Seminarska naloga 2

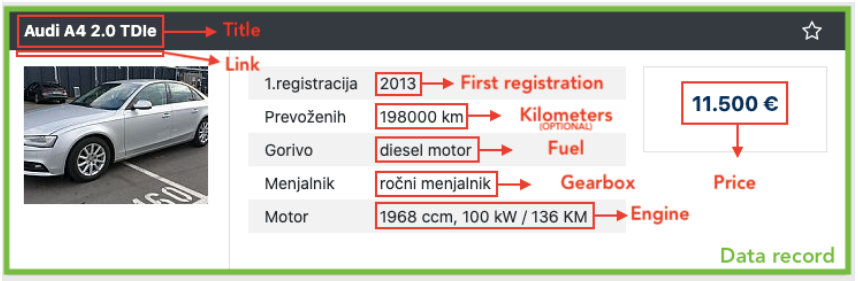
Neža Bačar, Ana Čepuran, Bojan Vrangeloski

1. Uvod

Cilj druge seminarske naloge pri predmetu Iskanje in ekstrakcija podatkov s spleta je bil implementacija treh različnih pristopov za pridobivanje strukturiranih podatkov s spleta. Implementirani pristopi so:

A. Uporaba regularnih izrazov,
B. Uporaba XPath izrazov,
C. Implementacija algoritma podobnega RoadRunner-ju.

Pridobiti smo morali podatke iz šestih spletnih strani (po dve podobni spletni strani na domeno), od katerih so bile štiri določene v naprej (dve spletni strani domene Overstock in dve spletni strani domene RTVSLO), poleg teh pa smo izbrali še dve spletni strani domene Avtonet. Gre za spletni strani, ki vsebujeta različno dolg seznam avtomobilov, ki predstavljajo iskane elemente s pripadajočimi podatki. Ena stran vsebuje 48, druga pa 99 zadetkov.



Slika 1: Primer iskanih podatkov na spletnih straneh domene Avtonet

2. Implementacija metod

2.1 Regularni izrazi (ang. Regular Expressions)

Za spletno domeno ***Overstock*** smo uporabili naslednje (glej spodaj) regularne izraze. Najprej smo definirali regularne izraze za vsak iskani podatek posebaj, nato pa smo jih z ustreznimi vmesnimi regularnimi izrazi povezali v skupen regularni izraz, ki poišče vse omenjene podatke hkrati.

```
title_regex = "<b>(.+-[kK]t[^<]+)</b>"

list_price_regex = "<b>List Price:</b></td><td align=\"left\"
nowrap=\"nowrap\"><s>([^<]+)</s>"

price_regex = "<span class=\"bigred\"><b>([^<]+)</b></span>"

saving_regex = "<span class=\"littleorange\">([$][^<]+)</span>"

content_regex = "<span class=\"normal\">([^<]+)"


regex = title_regex + "</a><br>[\s]+<table><tbody><tr><td
valign=\"top\"><table>[\s]+<tbody><tr><td align=\"right\" nowrap=\"nowrap\">" + \
```

```
        list_price_regex + "</td></tr>[\s]+<tr><td align=\"right\"
nowrap=\"nowrap\"><b>Price:</b></td><td align=\"left\" nowrap=\"nowrap\">" + \

        price_regex + "</td></tr>[\s]+<tr><td align=\"right\" nowrap=\"nowrap\"><b>You
Save:</b></td><td align=\"left\" nowrap=\"nowrap\">" + \

        saving_regex + "</td></tr>[\s]+</tbody></table>[\s]+</td><td valign=\"top\">" + \

        content_regex
```

Za spletno domeno ***RTV SLO*** smo uporabili naslednje (glej spodaj) regularne izraze, ki smo jih z ustreznimi vmesnimi izrazi povezali v skupen regularni izraz, ki poišče vse omenjene podatke hkrati.

```
title_regex = r"<h1>([^<]+)</h1>[\s]+<div class=\"subtitle\">([^<]+)</div>[\s]+<div
class=\"article-meta-mobile\">[\s]+"

author_regex = r"<div class=\"author-timestamp\">[\s]+<strong>([^<]+)</strong>([^<]+)</div>"

lead_regex = r"<p class=\"lead\">([^<]+)</p>"

content_regex = r"<\/div>[\s]*?<\/figure>[\s]*?<p([\s\S]*?)<div class=\"gallery\">"

regex = title_regex + author_regex + r"[\s\S]*" + lead_regex + r"[\s\S]*" + content_regex
```

Za spletno domeno ***Avtonet*** smo uporabili naslednje (glej spodaj) regularne izraze, katerih rezultate smo dodajali iterativno v posamezen iskani element oziroma *data record*.

```
link_regex = "<a class=\"stretched-link\" href=\"([^\"]+)\"></a>"

title_regex = "<div class=\"GO-Results-Naziv [^\"]+\">[\s]+<span>([^<]+)<\/span>[\s]+</div>"

data_regex = "<tbody>[\s]*<tr>[\s]+<td class=\"w-25 d-none d-md-block pl-
3\">[^<]+</td>[\s]+<td class=\"w-75 pl-3\">([^<]+)" \

             "</td>[\s]+</tr>(<tr>[\s]+<td class=\"d-none d-md-block pl-3\">[^<]+</td>[\s]+<td
class=\"pl-3\">([^<]+)</td>[\s]+" \

             "</tr>)?<tr>[\s]+<td class=\"d-none d-md-block pl-3\">Gorivo</td>[\s]+<td
class=\"pl-3\">([^<]+)</td>[\s]+</tr><tr>[\s]+" \

             "<td class=\"d-none d-md-block pl-3\">Menjalnik</td>[\s]+<td class=\"pl-3 text-
truncate\">([^<]+)</td>[\s]+" \

             "</tr><tr class=\"d-none d-md-table-row\">[\s]+<td class=\"d-none d-md-block pl-
3\">Motor</td>[\s]+<td class=\"pl-3 text-truncate\">[\s]+" \

             "([^<]+)</td>[\s]+</tr>[\s]+</tbody>[\s]+</table>"

price_regex = "<div class=\"GO-Results-(Top-)?Price-TXT-[^BbSsTt\"]*\">([^<]+)</div>"
```

2.2 Xpath

Za spletno domeno ***Overstock*** smo uporabili naslednje (glej spodaj) XPath izraze, kjer smo s spremenljivko ***i*** definirali iskani element na strani (gre za seznam elementov).

```
xpath_title = '/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()'

xpath_content = '/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()'

xpath_list_price = '/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()'

xpath_price = '/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()'

xpath_saving = '/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()'
```

Za spletno domeno ***RTV SLO*** smo uporabili naslednje (glej spodaj) XPath izraze. Gre za en sam iskani element na strani, zato v tem primeru uporaba spremenljivke ***i*** ni bila potrebna.

```
xpath_author = '//*[@id="main-container"]/div[3]/div/header/div[3]/div[1]/strong/text()')

xpath _published_time = '//*[@id="main-container"]/div[3]/div/header/div[3]/div[1]/text()')

xpath _title = '//*[@id="main-container"]/div[3]/div/header/h1/text()')

xpath _subtitle = '//*[@id="main-container"]/div[3]/div/header/div[2]/text()')

xpath _lead = '//*[@id="main-container"]/div[3]/div/header/p/text()')

xpath_content = '//*[@id="main-container"]/div[3]/div/div[2]/article//p//text()')
```

Za spletno domeno ***Avtonet*** smo uporabili naslednje (glej spodaj) XPath izraze, kjer smo s spremenljivko ***i*** definirali iskani element na strani (gre za seznam elementov), s spremenljivko ***j*** pa smo se sprehodili čez iskane podatke o elementu* (prva registracija, gorivo, motor, prvoženi kilometri (glej Sliko 1).

```
title_xpath = "//*[@id=\"results\"]/div[" + str(i) + "]/div[1]/span/text()"

price_xpath = "//*[@id=\"results\"]/div[" + str(i) + "]/div[6]/div[1]/div[1]/div/text()"

link_xpath = "//*[@id=\"results\"]/div[" + str(i) + "]/a/@href"
field_xpath = "//*[@id=\"results\"]/div["+str(i)+"]/div[4]/div/table/tbody/tr["+str(j)+"]/td[1]/text()"
data_xpath = "//*[@id=\"results\"]/div["+str(i)+"]/div[4]/div/table/tbody/tr[" +str(j)+ "]/td[2]/text()"
```

2.3 Avtomatska ekstrakcija podatkov (RoadRunner)

Implementacije avtomatske ekstrakcije podatkov nam ni uspelo implementirati do konca.

Pred zagonom metode roadrunner smo HTML datoteko (oziroma besedilo) ustrezno obdelali in odstranili vse nepotrebne token-e, kot je prikazano spodaj.

```
def clean_file(file):

    cleaner = Cleaner()

    cleaner.javascript = True  # we want to activate the javascript filter

    cleaner.style = True  # we want to activate the styles & stylesheet filter

    cleaner.kill_tags = ['head', 'img', 'iframe', 'nav', 'svg', 'figure', 'map']

    file = cleaner.clean_html(file)

    file = file.split()

    file = " ".join(file)

    return file
```

Uspeli smo pridobiti tokene s pomočjo HTMLParser (kot je prikazano spodaj), ki bi jih v nadaljevanju primerjali in uredili v končni regularni izraz, s katerim bi lahko pridobili podatke s spletne strani na način, kot smo to storili pri prvih dveh implementiranih metodah.

```
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        parsed_tokens.append(["s_tag", tag])
    def handle_endtag(self, tag):
        parsed_tokens.append(["e_tag", tag])
    def handle_data(self, data):
        if data != ' ':
            parsed_tokens.append(["data", data])
```

S krajšim primerom je bila implementacija uspešna, vendar ni delovala za daljše primere kot so primeri spletnih strani, ki jih je bilo potrebno obdelati. Implementacija bi morala iskati opcijske token-e, vendar zaradi nerazrešenega razloga rekurzija ni potekala pravilno in se je ustavila, če se je preveč zaporednih token-ov razlikovalo od iskanega.

```
def roadrunner(p1_tokens, p2_tokens, w, s, wrapper):

    """ END OF HTML FILE(S) """

    if w == len(p1_tokens) and s == len(p2_tokens):

        return wrapper

    p1_token = p1_tokens[w]

    p2_token = p2_tokens[s]

    """ MATCHING TOKENS """

    if match_found(p1_token, p2_token):

        wrapper.append(p1_token)

        return roadrunner(p1_tokens, p2_tokens, w + 1, s + 1, wrapper)

    """ FOUND DATA """

    if p1_token[0] == "data" and p2_token[0] == "data":

        wrapper.append(["data", "#PCDATA"])

        return roadrunner(p1_tokens, p2_tokens, w + 1, s + 1, wrapper)

    else:

        return wrapper
```

V primeru, da pridemo do konca obeh HTML datotek vrnemo končni wrapper oziroma ovojnico. V primeru, da najdemo ujemajoče se token-e (oba začetni / končni / data in enaka vrednost oziroma token), jih dodamo ovojnici. V primeru, da najdemo polje data, ovojnici dodamo vrednost #PCDATA. Na koncu ovojnico še preoblikujemo v regularni izraz in jo vrnemo kot končno

GitHub repozitorij projekta: https://github.com/anacepuran/IEPS/tree/main/pa2