3 metode ekstrakcije podatkov iz spletne strani

2. domača naloga pri predmetu WIER

Jakob Merljak, Kristjan Panjan, Blaž Dobravec

MENTOR: asist. prof. dr. Slavko Žitnik

1 Uvod

V drugi domači nalogi pri predmetu Iskanje in ekstrakcija podatkov iz spleta smo s tremi različnimi načini izluščili določene podatke iz treh spletnih straneh.
Tri spletne strani, s katerimi smo se ukvarjali so:

- SIOL
- RTVSLO
- OVERSTOCK

2 Opis izbranih strani, identifikacija elementov in podobnosti

Na danih straneh spletnega mesta Overstock se nahajajo podatki o cenah izdelkov. Strani spletnega mesta RTV Slo predstavljajo članke, ki imajo po pričakovanjih podobno strukturo. Enako velja za strani spletnega mesta Siol. Več informacij o najdenih elementih in podobnostih sledi v poglavjih o implementaciji regularnih izrazov in uporabi Xpath-a.

3 Implementacije

3.1 Implementacija 1: Regularni izrazi

3.1.1 Overstock

Overstock spletna stran se je malenkost razlikovala od drugih strani, saj smo tu morali pridobiti podatke o izdelkih, ki pa so se pojavili v večjem številu. Zato smo z uporabo regex iterirali skozi te izdelke in za vse poiskali zahtevane vsebine. Spletno stran Overstock smo pri RegEx implementaciji preiskali z naslednjimi izrazi:

- Title - `r"<b>([0-9]+-(.+)(?=\</b>))</b>`
- Last prices - `r"<s>(.*?)</s>"`
- Prices - `r"<b>([$€]\s*[0-9\.,]+)</b>"`
- Percents - `r"\(([0-9]+%\))"`
- Content - `r"<td valign=\"top\"><span class=\"normal\">(.*?)<\/b>"`
- Savings - `r"([$€]\s*[0-9\.,]+ )"`

3.1.2 RTV Slo

Pri RTV straneh sta bila na dveh straneh dva članka podobne strukture, zato smo samo poiskali specifične značke za iskane podatke in jih poskali preko RegEx izrazov. Uporabili smo naslednje:

- Author - `r"<div class=\"author-name\">(.*)</div>"`
- Title - `r"<h1>(.*)<\/h1>"`
- Date - `r"<div class=\"publish-meta\">(.*?)<br>"`
- Subtitle - `r"<div class=\"subtitle\">(.*)</div>"`
- Lead - `r"<p class=\"lead\">(.*)</p>"`
- Content - `r"<article class=\"article\">(.*?)<\/article>"`

Podobno kot pri RTV-ju smo naredili tudi pri Siol-ovih straneh, dva članka s podobno vsebino smo preiskali na podlagi značk in razredov teh značk. Uporabili smo naslednje RegEx izraze:

3.1.3 Siol

- Title - `r"<\s*h1[^>]*>(.*?)<\s*/\s*h1>"`
- Author - `r"<a href=\"/avtorji/(.*)>(.*)"`
- Date - `r"<span class=\"article__publish_date--date\">(.*?)</span>"`
- Intro - `r"<div class=\"article__intro js_articleIntro\">(.*?)</div>"`
- Content - `r"<div class=\"article__main js_article js_bannerInArticleWrap\">(.*?)</div>"`

3.2 Implementacija 2: Xpath algoritem

3.2.1 Overstock

Pri spletni strani Overstock smo za ekstrakcijo ustreznih podatkov uporabili pristop, kjer smo najprej zgenerirali vse objekte v seznam objektov, kjer posamezen objekt vsebuje vse elemente posameznega produkta, ki ga želimo nato pregledati.

objects = tree.xpath('//tbody/tr[(contains(@bgcolor, "ffffff") or contains(@bgcolor, "dddddd")) and count(td[@valign="top"]) = 2]/td[2]')

S pomočjo for zanke smo se nato sprehodili po objektih in izluščili ustrezne podatke:

Title = string(a/b/text())

ListPrice = string(table/tbody/tr/td[1]/table/tbody/tr[1]/td[2]/s/text())

Price = string(table/tbody/tr/td[1]/table /tbody/tr[2]/td[2]/span/b/text())

Saving, SavingPercent = string(table/tbody/tr/td[1]/table/ tbody/tr[3]/td[2]/span/text()).split()

Content = string(table/tbody/tr/td[2]/span/text())

Posamezen element smo nato shranili kot slovar in ga dodali v seznam ter seznam shranili v json format.

3.2.2 rtvslo

Pri ekstrakciji RTVslovenija smo uporabili relativne poti, kjer smo ustrezne datoteke identificirali na pogladi imena razreda.

Author = tree.xpath('//div[@class="author-name"]/text()')[0]

PublishedTime = (tree.xpath('//*[@id="main-container"]/div[3]/div/div[1]/div[2]/text()'))[0].lstrip().rstrip()

Title = tree.xpath('//header[@class="article-header"]/h1/text()')[0]

SubTitle = tree.xpath('//header[@class="article-header"]/div[@class=šubtitle"]/text()')[0]

Lead = tree.xpath('//header[@class="article-header"]/p[@class="lead"]/text()')[0]

3.2.3 siol

V razdelku SIOL smo pregled spletne strani s pomočjo xpath algoritma naredili podobno kot pri RTVSLO.

PublishedTime = remove_unwanted_characters(.join(tree.xpath('//span[@class= "article__publish_date"]/span[position()<4 and position()>1]/text()')))

Title = remove_unwanted_characters(tree.xpath('//h1[@class= "article__title"]/text()')[0])

Author = remove_unwanted_characters(tree.xpath('//span[@class= "article__author "]/a/text()')[0])

Lead = remove_unwanted_characters(tree.xpath('//div[@class= "article__intro js_articleIntro"]/p/text()')[0])

Content = remove_unwanted_characters(.join(tree.xpath('//div[@class= "article__main js_article js_bannerInArticleWrap"]/p[position()>0]/text()')))

3.3 Implementacija 3: Roadrunner

Za avtomatsko ekstrakcijo podatkov smo implementirali algoritem Roadrunner. Za osnovne smernice pri razvoju smo uporabili članek [1]. Za razčlenjevanje HTML kode smo uporabili knjižnico BeautifulSoup [2]. Naš algoritem je implementiran po naslednji psevdokodi:

```
parse HTML into BeautifulSoup object
remove whitespace and comment elements
from object
run recursive matching function
find iterators and return

recursive match function:
iterate over wrapper's and sample's children:
if both children are strings
    match them with regex for any number
    of any characters
else if both children are tags
    if both tags are of the same type
        match children recursively
else if wrapper's child is string and
sample's next child is string
    add optional regex for current samples'
    child
    keep wrapper's child for next
    iteration
else if sample's child is string and
wrapper's next child is string
    add optional regex for current wrapper's
    child
    keep sample's child for next
    iteration
at this point we can assume that both
children are tags and that they don't
match
if next child in sample is the same tag
as current wrapper tag
    add optional regex for current sample's
    child
    keep wrapper's child for next iteration
if next child in wrapper is the same tag
as current sample tag
    add optional regex for current wrapper's
    child
```

```
    keep sample's child for next iteration
if none of the above, add optional regex
for current sample's child

after above loop, add any remaining children
as optional regexes

algorithm for finding iterators:
for every child:
    remove all consecutive  children that
    exactly match child
    if any such children existed, surround
    current child with one-or-more regex
    recurse on child
```

Komentarje in "whitespace"odstranimo zato, ker so ti sicer del razčlenjenega drevesa, kar bi močno pokvarilo delovanje našega algoritma. Naš algoritem nato poskuša rekurzivno iskati ujemanja v dokumentu. Algoritem preverja, ali se istoležni nizi popolnoma ujemajo. Takšni nizi se ohranijo v ovojnici. Če se nizi ne ujemajo popolnoma, jih generaliziramo v kakršenkoli niz. Za to smo se odločili, ker smo želeli poenostaviti algoritem. Značke, ki se poimensko ujemajo, rekurzivno ujemamo dalje. Za poenostavitev algoritma ignoriramo atribute v značkah. Če se značke ne ujemajo mora algoritem generalizirati ovojnico tako, da določi opcijski element. Iskanja ponovljenih elementov (angl. in v članku *iterator*) na tej točki ne izvajamo, ker nam tega ni uspelo usposobiti. Vsaj nekaj generalizacije z iskanjem iteratorjev algoritem naredi na koncu, kjer v nastali ovojnici generalizira značke, ki se popolnoma ujemajo in so v neposrednem zaporedju.

Zaradi poenostavljene implementacije iskanja ponovljenih delov strani, ujemanja značk ter ujemanja nizov naš algoritem ne daje dobrih rezultatov. Vendar lahko v nastalih ovojnicah še vedno opazimo, da algoritem uspe generalizirati več nizov ter ponavljanj. V nastalih ovojnicah je po pričakovanjih veliko opcijskih elementov, ki so rezultat neuspelega iskanja ujemanja in posledično generalizacije v opcijski element. Nastale ovojnice so prevelike za to poročilo.

4 Zaključek

Pri tej nalogi smo uspešno izvedli pridobivanje informacij z danih strani z regularnimi izrazi in Xpath-om. Uspelo nam je implementirati tudi poenostavljen algoritem Roadrunner za generiranje ovojnic, ki pa se zaradi vseh poenostavitev ne odreže najbolje.

Literatura

[1] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.

[2] Leonard Richardson. Beautiful soup documentation. *Dosegljivo: https://www. crummy. com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]*, 2007.