International Journal of Future Generation Communication and Networking
Vol. 9, No. 4 (2016), pp. 171-178
http://dx.doi.org/10.14257/ijfgcn.2016.9.4.14

Network Information Target Search Model and Strategy Based on Web Vertical Crawler System

Bin Wang, Jian Zhang, Na Wang, Xiaohua Sun and Yanhui Wang

College of Information Science and Technology, Agricultural University of Hebei, Baoding, China
Department Economics and Management, Baoding Vocational and Technical College, Baoding, China
Department of Digital Media, Hebei Software Institute, Baoding, China
College of Life Sciences, Agricultural University of Hebei
wb900@126.com

Abstract

With the explosive growth of the web information, acquiring the target information quickly, precisely and effectively in a large number of network information is restricted by many factors. In response to it, this paper analyzed the key technician part in the network information search engine-web crawler technology and proposed a network information target search model based on the web vertical crawler system with discussion on the implementation of the corresponding search strategy. Firstly, it builds the structure of the web crawler system and analyzes different function models. Next, it discusses some crucial problems including the options of deleting duplicated URL, the strategies to choose duplicated URL deletion method and the control model of error probability estimate model to acquire the closest network information to the target information. Finally, it verifies and discusses the operability and effectiveness of the model and the implementation strategy through case study.

Keywords*: Web Vertical Crawler System, Network Information, Target Search Model, Target Search Strategy, Search Engine*

1. Introduction

As the computer science technology and Internet technology is developing rapidly, network information shows an explosive growth. Thus, it's important to acquire the target information quickly, precisely and effectively [1-2]. In response to it, many experts and scholars have studied and analyzed the search algorithm, search model, search tool and search system of the network information[3-5], which has obtained a series of research results. It enables people to develop search engine technology and related tools and systems [6-8]. However, the customers sometimes need the information to be specific. Although using the traditional search engine algorithm, technology and tool can acquire the expected network information, its efficiency and accuracy are limited because the search is based on key word. Thus, web crawler technology, which can acquire the web information based on the customers' needs and search the entire network, has become the research focus [9-11]. Based on the existing research results, this paper discusses and analyzes the path planning, error control, strategy implementation of the search as well the realization of the system.

2. Build the Structure of the Web Vertical Crawler System

From a narrow point of view, web crawler technology uses standard HTTP protocol to traverse the World Wide Web information, whose main goal is to download the page on

ISSN: 2233-7857 IJFGCN

the Internet to local and form an image copy of the Internet information. Besides, the fetching strategy, namely the sequence of the URL ready to be fetched, is also an important problem. So the basic procedure of realizing web crawler is: firstly pick up the seed URL, then download the URL to a specific web, finally find the link and process the page. This paper builds the basic structure of the web vertical crawler system based on the above procedure, as is shown in Figure 1.

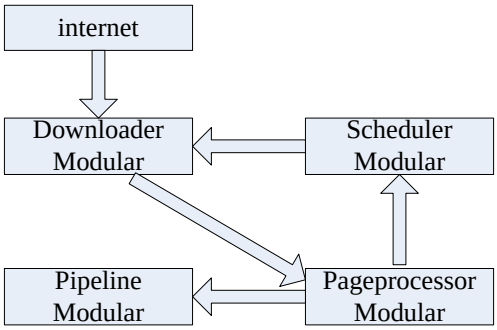

Figure 1. Basic Structure of the Web Crawler System

The Structure Consists of Four Models:

(1) Downloader module: it mainly downloads the page and prepare for further processing. This system uses Apache HttpClient as download tool.

(2) Page-processor module: it parses the page and extracts the useful information to the user, including extract URL, acquire the information on the page. Meanwhile, it also finds out the URL of the pages to be downloaded, which will be used for further download.

(3) Scheduler module: it manages the URL to be fetched and the sequence of the URL as well as deletes the duplicated URL. Using Bloom Filter can effectively delete the duplicated URL. For small-scale fetch, it only takes little RAM to store a large number of URLs.

(4) Pipeline module: It processes the fetched data and persists them into the data base or files, where the users can define them at their own will by the support of different implement interfaces. They can choose their own way to persist them, such as store them into data base or files, or output it to the console.

3. The Options and Strategies to Delete the Duplicated URL in the Web Vertical Crawler System

3.1. The Options to Delete the Duplicated URL

To avoid the scheduler modular fetching the pages repeatedly, deleting the duplicated URL, namely get rid of the links of the pages that have already been fetched, must be considered. This paper uses MySQL database to delete the duplicated URL. It uses the md5 value of the URL as the only index. It processes the input information by 512-bit group, and every group is further divided into 16 32-bit groups. After a series of processing, the output of the algorithm consists of 4 32-bit groups, which can form a 128-bit hashed value after cascade. This makes the search quicker and the table structure simpler. The algorithm comprises the following implementing steps:

Step1 filling: divide 512 by the length (bit) of the input information; if the residue is not 448, then we should fill it to be 448. The method is to fill a 1 and in 0. After the filling, the length of the information should be N*512+448(bit).

Step2 Record the length of the information: use 64-bit to store the length of the information before filling, which is N*512+448+64= (N+1)*512 (bit).

Step3 Add standard magic number: the physical sequence of the magic number is (A= (01234567)16, B= (89ABCDEF) 16, C= (FEDCBA98)16, D= (76543210)16), which corresponds to the definition in the program (A=0X67452301L, B=0XEFCDAB89L, C=0X98BADCFEL, D=0X10325476L).

Step4 Four-round loop computation: the number of the loop equals to that of the group (N+1).

Step4.1 Subdivide every 512 byte into 16 groups, making every group 64 Bit (8 byte).

Step4.2 Assume Mj is the number j sub group (from 0 to 15), <<<s means rotating s bit to the left, and then the four operations is:

FF(a,b,c,d,Mj,s,ti) represents a=b+((a+F(b,c,d)+Mj+ti)<<<s);

GG(a,b,c,d,Mj,s,ti) represents a=b+((a+G(b,c,d)+Mj+ti)<<<s);

HH(a,b,c,d,Mj,s,ti) represents a=b+((a+H(b,c,d)+Mj+ti)<<<s);

II(a,b,c,d,Mj,s,ti) represents a=b+((a+I(b,c,d)+Mj+ti)<<<s).

Step4.3 Proceed four-round computation Figure 1 shows the process.

The first round	The second round
The third round	The fourth round

Step 4.4 after the loop of every round, add a, b, c, d to A, B, C, D respectively and proceed the next loop.

3.2. Implementation Strategy of Deleting the Duplicated URL

Using MD5 encryption algorithm and MySQL database to deal with the deletion of the duplicated URL is expected to process a large number of URLs without taking too much RAM. So this paper uses Bloom Filter to delete the duplicated URL. In this way, every URL only takes up several bites and has nothing to do with the length of URL, which can save the RAM greatly. Bloom Filter has a certain rate of error (*e.g.* 1/1000 or 1/100, depending on the configuration), which will cause it miss to crawl some pages, but won't crawl repeatedly. The rational of its realization is: assume there is a set $S=\{x_1,x_2,\cdots,x_n\}$, containing n elements. Bloom Filter uses k independent hash function (Hash Function) to reflect every element in the set to $\{1,2,\cdots,m\}$. For any element x, $hi(x)$ will be set as $1(1\le i\le k)$, ($hi(x)$ is the location of the value which is reflected by number i hash function that x used) . But if a location has been set as 1 many times, then only the first time will count, the following reflection value will be seen as invalid. When it comes to judging if y belongs to the set, use hash function k times on y, if the location of all $hi(y)$ is $1(1\le i\le k)$, then regard k as an element in the set, otherwise it's not.

4. The Control Model of the Search Estimate Error Probability of Web Vertical Crawler

Bloom Filter has a certain rate of error when judging whether an element belongs to its set. Assume $kn<m$ and every hash function is random. When all the elements in set $S=\{x_1,x_2,\cdots,x_n\}$ are reflected to the array of m-bits by hush function, the probability of existing 0 (p') is:

$$p' = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m} \tag{1}$$

Where $\frac{1}{m}$ is the probability of any hash function choosing this bite, $\left(1 - \frac{1}{m}\right)$ is the probability of hash function not choosing this bite.

The approximate formula based on e:

$$\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^{-m} = e \tag{2}$$

To make it easier to analyze, set:

$$p = e^{-kn/m} \tag{3}$$

Set ρ as the proportion of 0 takes in the bit array, then the mathematic expectation of ρ: $E(\rho) = p'$。When the value of ρ is known, the error rate f is:

$$f = (1 - \rho)^k \approx \left(1 - p'\right)^k \approx (1 - p)^k \tag{4}$$

Where $(1 - \rho)$ is the proportion of 1 takes in the bit array, $(1 - \rho)^k$ is the area where hush functions with k degree picked 1.

M. Mitzenmacher has proven that the proportion of 0 takes in the bit array intensively distributed around its mathematic expectation value. Thus, add p and p' in the equation (4):

$$f = (1 - p)^k = \left(1 - e^{-kn/m}\right)^k \tag{5}$$

$$f = \left(1 - p'\right)^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \tag{6}$$

To determine the number of hash function when the error rate is the lowest, two mutex reasons of the hash function should be used: If the number of hash function is large, then the probability of getting 0 when inquiring an element doesn't belong to the set is large; on the other hand, if the number of hash function is small, then there will be more 0 in the bit array. To get the optimization number, $g = k \ln\left(1 - e^{-kn/m}\right)$, when g is the minimum, f will be the minimum, too. So g is:

$$g = -\frac{m}{n} \ln(p) \ln(1 - p) \tag{7}$$

According to the symmetry principle, when $p=\frac{1}{2}$, namely $k=\ln 2*(m/n)$, g is the minimum. Similarly, set $g^{'}=k\ln\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)$, $f^{'}$ will be the minimum as long as $g^{'}$ is the minimum, so $g^{'}$ is:

$$g^{'}=-\frac{1}{n\ln\left(1-\frac{1}{m}\right)}\ln\left(p^{'}\right)\ln\left(1-p^{'}\right) \tag{8}$$

According to the symmetry principle, when $p=\frac{1}{2}$, g is the minimum.

Under the condition that not exceeding a certain error rate, how many bites Bloom Filter at least need to have in order to represent set of n elements in the full set? Assume there are u elements in the full set and the maximum error rate is ε. X is the set of n elements from the full set. $F(X)$ is the bit array of X. Then for any element in the X x, there would be a definite result when inquiring x in $s=F(X)$, namely s can accept x. Obviously, because Bloom Filter introduces error s not only accepts elements from X, but also accepts $\varepsilon(u-n)$ errors. Thus, for a fixed bit array, it can accept $n+\varepsilon(u-n)$ elements in total, of which the number of s truly represents is n, so a fixed bit array can represent N sets:

$$N=\binom{n+\varepsilon(u-n)}{n} \tag{9}$$

There are 2^m different combinations of the m-bits array. So the number of sets they can represents is M:

$$M=z^m*\binom{n+\varepsilon(u-n)}{n} \tag{10}$$

n elements in the full set has altogether $\binom{u}{n}$ sets, so in order to represent all the sets of n elements, m-bit array must fulfill:

$$M=z^m*\binom{n+\varepsilon(u-n)}{n}\geq N=\binom{n+\varepsilon(u-n)}{n} \tag{11}$$

Namely:

$$m\geq\log_2\left(\binom{u}{n}/\binom{n+\varepsilon(u-n)}{n}\right)\approx\log_2\left(\binom{u}{n}/\varepsilon u\right)\geq\log_2\varepsilon^{-n}=n\log_2\left(\frac{1}{\varepsilon}\right) \tag{12}$$

It can be seen that when the error rate is lower than ε, m at least needs $n\log_2\left(\frac{1}{\varepsilon}\right)$ to represent every set of n elements.

It can be inferred that when $k=\ln 2*(m/n)$, the error rate f is the minimum. Now set $f\le\varepsilon$,

$$m\ge n\log_2\left(\frac{1}{\varepsilon}\right)/\ln 2=n\log_2 e*n\log_2\left(\frac{1}{\varepsilon}\right) \tag{13}$$

It can be seen that the results of the equation is $\log_2 e=1.44$ times larger than the previous lower bound $n\log_2\left(\frac{1}{\varepsilon}\right)$, which shows that when the number of hash functions is the most favorable, m at least has to be 1.44 times larger than the minimum in order to keep the error rate less than ε.

5. Case Validation and Analysis

The experiment uses breadth-first search method to crawl websites. It only needs one queue to store URL and uses memory data structure to realize URL queue, namely put the newly found URL to the end of the queue to realize breadth-first search. The web vertical crawler system provides various implement interfaces, such as store them into json file or persists them into the data base or files. The latter one, especially, needs users to design table according to their own need to store the results.

Besides, the main page parsing tool in the experiment is jsoup, which is a parser based on Java HTML. It can directly parse the URL address and HTML text content in the page. The API it provided can extract and operate data though DOM, CSS and methods similar to j Query. Meanwhile, this paper uses regular expression to extract the URL which users need, so the parsing of the page also supports regular expression. Based on the algorithm and strategy discussed in the paper, the experiment fetched the news on http://news.subaonet.com/importnews/ and stored it in the form of json. The regular expression of the page information to be fetched is (http://news.subaonet.com/importnews/\\d{1,2}\\.\\w*) and (http://news.subaonet.com/[0-9]+/[0-9]+/[0-9]+\\.\\w*). The screenshot of the experiment results after text and analysis is shown in Figure 2.

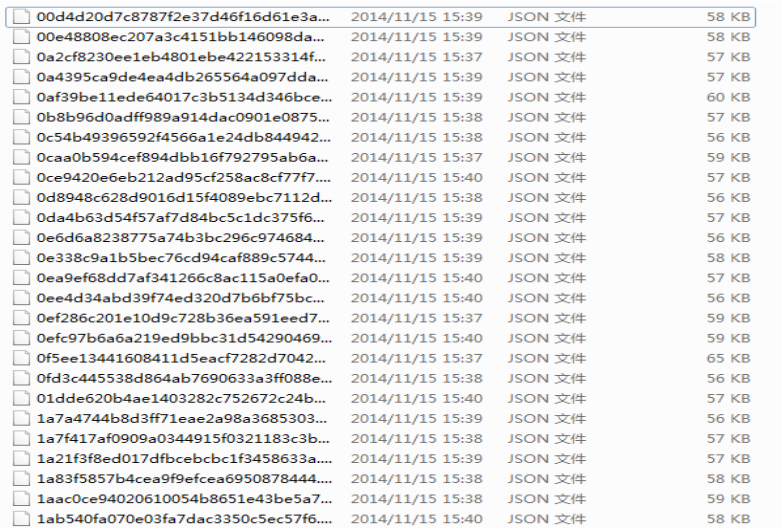

Figure 2. Screenshot of the Search Results of Network Information Based on Web Crawler System

The file is named after the MD5 encrypted value of the page. Besides, users can extract the headline; source URL; publish time and other relevant information of the network information to store them into relevant data structure in database. This paper will not go into detailed elaboration as the process is similar to the above.

6. Conclusion

It's hard to search among massive network information. In response to it, this paper proposes a network information target search model based on the web vertical crawler system and analyzes search strategy of relevant. This paper analyzes different function modules on the basis of building a structure of web crawler system. It also goes into detailed discussion on some crucial problems, including the options of deleting duplicated URL, the strategies to choose duplicated URL deletion method and the control model of error probability estimate model to acquire the closest network information to the target information. Finally, it verifies the operability and effectiveness of the model and the implementation through case study, providing a new method of search among massive network information on the Internet.

Acknowledgment

This work is supported by rural informatization engineering technology research center of Hebei province,2014 annual plan for scientific research and development of Baoding support project(Grant No.14ZS004),2015 annual plan for scientific research and development of Baoding support project(Project: Agricultural products traceability management system based on IOT),2015 University-enterprise cooperation project(Mobile version of the fuzzy query system of wild plants in Hebei Province),The authors also wish to thank the anonymous reviewers and editor-in-chief for their valuable suggestions to improve the quality of this paper.

References

[1] G. Haralabopoulos, I. Anagnostopoulos and S. Zeadally, "Lifespan and propagation of information in On-line Social Networks", A case study based on Reddit [J], Journal of Network and Computer Applications, vol. 56, no. 10, **(2015)**, pp. 88-100.
[2] K. Gërxhani, J. Brandts and A. Schram, "The emergence of employer information networks in an experimental labor market [J]", Social Networks, vol. 35, no. 4, **(2013)**, pp. 541-560.
[3] A. Ortiz-Cordova, Y. Yang and J. J. Bernard, "External to internal search: Associating searching on search engines with searching on sites [J]", Information Processing & Management, vol. 51, no. 5, **(2015)**, pp. 718-736.
[4] S. S. Liaw, W. C. Chang, W. H. Hung and H. M. Huang, "Attitudes toward search engines as a learning assisted tool, approach of Liaw and Huang's research model [J]", Computers in Human Behavior, vol. 22, no. 2, **(2006)**, pp. 177-190.
[5] T. Perez, N. L. V. Calazans and C. A. F. De Rose, "System-level impacts of persistent main memory using a search engine ［J］", Microelectronics, Journal, vol. 45, no. 1, **(2014)**, pp. 211-216.
[6] Y. Ke, L. Deng, W. Ng, D. L. Lee, "Web dynamics and their ramifications for the development of Web search engines [J]", Computer Networks, vol. 50, no. 10, **(2006)**, pp. 1430-1447.
[7] B. J. Jansen and P. R. Molina, "The effectiveness of Web search engines for retrieving relevant ecommerce links [J]", Information Processing & Management,vol. 42, no. 4, **(2006)**, pp. 1075-1098.
[8] B. I. Judit, M. H. Mazlita and M. Levene, "Methods for comparing rankings of search engine results [J]", Computer Networks, vol. 50, no. 10, **(2006)**, pp. 1448-1463.
[9] A. A. Fatemeh and A. Selamat, "An Architecture for a focused trend parallel Web crawler with the application of clickstream analysis [J]", Information Sciences, vol. 184, no. 1, **(2012)**, pp. 266-281.
[10] Q. Bai, G. Xiong, Y. Zhao and L. He, "Analysis and Detection of Bogus Behavior in Web Crawler Measurement [J]", Procedia Computer Science, vol. 31, **(2014)**, pp. 1084-1091.
[11] D. Stevanovic, A. An and N. Vlajic, "Feature evaluation for web crawler detection with data mining techniques [J]", Expert Systems with Applications, vol. 39, no.10, **(2012)**, pp. 8707-8717.
[12] M. Mitzenmacher, "Compressed Bloom Filters [J]", IEEE/ACM Transactions on Networking, vol. 10, no. 5, **(2002)**, pp. 604-612.

