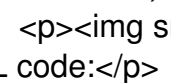


One of most important tasks in information retrieval science is to extract useful content from Web pages. The problem is Web pages (usually HTML) contain some information (text) with tons of information additional texts - navigation structures, advertising etc. From point of view of information retrieval tools each HTML page has main (useful) content and helpful information that is good when viewing Web, but not when extracting the data. The task is to filter useful content from HTML without knowing of structure of the page. It is not difficult to get useful content when structure of HTML page is known. For example, with using

[WebScrapers](index.php?option=com_content&view=article&id=53:extracting-data-from-html-pages-with-perl&catid=38:perl&Itemid=56). But it is not possible to create templates or REGEXP expressions for every page in the Web. There is simple solution for this task.

As it is known, HTML page can be presented as tree view.

Example:  Or another example as HTML code:

```
<div id="container">  
<h1>Main Heading</h1>  
<p>Most programmers  
<em>rely</em> on caffeine</p>  
<p>Most programmers like Anime</p>  
<div class="lister">  
<h2>The 'P' Interpreted Languages</h2>  
<ul>  
<li>Perl</li>  
<li>PHP</li>  
<li>Python</li>  
</ul>  
</div>
```

My idea of extracting useful content is to represent HTML page as tree of nodes and then check the list of one level nodes to see how text is distributed between them. Then take node with major text part and process its subnodes with the same procedure. Let look at example.

```
<html>  
<head>  
<title> My New Web Page </title>  
</head>  
<body>  
<table>  
<tr><td colspan="2"><h1> Welcome to My  
Web Page! </h1></td></tr>  
<tr><td width="200"><a>Menu item 1</a>  
<br></td><td id="maincontent"><p><br />This page illustrates how you can write proper HTML  
<br />using only a text editor, such as Windows Notepad. You can also <br />download a free  
text editor, such as Crimson Editor, which is <br />better than Notepad.<br /><br />  
<br />There is a small graphic after the period at the end of this sentence. <br /> The graphic is in a  
file. The file is inside a folder named "images."<br /></td></tr>  
</table>  
</body>  
</html>
```

Link: <http://www.yahoo.com/> Yahoo! Another link: <tableexample.htm> Another Web page Note the way the BR tag works in the two lines above. <index.htm> HTML examples

For any HTML page it make sense only to look at body of the document - everything that is in `<body>...</body>` tags. So we have top level node in the tree - tag `<strong>body</strong>`. My algorithm of extracting useful data is:

- Get root node (body tag) and put it for processing to recursively procedure that begins at step 2.
- Calculate length of clear text in node.
- Get list of subnodes for node.
- If there is only 1 subnode then go to step 2 to process this subnode.
- For each subnode calculate length of clear text in the subnode.
- Get coefficient (K) of distribution of a text between subnodes.
- If K is low (<5%) it means that text is uniformly distributed between subnodes and current node is exact main content of the page.
- If K is not low (>=5%) then choose the subnode with longest clear text, and apply the same

procedure for this subnode (go to step 2). There is recursive call of content extracting function.

There are different ways to calculate coefficient of distribution of a text. And different value can be used as signal to stop. I used 5%.

Let look how this algorithm will work with example above.

First it will process node - `<strong>body</strong>`.

`<strong>Body</strong>` has 1 subnode - `<strong>table</strong>`. So just take this subnode to procedure.

Table has 2 subnodes - `<strong>tr</strong>` 2 times. One of them (second) has much more text than first. Coefficient of distribution of a text will be high there. So, next node to process is second `<strong>tr</strong>`.

Let's go deeper. The `<strong>tr</strong>` has 2 subnodes - `<strong>td</strong>` 2 times. The second has much more text. It will be used in next step.

`<strong>Td</strong>` tag (current node) has few subnodes - `<strong>p</strong>` few times. In this case coefficient of distribution of a text will be low. Because, part of the text is in each p tag. So, at this step procedure will stop and return HTML code in this `<strong>td</strong>` as main (useful) content of the page.

Calculating of the coefficient (K).

Let  $L(i)$  is length of text in subnode number  $i$ ,  $i=1..n$ . Where  $n$  is count of subnodes of node.

FTL is length of all text in node (just text after removing HTML tags).

LV is linear variance of list  $L(i)$ ,  $i=1..n$ .

And at last  $K=100*LV/FTL$ .

I have created simple tool to demonstrate how this works. It is written with Perl. [Getting useful content of the Web page](cgi-bin/science/parsing/1/getcontent.cgi).

Try to post some urls there. It is not ideal. But works fine for all tests that I did.