification training sets, provided by Peking University Computer Network and Distributed Systems Laboratory, were used to train the classifier of crawler. And the results were stored in a file. Then, several well-known portals like SINA (http://www.sina.cn), 163 (http://www.163.com) and TOM (http://www.tom.com) were as seed URLs for crawling.

For a variety of topics, we study the harvest ratio to see if it is high enough to warrant a focused crawl. Focused crawlers based on Naive Bayes classifier with kernel density estimation, Naive Bayes classifier with normal density estimation and PageRank algorithm were implemented respectively. In addition, a Breadth-First Search (BFS) crawler was also implemented for performance comparison.

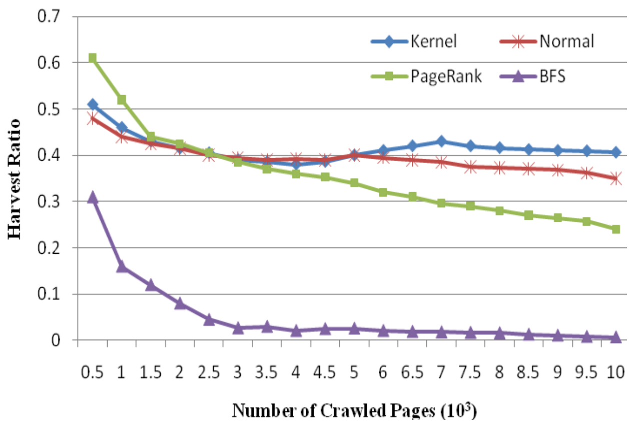

Figure 2. Crawling harvest ratio

Figure 2 shows the performances of the four focused crawlers. The results show that in early stages of the crawling the harvest ratio of our focused crawler is not so high compared to PageRank crawler, but enhancement in the performance appeared after crawling first few thousand pages. It is meaningful that there are not so much pages in URL queue in primary steps of crawling process and so there is some noise. But as the number of downloaded pages increases, the chart would be smoother and harvest ratio increases. Compared to FBS, the harvest ratio of our focused crawler is high. Within the first few thousand pages fetches, BFS is completely lost in web terrain.

V. Conclusion & Future Work

In this paper, a focused crawler based on Naive Bayes classifier is proposed to get the pages relevant to a topic. From the experimental results, we can conclude that our approach has better performance than the PageRank crawler and BFS crawler. The crawler has solved the "trap" problem effectively.

Although the initial results are encouraging, there is still a lot of work to do for improving the crawling efficiency. A major future work is to do more extensive test with large volume of web pages. Future work also includes development of distributed focused Crawler system.

Acknowledgment

This project was supported by Major State Basic Research Development Program of China (“973” Program) (Number 2007CB311106). The research presented in this paper would not have been possible without our institutes, at Sichuan University, Chengdu. We wish to thank the reviewers for their valuable suggestions.

References

[1] Netcraft: there are over 1.6 billion Websites in the World Wide Web (WWW). [EB/OL]. http://it.hexun.com/2008-03-28/104849372.html.

[2] The Indexed Web contains at least 21.7 billion pages (Wednesday, 11 November, 2009) [EB/OL]. http://www.worldwidewebsize.com/index.php?lang=EN.

[3] CNNIC: 23nd Survey for Chinese internet development [EB/OL]. http://www.cnnic.net.cn/uploadfiles/pdf/2009/1/13/92458.pdf

[4] P.M.E. De Bra, R.D.J. Post. Information Retrieval in the World Wide Web: Making Client-based searching feasible. Computer Networks and ISDN Systems, 1994, 27(2): 183-192.

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

[6] J. Cho, H. Garcia-Molina, L. Page. Efficient Crawling Through URL Ordering. In Proceedings of the 7th International WWW Conference, Brisbane, Australia, April 1998.

[7] [7] S. Bri, L. Page. The anatomy of large-scale hypertext Web search engine. Proc of World-Wide Web Conference, Brisbane, Australia, 1998, 107-117.

[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] S. Ganesh, M. Jayaraj, V. Kalyan, S. Murthy and G. Aghila. Ontologybased Web Crawler. IEEE Computer Society, Las Vegas – Nevada –USA, pp. 337-341, 2004.

[10] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 1999, 46(5), 604-632.

[11] LUO Xin; XIA De-lin; YAN Pu-liu. Improved feature selection method and TF-IDF formula based on word frequency differentia. Computer Applications, 2005, 25(9): 2031-2033.

[12] [12] PENG F P, DALE SCHUUMANS D, WANG shao-junet al. Augmenting Naive Bayes Classifiers with Statistical Language Models[J].Information Retrieva,l2004,7(3): 317-345.

[13] JOHN G H, LANGLEY P. Estimating continuous distributions in bayesian classifiers[C] Proc of the 11thAnnualConference on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 1995: 338-345.

[14] Wang, W. J., Xu, Z. B., Lu, W. Z., and Zhang, X. Y.. Determination of the Spread Parameter in the Gaussian Kernel for Classification and Regression. Neurocomputing, 2003, 55(1):643–663.

[15] Allan Heydon and Marc Najork. Mercator: A scalable, extensible Web crawler. World Wide Web, 1999, 2(4):219–229.

[16] A Standard for Robot Exclusion[EB/OL]. http://www.robotstxt.org/wc/norobots.html.