Deep Web Mining Using C# Wrappers

Rakesh Kumar Baloda[1], **Praveen Kantha**[2]

[1, 2]BRCM College of Engineering and Technology, Bahal - 127028, Bhiwani, Haryana, India

Abstract: ***World Wide Web (Internet) has immense collection of information that can be extracted for building knowledge base and business intelligence purposes. Generally that valuable information lies deep inside web databases and is not accessible directly through surface web crawling methods. This information can only be accessed via a focused crawler or wrapper program customized for a particular website. The wrapper can submit a set of values for form fields and imitate user actions such as mouse click or link navigations as performed on a web browser, thus saving the response page received from a web server and can then after extract information such as table data, links, image URLs etc after parsing the DOM structure of the document. In this research paper we propose a C# crawler that can crawl a basic website and a set of related procedures (wrapper) which can extract (or mine) data from that resource by making use of regular expressions (Regex) patterns.***

Keywords: Deep Web, Crawling, Wrappers, Regular Expressions and Information Extraction

1. Introduction

A. Deep Web(Hidden Web)

The deep web, invisible web, or hidden web is a part of the World Wide Web whose contents are not indexed by standard search engines. The deep web is opposite to the surface web. It also refers to web pages that are dynamically generated from underlying databases. These web pages are generated on - the - fly (at runtime) by the scripts running on web server as a response to user query or search results. Conventional search engines cannot index the "hidden web" [9].

B. Crawling

A Web crawler is a program that automatically traverses the Web’s hyperlink structure and downloads each linked page to a local storage. Crawling (Document Retrieval) is often the first step of Web mining process. Broadly classifying, crawlers can be divided into two categories: universal crawlers and topic crawlers. A universal crawler downloads all pages irrespective of their contents, while a topic crawler downloads only pages of certain topics (relevant to user’s search query) [1].

C. Wrappers

Wrapper in data mining is a program that extracts content of a particular information source and translates it into a relational (structured) form [10].

For extracting information from a web page, a mechanism must be set up to take as inputs, the HTML source code of the web page and some method or pattern for extracting information from that code. One such method is the use of wrappers [3]. They function by given delimiters (parameters) relating to information that is to be extracted. When extracting information from a web page, these parameters will be HTML tags that surround the required information.

A web data mining wrapper can be constructed using any programming language that supports features like Network/Socket programming and String/Text processing (with Regular Expressions).

D. Regular Expressions (Regex)

A *Regular Expression (Regex)* is a pattern that the regular expression engine attempts to match in input text. A pattern consists of one or more characters, literals, operators or constructs. Regex provides a concise and flexible means for matching strings of text, such as particular characters, words or patterns of characters.

A regular expression is written in a formal language that can be interpreted by a regular expression processor, a program that serves as a Parser generator or examines text and identifies parts that match the provided specification (pattern).

Few basic patterns and their corresponding match description are as follows:

Basic Patterns Matches

\d	a single digit (0-9)
\w	alphanumeric (a-z, A-Z, 0-9 and _)
\s	tabs, spaces, newline, carriage return
.	Any one character
+	One or more of the previous character
*	Zero or more of the previous character
?	Zero or one of the previous element (Makes it optional)

Other special characters:

\	Matches the literal that follows \
^	Match must start at the beginning of line.
$	Match must occur at the end of string.
( )	Captures the matched sub-expression.
{n}	Matches the previous element n times.
[ ]	Defines a character class (matches any one of a set of characters.
|	Matches any one element. Ex: colo(r|ur)

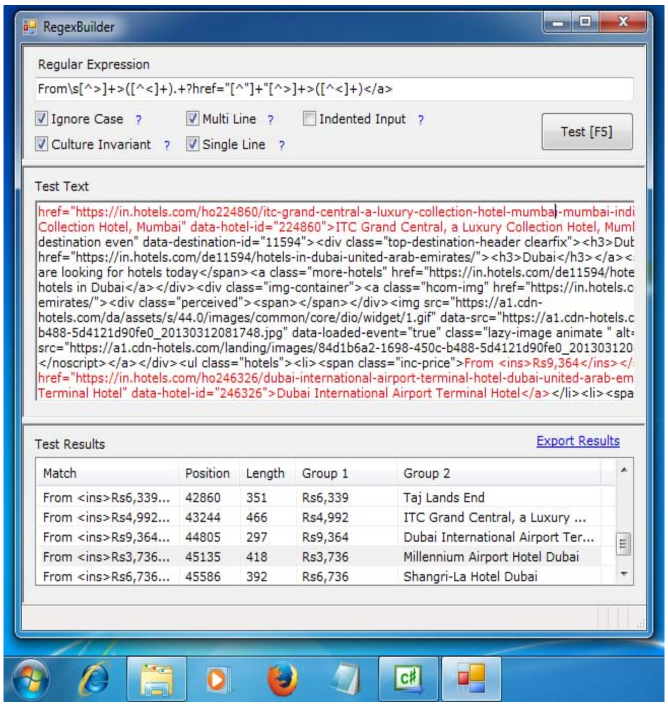
Example:

Matching a valid phone number (206) 443-6836

```
\(\d\d\d\)\s\d\d\d-\d\d\d\d
Regex: \(\d{3}\)\s\d{3}-\d{4}
```

Regular Expressions (Regex) pattern matching can be effectively used for web content mining.

We programmed a Regular Expression based data matcher and extractor using C# language. Screenshot image of the RegexBuilder application is shown as below:



Regular Expression

```
From\s[^>]+>([^<]+).+?href="[^"]+"[^>]+>([^<]+)</a>
```

☑ Ignore Case ? ☑ Multi Line ? ☐ Indented Input ?
☑ Culture Invariant ? ☑ Single Line ?

Test Text

```
href="https://in.hotels.com/ho224860/itc-grand-central-a-luxury-collection-hotel-mumbai-mumbai-indi
Collection Hotel, Mumbai" data-hotel-id="224860">ITC Grand Central, a Luxury Collection Hotel, Muml
destination even" data-destination-id="11594"><div class="top-destination-header clearfix"><h3>Dut
href="https://in.hotels.com/de11594/hotels-in-dubai-united-arab-emirates/"><h3>Dubai</h3></a><
are looking for hotels today</span><a class="more-hotels" href="https://in.hotels.com/de11594/hote
hotels in Dubai</a></div><div class="img-container"><a class="hcom-img" href="https://in.hotels.c
emirates/"><div class="perceived"><span></span></div><img src="https://a1.cdn-
hotels.com/da/assets/s/44.0/images/common/core/dio/widget/1.gif" data-src="https://a1.cdn-hotels.c
b488-5d4121d90fe0_20130312081748.jpg" data-loaded-event="true" class="lazy-image animate " alt=
src="https://a1.cdn-hotels.com/landing/images/84d1b6a2-1698-450c-b488-5d4121d90fe0_201303120
</noscript></a></div><ul class="hotels"><li><span class="inc-price">From <ins>Rs9,364</ins></
href="https://in.hotels.com/ho246326/dubai-international-airport-terminal-hotel-dubai-united-arab-em
Terminal Hotel" data-hotel-id="246326">Dubai International Airport Terminal Hotel</a></li><li><spa
```

Test Results

Match	Position	Length	Group 1	Group 2
From <ins>Rs6,339...	42860	351	Rs6,339	Taj Lands End
From <ins>Rs4,992...	43244	466	Rs4,992	ITC Grand Central, a Luxury ...
From <ins>Rs9,364...	44805	297	Rs9,364	Dubai International Airport Ter...
From <ins>Rs3,736...	45135	418	Rs3,736	Millennium Airport Hotel Dubai
From <ins>Rs6,736...	45586	392	Rs6,736	Shangri-La Hotel Dubai

Figure 1: RegexBuilder, (a Regular Expression based Data Extractor)

Such a tool is helpful in writing error free Regex patterns and quickly checking these against test data when creating a supervised wrapper for data extraction.

2. Web Data Extraction System

A web data extraction system is a software system that automatically and repeatedly extracts data from web pages with changing content and delivers the extracted data to a database or some other software application [6].

The task of web data extraction performed by such a system is usually divided into five different functions:

1) *Crawling:* (or web interaction), which comprises mainly the navigation to usually pre-determined target web pages containing the desired information and storing those resources to local system.
2) *Support for wrapper generation and execution:* where a wrapper is a program that identifies the desired data on target pages, extracts the data and transforms it into a structured format.
3) *Scheduling:* which allows repeated application of previously generated wrappers to their respective target pages.
4) *Data Transformation:* which includes filtering, transforming, refining, and integrating data extracted from one or more sources and structuring the result according to a desired output format (usually XML or relational tables); and
5) *Data Loading*: delivering the resulting structured data to external applications such as database management systems, data warehouses, business software systems, content management systems, decision support systems, RSS publishers, email servers, or SMS servers.

Alternatively, the output can be used to generate new web services out of existing and continually changing web sources.

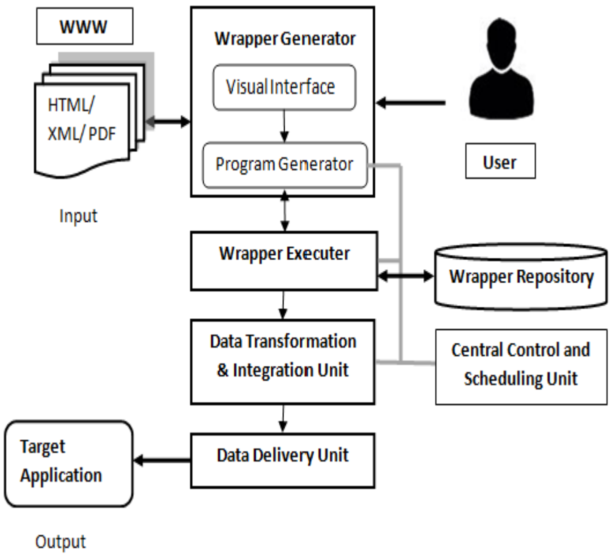


Figure 2: Architecture of a typical Web Data Extraction System

Figure 2 depicts a high-level view of a typical full-fledged semi-automatic interactive web data extraction system. This system comprises several tightly connected components and interfaces three external entities: (1) the Web, which contains pages with information of interest; (2) a target application, to which the extracted and refined data will be ultimately delivered; and (3) the user, who interactively designs the wrapper [6].

3. Steps for writing a script for web crawling.

Strategic approaches may be taken to target deep web content. With a technique called *screen scraping*, specialized software may be customized to automatically and repeatedly query a given Web form with the intension of aggregating the resulting data. Data extracted from the results of one Web form submission can be taken and applied as input to another Web form thus establishing continuity across the Deep Web in a way not possible with traditional web crawlers [6].

Here we describe a few general steps that may be followed in the order specified, to crawl a basic webpage:

(1) *Website Analysis*: First the target website needs to be analysed for :
    (a) Protocol used (http or https), the top level domain URI (Uniform Resource Identifier), Request method (GET / POST) used in the Search (Query) Form. HTTP Network Packet Analyser software such as Wireshark or HttpFox (a packet sniffer for Mozilla Firefox

browser) comes as handy tool for this as well as for studying the website navigation and pagination behaviour.

(b) Identify the key-value pairs (parameters) necessary in the Request of form submission.

(c) Observe the HTTP Request Headers of the browser, The Response Headers from the Server, type of response received (text/html, XML, JSON). This is required for making any adjustment in the Request.Accept parameters while submitting the same request via the Intelligent Agent.

(d) Check whether the target page (that contains the desired data) has been received directly by the request submission or any URL redirection (Status Code: 301/303) is involved in-between.

(2) *Request Submission via Intelligent Agent:* Submit an http request through the crawler (wrapper) as identical to the request submitted by the web browser [Analysed in step 1(b) via network monitoring tool]. Make sure that the submitted URL is in encoded form.

(3) Store the response (received from the Server) on your Local System as .HTML file or generate its Snapshot.

(4) *Comparison with the Live Site*: Compare the HTML source code of the received response with that of the live site or records captured by the Network Analyser. Make adjustments in Request Headers like User-Agent, Accepts etc (if necessary).

4. Sample code for crawler script

(1) Crawling a simple webpage :

The URI class in C# checks to see if the URL is valid.

To open a Stream, we must first obtain a WebRequest object. This object can be obtained by calling the Create function of the HttpWebRequest class. The Create function accepts a URI to specify which page to be downloaded.

From the WebRequest object we can obtain an HttpWebResponse object. The HttpWebResponse object allows us to obtain a stream by calling GetResponseStream. The below code demonstrates how this is accomplished [7].

```
Uri url = new Uri("http://www.travelocity.com");
WebRequest http = HttpWebRequest.Create(url);

HttpWebResponse response =
HttpWebResponse)http.GetResponse();

StreamReader stream =
new StreamReader(response.GetResponseStream(),
System.Text.Encoding.ASCII );

String result = stream.ReadToEnd();
Console.WriteLine( result );

response.Close();
stream.Close();
return result;
```

The StreamReader class provides the ReadToEnd function that will read until the end of the stream has been reached.

(2) Saving the HTML page to your local disk :

```
Console.WriteLine( result );
/* String variable result contains the HTML code of the
webpage just crawled using previous code. */

using (StreamWriter sw = new
StreamWriter("DownloadedPage.html"))
        {
           sw.WriteLine(result);
        }
```

Above code saves the crawled webpage as "DownloadedPage.html" (on local system) using the StreamWriter Class.

(3) Sending a POST Request: [8]

```
public static void Main ()
{
// Create a request using a URL that can receive a post.
WebRequest request =
WebRequest.Create("http://www.xyz.com/PostAccepter.
aspx");

request.Method = "POST";
// Create POST  data and convert it to a byte array.
string postData = "This  is  a test that posts this string  to a
Web server.";
byte[] byteArray = Encoding.UTF8.GetBytes(postData);
request.ContentType = "application/x-www-
formurlencoded";
// Set the ContentLength property of the WebRequest.
request.ContentLength = byteArray.Length;

// Get the request stream.
Stream dataStream = request.GetRequestStream();
// Write the data to the request stream.
dataStream.Write(byteArray, 0, byteArray.Length);
// Close the Stream object.
dataStream.Close();

// Get the response.
WebResponse response = request.GetResponse();
Console.WriteLine
(((HttpWebResponse)response).StatusDescription);
// Get the stream containing content sent by the server.
dataStream = response.GetResponseStream();
// Open the stream using a StreamReader for access.
StreamReader reader = new StreamReader(dataStream);
// Read the content.
string responseFromServer = reader.ReadToEnd();
Console.WriteLine (responseFromServer);

reader.Close ();
dataStream.Close ();
response.Close ();
 }      // End of Main method.
```

Identifying the Browser Type

The browser type can be determined from the user-agent HTTP request header. We can easily set the value of this, or any HTTP request header using the set method of the Headers collection.

```
http.Headers.Set("user-agent", "A C# Crawler");
```

Calling Sequence

A variety of operations can be performed on the HttpWebRequest and HttpWebResponse classes. However, these operations must follow a very specific order.

All request information must be set before we begin working with the response. The general order that we should follow is:

- Step 1: Obtain a HttpWebRequest object.
- Step 2: Set any HTTP request headers.
- Step 3: POST data, if this is a POST request.
- Step 4: Obtain a HttpWebResponse object.
- Step 4: Read HTTP response headers.
- Step 5: Read HTTP response data.

If we ever face a bug where it seems the request headers are being ignored, then there is a need to check whether we are not already calling a method related to the response before setting the header. All headers must be set before the request is sent.

5. General Steps for Data Extraction

Once we have the desired pages (HTML or XML) crawled to our local system, the data extraction process can be initiated.

Steps for Manual (Supervised) Wrapper Induction are as follows:

(1) Place the pages which have a common style of appearance (html visual elements) into a common directory (Extraction Folder).

(2) Select a page, view its source code using Adobe Dreamweaver (EditPlus or any other source code editor/syntax highlighter). It helps in identifying logical separators between different data element sections and records. Identify the html/xml tags which acts as a separator (generally like <hr />, </tr> or closing </div> tags).

(3) *Page Content Filtering :* Eliminate advertisement content, header – footer, navigation menu tags using the proper extraction/string processing functions available with the wrapper development tool, So that only the desired content portion is available for extraction. This step is also known n as Pre-processing before Data Extraction.

(4) *Page Chunking:* Divide the data content portion into logically separated records (blocks) such as a array of text items (String array) based on the separator tag [as Identified in Step 2]. An auto increment Chunk_id should be generated for each record chunk.

(String) Text_to_Array (source_content, <separator_tag | Regex Pattern>)

(5) Write Regular Expression (Regex Patterns) to extract individual data records (data sets).

```
For( int i = 1;i < chunks.count ;i++ )
{
Console.WriteLine(Regex.Match ( source_chunk,
<Regex_pattern>));
}
```

(6) Export the extracted records as .CSV file and Upload to database/data warehouse.

6. Overview of Data Extraction Functions

Method (or Function) for extracting String (text content) between two HTML/XML tags (represented by token1 and token2). Here Count represents that after how many occurrences of token1 we should start the lookup.

```
public String Extract(String str,
   String token1, String token2, int count)
{
   int location1, location2;
   location1 = location2 = 0;
   do
   {
   location1 = str.IndexOf(token1, location1+1);
   if (location1 == -1)
     return null;
   count--;
   } while (count > 0);

   location2 =
   str.IndexOf(token2, location1 + 1);

   if (location2 == -1)
   return null;
   location1 = location1 + token1.Length;
   return str.Substring(location1,
     location2 - location1 );
}
```

Alternatively we can make use of Regular Expressions to extract text:

(a) Text before any specified tag.

```
String regexBefore = @"(.*)" + regexPattern;
Console.WriteLine(Regex.Match( source, regexBefore,
   RegexOptions.IgnoreCase));
```

(b) After any specified tag.

```
String regexAfter = @"" + regexPattern + "(.*)";
Console.WriteLine(Regex.Match(source, regexAfter,
   RegexOptions.IgnoreCase));
```

(c) Between any two html tags.

```
String regexBetween = @"" + regexPattern + "(.*)" +
regexPattern2 ;

Console.WriteLine(Regex.Match(source, regexBetween,
   RegexOptions.IgnoreCase));
```

7. Applications of Web Data Mining

Web data mining can be used in the following sectors:
1) Business and Competitive Pricing Intelligence.
2) Publishing and Media.
3) Information Technology sector and Internet.
4) Banks, Insurance and Financial markets.
5) Market Analysis, Fraud Detection, and Customer Retention.
6) Social Web Mining (Twitter, Blog, RSS Feed), Public Administration and Legal Documents.
7) Pharmaceutical, Research and Healthcare.

8. Conclusions

This paper focuses on approaches to extract valuable information from the web (or unstructured textual information), combining methods from information retrieval and text mining.

We have also discussed a prototype of a general web data extraction system in C# language and its application in deep web mining. Such a wrapper can crawl (or fetch) and extract real time information from web (required in case of Stock Exchange or Forex Trading etc). Traditional search engines fail to provide real time vital information as they crawl, store and index the web pages periodically (on timely basis, say 15 days, 1 week or 24 hours at the most).

Our further work involves an evolution of this system into a script/query based crawler, a click - point facility for auto code generation, browser automation and data extraction system.

References

[1] Ankita Dangre, Vishakha Wankhede, Priyanka Akre, Puja Kolpyakwar Design and Implementation of Web Crawler, International Journal of Computer Science and Information Technologies (IJCSIT), Vol.5(1), 2014, 921-922
[2] T. Sunil kumar, Dr. K. Suvarchala: Web Data Mining Challenges and Application for Information Extraction - IOSR Journal of Computer Engineering (IOSRJCE), Volume 7, Issue 3 (Nov-Dec.2012), PP 24-29.
[3] Nicholas Kushmerick, Daniel S.Weld, Robert Doorenbos, "Wrapper Induction for Information Extraction", IJCAI-97.
[4] Rakesh Kumar Baloda, Praveen Kantha: Deep Web Mining and its Application in Business Intelligence, IJCSIT Vol. 7 (4), 2016, 1936-1939
[5] Bing Liu, Web Data Mining – Exploring Hyperlinks, Contents and Usage Data, Springer 2007.
[6] Robert Baumgartner, Wolfgang Gatterbauer, Georg Gottlob: Web Data Extraction System.
[7] Jeff Heaton, HTTP Programming Recipes for C# Bots (2007), Publisher : Heaton Research Inc. St. Louis
[8] How to: Send Data Using the WebRequest Class – MSDN
[9] Wikipedia(www.wikipedia.org) https://en.wikipedia.org/wiki/Deep_Web
[10] Wikipedia: https://en.wikipedia.org/wiki/Wrapper_(data_mining)

Author Profile

Rakesh Kumar Baloda received his B.E. degree in Computer Science & Engineering from BRCM College of Engineering and Technology Bahal in year 2004. After that he worked with Electrobug Technologies Ltd. and QL2 Software on Web Data Mining, Extraction and Business Intelligence projects. He is now pursuing his M.Tech in Computer Science & Engineering from BRCM College of Engineering & Technology Bahal, Bhiwani, Haryana (India). His areas of interests include web programming, real-time web data mining and extraction.

Praveen Kantha holds MCA and M.Tech degree in Computer Science & Engineering. He is currently working as an Assistant Professor in Computer Science & Engineering department of BRCM College of Engineering & Technology Bahal, Bhiwani, Haryana (India). He is passionate about teaching and research work. His areas of interests include DBMS, Data Mining, Network Security and Cryptography.