Ekstrakcija podatkov

Skupina **DOMACI-NJOKI**: Niki Bizjak, Bojan Vrangeloski, Uroš Škrjanc

***Povzetek*—Cilj prve seminarske naloge je bil napisati pajka, ki zna s spleta prenašati spletne vsebine in jih shranjevati v podatkovno bazo. Pred iskanjem podatkov po taki podatkovni bazi, pa je treba prenesene vsebine najprej prečistiti in iz njih izluščiti pomembne informacije. V drugem seminarju si bomo ogledali tri različne načine za ekstrakcijo podatkov iz HTML vsebin.**

I. Uvod

Druga seminarska naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta je namenjena pridobivanju informacij iz vsebin, ki jih je s spleta prenesel pajek. V seminarju smo si ogledali tri različne načine ekstrakcije - dva taka, ki zahtevata veliko uporabnikovega poseganja in razumevanja strukture strani in tretjega, ki zna informacije prepoznati in izluščiti avtomatično, s primerjavo podobnih spletnih strani.

Ekstrakcijo podatkov smo izvajali na štirih različnih spletnih straneh, ki jih glede na strukturo razdelimo na dve skupini:

- **list** - stran vsebuje *seznam* izdelkov, artiklov, člankov, ipd.
- **detail** - stran prikazuje podatke za en izdelek, članek, ipd.

Tabela I, prikazuje naše testne strani, razvrščene glede na tipa strani, opisana zgoraj.

Stran	Tip
`overstock.com`	list
`rtvslo.si`	detail
`bolha.com`	list
`avto.net`	detail

Tabela I: Tip strani za posamezno domeno, nad katero smo izvajali ekstrakcijo podatkov

II. Regularni izrazi

Regularni izrazi nam omogočajo učinkovito iskanje informacij v nizih z uporabo končnih avtomatov. Regularni izrazi se lahko uporabijo za preverjanje, če posamezen niz vsebuje iskan vzorec. Pri implementaciji ekstrakcije z regularnimi izrazi, smo si pomagali s vgrajeno Python knjižnico `re`.

Za ekstrakcijo podatkov s strani Overstock, smo uporabili regularne izraze, prikazane na sliki 1. Najprej smo za vsako iskano polje na strani definirali svoj regularni izraz, nato pa smo jih z ustreznimi vmesnimi regularnimi izrazi, povezali v skupen izraz, ki zna hkrati poiskati vse podatke. Na podoben način smo ekstrahirali podatke tudi iz ostalih spletnih strani. Regularne izraze za spletno stran RTV SLO lahko vidimo na sliki 2, slika 3, pa prikazuje regularne izraze za spletno stran `Avto.net`.

```
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
```

Slika 1: Regularni izrazi, uporabljeni za ekstrakcijo podatkov s strani Overstock

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

Slika 2: Regularni izrazi, uporabljeni za ekstrakcijo podatkov s strani RTV SLO

III. XPath

XPath je poizvedovalni jezik, ki je bil razvit za namene ekstrakcije podatkov iz XML podatkovnih struktur. Izrazi so sestavljeni v obliki datotečnih poti in lahko vsebujejo pogoje za spust po določeni veji drevesa, klice funkcij nad vejami, ipd. Slike 4, 5, 6 in 7 prikazujejo XPath izraze, ki smo jih uporabili za ekstrakcijo podatkov iz izbranih spletnih strani.

Največje težave sta predstavljali strani `Avto.net` in Overstock, saj elementi niso bili semantično označeni. Ostale strani imajo namreč z razredi ali identifikatorji označene elemente na strani, kar olajša iskanje, saj lahko take elemente hitro najdemo s filtriranjem. Spletna stran Overstock je sestavljena iz vgnezdenih tabel, kar še posebej oteži iskanje, saj je potrebno poznati indekse elementov, ki jih iščemo. Če se taka

```
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>"
```

Slika 3: Regularni izrazi, uporabljeni za ekstrakcijo podatkov s strani Avto.net

```
title = '/tr/td[2]/a/b/text()'
list_price = '/tr/td[2]/table//table/tbody/tr[1]/td[2]/s/text()'
price = '/tr/td[2]/table//table/tbody/tr[2]/td[2]/span/b/text()'
saving = '/tr/td[2]/table//table/tbody/tr[3]/td[2]/span/text()'
content = '/tr//span[@class="normal"]//text()'
```

Slika 4: XPath izrazi, uporabljeni za ekstrakcijo podatkov s strani Overstock

```
title = '//header/h1/text()'
subtitle = '//header//div[@class="subtitle"]/text()'
lead = '//p[@class="lead"]/text()'
author = '//div[@class="author-name"]/text()'
published_time = '//div[@class="publish-meta"]/text()'
content = '//div[@class="article-body"]//*[not(name()="script")]/text()'
```

Slika 5: XPath izrazi, uporabljeni za ekstrakcijo podatkov s strani RTV SLO

stran le malo spremeni, bi naši XPath izrazi ne delovali več.

IV. Avtomatična ekstrakcija podatkov

Na spletu se od uvedbe programskega jezika PHP naprej pojavlja dosti dinamičnih spletnih strani. Take strani hranijo podatke v podatkovni bazi in glede na dostopan URL naslov izvedejo poizvedbo v podatkovni bazi in na podlagi rezultata prikažejo podatke v uporabniku prijazni obliki. Pri avtomatični ekstrakciji podatkov, bi radi na podlagi dveh strani, ki imata enako strukturo, a drugačno vsebino, zgradili program, ki zna samodejno pridobiti podatke.

V našem primeru smo za avtomatično ekstrakcijo podatkov uporabili algoritem RoadRunner [1], [2]. Ta sicer v originalni implementaciji deluje na seznamu žetonov (angl. token), v našem primeru pa smo za predstavitev HTML vsebin uporabili kar DOM drevo. Algoritem deluje tako, da v pomnilnik prebere dve datoteki - ovojnico (angl. wrapper) in drugi dokument. S pomočjo funkcije Generalize, skuša nato ovojnico ustrezno uskladiti z dokumentom tako, da išče istoležne elemente, v njih išče polja (angl. fields) in opcijske elemente (angl. optionals).

Implementiran algoritem zna na strani prepoznati polja, tj. elemente, ki se pojavijo na obeh straneh, a imajo drugačne vrednosti. Prav tako zna program prepoznati opcijske elemente, tj. elemente, ki se pojavijo v enem izmed dokumentov, v drugem pa ne. Našo implementacijo bi lahko izboljšali tako,

```
title = '//h3//text()'
price = '//div[@id="FINANCE"]/following-sibling::div//p/text()'
date_published = '//i[contains(@class, "fa-calendar")]/parent::div/text()'
number_of_views = '//i[contains(@class, "fa-bar-chart")]/parent::div/text()'
seller = '//div[contains(text(), "Prodajalec")]/following-sibling::div/following-
  sibling::div//strong/text()'
```

Slika 6: XPath izrazi, uporabljeni za ekstrakcijo podatkov s strani Avto.net

```
title = '/li//h3[@class="entity-title"]/a/text()'
price = '/li//li[@class="price-item"]/strong/text()'
location = '/li//div[@class="entity-description-main"]/text()'
date_posted = '/li//time/text()'
```

Slika 7: XPath izrazi, uporabljeni za ekstrakcijo podatkov s strani Bolha

```
<html>
    Books of: <b>PCDATA</b>
    <img optional="True" src=".../"/>
    <ul>
        <li>
            <i>Title:</i> #PCDATA
        </li>
        <li>
            <i>Title:</i> #PCDATA
        </li>
        <li optional="True">
            <i>Title:</i> Javascript
        </li>
    </ul>
</html>
```

Slika 8: Rezultat izvajanja avtomatične ekstrakcije podatkov

da bi dodali prepoznavanje iteratorjev, tj. elementov, ki se ponavljajo. Tega dela nam zaradi pomanjkanja časa, ni uspelo implementirati.

Slika 8 prikazuje rezultat izvajanja avtomatične ekstrakcije podatkov nad primerom iz članka [1]. Naš program opcijske elemente označi z atributom `optional="True"`. Kot lahko vidimo, je naša implementacija ustrezno prepoznala polja in opcijske elemente.

V. Zaključek

Pisanje regularnih in XPath izrazov za pridobivanje podatkov je zamudno in od razvijalca zahteva poznavanje oziroma razumevanje strukture spletne strani. Kljub temu, pa daje tak način ekstrakcije podatkov boljše rezultate, saj podatke označimo oziroma umestimo v nek kontekst. Razvijalec lahko npr. na mestih kjer se pojavijo cela števila, le-ta iz nizov pretvori v številčno predstavitev in s tem še dodatno opiše podatke. Avtomatična ekstrakcija podatkov je princip, pri katerem sistem na strani sam najde polja, ki vsebujejo podatke. Rezultati, tj. neoznačene izluščene informacije pa so v tem primeru pogosto pomanjkljivi oziroma težko razumljivi.

Prva metoda je uporabna predvsem v primerih, ko je potrebno izluščiti podatke iz ene same spletne strani. Pisanje izrazov je časovno potratno in je smiselno, kadar vemo, da se bo struktura spletne strani spreminjala zelo redko.

Druga metoda je zelo uporabna v primeru, ko imamo za ekstrahirati podatke iz ogromne količine različnih strani in smo za prihranek časa, pripravljeni žrtvovati natančnost.

Literatura

[1] V. Crescenzi, G. Mecca, and P. Merialdo, "The roadrunner project: Towards automatic extraction of web data," 01 2001.

[2] ——, "Automatic web information extraction in the road r unner system," in *International Conference on Conceptual Modeling*. Springer, 2001, pp. 264–277.