University *of Ljubljana*
Faculty *of Computer and Information Science*

Programming Assignment 2: Data Extraction

Tim Vučina

Abstract
We present our implementation for the course project and roughly outline the problems and solutions during the development. We analyze and show two general approaches to extracting data from the website as well as the automatic union-free type wrapper generation based on the RoadRunner algorithm.

Keywords
regular expressions, XPath, RoadRunner

Advisors: Slavko Žitnik

Introduction

Data extraction is the act or process of retrieving data out of (usually unstructured or poorly structured) data sources for further data processing or data storage. Typical unstructured data sources include web pages, which were also the main source for our assignment. This growing process of data extraction from the web is commonly referred to as ”Web data extraction” or ”Web scraping”. The act of adding structure to unstructured data from the websites in the form of HTML documents can be done in a number of different ways. In this assignment we explore three possible ways of data extraction or wrapper generation:

(A) Data extraction using regular expressions only.

(B) Data extraction using XPath only.

(C) Generation of extraction rules using automatic Web extraction algorithm (RoadRunner).

Web Pages Description

In order to test our algorithm implementations we chose two web pages from the same site in addition to the 4 provided examples in the assignment instructions. The site that we chose was Ceneje.si. The chosen web pages contain some common elements such as item name, top price and discount as well as a list of data records of individual offers for a particular item. The structure of the extracted data is more clearly presented in Figure 1. The data items are marked with red squares while the data records are located inside blue boxes which represent the repeating elements inside of a list.

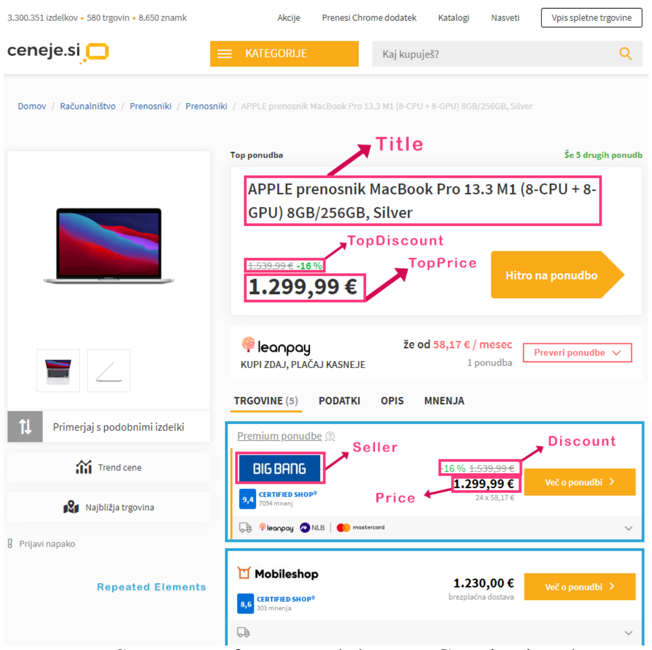


Figure 1. Structure of extracted data on Ceneje.si web page.

Regular expressions implementation

In the **A.py** script we implemented data extraction with regular expressions. It was not entirely possible to extract only text from all of the marked regions of the web pages and that is why some of the outputs include structural elements as well. The regular expressions that were used to extract data from all three web sites are listed in Table 1 in section Expression tables.

XPath implementation

Simmilarly to the regular expressions, in the **B.py** script we implemented data extraction with the use of XPath. Similar to regex implementation, the extraction of text from the web pages was not possible without some intermediate XPath expressions which returned lists of elements from which further text extraction was needed. The XPath expressions that were used to extract data from all three web sites are listed in Table 2 in section Expression tables.

RoadRunner implementation

For the automatic web extraction algorithm we chose the RoadRunner algorithm. Our implementation is located in script **C.py**. We implemented our own HTML tokenizer which is needed for the input to the RoadRunner algorithm. The RoadRunner itself was implemented recursively as presented in the paper in the instructions. The pseudocode for the algorithm is shown below.

Algorithm 1 RoadRunner

```
$A, B \leftarrow$ Tokenize $A, B$
while not end of tokens do
  if $TAG$ mismatch then
    $match \leftarrow try\_match\_iterator(A, B)$
    if $match$ was found then
      $wrapper \leftarrow generalize\_wrapper(match)$
    else
      $match \leftarrow try\_match\_optional(A, B)$
      $wrapper \leftarrow wrapper + match$
    end if
  else if $STRING$ mismatch then
    $wrapper \leftarrow wrapper + \#PCDATA$
  else $\{A$ matches $B\}$
    $wrapper \leftarrow wrapper + A$
  end if
  $A \leftarrow next(A)$
  $B \leftarrow next(B)$
end while
```

The *try_match_iterator()* function consists of two steps:

1. Square Location by Terminal–Tag Search
2. Square Matching

The terminal tag search is simply performed as a walk along the token list to the nearest terminal tag. The terminal tag is assumed to be the preceding token to the mismatch location.

Here we also had to accommodate for there being another opening tag of the terminal tag before the actual terminal tag which marks the end of the square candidate. We solved the problem by tagging the tokens as *open* or *close* type of tags and than looking for one that correctly matches the terminal tag.

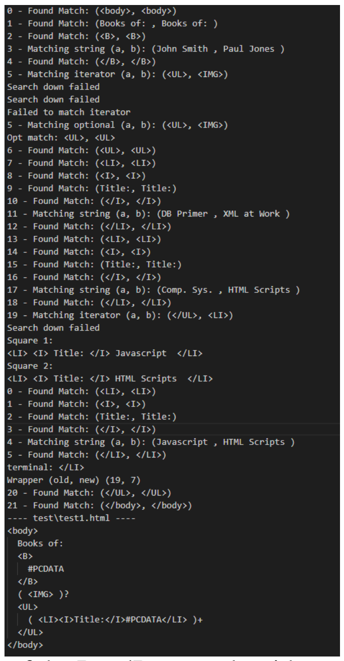
The square matching was done by first identifying where the upper square candidate starts and then recursively calling the RoadRunner algorithm with lists of tokens containing only the square parts.

The algorithm was extremely difficult to implement so that it would work flawlessly even on larger web pages. It seems that it misses a lot of important information or structure due to elements not being positioned according to its assumption.

For example the iterator matching assumes that the preceding token is from where the mismatch occurs is the terminal tag. This assumption makes it hard to identify iterator elements if there is some breaking element in between them in a list.

It also compares the found square candidate only with the previous similar consecutive element which means that information about optional elements or other mismatches in previous elements is lost after the wrapper is generalized with the first found match. Another disadvantage is quickly present where a mismatch occurs between a string and a tag, where the most fitting solution seems to be that we treat the tag as an optional element but of course this is not always the case. The iterator often fails to recognize the right portion of the list as the repeating element and instead takes a part of the end of the previous element an a portion of the next element as the matching square. This often leads to undesired outputs that bear no real information about the difference between the pages.

During the development of the algorithm we had to make a lot of minor adjustments and assumptions about the structure of the documents which are not universal in terms of what the proper HTML structure enforces. The intermediate steps of the algorithm are shown in Figure 2.



```
0 - Found Match: (<body>, <body>)
1 - Found Match: (Books of: , Books of: )
2 - Found Match: (<B>, <B>)
3 - Matching string (a, b): (John Smith , Paul Jones )
4 - Found Match: (</B>, </B>)
5 - Matching iterator (a, b): (<UL>, <IMG>)
Search down failed
Search down failed
Failed to match iterator
5 - Matching optional (a, b): (<UL>, <IMG>)
Opt match: <UL>, <UL>
6 - Found Match: (<UL>, <UL>)
7 - Found Match: (<LI>, <LI>)
8 - Found Match: (<I>, <I>)
9 - Found Match: (Title:, Title:)
10 - Found Match: (</I>, </I>)
11 - Matching string (a, b): (DB Primer , XML at Work )
12 - Found Match: (</LI>, </LI>)
13 - Found Match: (<LI>, <LI>)
14 - Found Match: (<I>, <I>)
15 - Found Match: (Title:, Title:)
16 - Found Match: (</I>, </I>)
17 - Matching string (a, b): (Comp. Sys. , HTML Scripts )
18 - Found Match: (</LI>, </LI>)
19 - Matching iterator (a, b): (</UL>, <LI>)
Search down failed
Square 1:
<LI> <I> Title: </I> Javascript  </LI>
Square 2:
<LI> <I> Title: </I> HTML Scripts  </LI>
0 - Found Match: (<LI>, <LI>)
1 - Found Match: (<I>, <I>)
2 - Found Match: (Title:, Title:)
3 - Found Match: (</I>, </I>)
4 - Matching string (a, b): (Javascript , HTML Scripts )
5 - Found Match: (</LI>, </LI>)
terminal: </LI>
Wrapper (old, new) (19, 7)
20 - Found Match: (</UL>, </UL>)
21 - Found Match: (</body>, </body>)
---- test\test1.html ----
<body>
  Books of:
  <B>
    #PCDATA
  </B>
  ( <IMG> )?
  <UL>
    ( <LI><I>Title:</I>#PCDATA</LI> )+
  </UL>
</body>
```

Figure 2. Run of the RoadRunner algorithm on a test example.

Overstock.com	
Title:	`/(?<=<a.*?<b>).*?\n.*?(?=<\/b><\/a><br>)/`
Content:	`/(?<="normal">).*?(?=<br>)/s`
ListPrice:	`/(?<=<s>)\$\d*,?\d*\.?\d*/`
Price:	`/(?<=<b>)\$\d*,?\d*\.?\d*/`
Saving:	`/(?<="littleorange">)\$\d*,?\d*\.?\d*/`
SavingPercent:	`/\(\d*%\)/`
Rtvslo.si	
Title:	`/(?<=news-container.*?<h1>).*(?=<\/h1>)/s`
Subtitle:	`/(?<=subtitle">).*?(?=<\/)/`
Lead:	`/(?<=lead">).*?(?=<\/)/s`
Content:	`/(?<=<article.*?<p.*?>).*?(?=<div class="gallery")/s`
Author:	`/(?<=author-name">).*?(?=<\/)/`
PublishedTime:	`/(?<=publish-meta">\n\t*?)\d.*?(?=<br)/`
Ceneje.si	
Title:	`/(?<=<h1 class="top-offer__name">\n\s).*?(?=\n\s<\/h1>)/s`
TopPrice:	`/(?<=top-offer__grid.*?priceB">\n\s).*?(?=\n\s<\/div>)/s`
TopDiscount:	`/(?<=top-offer__discount">\n\s).*?(?=\n\s<\/div>)/s`
Seller:	`/(?<=data-sellername=").*?(?=")/`
Price:	`/(?<=data-cclass.*)\d*\.?\d{1,3},\d{2}\s.\n/s`
Discount:	`/(?<=discountPrice">).*?(?=<\/div>)/s`

Table 1. Regular expressions that were used in our implementation.

Overstock.com	
Title:	`//tr[@bgcolor]//a/b/text()`
Content:	`//tr[@bgcolor]//span[@class="normal"]/text()`
ListPrice:	`//tr[@bgcolor]//s/text()`
Price:	`//tr[@bgcolor]//span[@class="bigred"]/b/text()`
Saving:	`//tr[@bgcolor]//span[@class="littleorange"]/text()`
SavingPercent:	`//tr[@bgcolor]//span[@class="littleorange"]/text()`
Rtvslo.si	
Title:	`//header//h1/text()`
Subtitle:	`//header//*[@class="subtitle"]/text()`
Lead:	`//header//*[@class="lead"]/text()`
Content:	`//article//p/text()`
Author:	`//*[@class="author-name"]/text()`
PublishedTime:	`//*[@class="publish-meta"]/text()`
Ceneje.si	
Title:	`//h1[@class="top-offer__name"]/text()`
TopPrice:	`//*[contains(@class, "priceB")]/text()`
TopDiscount:	`//*[@class="top-offer__discount"]/*/text()`
Seller:	`//@data-sellername`
Price:	`//*[@class="price-value"]/text()[2]`
Discount:	`//*[@class="discountPrice"]`

Table 2. XPath expressions that were used in our implementation.

Results

The output wrappers for each web page pair are shown below. In order to fit them in the report only the most important parts of the outputs are shown. Whole outputs are available on Github repository inside the outputs folder.

0.1 Test

```
<body>
  Books of:
  <B>
    #PCDATA
  </B>
  ( <IMG> )?
  <UL>
    ( <LI><I>Title:</I>#PCDATA</LI> )+
  </UL>
</body>
```

0.2 Overstock

```
<body><input>
    ( <form></form> )?
    <table><tbody>
    ...
    <img></td></tr>
        ( <form></form> )?
    <tr><td><table><tbody
    ...
    <span>Search:</span>
    <select><option>#PCDATA</option><option>#PCDATA</option> ...
    <table><tbody>
        ( <tr><td></td><td><a>#PCDATA</a></td></tr> )+
    </tbody></table>
    ...
    ( <tr><td><span><b>#PCDATA</b></span></td><td><b>#PCDATA</b>#PCDATA( <b>#PCDATA</b> )+
    ( <a>Discount</a> )?( <b>296</b> )?( <a>Newest First</a> )?
    ( <a>First Page</a> )?( <a>Price</a> )?( <a>Previous 15</a> )?
    ( <a>Quantity</a> )?( <a>Next 15</a> )?( <a>Markdowns</a> )?</td></tr> )+
    ...
    ( <tr><td><table><tbody><tr><td><a><img></a></td></tr>
    <tr><td><a>More Info...</a></td></tr></tbody></table></td>
    <td><a><b>#PCDATA</b></a><br><table><tbody><tr><td><table>
    <tbody><tr><td><b>List Price:</b></td><td><s>#PCDATA</s></td></tr>
    <tr><td><b>Price:</b></td><td><span><b>#PCDATA</b></span></td></tr>
    <tr><td><b>You Save:</b></td><td><span>#PCDATA</span></td></tr>
    </tbody></table></td><td><span>#PCDATA<br><a><span><b>Click here to purchase.</b>
    </span></a></span><br></td></tr></tbody></table></td></tr> )+
    ...
    ( <tr><td><span><b>#PCDATA</b></span></td><td><b>#PCDATA</b>#PCDATA( <b>#PCDATA</b> )+
    ( <a>Discount</a> )?( <b>296</b> )?( <a>Newest First</a> )?( <a>First Page</a> )?
    ( <a>Price</a> )?( <a>Previous 15</a> )?( <a>Quantity</a> )?( <a>Next 15</a> )?
    ( <a>Markdowns</a> )?</td></tr> )+
    ...
    </map>
</body>
```

0.3 Rtvslo

```
<body><div><div><div>
...
( <span></span><span></span><span></span><span></span></div></div><div></div><div><div> )?
<div><div>
( </div>...<a>Ture avanture </a></span><span><a>Kulinarika </a></span><span><a>Lepota bivanja </a
    ></span><span><a>Avtomobilnost </a></span><span><a>Moda </a></span><span><a>196x ljubezen </a
    ></span><span>3. 4. 2019 | 15.25</span>...<div> )?
...
</div><div><div><strong>
  Miha Merljak
```

```
</strong>
#PCDATA
</div><div>
Ljubljana - MMC RTV SLO
</div></div>
( <figure><a><img></a><figcaption><span></span>XC40 se v mestu ...</p></header><div> )?
( <figure><a><img></a><figcaption><span></span>Audi A6 ...</figcaption></figure> )?
...
<div>
  ( <div>Miha Merljak</div> )?
  ( <br>Ljubljana - MMC RTV SLO  )?
</div><div>
  ( <div></div> )?
  ( <br>Ljubljana - MMC RTV SLO </div><div> )?
  ( <div></div> )+
 ...
 ( <img></div></figure><p>Volvo se je nizjih ...><div> )?
      <div>
        ( <a><img></a></div><div><img></div></div></figure><p></p><p>Samo poglejte njegovo masko to
             ogromno ...<div> )?
        ( <figure><a><img></a><figcaption>MMC RTV SLO </figcaption></figure> )+
     </div><div></div><div></div></div><div><figure><a><img>
    <figcaption>
    #PCDATA
    </figcaption></figure><figure><a><img></a><figcaption>
    #PCDATA
    </figcaption>
...
 ( <a>#PCDATA</a> )+
...
( </div><div><h3><svg><g><circle>...</circle></svg><a>Avtomobilnost</a></h3></div><div> )?
<div><div><div><a><img></a><h3><span><a>
<a>Avtomobilnost</a></span>
<a>#PCDATA</a></h3>
<p>#PCDATA</p>
...
</iframe><img></body>
```

0.4 Ceneje

```
<body><div><main></main></div><div><div><nav><div><span><a>
    Akcije
    </a></span><span><a>
    Prenesi Chrome dodatek
    </a></span><span><a>
...
<a>
  Racunalnistvo
</a>
/
<a>
  Prenosniki
</a>
/
<a>
  Prenosniki
</a>
/
<span>
  #PCDATA
</span>
...
<div><img></div><div><div>
    ( </div><div> )?
    <div>
      ( <div><a><img></a></div> )+
    </div></div>
  ( <div></div> )?
</div></div><div><a><i></i><span>
```

```
    Primerjaj s podobnimi izdelki
  </span>
...
Top ponudba
</div><div>
    #PCDATA
</div></div><div><h1>
    #PCDATA
</h1><div><div><div><div><span>
    #PCDATA
</span><span>
    #PCDATA
</span></div><div>
    #PCDATA
</div></div></div><div><a><span>
Hitro na ponudbo
...
<table><tbody>
    ( <tr><td><a><img></a></td><td>
        <a><span>ze od <span>#PCDATA</span></span></a>
    </td><td><a><button>Vec o ponudbi <i></i></button></a></td></tr> )+
</tbody></table>
...
Trgovine
<span>
    #PCDATA
</span></a></li><li><a>
    Podatki
</a></li><li><a>
    Opis
</a></li><li><a>
    Mnenja
...
</span></td><td><a>
#PCDATA
</a></td></tr><tr><td><span>
Barva
</span></td><td><a>
#PCDATA
</a></td></tr><tr><td><span>
Dodan operacijski sistem
</span></td><td><a>
#PCDATA
</a></td></tr><tr><td><span>
Model graficne kartice
</span></td><td><a>
#PCDATA
...
( <span>Velikost zaslona (v ")</span></td><td><a>17.3 "</a></td></tr><tr><td><span>Vrsta naprave</
    span></td><td><a>gaming</a></td></tr><tr><td> )?
( </div><div><span><a><b>Prijavi napako</b></a></span></div><p>Ceneje.si je vodilni portal ... ce
    zelis kvaliteten izdelek po najboljsi ceni. )?
( <br> )?
...
<body>
```