Design of Data Capture Program Based on Web Crawler Technology

Rui-xing CHEN, Yi CHEN, Chun-yuan DENG and Di MO

Electronic Engineering and Automation College, Guilin University of Electronic Technology, China

Keywords: Web crawler, Data capture, Internet of things, Programming, Python.

Abstract. With the advent of the Internet of things era and the vigorous development of electronic information era, the network information resources are growing exponentially. Faced with the demand of obtaining useful information, based on the general web crawler technology, this paper uses Python software to design a deep and optimized web crawler data fetching program. This crawler program can effectively solve a series of problems, such as waiting time, information overlapping and information incompleteness, so that the crawler has good performance.

Introduction

The general search engine deals with Internet web pages, and the number of web pages that can be indexed currently has at least 21.7 billion pages [1]. How can an efficient download system be designed to transmit such a vast amount of web page data locally, creating a mirror backup of web pages on the Internet? Although web crawler technology can play such a role, with the continuous development of the Internet of things, the traditional general web crawler technology has been unable to meet the ever-changing Internet of things network and people's diversified pursuit. In this paper, the deep network crawler technology with the effect of de-duplication can be used to quickly and effectively capture data for a large and miscellaneous object, so as to achieve the non-repeatability and integrity of data capture.

Structure Design of Web Crawler

Web crawler [2] is a program that automatically crawls a web page and extracts its content. In the case that one or more uniform resource locator URL seed sets are set, data is collected from the seed web page. In the process of fetching web content, new urls are constantly put into the URL queue to be crawled until certain conditions are met. The stop condition includes cases where the crawl queue is empty and the specified number of crawls is reached.

According to literature [3], the main crawling strategies are as follows: 1) width preference and convenience strategy, 2) depth-first convenience strategy, 3) number of reverse connections, 4) PartialPageRank, 5) OPIC, 6) major station priority strategy, etc. The breadth-first convenience strategy is easy to implement but can lead to undesirable effects such as denial-of-service attacks. The depth-first convenience strategy can easily crawl to something similar to the target, but you need to set the depth, otherwise you may not find a solution. The number of backlinks will be crawled according to the importance of the web page, the drawback is that the amount of data is not large enough in the case of the effect. Partial PageRank will choose the evaluated URL as the priority for crawling, but there is a problem of large amount of calculation and slow response. The advantages of the OPIC are the same as the number of backlinks, but the disadvantage is Partial PageRank. Although the strategy of station priority has high crawling efficiency, the effect of previous crawling data is not obvious.

In order to make data fetching have timeliness, reliability, robustness and expansibility, a deep priority strategy of general network crawler architecture is selected for data fetching [4]. Its general network crawler architecture is shown in figure 1.
It can be seen from figure 1 that the working path of the depth-first search strategy starts from a set seed URL, and enters and analyzes other URLs within the selected web page. Search and analyze the web page you enter by selecting a best match URL. This loop tracks deeper URLs and web content.

### Web Crawler Optimization and Programming

#### Optimization Scheme Based on PageRank Value

Although the links provided by some portal websites are of high value, their PageRank value is also high [5]. However, as the depth-first strategy is dug up layer by layer, the web value and PageRank will decrease accordingly, and its linear relationship is shown in table 1.

#### Table 1. Table of page depth, number of pages and importance.

<table>
<thead>
<tr>
<th>Web page depth</th>
<th>Number of pages</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>600</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>More than 6000</td>
<td>Quite tiny</td>
</tr>
</tbody>
</table>

Web crawlers that work on the depth-first principle in order to fully crawl Web pages, often the crawler will repeatedly crawl the same link or all links related to hyperlinks, which will lead to the phenomenon of repeated crawling. Therefore, how to avoid such problems has become the focus of the depth-first strategy. For some repeated fetching problems of web crawler, the improvement method of PageRank value in reference [6] is improved: if there is a link to webpage A for webpage T, it indicates that the owner of T thinks that A is more important, thus assigning part of the importance score of T to A. The importance score is: \( \text{PR(T)}/\text{C(T)} \). Where PR(T) is the PageRank value of T. C(T) is the total number of external links of T, and the PageRank value of webpage A is the sum of a series of points of importance similar to T.

The formula for calculating the PR score is as follows:

\[
\text{PR}(A) = (1-d) + d \sum_{i=1}^{n} \frac{PR(t_i)/C(t_i)}{n}
\]

\[ (1) \]

\( \text{PR(A)} \) represents the PR score of webpage A; D is the damping coefficient, which is usually 0.85. PR (t_i) represents the PR score of the external linked website the ‘t_i’ itself. C(t_i) represents the number of external links owned by the external link site. Expand equation (1) to get the following formula (2):

\[
\text{PR}(A) = (1-d) + d(\text{PR}(t_1)/C(t_1) + \ldots + \text{PR}(t_n)/C(t_n))
\]

\[ (2) \]

#### Design of Deep Web Crawler Algorithm

Conventional web crawlers cannot find information and rules hidden in ordinary web pages in their operation, and lack certain initiative and intelligence [7]. For example, pages requiring user account login, pages with advanced AI authentication and pages containing page number navigation cannot be crawled. In view of these deficiencies of conventional web crawlers, the design of deep crawler makes its structure to be improved and optimized, adding improved PageRank value, form analysis...
and page status keeping three parts. By analyzing and summarizing the captured web page, we can get the high PageRank type that the web page is ordinary or can be excavated deeply. Form parameters are calculated and submitted for high PageRank pages, allowing the crawler to get more pages. The algorithm flow chart of the depth crawler is shown in figure 2.

![Algorithm flow chart of deep crawler.](image)

A typical crawler will run through all hyperlinks immediately after downloading the page, while a deep crawler uses certain algorithms to classify them. The depth crawler takes different methods to calculate query parameters for different categories and submits the parameters to the server again. If the query parameters submitted are correct, the hidden pages and links will be obtained [7].

**Experimental Simulation Effect and Analysis**

According to the above design scheme, the data capture program based on web crawler technology is realized by using Python programming language. The initial URL is an internal website of a campus, and the target is the annual water consumption of the dormitory building. PageRank is set to 3. The user operation interface designed is shown in figure 3, left. Enter the three parameters into three corresponding positions in the interface, and then click the start button to fetch the data. Data capture results are collated and analyzed, as shown in the right figure of figure 3.

![User interface design diagram and data capture diagram.](image)

The figure on the right of figure 3 gives a very intuitive understanding of the annual water consumption of the male dormitory from block A to block P. One column is the total water consumption of all the girls’ dormitory building. This program after clearly specified the search target and the initial URL, can be in the network smooth data capture and collation. When a crawler encounters a hidden web page or a page that requires an account to log in, it can also grab the
content of the page. The data captured by the program is paired with data from the school network center database, such as table 2 and table 3.

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web crawler</td>
<td>11141.2L</td>
<td>2421.7L</td>
<td>4318.4L</td>
<td>1879L</td>
<td>4040.2L</td>
<td>2191.4L</td>
<td>1411.7L</td>
<td>5228.4L</td>
</tr>
<tr>
<td>Database</td>
<td>11141.2L</td>
<td>2421.7L</td>
<td>4318.4L</td>
<td>1879L</td>
<td>4040.2L</td>
<td>2191.4L</td>
<td>1411.7L</td>
<td>5228.4L</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web crawler</td>
<td>1376.7L</td>
<td>1779.3L</td>
<td>3163L</td>
<td>2042.9L</td>
<td>1853.8L</td>
<td>1213.8L</td>
<td>953.7L</td>
<td>890.2L</td>
<td>646L</td>
</tr>
<tr>
<td>Database</td>
<td>1376.7L</td>
<td>1779.3L</td>
<td>3163L</td>
<td>2042.9L</td>
<td>1853.8L</td>
<td>1213.8L</td>
<td>953.7L</td>
<td>890.2L</td>
<td>646L</td>
</tr>
</tbody>
</table>

Through the comparative analysis of table 2 and table 3, it can be found that the overlapping and incomplete acquisition of target information can be well solved in the process of data capture. But there is a risk to the application. The program frequently visits the target site within a time period in order to achieve accuracy and completeness. As a result, the visiting time interval of the website is too short, which causes excessive load on the visited website.

**Summary**

This paper designs a crawler application based on the content of the Internet of things target data. After paying attention to the restriction protocol of website to crawler and being aware of being blocked by IP address of the other party, the program can analyze web pages and download content, solve the problems such as repeated operation of URL and improvement of in-depth optimization strategy, and achieve a better target effect.

**Acknowledgement**

This research was financially supported by the Innovation Project of GUET Graduate Education (Project Number: 2017YJCYX103).

**References**


