HK.NCCP
International Journal of Intelligent Information and Management Science
ISSN: 2307-0692 Volume 4, Issue 6, December 2015

Web Content Extraction Technology

Chua sheng WU
Software College, University of Science and Technology Liaoning, CHINA

Abstract: In this information era, we are facing the knowledge explosion, and the information on the Internet is multifarious. It is not convenient enough for us to access to information directly on cell phones due to their limitation. Based on parsing a web page with regarding it as a DOM (Document Object Model) tree, we extract the valuable information with considering three factors: structure, content and programming habits. For illustration, 28 websites are utilized to show the feasibility of the method in web information extraction, and we design the mobile client to present the web content on the cell phones. The Practice has proved that using the web page extraction technology related to this article to browse the corresponding news websites, only consumed 8% of cell phone traffic of the existing mobile phone browser did. And the user experience is improved. This method can help people to get rid of costing too much on the cell phone traffic, redundant information, complicated operations and so on.

Keywords: Information extraction; Android; Soup; DOM

1. Introduction

The 21st century is the era of information and the era of data rapid expansion. How to fast and efficient accessing to information becomes an important issue to us. Network is the most commonly used way of obtaining information for us in today's society. Browsing the web with smart phone is the best choice to meet the requirements of our access to information anytime and anywhere.

Comparing with computer, portability and easy access are the advantages of a cell phone, but its unavoidable disadvantage is that it has a smaller screen than a regular computer, and the cost of cell phone traffic is expensive. When using cell phones to browse the webs, which are designed by the size of the computers' screen, we need to Keep zooming in and zooming out to browse the whole picture of the web and read the specific information we need, which will not offer us a delightful experience. What is more, all of us have almost been through the counting and controlling cell phone traffic Month-end. Therefore, every cell phone users hope to be able to browse the web matching their screen with less traffic cost. At present, there are many websites specially developed the mobile version of their site to solve this problem. However, this "Sweep before your own door" approach is a large consumption on resource, which is not really effective to solve this problem. This paper puts forward an effective solution to solve this problem and verify the feasibility of this solution through extracting information from 28 websites.

2. Related Work

The existing web information extraction methods can be divided into five main categories: 1. the method of information extraction based on natural language processing. This method regard the web page as plain text, has a slow processing speed, needs a lot of samples. The text in the web page does not contain complete sentences, so the using range of this method is small. 2. The method of information extraction based on wrapper induction approach has strong pointedness, low expansibility and poor reusability, only can be applicable to a kind of web page. For different web pages, a lot of the wrappers are need. 3. The method of information extraction based on the ontology, this way has less dependence on the structure of web pages, but it need domain experts to create a clear Ontology in this field in detail, and its workload is big. 4. The method of information extraction based on HTML structure, its flexibility is strong, but it is only suitable for the page, which contains obvious regional structure. 5. The method of information extraction based on web query, it has good versatility and scalability, but there is no guarantee for its accuracy. Our goal is to extract information from web pages and present it on smart phones; this technology must have the following characteristics: fast processing speed, strong scalability, high accuracy. So the above methods are not suitable for solving our problem.

At present, most of the research papers in the field of web information extraction are aimed at a particular type of website. Such as MDR [1] and Delta [2] are for commodity list or form information, these particular web sites often carries with formatting information, such as record information, commodity price information, etc. In order to get a general method, recently, some scholars put forward: This problem can be solved by establish a template database [3]. However, due to the time of template matching is costly, so this method is not suitable for

browsing the web. As a result, the existing cell phone browser reduces the network traffic just by reducing the Image resolution and doesn't deal with the content for webs which don't have phone version. Our method takes the structure, the content, and the programing habits into consideration. Its processing speed is good, because it doesn't rely on a lot of templates. Its scalability and accuracy is good, because it contains a lot of rules, which is confirmed with each other. Our method can handle both strong structured and loose structured web pages with a reasonable time, so it meet the need of using mobile phones to browse the web.

3. Overall Architecture

We will add a server with the function of filtering (hereinafter referred to as proxy server) between the cell phone and the server of the goal page (hereinafter referred to as the target server), therefore, the users can get useful web content by accessing the server we added instead of accessing the target server directly. We can see the flow Chart below (Fig. 1):

The proxy server will parse the request from the client and identify the requested content, then respond directly with its resources if contents have been stored. If the requested resource is not in the proxy server, it will put forward a request to the target server, wait for the target server's response, parse the related resource, extract valuable information, establish the image file and store it locally, then respond the client's request accordingly. When other users request the same page next time, it can be responded directly by the proxy server. In order to prevent storing too much content in the proxy server, we also delete the web image files, which are requested with low frequency.

The proxy server defined in this paper is not the same as the traditional one. It doesn't only use traditional caching technology, but also its cached information which is extracted from web pages, is only part of the original web page.

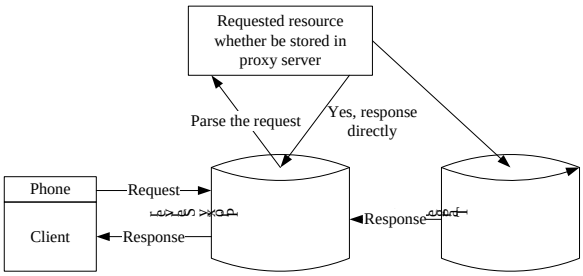

Figure 1. The overall structure of the method

Web can be seen, as a DOM tree (shown in Figure 2), the images and text in the web are nodes in the tree. Extracting content from a web page can be regarded as picking some nodes from the tree. The core work of the content extraction is how to select useful node and eliminate useless nodes at the same time.

We need to choose a parser to complete the work of extracting content from a web. There are many kinds of parser and we finally chose the Soup [4]. Soup is a Java based parser, which can parse a web both with its URL and with its HTML text content. It provides a set of efficient APIs that can be used to extract and operate data through the DOM, CSS and a method similar to the operation of the jQuery. We can directly load a Document object through the URL of a website, such as using the following code:

Document doc = Soup. Connect ("http://example.com/"). Get ();

We will get the content of http://example.com/, and store it in the document object. Then, we can find the elements through the DOM method or the selector method [5]. Through the two methods and the combination of two methods above, we can locate any position in the document. A web site generally can be divided into two categories, list page and articles page, we will show you how to deal with them respectively.

3.1. The Process of List Page

For list page, what we need includes three items: the titles of news, the release time of news (sometimes may not exist) and the link addresses of news. Because the news headlines are some links (these links are used to jump to the content pages), so we will

Pick out all nodes, which are links with the following code:

Elements choose = doc. Select ("a");

After getting these links, we need to find which of them useful news are and which are not what we need. For this discrimination, we are going to carry out a comprehensive approach:

For the beauty of the website, we need to set the CSS file to decorate it. CSS file uses different styles for different categories of nodes while the same style for the same category of nodes. For news in the list, the nodes often have the same "class" attribute value. Therefore, we can select similar nodes according to its "class" attribute value with using the method <select (".class Name ")>.

We can locate the news nodes according to the literals. The literals of the news list node are more than the other nodes and have a gathered state. Therefore, we can use the method <text ()> to obtain the text of a node, if a node's literal is significantly higher than others, we can mark it as a news node.

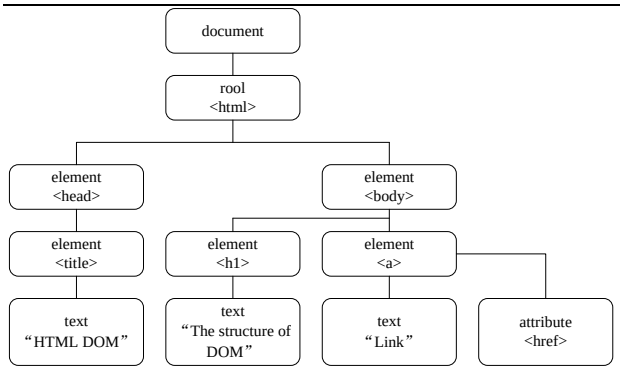

Figure 2. An example of the DOM tree

All link nodes have "her" attributes, which are used to determine the URL of the target links. Similar news should be stored under the same path on the server, so the first half of their URL is the same, different from the other kind of nodes at the same time. Since people use meaningful strings to name the directory, so the strings such as "News", "list" are often appear in the "her' path. We can use the method <attar ("Attribute Key")> to obtain the attribute values of a node.

Based on the above, we can locate the news list of each website. Then, we need to get the corresponding link address and the release time of the news. URL can be directly get from the "her" attributes (sometimes we should change the relative path to the absolute path by adding a prefix), with simply calling the Soup method <attar ("her ")>. Due to the news release time has a close relationship to the news; it is often near to the news node, appearing in the DOM tree as the child node or the sibling node of the news node. Then we pick out right ones among these nodes according to the format of release time (Numbers separated with space mark). After this we can conveniently get the release time. As shown in Figure 3, we have finished the first step in the conversion.

For the article page, what we need from it including: the title of the article, the released information of the article (including released time, author information, etc., which sometimes may not exist), the text of the article, the pictures in the article.

Figure 3. An example of extracting information from a web page

How to locate the body of the article: Article body content is generally in the core of the entire site, and has the characteristics of big literals. So the parent node's literals in the position should be significantly higher than that of the other nodes. First of all, you can use the method <parent ().text ()> to get the text of the parent node, if its literals are significantly higher than the other nodes, this parent node can be thought as at the core of the text. And in order not to regard the entire page as a core, a node should be compared with its sibling nodes layer-by-layer with a bottom-up approach. If its literals are significantly higher than its sibling nodes, it can be regarded as the core. Once the core node is identified, its sub-nodes are often the main body of the article. Regarding the parent node as the root of a DOM tree, as the body of the article in general won't be center aligned, so we use the method select (" p [alive! = center]") 〉 to select all nodes which are tagged as paragraph from the parent node. It is the body of the essay (Sometimes may contain the title or the release information, etc.).

The title of the article often locates above the core node and inside its parent node. Under normal circumstances, the article title is set tag class = title and is center- aligned, so we can get the title with the method select ("p [class = title], div [class = title] ")> or select (" p [alive=center], span [class=title]")> based on recursion. If the returning value of these methods is empty, we will locate to the grandfather node, call again until the returning values aren't empty. If there is more than one returning value, the first one shall be the title, the others and subheadings.

As the release information of the article is center aligned, we deal with it with the similar method: 〈 select ("alive=center").text ()>, but we should pay attention to a case that the returning value is the title. We need to compare the first returning value with the title of the article, if they are the same, and then abandon the former one.

The layout of image is difficult to locate, but it is usually below the title. So we can select images with the method select ("imp")>, after locating the article title.

We will package the transferred text to get XML (extensible Markup Language) files for the client to recognize them. We tag the labels according to our needs of transmission and processing. For example, we use the following form for list page:

3.3. Image Compression

The size of Image is much larger than text's, so images processing is very important. We get rid of the useless

images directly in the parsing process, so we will only deal with the useful images. In order to adapt to the cell phone screen display, reduce traffic burden, we will compress the images. And we will compress the images to different levels for different cell phones according to the specific screen pixel information we get from them, so cell phone users can save their mobile traffic and get the most suitable image size when they view the pictures.

4. Experiment Evaluation

According to the above methods, we designed the client on android mobile phones and tested the home pages of Nunki University and its department's websites.

Figure 4. An example of tagging the text

Figure 5. The presentation of using our method to show the web page on phones

Seen from the diagram above, using our solution has improved our reading experience; the original websites are difficult to read directly on cell phones, after treatment by our system they become concise and easy to read. We can see a record of actual test results (a small part) in detail in the below table 1:

we can see the data in the table 1, after using our solution, the average consumption of the traffic on visiting the listed pages is only 8% of the cost of UC Browser, the maximum consumption of traffic is less than 16.3% of UC Browser's way.

For the article page, the test results are as follows (Table 2):

We can see the data in the table 2, after using our solution, the average consumption of the traffic on visiting the article pages is only 56% of the cost of UC Browser. And the more pictures a web page containing, the more traffic will be saved.

We also measured the consumed time of the two methods to discover that they have little difference, and our method is slightly better.

The causes are as follows: For the user's request, only the first user's request needs to be transmitted to the target server by the proxy server to obtain corresponding returns for making the corresponding image documents on the proxy server. Image document returned by the proxy server will transfer more quickly since it has less content compared to the original pages.

5. Conclusion

This article provides the technology of extracting content from the web and displaying it on the cell phones, and it has a certain universal applicability. Because of typifying of web pages on the overall structure and the similarity of the writing among coders, our method can be simply extended to more

Other types of web pages. It provides a good idea for how to show webpage on a phone better in the practical work. We are facing the challenge of big data in this Information era, how to extract the most valuable information from numerous and disorderly data for the users, is one of the most important challenges to each operator.

Table 1. Actual test results of listed page in detail

Tested websites	Traffic consumed by UC browser	Traffic consumed by our solution
Nankai university	236.36KB	6KB
Nankai news	1.26MB	25 KB
College of arts	40.96KB	6 KB
School of history	92.16KB	4 KB
Philosophy school	0.99MB	10 KB
Law school	30.72KB	5 KB
Zhou Enlai school of government	71.68KB	11 KB

Table 2. Article page results

The sample pages	Traffic consumed by direct access	Traffice consumed by our solution
1	37.8KB	15.1 KB
2	30 KB	13.7 KB
3	4.3 KB	3.1 KB
4	191.7 KB	45.3 KB
5	11.4 KB	10.0 KB
6	2.6 KB	1.6 KB
7	2.8 KB	1.9 KB

We believe that our approach would give operators a beneficial inspiration, help every cell phone user to get better, more high- quality service.

References

[1] Bali, Grossman, Y. Zhao. Mining Data Records in Web Pages [C]. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, US. pp, 601-606(2013)

[2] Y.Zhai and Bali. Web Data Extraction Based on Partial Tree Alignment [C]. In Proceedings of the 14th international conference on World Wide Web. ACM, New York, NY, US. pp, 76- 85(2015)

[3] F.Qiao. Web information extraction technology based on template web crawler [D]. Chengdu: University of Electronic Science and Technology of China (2012) (in Chinese)

[4] Jonathan Hedley. Soup cookbook (EB/OL). http://jsoup.org/cookbook/.(2013)

[5] W.Andrew.Beginning regular expressions [M]. Hungry Minds Inc., US. pp, 34-60 (2005)

Subscriptions and Individual Articles:

User	**Hard copy:**
Institutional:	800 (HKD/year)
Individual:	500 (HKD/year)
Individual Article:	20 (HKD)