Design and Implementation of a Web Crawler System based on an Adaptive Page-Rank algorithm

Xin Zhang*, Zhi Cheng, Chen Zhang

Department of Big Data and Information Engineering, Hefei University, He Fei, An Hui Province,230601, China

*Corresponding author’s e-mail: zxscee@163.com

Abstract. Web crawlers have the ability to automatically extract web page information, but there exists the issue that some pages reuse keywords to improve their search rankings. Therefore, we propose an adaptive Page-rank algorithm to build a crawler system to resolve the issue mentioned above. Specifically, we generate a relationship matrix based on the crawled web page access relationships, and then an probability matrix based on the number of web pages is generated iteratively, and finally the web pages crawled are displayed in descending order of calculated weights. Besides, we propose to control the iterative process in Page-rank with the coherence of anchor texts. The system uses Python language to realize the functions of web crawling. Experimental results demonstrate that this system has a high speed in data collection. Comparing with Hints and classical Page-rank crawler systems, The results show that the proposed method outperforms in precision and recall.

1. Introduction
With the rapid increase of network information, how to satisfy the users’ needs accurately from massive web resources is a great challenge [1]. Web crawlers can automatically grab Internet data and download to local storage through established rules. Data is crawled from major websites all year round to provide services for search engines. However, due to the fact that some web-pages reuse keywords to improve their search rankings, crawling results are often deceptive. Thus, the user needs cannot truly be satisfied and the efficiency of crawlers is therefore reduced.

Most research in web crawlers focuses on improving the speed of crawlers. For example, [2-3] propose to optimize URLs and storage structure to enhance the performance of crawling programs. However, it is difficult to balance the recall rate and the precision rate in these researches and the deception in keywords using still exists. In this paper we design an adaptive Page-rank algorithm, and propose a keyword-based and Page-rank algorithm search strategy. The algorithm calculates coherence of anchor text among each group in the process of webpage crawling and intelligently stops when the coherence is lower than a predefined threshold. At the same time, web-pages are stored according to the weights of Page-rank. Experimental results prove that the proposed system outperforms both in precision and recall.

2. Related Work
Web crawler [4-5], also known as web spider, is able to extract web page information automatically by specifying the rules in advance. To get access to network resources, the crawler program sends HTTP requests by imitating the browser firstly. Different crawler strategies are adopted in the crawler program, such as breadth-first or depth-first strategies. The operation of crawling from the current web...
The research in web crawler is mostly in improving the performance. For example, to enhance the efficiency of collecting data, some research optimizes the crawler structure from the DNS resolution stage and the page collection stage [2]. Although the performance of the crawler is improved, the deception of the crawl results still exists. In [3], researchers attempt to make using of the Rabin fingerprint algorithm to optimize the URL, but the accurate rate loses in this method. To achieve the purpose of improving crawler efficiency, a MD5 encryption algorithm for URL deduplication and a hash linked list for DNS cache are designed in this literature [6].

However, due to the issues of "tunnel phenomenon" on the Internet, it is difficult to keep a balance between recall and precision. Thus, some studies have proposed to utilize the Page-rank algorithm to resolve the above problems for subject-oriented crawling. In order to overcome the theme drift in theme crawler, the literature [7] designs an improved Page-rank method based on genetic algorithm for web search. The research applies the existing personalized Page-rank algorithm in subject-oriented crawling, and proposes a search strategy based on anchor text content [8]. Through this algorithm, the topic crawler can crawl as many web pages as possible with high topic value. Literature [9] uses the pre-set topic keywords and the keywords of the currently crawled web page to establish a vector space model, and calculates the topic correlation of the web page according to the cosine theorem. Then it is taken as a parameter into the original Page-rank formula to achieve the balance between recall and precision.

This paper designs an adaptive Page-rank algorithm, and proposes a search strategy based on keywords and Page-rank algorithm. The algorithm calculates the coherence of anchor text of web-pages in the process of webpage crawling, therefore, the crawling program is able to intelligently stop according to the predefined iterative loops or when the coherence is lower than the predefined threshold. Finally, web-pages are stored according to the weight of Page-rank. Experimental results demonstrate that the crawling pages meet users' need with less deception and a good balance is reached between recall and precision.

3. The improved Page-rank web crawler System
In this system, the Python language is utilized to implement the proposed improved Page-rank web crawler System. In the following sections, we introduce the system in detail.

3.1. Resources acquisition and web page parsing
The main objects of the crawler system are URLs. In order to find the rules of web-pages, the system analyzes the source codes structure of web-pages through means of regular expressions and BeautifulSoup in lxml format, which is provided by Python. In the process of analyzing the webpage codes, developer tools are used to determine the category of each webpage element. For example, the categories of homepage, author, and contribution of a web page are shown in Table 1.

| Element  | Tag   | Category   |
|----------|-------|------------|
| homepage | li    | one item   |
| author   | ul    | author-select-li |
| contribute | span | eight item |

3.2. Obtaining method
The crawler program has to perform IO operations both in the processes of making page requests and storing the web-pages when they are stored in local or database. In order to improve CPU utilization, the proposed system adopts an asynchronous distributed crawling strategy. Two models, namely the asyncio module and the aiohttp module in Python, are utilized to realize the asynchronous function. To
create an asynchronous crawling task, the ClientSession() method is called from this module to store all the web-pages to be crawled, thus, in our system you can directly crawl web-pages again without waiting for the analysis completion of the former web-page. Distributed crawling is performed in a distributed manner based on the Hadoop platform. The distributed crawling system obtains information through the homepage of a website. Due to the fact that the security of a website may not be high, the remote host of a website will be shut down when the website is crawling by two many robots, we preset the number of crawling pages. Regular expressions are used when acquiring web page information. For example, to obtain a URL which has a href in the form of `re.compile ('^ /.+? / $')`, the regular expression is interpreted as: href with the form "(any information) /". (Any information) / $" will be obtained. This data is the key of the hyperlink in href. Then the urljoin() function is called, which is used to combine web pages and automatically match and combine the information of the two URLs. Thus, the information can be matched and combined between the homepage of a website and the key elements of a hyperlink. Besides, the address is transformed from the original relative address to the absolute address. Finally, the matching result is stored in the set() collection, set() automatically analyze whether the webpage has been parsed. If the webpage has been parsed, it is not stored in the url set, otherwise it is stored.

3.3. The adaptive page-rank method
Page-rank is a static algorithm that can calculate the weights of web pages offline regardless of the query. Its advantage is that it can effectively reduce the amount of calculation during online query and improve the efficiency of response. In this paper, we design an improved Page-rank method, in which the iterative process of the algorithm is controlled by a predefined threshold or the coherence among anchor texts of different web-pages. Firstly, to obtain the access relationship among web-pages, the proposed algorithm parses web-pages cyclically in the website, and stores the links among these web-pages in a two-dimensional dictionary. Then, we generate a relationship matrix. In this matrix, in order to effectively distinguish each URL, two dictionaries are defined, namely index node and node index, based on the enumerate function in Python. After the dictionary is generated, we define a zero matrix M with length n * n (n: the number of crawled web pages). Two levels of loop nesting form a relationship matrix. The outer loop uses the Key of the two-dimensional dictionary as an index, and the inner loop obtains the value corresponding to the Key. Thirdly, an initial probability matrix is generated. Iteratively According to different situations, the system can generate an equal-distance probabilities matrix and the first element full acquisition probability matrix. The former assigns the value of 1/n to each element in the zero matrix, and the latter assigns the probability of ‘1’ to the first element in the zero matrix. Next, the weight of each page is calculated as follows.

\[
PR(p_i) = \alpha \sum_{p_j \in M_{p_i}} PR(p_j) \frac{1}{L(p_j)} + \frac{1-\alpha}{N}
\]

Where \(M_{p_i}\) is the collection of all web pages that have links to the web page \(p_i\), \(L(p_j)\) is the number of web pages linked from the web page \(p_j\), \(N\) is the total number of web pages, \(\alpha\) is generally taken as 0.85 empirically. Then, a new probability matrix is updated with new computed weight.

The process of generating probability matrix and calculate page weight is iterative until the halting criterion is met. In our method, the page-rank algorithm stops until the error of two iterations before and after is less than 0.001 or the coherence among web pages with high-weight are lower than the specified value. Specifically, the anchor texts of web pages with high-weight are clustered into two groups, and then we use cosine similarity to measure the coherence of each group. Cosine similarity is computed between the feature vectors constructed from the seed set and the anchor text extracted from the crawled web page by means of the tf-idf method. The proposed page-rank algorithm is shown in Figure1.
4. Experiments

4.1. Experimental settings
The experimental environment of the proposed crawling system is as follows.

Table 2. Experimental environment.

| Software        | Configuration          |
|-----------------|------------------------|
| Operating System| Windows 10             |
| CPU             | 1.1GHZ                 |
| RAM             | 8.0G                   |
| JDK             | 1.8                    |
| Server          | Apache Tomcat 8.0      |

Our crawling experiments are divided into two stages. The first stage tests the ability of the system's crawling. We chose two domestic portals, China National Geographic Network (http://www.dili360.com/), Baidu Encyclopedia homepage (https://baike.baidu.com/), and then the experiments are tested in a whole network mode. If the test state of the system is set to work in one thread, after running for about 24 hours, the collected results are shown in Table 3.

Table 3. The collected results of our crawling system.

| Run time  | # Pages found | #Pages download | Download rate     |
|-----------|---------------|-----------------|-------------------|
| 24(hours) | 1,152,000     | 460,800         | 4000/minute/thread |

The second stage is the crawler precision and recall test. Taking China National Geographic Magazine as an example, we analyze the accuracy rates and recall rates of geo-landscapes, hydrological landscapes, biological landscapes, meteorological and astronomical landscapes. The precision rate is the percentage of the number of related web-pages in the extracted pages and the total number of extracted pages in the results given to the crawling topic. The recall rate is the percentage of the number of related web-pages in the extracted pages and the total number of pages in the web given to a crawling topic.

The calculation of the recall rate is more difficult, because it is impossible to obtain the distribution of the theme resources of the entire Web, so it is difficult to know the total number of related pages in
the entire web source. Therefore, this article uses an alternative method, which is raised by Srinivasan [10]. It is calculated by dividing the sum of all target recall rates by the number of crawled pages.

4.2. Results and Discussions

In our experiments, Hints and the classical page-rank algorithm are taken as the baseline. The precision is shown in Table 4, as follows. We can see that our proposed method behaves well in all cases, although the precision is a little lower when grabbing geographical landscape. We analyze that some web pages of geographical landscape are mixed with hydrological landscape, which results in a lower precision. The recall rates are shown in Figure 2. It is easily demonstrated that the proposed Page-rank algorithm outperforms in most cases. But we can also see that the recall can be enhanced further more. It is suggested that the stop rules of coherence among each group and the predefined number of crawling web-pages in the proposed method limit the recall of related web-pages.

Table 4. The precision of the proposed method comparing with the baselines in the top-100 web-page.

| Precision               | Geo-landscape | Hydrological landscapes | Biological landscapes | Meteorological & astronomical landscapes |
|-------------------------|---------------|-------------------------|-----------------------|------------------------------------------|
| Improved PageRank       | 0.52          | 0.73                    | 0.79                  | 0.84                                     |
| Classical PageRank      | 0.46          | 0.70                    | 0.75                  | 0.73                                     |
| Hints                   | 0.47          | 0.64                    | 0.68                  | 0.69                                     |

Figure 2. The recall rates of different algorithms with 1000 crawling pages.

5. Conclusions

In this paper, an adaptive Page-rank web crawler system is designed. The algorithm can automatically determine whether to perform the next iteration by calculating the coherence in groups during the web crawling process. In the process of web page storage, crawl results are stored according to the weight computed by Page-rank. Experimental results show that the system has significant advantages in web crawling ability and accuracy. However, the disadvantage is that due to the setting of the data dictionary structure during the crawling process, the performance of analyzing the link relationship between web pages is low. Thus, the number of crawled web pages is limited in our system, which leads to a low recall rate.
Acknowledgements
This work is supported in part by the Natural Science Foundation of Hefei University (18ZR07ZDA,19ZR04ZDA), National Nature Science Foundation of China (Grant No. 61806068), the natural science research key project of Anhui university (Grant No. KJ2018A0556), the grant of Natural Science Foundation of Hefei University (Grant No. 16-17RC19,18-19RC27).

References
[1] Page L., Brin S., Motwani R., Winograd T. (1999) The PageRank Citation Ranking: Bringing Order to the Web. In: the web conference.
[2] Yin J, Yin Z.B., Huang H. (2008) Efficiency bottlenecks analysis and solution of web crawler. Computer Applications, 28(5): 1114-1119.
[3] Liang Z.Y., Zhang L.C., Huang H.(2008) Duplicated URL detection based on Rabin’s fingerprint method. Computer Applications, 28: 185-186.
[4] Lee H.T., Leonard D. (2008) IRLbot: Scaling to 6Billion Pages and beyond. In: World Wide Web. pp. 427-436.
[5] Tan X.W., Zhang Y, Pan Z.J.(2017) Research and Implementation of Web Crawler Based on Learning Resources. In: CMEE. pp. 99-104.
[6] Yan L, Ding B, Yao Z.M.(2015) Design and optimisation of MD5 duplicate elimination tree-based network crawler. Computer Applications and Software, 32(2): 325-329
[7] Yan L.L, Gui Z.J, Du W.C, Guo Q.J. (2011) An Improved PageRank Method based on Genetic Algorithm for Web Search. Procedia Engineering, 15:2983-2987.
[8] Hu W. (2012) Research and design of topic crawler based on PageRank algorithm. Wuhan University of Technology.
[9] Zhang X., Zhou M.Q., Li Z.J., Dong L.L.(2010) Focused crawler based on PageRank and Bagging. Computer Engineering and Design, 31(14): 3309-3312.
[10] Srinivasan P., Menczer F., Pant G.(2005) A General Evaluation Framework for Topical Crawlers. Information Retrieval, 8: 417-447.