Data Extractor

Timotej Kovač

I. Introduction

In this paper we provide and discuss results gathered when trying to implement data extraction algorithms using regular expressions, XPath and automatic web extraction. The later was implemented using RoadRunner approach for automatic web extraction [1], though the algorithm does not strictly follow their implementation approach but rather uses their general observations.

II. Selection of optional web pages

For our first website we have chosen a detailed description page of a product available on web store Mimovrste [2]. Here we have chosen some interesting fields that we might be useful as shown in figure 1. These were:

- **TAGS***, which further describe the item as having a discount, being a recommended product, etc.";
- **TITLE**;
- **DESCRIPTION**;
- **OLD PRICE***, which states the price before the now discounted price;
- **PRICE**, which states the current price;
- **SAVINGS***, which represents the percentage saved;
- **AVAILABILITY**, which states when the product will be available for shipment.

Fields above marked with an asterisk don't appear always and are therefore optional.

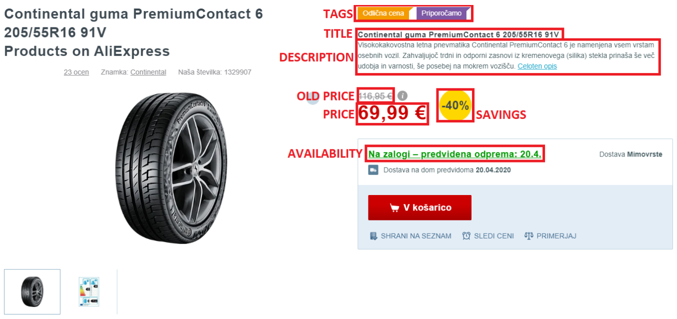
Continental guma PremiumContact 6 205/55R16 91V
Products on AliExpress
TAGS
Odlična cena
Priporočamo
TITLE
Continental guma PremiumContact 6 205/55R16 91V
DESCRIPTION
OLD PRICE
PRICE
69,99 €
-40%
SAVINGS
AVAILABILITY
Na zalogi – predvidena odprema: 20.4.
V košarico

Figure 1. Web store mimovrste.si with tagged fields that we used for extraction of web content.

For our second website we have chosen a page containing multiple items in a grid pattern on a web site Ceneje [3]. Here we have chosen the fields listed bellow:

- **IMAGE**;
- **TITLE**;
- **MIN PRICE**, which states the minimal price in all of the stores that provide the product;
- **NUMBER OF STORES**, which states the number of stores that provide the product;
- **ACTION**, which states what the button does either takes the user to a particular web store or to a list of web stores still on the same ceneje.si domain.

Here none of the fields are optional but some do vary as they may contain some other words in front of the fields or are ads which have a slightly different structure. This too can be seen in figure 2.

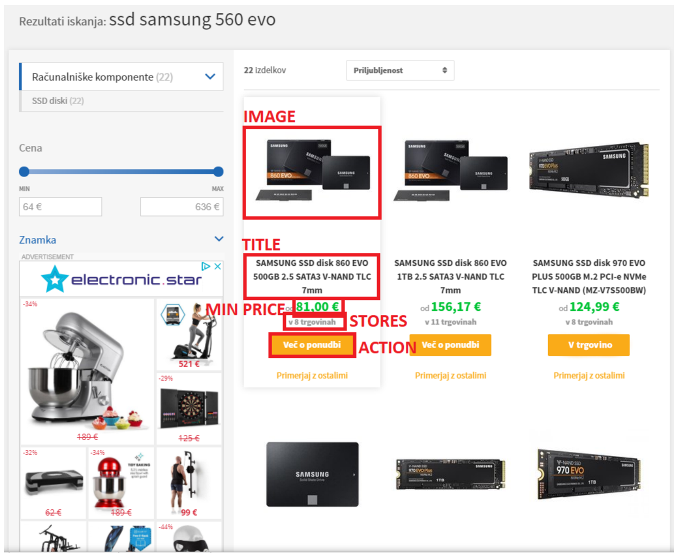
Rezultati iskanja: ssd samsung 560 evo
Računalniške komponente (22)
Cena
Znamka
22 izdelkov
Priljubljenost
IMAGE
TITLE
SAMSUNG SSD disk 860 EVO 500GB 2.5 SATA3 V-NAND TLC 7mm
SAMSUNG SSD disk 860 EVO 1TB 2.5 SATA3 V-NAND TLC 7mm
SAMSUNG SSD disk 970 EVO PLUS 500GB M.2 PCI-e NVMe TLC V-NAND (MZ-V7S500BW)
MIN PRICE
81,00 €
156,17 €
124,99 €
STORES
ACTION
Več o ponudbi
V trgovino
Primerjaj z ostalimi

Figure 2. Web site ceneje.si with tagged fields that we used for extraction of web content.

III. Regular Expressions Implementation

Bellow is the list of all regular expressions that were used on the target web pages.

A. *rtvslo.si*

```
Title:  <h1>(.*?)</h1>
SubTitle: <div class=\"subtitle\">(.*?)</div>
Lead: <p class=\"lead\">(.*?)</p>
Content: <div class=\"article-body\">(.*?)</div>[ ]
*<div class=\"article-column\">
Author:  <div class=\"author-name\">(.*?)</div>
PublishedTime: <div class=\"publish-meta\">(.*?)<br>
```

B. *overstock.com*

```
Title(s): <td valign=\"top\">\W*<a.*?PROD_ID=([0-9]+)\"
.*?<b>(.*?)</b></a>
Content(s): <td valign=\"top\">\W*<a.*?PROD_ID=([0-9]+)
\".*?<span class=\"normal\">(.*?)<br>
ListPrice(s): <td valign=\"top\">\W*<a.*?PROD_ID=
([0-9]+)\".*?<s>(.*?)</s>
Price(s): <td valign=\"top\">\W*<a.*?PROD_ID=([0-9]+)
\".*?<span class=\"bigred\"><b>(.*?)</b>
Saving(s):
SavingPercent(s): <td valign=\"top\">\W*<a.*?PROD_ID=
([0-9]+)\".*?<span class=\"littleorange\">(.*?) \((
[0-9]{0,2}\%)\)</span>
```

C. *mimovrste.si*

```
Title: <h3.*?>(.*?)</h3>
Description: <p.*?itemprop=\"description\".*?>(.*?)<a
OldPrice: <del.*?class=\"rrp-price\".*?>(.*?)</del>
Price: <b class=\"pro-price.*?>(.*?)</b>
```

```
Availability: <a data-sel=\"availability-detail\".*?>
(.*?)</a>
Tags: <em class=\"label.*?>(.*?)</em>
Savings: <div class=\"label--round-sale.*?>(.*?)</div>
```

D. ceneje.si

```
Image(s): <div class=\"innerProductBox\">.*?<img.*?
alt=\"(.*?)\".*?src=\"(.*?)\"
Titles(s): <div class=\"innerProductBox\">.*?<img.*?
alt=\"(.*?)\".*?<h3>\W*<.*?>(.*?)</.*?>
MinPrice(s): <div class=\"innerProductBox\">.*?<img.*?
alt=\"(.*?)\".*?<b>(.*?)</b>
Store(s): <div class=\"innerProductBox\">.*?<img.*?
alt=\"(.*?)\".*?class=\"qtySellers\">\W*<b>(.*?)</b>
Action(s): <div class=\"innerProductBox\">.*?<img.*?
alt=\"(.*?)\".*?<div class=\"rBox\">\W*<.*?>(.*?)</.*?>
```

IV. XPath Implementation

Bellow are the lists of XPath expressions for every of the targeted pages.

A. rtvslo.si

```
Title: //*[@id=\"main-container\"]/div[3]/div/header/
h1/text()
SubTitle: //*[@id=\"main-container\"]/div[3]/div/
header/div[2]/text()
Lead: //*[@id=\"main-container\"]/div[3]/div/header/
p/text()
Content: string(//*[@id=\"main-container\"]/div[3]/div/
div[2])
Author: //*[@id=\"main-container\"]/div[3]/div/div[1]/
div[1]/div/text()
PublishedTime: //*[@id=\"main-container\"]/div[3]/div/
div[1]/div[2]/text()[1]
```

B. overstock.si

```
Title(s):  /html/body/table[2]/tbody/tr[1]/td[5]/
table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/
tr[" + str(i) + "]/td[2]/a/b/text()
Content(s): /html/body/table[2]/tbody/tr[1]/td[5]/
table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/
tr[" + str(i) + "]/td[2]/table/tbody/tr/td[2]/span/text()
ListPrice(s): /html/body/table[2]/tbody/tr[1]/td[5]/
table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/
tr[" + str(i) + "]/td[2]/table/tbody/tr/td[1]/table/
tbody/tr[1]/td[2]/s/text()
Price(s): /html/body/table[2]/tbody/tr[1]/td[5]/
table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/
tr[" + str(i) + "]/td[2]/table/tbody/tr/td[1]/table/
tbody/tr[2]/td[2]/span/b/text()
Saving(s):
SavingPercent(s): /html/body/table[2]/tbody/tr[1]/td[5]/
table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/
tr[" + str(i) + "]/td[2]/table/tbody/tr/td[1]/table/
tbody/tr[3]/td[2]/span/text()
```

C. mimovrste.si

```
Title: //*[@id=\"content\"]/div/article/div[1]/
section[2]/h3/text()
Description: //*[@id=\"content\"]/div/article/div[1]/
section[2]/p[2]/text()
OldPrice: //*[@id=\"content\"]/div/article/div[1]/
section[2]/div[3]/div[1]/div[1]/div/del/text()
Price: //*[@class=\"price-wrapper\"]/div[1]/
div[1]/b/text()
Availability: //*[@class=\"delivery-wrapper\"]/a/
text()
Tags: //*[@id=\"content\"]/div/article/div[1]/
section[2]/p[1]/em[" + str(i) + "]/text()
Savings: //*[@id=\"content\"]/div/article/div[1]/
section[2]/div[3]/div[1]/div[2]/text()
```

D. ceneje.si

```
Image(s): //*[@id=\"productGrid\"]/div[" + str(i) + "]
/div/div[1]/a/img/@src
Title(s): //*[@id=\"productGrid\"]/div[" + str(i) + "]
/div/div[2]/h3/a/text()
MinPrice(s): //*[@id=\"productGrid\"]/div[" + str(i) + "]
/div/div[2]/p/a[1]/b/text()
Store(s): //*[@id=\"productGrid\"]/div[" + str(i) + "]
/div/div[2]/p/a[2]/b/text()
Action(s): //*[@id=\"productGrid\"]/div[" + str(i) + "]
/div/div[3]/a/text()
```

V. Automatic Web Extraction Implementation

A. Pseudo code

In our approach we first generated DOM structures from both input HTML pages. Then we cleaned them by removing any <head>, <style> and <script> tags. After that the auto_ex function was called to produce a union–free regular expression. When checking for node mismatches we mainly relied on the type of tag the node represented, the id of the node, if it didn't contain more than 3 numbers which generally described an ad node, and sequential attributes of the tag to determine if the two nodes in separate trees are the same. The pseudo code of the function is described bellow.

```
def auto_ex(tree1, tree2):
for child in tree1:

        # Check if there is any node on count position
 in tree2
        if count >= len(node_2.getchildren()):
            wrapper += "(<" + str(child.tag).upper()
+ "... >)?"
            continue

        target = node_2.getchildren()[count]

        # Handle tag mismatches
        if mismatch(child, target):
            if contains(node_2, count, child):
                wrapper += "(<" + target.tag.upper()
+ "... >)?"
                count += 1
                target = node_2.getchildren()[count]
            else:
                wrapper += "(<" + str(child.tag).upper()
 + "... >)?"
                continue
        # Add starting tag
        new_tag = "<" + str(child.tag).upper() + ">"
        # Add text inside of the tag
        if child.text is not None:
            new_tag += get_smt(child.text, target.text)

        # Handle children of this node
        new_tag += auto_ex(child, target)

        # Add a closing tag
        new_tag += "</" + str(child.tag).upper() + ">"

        # Handle text after the current node
        if child.tail is not None:
            tail = child.tail
            if len(tail) > 0:
                new_tag += get_smt(tail, target.tail)

        # Replace multiple occurrences with a special
tag
        if new_tag == prev_tag:
            wrapper = re.sub(new_tag + "$", "(" +
new_tag + ")+", wrapper)
            continue

        wrapper += new_tag

        prev_tag = new_tag
        count += 1

        # If this is the last node in tree1 but there are
 still nodes in tree2 this one must be optional
        if node_exists(tree2, level, count):
            wrapper += "(<" + node_that_exists_in_tree2
 + "... >)?"

    return wrapper
```

B. Results

C. rtvslo.si

`%TODO: Results`

D. overstock.com

`%TODO: Results`

E. mimovrste.si

`%TODO: Results`

F. ceneje.si

`%TODO: Results`

VI. Conclusion

We were successful in implementing all of the above listed approaches and have gathered good results. Our automatic web extraction approach could of course be improved with additional information gathered from other articles that tackle with the same problems or with testing the algorithm on more websites.

References

[1] V. Crescenzi, G. Mecca, P. Merialdo *et al.*, “Roadrunner: Towards automatic data extraction from large web sites,” in *VLDB*, vol. 1, 2001, pp. 109–118.

[2] “mimovrste=) | računalništvo, prenosniki, GSM telefoni, avdio-video,” Accessable: https://www.mimovrste.com/, 2020, [Accessed: 24. 04. 2020].

[3] “Ceneje.si - Prva misel pred nakupom,” Accessable: https://www.ceneje.si/, 2020, [Accessed: 24. 04. 2020].