A Study of Web Information Extraction Technology Based on Beautiful Soup

Chunmei Zheng[1*], Guomei He[1], Zuojie Peng[2]

[1] School of Information Engineering, China University of Geosciences, Beijing, 100083, China.
[2] Tencent, Inc., Beijing, 100083, China.

* Corresponding author. Tel.:15210831696; email: zcm@cugb.edu.cn
Manuscript submitted May 14, 2014; accepted July 30, 2015.
doi: 10.17706/jcp.10.6.381-387

Abstract: In the context of comparative analysis of common web information retrieval technologies, this article discusses the principles and applications of Beautiful Soup, a vertical information search technology based on DOM tree structure. Supported by actual system examples and centering on the system architecture and core technology, this article discusses how to use Beautiful Soup to conduct deep information retrieval for partially structured webpage data, obtain directional information, reorganize the information, and then send the information to users via text message. The test results demonstrate that the web crawler achieved over 95% accuracy, satisfying the needs for commercial application.

Key words: DOM, information collection, vertical information retrieval, beautiful soup, customize.

1. Introduction

Since its emergence in 1991, the World Wide Web has quickly become a huge space for globalized information. In February of 1992, there were roughly 2.8 million web servers, storing 8×108 web pages, reaching 15 terabytes of information [1]. In the 21st century, the amount of information has reached a level so high that it's difficult to quantify. It's already become difficult for users browse the web to find the information they are seeking. Web information technology emerged from this necessity, assisting users to efficiently find file subsets related to their search query from within the vast amount of information. It has also become a hot research topic.

Nowadays, web search technology is relatively perfected. For example, the bellwether Google has impressive achievements in this area. But amidst the explosive growth of use of web information, multiple challenges have emerged, including the speed of updates and customized requests, amongst others. Facing these challenges, customized web crawlers that adapt to special topics were invented [2]. At present, search engines based on topic web crawlers (fourth generation search engines) have already become a hot research topic in present-day web information mining. With this as a background, this article uses comparative analysis to examine the pros and cons of current popular web information retrieval and mining technology, as well as using Beautiful Soup to design and develop a customized information delivery system for applications with university students.

2. Advantages and Disadvantages of Common Web Information Retrieval Technology

Research on web information retrieval technology began in the 20th century and developed into the following mainstream technologies.

2.1. Statistics-Based Webpage Information Retrieval Technology

Statistics-based webpage information retrieval technology uses the information included in each node of DOM tree species to locate the needed information node. This method is applicable is many areas, but when designating the parameters for accepting or rejecting key information, the threshold values are highly random. At the same time, because HTML files lack strict grammar standards, there are countless different HTML writing methods, which reduces the usability of this method in practical applications for customized retrieval.

2.2. Machine Learning and Heuristic Information Retrieval Technology

Information retrieval based on induction was first proposed in 1996 by N. Kushmeeric of the University of Washington [3]. This method needs users to provide many example web pages, which increases the burden of the user. N. Ashish and others joined the heuristic method [4]. Source [5] proposed webpage information retrieval based on active learning. And source [6] suggested that the effectiveness of a tag tree would be relatively good, but it would be fairly expensive to develop and have other shortcomings, such as human interference and partially structured data structures, which are difficult for machine processing. The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

2.3. Piecemeal Deletion Method Based on HTML Elements

Representative of this method is the technology presented by Ashish [7], which calculates the information entropy of each block of content in order to extract useful pieces of information and removes noise [8], [9]. This method requires, within the virtual system, a large amount of statistical computing to obtain the probability, and assumes webpage content is concentrated within a tab, which makes this method highly limited.

2.4. Information Retrieval Technology Based on the Hidden Markov Model

L. Moorn proposed information retrieval technology based on the Hidden Markov Model in 2000 [10]. This technology makes the states of the retrieved information into different properties from the Hidden Markov Model. This method is applicable for structuring the retrieval information, and its precision is relatively high. Moorn and Liu Yunzhong [11], [12] use this technology to make a system that targets the headers of English-language documents, but the system's flexibility is very poor.

2.5. Information Retrieval Technology Based on DOM Structure

Web information retrieval technology is an automatic training research method, and at present it is the most researched web information retrieval technology and the best developed method. Beautiful Soup is an outstanding example of the use of these algorithmic principles.

3. Beautiful Soup

Beautiful Soup is a Python database based on the foundation of HTML/XML analytics engine, used for extracting, analyzing, and editing information in the DOM tree of webpages. It supplies a series of concise DOM visitor interfaces, helping developers quickly build a system prototype and obtain experiment data. Additionally, it has high cross-platform flexibility.

3.1. Structural Characteristics

Users of Beautiful Soup can, according to their needs, install specific HTML/XML analytics engines (e.g., xml, html5lib, etc.). Taking lxml as an example, users can initialize Beautiful Soup with the following call: Beautiful Soup (markup,"lxml").

After initializing, Beautiful Soup obtains all the corresponding DOM tree structures for HTML documents. Then using a built-in series of interface functions that correspond with DOM API, it visits, obtains, and edits the property values or sets of the designated nodes of the DOM tree. For example, obtaining the content within the tag <b> can be done as shown in Fig. 1.

```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```

Fig. 1. Obtaining content within a tag.

Beautiful Soup not only has a built-in series of DOM API, but also many newly added interfaces that standard DOM APIs don't have, making calls easier for developers. Following from Fig. 1, users can call for the type of content within a tag:

```
type(comment)
# <class 'bs4.element.Comment'>
```

Fig. 2. Obtaining type of content within a tag.

3.2. Working Principles

Beautiful Soup will analyze any document passed to it, including all kinds of HTML/XML documents and normal documents. But only with HTML and XML documents that meet specifications will Beautiful Soup generate a corresponding DOM tree and carry out a series of API calls to obtain the designated data.

Using HTML as an example, Beautiful Soup first uses a HTML analytics engine to convert the HTML document into a DOM tree, and then the user can use Beautiful Soup's supplied inquiry and editing functions to operate a DOM tree.

```
def _findAll(self, name, attrs, text, limit, generator, **kwargs):
        if isinstance(name, SoupStrainer):
            strainer = name
        else:
            strainer = SoupStrainer(name, attrs, text, **kwargs)
        results = ResultSet(strainer)
        g = generator()
        while True:
              i = g.next()
              if  i:
                found = strainer.search(i)
                    if  found:
                   results.append(found)
                           if limit and len(results) >= limit:
                            break
        return results
```

Fig. 3. Example of _findAll.

In the majority of inquiry interfaces supplied by Beautiful Soup (e.g., findPrevious, findPreviousSibling, findParend, findParents, etc.), all will call the two functions _findOne and _findAll, which is to say that all inquiries take place within these two functions. When Beautiful Soup defines these functions, Python's lengthening parameter list technology can carry out different DOM API functions with a different number of parameters and values in order to call _findOne and _findAll, thereby achieving two different goals. For example, _findOne can search designated tag or the designated content of a tag, while _findAll can search all tags or content that meet its specified conditions within the whole DOM tree. In contrast to a depth-first search, when _findAll traverses the DOM tree it employs a breadth-first search and can thereby avoid overusing memory when searching oversized DOM trees. _findAll takes place in the following manner:

```
def getPresentation(self):
  base_url = 'http://my.yingjiesheng.com/xuanjianghui_province_'
  for i in range(1, 35):        # Take out 34[1-34] each province's future two days of hiring
info
    url = base_url + str(i) + '.html'
    page = self.getRes(url)
    soup = BeautifulSoup(page)
    countdowns = soup.findAll('div', {'class': 'list_topic'})
          y_m_d2, y_m_d3 = '', '';            # Record the 2nd and 3rd day of the presentation
dates
    first, second = -1, -1
    day = 1
    for countdown in countdowns:
         cd = string.atoi(countdown.contents[0].contents[2].string)
          if cd > 2: break            # If the countdown is over 2 days, temporarily don't consider it
          elif cd = = 1:            # Hold presentation on 2nd day (countdown = 1 day)
               first = day
               y_m_d2 = countdown.contents[1].string
          elif cd == 2:                    # Hold presentation on 3rd day (countdown = 2 days)
               second = day
               y_m_d3 = countdown.contents[1].string
               day = day + 1
    if first != -1:
        tables = soup.findAll('table', {'class':'campusTalk'})[first]
        trs = tables.findAll('tr')
        for tr in trs:
            tds = tr.findAll('td')
            city = tds[0].a.string.strip()
            school = tds[1].a.string.strip()
            addr = tds[2].string.strip()
            inc = tds[3].a.string.strip()
            pdate = y_m_d2 + ' ' + tds[4].string
            self.presentations.append(CPresentation(city, inc, school, pdate, addr))
    if second != -1:
        tables = soup.findAll('table', {'class':'campusTalk'})[second]
        trs = tables.findAll('tr')
        for tr in trs:
            tds = tr.findAll('td')
            city = tds[0].a.string.strip()
            school = tds[1].a.string.strip()
            addr = tds[2].string.strip()
            inc = tds[3].a.string.strip()
            pdate = y_m_d3 + ' ' + tds[4].string
            self.presentations.append(CPresentation(city, inc, school, pdate, addr))
            self.py_log.get_file_name(), self.py_log.get_line())
```

Fig. 4. Creating URLs and information retrieval.

4. Beautiful Soup Applications in the KKT System

KKT (http://www.knockknockingting.com) uses Beautiful Soup to design and develop a package of information customization systems. It is aimed at providing accurate customized hiring information for university students, and it performs real time delivery with text messages. Below is an introduction to the design and working principles of this information search system.

In order to lower the repetition rate of posting presentations on large hiring websites, we chose clearly structured websites that are convenient for obtaining information to serve as search resources. Additionally, the newly graduated student websites didn't use dynamic technology (e.g., Ajax) to present data, thereby making it easy for web crawlers to gather information. First, the newly graduated student website has different URL addresses for different geographic regions, whereby numbers are used to distinguish provinces and regions. For example, 1 represents Beijing and 2 represents Tianjin:

http://my.yingjiesheng.com/xuanjianghui_province_1.html

http://my.yingjiesheng.com/xuanjianghui_province_2.html

Using the different URLs for different regions, a web crawler can use HTTP to send requests to the server and thereby obtain the local presentation information. Then users can use Beautiful Soup to analyze and retrieve information from webpage data that has been converted into a DOM tree. The code for creating the URLs and information retrieval is listed separately in Fig. 4.

5. Experiment and Analysis of Results

5.1 Experiment Data Collection

The search update rate for the presentation system is once every 24 hours. After users using the KKT system enter their information, the supporting database can search the entered data. If the user passes the buddy authentication and has legitimate customized information, then the user will receive a customized information reminder. Because the presentations are in Chinese, the system inquiries only generate information in Chinese, sending it via text message to the user's cell phone. But according to publishing house requirements, Chinese cannot appear within the article, therefore the image for customized information results cannot be shown here.

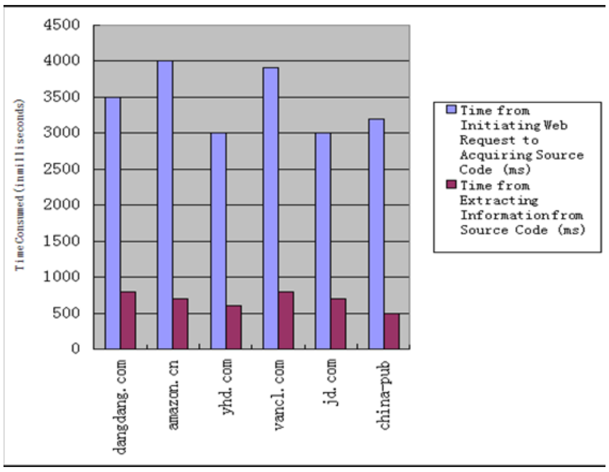

Fig. 5. Large website information retrieval performance comparison.

5.2 Data Analysis

Combining the data responses from experimental users, using Beautiful Soup for information retrieval achieved an accuracy of nearly 100%, mainly benefiting from the information retrieval rules of website in-depth customization. Over four months of service, only two abnormalities occurred in obtaining information. One time was due to the website's failure to obtain source code which led to an information delivery failure. In this instance, the filtering of information submitted by users could be strengthened and the filtering conditions could be made stricter. Overall, in order to increase working efficiency while guaranteeing data accuracy, one must design robust and comprehensive information retrieval rules as well as consider and resolve the data abnormalities.

Besides retrieval precision, retrieval efficiency is also a key metric for assessing an information search and retrieval system. Fig. 7 is in the same web conditions, after testing several large B2C websites which have been converted to DOM trees, using Beautiful Soup to conduct a time comparison for traversal searches. B2C websites were tested because they have large quantities of information and complex structures, which allows for a good measurement of the working efficiency of the information search system. Time was recorded based on the start-stop time of code execution.

From the data in the Fig. 5, the time for information retrieval is much lower than the time for web requests. Thus, besides not creating bottlenecks, Beautiful Soup also satisfies users with its retrieval efficiency.

6. Conclusion

Building on the foundation of popular website research within China, this paper integrates the needs of recent university graduates, developing a completely new vertical search web crawler system based on Beautiful Soup technology. This system is already online, and users can access http://knockknockting.com (KKT) to visit the system and customize their search inquiries. Thus far, the KKT system has already stably and effectively supplied tens of experimental users with several months of customized information service. Through high search efficiency and accuracy, stable system performance, and good operation, the web crawler technology has demonstrated its effectiveness. Currently, the system is still in an experimental stage, and before commercialization it still needs increased efficiency as well as improved load balancing and distributed web crawling.

Acknowledgements

The research work was supported by “the Fundamental Research Funds for the Central Universities”.

References

[1] Lawrence, S., & Giles, C. L. (1999). Accessibility and distribution of information on the web. *Nature, 400,* 107-109.

[2] Peng, T. (2008). Key research problems in vertical information searching. Academic Paper.

[3] Chang, H.-Y., Zhu, Z.-Y., Chen, Y., Zhang, P., & Zeng, L.-F. (2010). Content extraction technique for for web pages based on HTML-tags. *China Computer Engineering & Design, 31(24),* 5187-5191.

[4] Pu, Y.-D., Guan, Y., & Wang, Q. (2008). Research on webpage main text retrieval methods based on data mining thinking. *Computer Knowledge and Technology*, *1*, 120-123.

[5] Liu, B.-Q., Wang, Y.-H., Ge, D.-M., & Li, J. (2007). Extracting text content from Chinese webpages ased on parsing structural tree. Academic Paper.

[6] Liu, T. (1999). Research on automatic abstracting based on text multilevel dependency structure. *Journal of Computer Research & Development, 36(4),* 22-24.

[7] Ashish, N. (1997). Knob lock C1 wrapper generation for semi-structured internet sources. *Proceedings of Workshop on Management of Semi-Structured Data* (pp. 10-17). Tucson, Arizona.

[8] Zhu, C.-L., *et al*. (2005). Information extraction from webpages based on active learning. *Intelligence Journal*, *23(6),* 667-671.

[9] Zhang, S.-Y., Du, G.-Y., & Zhu, Z.-Y. (2004). Research on partially structured information retrieval tecnology. *Journal of Systems Engineering and Electronics, 26(5),* 610-612.

[10] Lin, S. H., & Ho, J. M. (2002). Discovering informative content. Blocks from web documents. *Proceedings of ACM SIGKDD*.

[11] Moorn, L. (1999). Record-boundary discovery in web-documents. *Proceedings of the 1999 ACM SGIMOD*. Philadelphia, Pennsylvania, USA.

[12] Liu, Y.-Z., Lin, Y.-P., & Chen, Z.-P. (2004). Text information extraction based on the hidden markov

model. *Journal of System Simulation, 6(4),* 535-541.

Zheng Chun Mei received the PhD degree from China University of Geosciences, China in 2014. She is a lecture in the China University of Geosciences, China. Her research interests include computer sciences courses, including C++, data structure, and operating systems.

Guomei He received the graduate degree from School of Information Engineering, China University of Geosciences, China.

Zuojie Peng received the graduate degree from School of Information Engineering, China University of Geosciences, China.