2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.25

Eliminating Noisy Information in Webpage through Heuristic Rules

Xin Qi * and JianPeng Sun

College of Computer Science and Technology, Wuhan University of Technology Wuhan, China

Abstract. Web pages typically contain a large amount of information that is not part of the main contents of the pages. The noisy information harms Web information processing. In this paper, we propose a novel method to deal with Web page noise. Since no training data set and artificial annotation, this technique has a very wide of versatility. The noise elimination technique based on the following observation: a web page contains many information block, while the topic blocks usually have the characteristics of aggregate. Based on this observation, we propose a tree structure, called satisfiable sub-tree. The process of eliminating noisy information translates into finding the satisfiable sub-tree. The proposed method is evaluated with several portals website. Experimental results show that our method is able to effectively detect and eliminate noises in Webpage.

Keywords: Noise elimination, DOM Tree, Information Block, Heuristic Rules

1. Introduction

Since the birth of the World Wide Web, after just a few decades it has become the world's largest public data sources. In order to efficiently obtain the required data from the mass of information, people developed a variety of search engines, web crawlers. However, these tools there are many deficiencies. Larger portion of the returned information is often not what we expected, not only a waste of valuable time and reduces the user experience. In addition to the complexity of human language itself, the key factor is the page filled with too much noise, such as navigation directory, ad links, copyright notices and so on. The noise harms web mining, and thus how to effectively eliminate the web noise has become a hot issue in the current.

2. Related Work

For elimination of noisy information in web pages, people have proposed many innovative approaches; based on the basic principles can be divided into two categories: methods based on DOM tree and methods based on machine learning.

In [1], they proposed the use of semi-automatic wrapper induction method. This method focused on learning extraction rules from the manual annotated page or data record set. In addition to time-consuming and labor-intensive manual annotation, does not have the versatility, the wrapper verification and repair is also facing a major challenge.

In [2], they proposed the first to use iterative algorithms between the same layer tables to eliminate noisy information around the main content in web pages. Then through the Levenshtein distance between two strings to eliminate noisy anchor text. However, in the current mainstream standard XHTML web design is no longer use the table positioning technology, instead using div + css technology. And alone by calculating the anchor text and page title text similarity to determine if it is noise, the accuracy is not high.

* Corresponding author:
E-mail: qixin.whut@gmail.com

In [3], they proposed eliminating noisy information based on a page layout analysis. The first step is to use the VIPS algorithm to identify all page blocks, and then based on a set of heuristic rules to eliminate the noisy blocks. However, the realization of VIPS algorithm is more complex, time expensive, and some of the current implementation of the algorithm is based on IE kernel, so there are still some practical limitations.

In [4], they proposed a web noise filtering method based on a DOM tree. Based on the integrated use of regular expressions, position and size of the node, it conducts the noise node identification. Using regular expressions the effect of eliminating noise is not good, because a lot of noisy nodes do not meet the pre-written regular expressions. In addition, as the diversity of web design, position and size of the noisy node is also difficult to determine a common threshold.

Because Web information dissemination, structural design has great arbitrariness, it has not found an effective method of eliminating noisy information. Inspired by previous work, combined with practical experience, this paper presents a new approach. The basic idea is to block web page information, the characteristics of the theme aggregation. And page segmentation and noise identification together to ensure the accuracy and noise elimination efficiency.

3. Fundamentals of the Proposed Technique

In general, the Web can be divided into three categories: HUB pages, picture pages and topic pages [5]. The content of HUB page is hyperlink list; they are not the interested object of the general Web mining system.

3.1. Overview

Through analysis of several major domestic portals and found that often blocks Web pages, and the same topics are often placed in a block. Figure 1 shows a typical topic page segmentation models, usually the topic information block is B2, B5 for copyright information block and so on. This design pattern consistent with people's habits of mind, ease of maintenance, as well brings opportunities to eliminate noise in pages. Based on the above analysis, noise elimination of a web page can be divided into the following two steps: identify the various information blocks, eliminating noise block. Recognition of information blocks using the VIPS algorithm has been proposed, but there are still many deficiencies in the algorithm. The method proposed in this paper brings together two steps to ensure the accuracy and noise elimination efficiency.

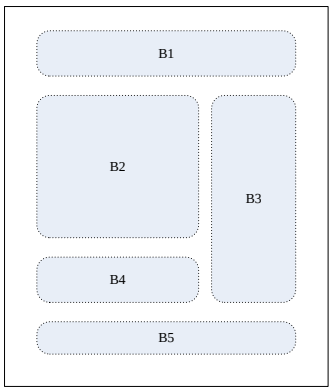

Fig. 1. A typical block pattern of topic page

3.2. Fundamental Idea

Using HTML tags in page to build the DOM tree (also known as the tag tree) is a necessary step of many information extraction methods. This method also is based on the tag tree. Assume that a given page P has N information block, that is $B=\{b_i \mid 1\le i\le N\}$. For each information block b_i, the corresponding sub-tree st_i can be found in the DOM tree of P. Consequently, on the basis of B, a sub-tree set can be constructed, that is $ST=\{st_i \mid 1\le i\le N\}$. $root(st_i)$ is defined as root node of sub-tree st_i. Root node set of sub-tree ST is $ROOT(ST)=\{root(st_i) \mid 1\le i\le N\}$.

For each element $st_i(1\le i\le N)$ of ST, If both of the following conditions can be satisfied that it is called:

- In G-generation child node of st_i, at least one child node is a text node $TNode_k$, and $LENGTH(TNode_k)\ge\alpha$.

- n text nodes, m anchor text nodes in st_i ($m \le n$,anchor text is a special text), if:

$$SL_{text} = \sum_{j=1}^{n} LENGTH(TNode_j)$$

$$SL_{anchor} = \sum_{j=1}^{m} LENGTH(TNode_j)$$

Then there is $SL_{text} \ge \beta$ and $SL_{anchor} \le \gamma \times SL_{text}\ (0 \le \gamma \le 1)$.

Where, $LENGTH(TNode_x)$ denotes the length of the string in text node x,

SL_{text} denotes sum of the length of all text nodes, and SL_{anchor} denotes sum of the length of all anchor text node.

All satisfiable elements in ST construct a set, called to Satisfiable sub-tree Set, namely:

$SST = \{ st_i \mid st_i \in ST \wedge st_i \_ is \_ satisfiable, 1 \le i \le N \}$

$\forall st_i, st_j \in SST$, if st_i is an element of st_j, it should be deleted from SST.

The result can be the largest Satisfiable sub-tree Set $SST_{\max} = \{ st_i \mid st_i \in SST, 1 \le i \le N \}$, and $ROOT(SST_{\max}) = \{ root(st_i) \mid st_i \in SST, 1 \le i \le N \}$.

So, the process of noise elimination in a web page can be described as follows:

First, find the elements of $ROOT(SST_{\max})$, then depth-first traversal sub-tree corresponding to each element node, at the same time printing the text node.

An example given below illustrates the above can be more intuitive process. For simplicity, only the necessary HTML fragment is given.

```
<body>
    <h2>News title</h2>
    <div>
        <p><b>Paragraph one</b>
         <p align = "center"><img src = "Imgpath"/>
        <p>Paragraph two
         <p><a href = "Infopath">Related Info</a>
    </div>
    <div>
         <a href = "link1">Link1</a>
         <a href = "link2">Link2</a>
         <a href = "link3">Link3</a>
    </div>
</body>
```

This is a typical HTML fragment, and corresponding DOM tree fragment shown in Figure 2.

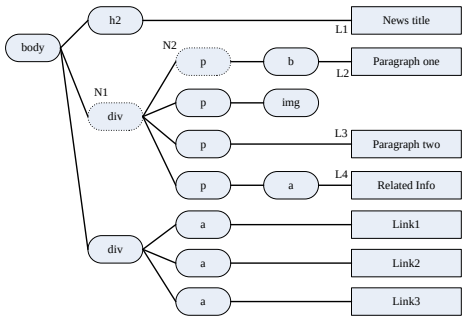

Fig. 2. A fragment of DOM tree

G, α, β, γ respectively take the appropriate threshold, so that is:

$ROOT(SST) = \{N_1, N_2\}$

$ROOT(SST_{\max}) = \{N_1\}$

Then, N_1 is the root node to depth-first traversal. It is noteworthy that the method discards the L1 node which is very important. Fortunately, information in L1 node can be obtained by analyzing title tag, because both the content often is the same.

3.3. Implementation of the Proposed Technique

3.3.1. Webpage Parsing

Web page parsing is the process of establishing the corresponding DOM tree based on a given HTML page, but the HTML language itself has a high degree of flexibility, allowing the realization of the process becomes very complicated. There are already many excellent open source project can help us accomplish this work. This paper used a very popular HTML parser NekoHTML [6]. NekoHTML can often revised HTML coding errors, and allows developers to use standard XML interface to operate on the HTML document.

3.3.2. Webpage Filtering

Before eliminating noisy information, filtering the DOM tree can effectively improve the noise elimination efficiency. The main targets of the process include JavaScript, css, applet, comment, etc. Implementation, the root node of DOM tree is the starting point of depth-first traversal. During recursive process, determine whether each node should be cut.

3.3.3. Elimilating noisy information

Implementation as follows:

```
Vector<Node> resVec = new Vector<Node>();
public String getTheme() {
deNoise(body);
getMaxSet(resVec);
   if(resVec.isEmpty()) {
      return "";
   } else {
      StringBuilder sb = new StringBuilder();
      for(Node node : resVec) {
         sb.append(getText(node).trim());
      }
      return sb.toString();
   }
}
```

Which, resVec to store the root nodes of the Satisfiable sub-tree, body corresponds to the body node in DOM tree, getMaxSet(resVec) returns the largest Satisfiable sub-tree Set, getText(node) returns all text content of all sub-tree whose root is node.

```
private void deNoise(Node root) {
if(root == null){
      return;
} else {
short nodeType = root.getNodeType();
if(nodeType == Node.TEXT_NODE) {
String textline = root.getNodeValue().trim();
if(textline.length() >  ) {
Get up along the DOM tree to G-generation parent node:pgNode;
if(pgNode is Satisfiable ) {
                  resVec.add(pgNode);
    }
   }
} else if(nodeType == Node.ELEMENT_NODE) {
      NodeList children = root.getChildNodes();
      for(int i = 0; i < children.getLength(); ++i) {
                  deNoise(children.item(i));
                  }
      }
}}
```

This method depth-first traversal the DOM tree, starting from the text node to find the Satisfiable sub-tree. The result is stored in variable resVec.

4. Experimental Results and Analysis

In order to evaluate the effectiveness of this noise elimination method, Web crawler respectively, from Sina, Tencent, Sohu and Netease four portals, crawled 1000 pages as a test set. Evaluate hardware and software environment is: JRE 6 Update 22, Fedora Core 7, Dual Core, Clock Speed 2000 MHz CPU, 1GB RAM.

It was found that when the value of G, α, β, γ is 2, 40, 100, 0.3, the better noise elimination, and the average speed is about 30 pages/sec. Taking into account the time of file I/O and web page parsing, this speed is acceptable. In order to more accurately evaluate the algorithm, the effect of noise cancellation is divided into three levels:

Level A means that all noise can be eliminated without loss of information.

Level B means that most of noise can be eliminated without loss of information.

Level C means that noise can not be eliminated with loss of information.

Respectively, from the results of four portals randomly selected 100 samples for analysis and evaluation, statistical results shown in Table 1:

TABLE 1. EXPERIMENTAL RESULTS

Web Portal	Level		
	A	B	C
http://www.sina.com.cn	94	1	5
http://www.qq.com	97	0	3
http://www.sohu.com	94	2	4
http://www.163.com	95	2	3

Table 1 show that this method can effectively eliminate noisy information in about 95% of the pages. 1% of the page noise elimination is ineffective, mainly because some of the noise information and the main content together in one information blocks, the block can not properly be treated differently. Another 3% of the page can not be processed; such pages include short news, forums, etc. Because main content in these web pages is small or less obvious, leading to algorithm failure.

5. Conclusions

In this paper, we proposed an effective method to eliminate noisy information in web pages. The basic idea of this method is web page blocking and topic aggregation. A set of heuristic rules is defined. By bottom-up way, the Satisfiable sub-trees are found in the DOM tree. Experiments show that this method can effectively and efficiently deal with the noise of the web pages and does not depend on the specific page mode, with good versatility. However, this method is not very good for web page that has less main content, which needs further research and improvement.

6. References

[1] Liu Bing, *Web Data Mining*, Tsinghua University Press, 2009.

[2] An Algorithm for Noise Reduction in Web Pages Based on a Group of Content-related Rules, New Technology of Library and Information Service, 2008(003): p. 51-54.

[3] Lei FU, Yao MENG, Yingju Xia, etc.Web Content Extraction based on Webpage Layout Analysis[C]. Kiev，Ukraine：The 2nd International Conference on Information Technology and Computer Science，2010：40-43.

[4] ZHENG Yan, CHENG Xiao-chun, CHEN Kai. Filtering noise in Web pages based on parsing tree[J].ScienceDirect, 2008, (Suppl.)：46-50.

[5] Li XiaoMing, Yan HongFei, Wang JiMing, Search Engine - principles, technologies and systems, Peking:Science Press, 2004

[6] Clark, A. CyberNeko HTML Parser. Available from: http://people.apache.org/~andyc/neko/doc/index.html.