EVALUATION OF HAP, ANGLESHARP AND HTMLDOCUMENT IN WEB CONTENT EXTRACTION

Erdinç Uzun
Namık Kemal University

H. Nusret Buluş
Namık Kemal University

Alpay Doruk
Namık Kemal University

Erkan Özhan
Namık Kemal University

Abstract

With the DOM, programming languages can access and change all the HTML elements of a web page. There are several libraries for instantiating the DOM. In this study, we compare three different well-known .NET libraries, including HAP (Html Agility Pack), AngleSharp and MS_HtmlDocument to extract content from web pages. The experimental results indicate that AngleSharp achieves the best results with average 5.54 ms for preprocessing of the DOM and average 0.46 ms for extracting of a content from the DOM.

Keywords: HTML, DOM, Web Extraction Performance, .NET C#

INTRODUCTION

Over the years and with the increase of internet usage, the amount of contents such as main text, summary text, user information, date information, advertisements, menus and related links on a web page has also increased. Web content extraction [1] is the process of accessing required contents from web documents. This process is important for indexing and storage operations on the computer. In this way, the performance of search process can be accelerated. In this study, we will explain how to access this content via ID and CLASS. Moreover, three libraries that can be used extraction process are compared.

Web content extraction methods can be classified into three classes: wrapper-based methods[2], DOM (Document Object Model)-based method [3] and Machine learning-based methods [4]. Wrapper in web content extraction is a program that extracts content of a particular information from web pages. DOM-based methods utilize structure, tags and attributes of HTML. Machine learning-based methods are on state-of-the-art machine learning algorithm and these methods require labeled data obtained from the DOM. This paper focuses on creation and search time of DOM.

In the DOM, everything is a node containing nodes, tags, attributes, texts. Moreover, everything including main text, summary text, graphics, audio, video, links, advertisements, contents, etc. on the web page is in a specific node. In all browsers, when an HTML document is loaded into a web browser, the DOM is created. With JavaScript scripting language, you can reach all the nodes by using the DOM. In this study, three libraries included HAP [6], AngleSharp [7], MS_HTMLDocument [8] developed to reach the DOM via a programming language .NET (C#) will be compared.

HIERARCHY OF DOM

A web page is a text document included HTML tags, HTML attributes, scripts and contents. This document is parsed by a web browser to display results in the browser window. After parsing, the DOM is created. The DOM is an object-oriented representation of the web page, which can be modified with a scripting language like JavaScript and programming languages such as C#, Java, Phyton and etc.

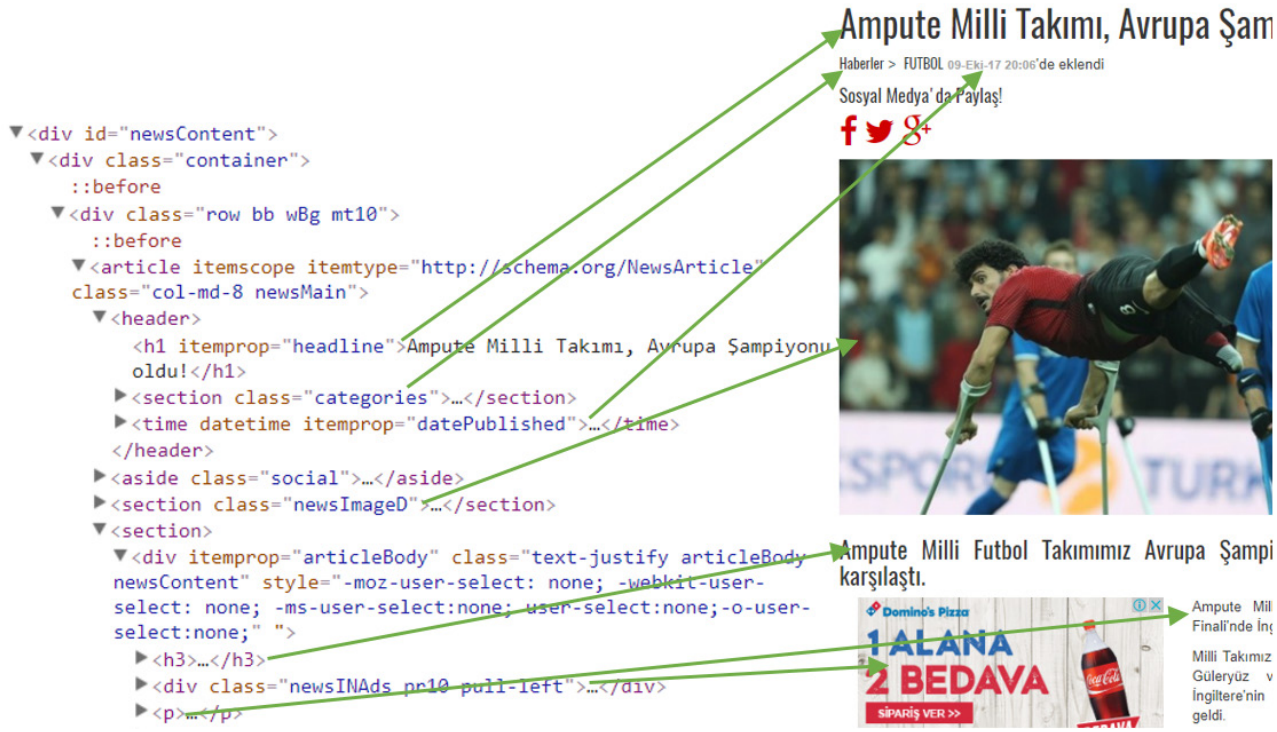

Fig. 1. *Relationship between the DOM and the web page*

There is a relationship between the DOM and the web page as shown in Fig. 1. For example, <h1> is an HTML element that defines the most important heading. Moreover, this element have an attribute (itemprop) and value of attribute (headline) that provide additional information for the element. There are a lot of HTML elements such as <section>, , <aside>, <h3>, <p> and etc. in a web page. Fig. 1 shows only a little part of a web page. Additionally, these elements have attributes such as id, class, itemprop, style and etc. id and class are widely used attributes useful for applying styles and manipulating an element with DOM and JavaScript. These information can be used for extraction process.

Web pages in a web domain have similar HTML elements. When the appropriate element is selected from a web page, it can be used to extract other web pages of domain. In experiments, we have downloaded 100 different web pages for 20 different web domain. Moreover, we have selected 38 different elements for obtaining required web content. Table 1 indicates example elements (as rules) for Fig. 1.

Table 1. *An example rules for extraction*

HTML Element	Description
<h1 itemprop="headline">	Heading
<div itemprop="articleBody" …>	Main Text

WEB EXTRACTION LIBRARIES

In web browsers, JavaScript is used for accessing or manipulating an element. However, in programming languages there are several libraries for this task. In this paper, we utilize three different libraries included HAP (Html Agility Pack), AngleSharp and MS_HtmlDocument in .Net framework.

In HAP, a DOM is first created with HTML Parser of HAP. Then, selectors allow you to reach elements from the DOM by using the XPath expression obtained from rules (see Code 1). XPath, is a W3C Recommendation, uses "path like" syntax to identify elements in

Code 1. *Extraction with HAP*

```
public List<string> Extract_with_HAP(string tagname,
string source){
string tagname_xpath = ToXPath(tagname);
List<string> list_sonuc = new List<string>();
HtmlDocument htmlDoc = new HtmlDocument();
htmlDoc.LoadHtml(source);
HtmlNodeCollection _htc =
htmlDoc.DocumentNode.SelectNodes(tagname_xpath);
if (_htc != null){
    foreach (HtmlNode node in _htc)
        list_sonuc.Add(node.InnerHtml);
}
else   return null;
return list_sonuc;
}
```

Code 2. *Converting HTML Element to XPath*

```
Public string ToXPath(string tagName){
  HtmlDocument htmlDoc = new HtmlDocument();
  htmlDoc.LoadHtml(tagName);
  HtmlNode node =
    htmlDoc.DocumentNode.SelectNodes("//*")[0];
  var attributes = node.Attributes.Any() ? "[" +
    string.Join(" and ", node.Attributes.Select(o => "@"
    + o.Name + "='" + o.Value + "'")) + "]" : "";
  return "//" + node.Name + attributes;
}
```

a web page. We have developed a method that converts HTML element to XPath expression (see Code 2). For example, <h1 itemprop="headline"> converts to XPath expression like //h1[@itemprop='headline'].

In Code 1 and 2, HtmlDocument is class of HAP. LoadHTML method loads given HTML document to instance of HtmlDocument. DocumentNode.SelectNodes method return nodes for a given HTML. In code 1, only one node is returned. node.Attributes.Select method prepare a string that contains key and value of attributes, respectively.

AngleSharp has angle brackets parser library to construct a DOM based on the official W3C specifications. AngleSharp exposes all DOM lists and allows you to use LINQ (Language-Integrated Query). LINQ is a .NET Framework component that adds native data querying capabilities to .NET languages.

Code 3. *Extraction with AngleSharp*

```
public List<string> Extract_Tag_with_AngleSharp(
string tagName, string source){
List<string[]> res =
NodenameAndAttributes(tagName);
List<string> list_sonuc = new List<string>();
var parser = new HtmlParser();
var document = parser.Parse(source);
    if (document != null){
      string[] nodename = res[0];
      List<AngleSharp.Dom.IElement> temp =
document.All.Where(m => m.LocalName ==
nodename[0]).ToList();
      for (int i = 1; i < res.Count; i++){
        string[] att = res[i];
        temp = temp.Where(m => m.Attributes[att[0]] !=
null && m.Attributes[att[0]].Value == att[1]).ToList();
      }
    if (temp != null){
      foreach (AngleSharp.Dom.IElement node in temp)
        list_sonuc.Add(node.InnerHtml);
    else return null;
    }
    else return null;
return list_sonuc;
}
```

Code 4. *Extraction with MS HtmlDocument*

```
public static List<string>
Extract_Tag_with_HTMLDocument(string tagName,
string source){
    List<string[]> res =
TagProcessing.NodenameAndAttributes(tagName);
    List<string> list_sonuc = new List<string>();
    HTMLDocument doc = new HTMLDocument();
    IHTMLDocument2 doc2 = (IHTMLDocument2)doc;
    doc2.clear();
    doc2.designMode = "On";
    doc2.write(source);
    if (null != doc){
      string[] nodename = res[0];
      nodename[0] = nodename[0].ToUpper( );
      for (int i = 1; i < res.Count; i++){
        string[] att = res[i];
        if (att[0]=="id"){
list_sonuc.Add(doc.getElementById(att[1]).innerHTML);
          return list_sonuc;
        }
    }
foreach (IHTMLElement element in
doc.getElementsByTagName(nodename[0])){
        bool sonuc = true;
        for (int i = 1; i < res.Count; i++){
        string[] att = res[i];
        if (att[0] == "class"){
          if(element.className != att[1]){
            sonuc = false;
            break;
          }
        }
        else{
          if(element.innerHTML != null){
          string tag_temp =
element.outerHTML.Substring(0,
element.outerHTML.IndexOf(">"));
              if (!(tag_temp.Contains(att[0]) &&
tag_temp.Contains(att[1]))){
                                sonuc = false;
                                break;
                }
            }
            else{
             sonuc = false;
             break;
                }
            }
        }
        if (sonuc)
          list_sonuc.Add(element.innerHTML);
      }
    }
    else return null;
return list_sonuc;
}
```

In Code 3, NodenameAndAttributes function returns node name and attributes of a tag to the string list (res). HtmlParser is main class of AngleSharp. It has methods which carry the parsed DOM. Parse function creates the DOM. All property of document.All returns all nodes that are contained in a web page and where function in document.All.Where is used for efficient extraction with LINQ statements.

MS HtmlDocument provides a wrapper around the DOM. Then, you can use HTMLDocument methods for accessing the desired element. In .NET, HtmlDocument is the base class of IHTMLDocument2 so we create doc2 instance for DOM. Write function of IHTMLDocument2 is utilized for constructing DOM. Then, in searching if attribute is ID, the function returns the result of getElementById. Otherwise, the function takes all nodes into account with the result of getElementsByTagName. In this case, if all attributes and their values are equal, the function adds this content to the list.

EXPERIMENTS

5 web pages for every 20 different online newspapers such as Milliyet, Sabah, Pravda, Corriere, Dbalears, Dnevnik, JapanTimes are used in the experiments. At least two rules (Html Element) have been prepared for each domain. Then, these rules are tested five times for the relevant domains in three different libraries. Table 2 gives information about file size of web pages and character size of rules in dataset. Table 3 indicates average result of three different libraries.

Timing performance tests are done at Core i5-3.2Ghz with 8GB memory configurations. In Table 3, creation times and extraction times of DOM are examined for three libraries. AngleSharp, allows you to use LINQ, has the best results with average 5.54 ms for preprocessing of the DOM and average 0.46 ms for extracting of a content from the DOM.

Table 2. *Information about dataset*

	Size of Files (KB)	**Number of Tags (Character Size)**
Average	142.8	33.9
Minimum	21.1	18
Maximum	905.9	127

Table 3. *Results of tree libraries*

	Preprocessing (ms)	**Extracting (ms)**
HAP	9.79	0.51
AngleSharp	5.54	0.46
MS_HtmlDocument	28.63	115.30

MS_HtmlDocument has the slowest search results with 115.30 ms because it does not have a special search function.

CONCLUSION

DOM creation and search on the DOM are important issue in terms of web content extraction. In this study, we have introduced three different libraries included HAP, AngleSharp and MS_HtmlDocument used by the .NET languages. The experimental results show that AngleSharp is a better library than the others.

We are considering using AngleSharp when we need DOM for web content extraction. All codes are an open-source and available via the github:

https://github.com/erdincuzun/SMA.NET.

ACKNOWLEDGEMENTS

The authors acknowledge the support received from the Namık Kemal University Research Fund.

REFERENCE

[1] Rahman, A.F.R., Alam, H. and Hartono, R., "Content extraction from HTML documents", International workshop on Web document Analysis, pp.7-10, 2001.

[2] Flesca, S., Manco, G., Masciari, E., Rende, E., Tagarelli, A. "Web wrapper induction: a brief survey", In: AI Communications, vol. 17, pp. 57–61. IOS Press, Amsterdam, 2004.

[3] Álvarez-Sabucedo, L. M., Anido-Rifón, L. E. and Santos-Gago, J. M., "Reusing web contents: a DOM approach", Softw: Pract. Exper., 39: 299–314. doi:10.1002/spe.901, 2009.

[4] Fu, L., Meng, Y., Xia Y. and Yu, H., "Web Content Extraction based on Webpage Layout Analysis", Second International Conference on Information Technology and Computer Science, Kiev, 2010, pp. 40-43, 2010.

[6] http://html-agility-pack.net/

[7] https://github.com/AngleSharp/AngleSharp

[8] https://msdn.microsoft.com/tr-tr/library/system.windows.forms.htmldocument(v=vs.110).aspx