Seminarska naloga 2: Ekstrakcija podatkov

Erik Rakušček, Jan Šmid, Qichao Chen
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

I. UVOD

V tem poročilu so opisane 3 metode spletne ekstrakcije podatkov in njihove implementacije. Izvorna koda je dostopna na [1].

II. OPIS STRANI

Dodatno smo si izbral še 2 strani s portala ceneje.si. Struktura teh dveh strani je prikazana na sliki 1. Podatki, ki smo jih pridobili so:

- category
- numberOfListings
- title
- price
- store

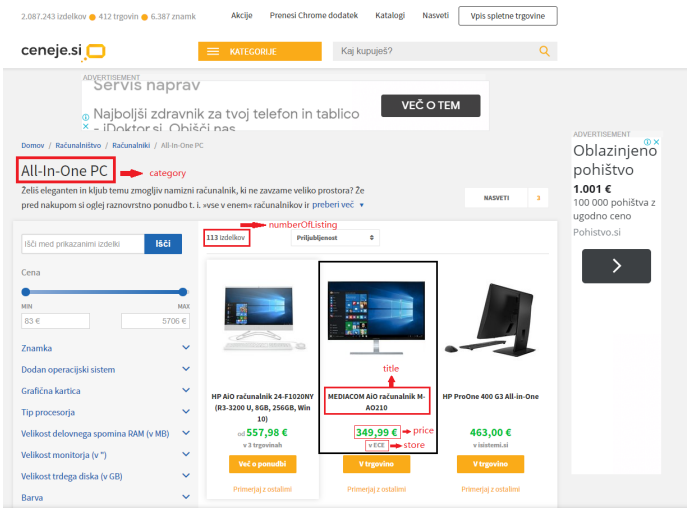

Fig. 1. Primer ceneje.si spletne strani

III. IMPLEMENTACIJA

A. Regular expressions

1) Rtvslo.si:

- author:

```
<div class=\"author-name\">
([a-zA-Z]+\s+[a-zA-Z]+)"
```

- publishedTime:

```
<div class=\"publish-meta\">
\s+([0-9]*.\s+[a-z]+\s+[0-9]*
\s+[a-z]+\s+[0-9]+\:[0-9]+)
```

- title:

```
<header class=\"article-header\">
[\s\S]+?<h1>([\s\S]+?)<\/h1>
[\s\S]+?<\/header>
```

- subTitle:

```
<header class=\"article-header\">
[\s\S]+?<div class=\"subtitle\">
([\s\S]+?)<\/div>[\s\S]+?<\/header>
```

- lead:

```
<header class=\"article-header\">
[\s\S]+?<p class=\"lead\">([\s\S]+?)
<\/p>[\s\S]+?<\/header>
```

- content:

```
<article class=\"article\">[\s\S]*?
(<p[\s\S]*?>([\s\S]*?)<\/p[\s\S]*?>
[\s\S]*?)*<\/article>
```

2) overstock.com:

- title:

```
<tr bgcolor=\"#[f|d]+\">[\S\s]*?<a
href=\"http://www\.overstock\.com
/cgi-bin/d2\.cgi\?PAGE=PROFRAME&amp
;PROD_ID=[0-9]+?\"><b>([\S\s]+?)<\/b>
```

- listPrice:

```
<tr bgcolor=\"#[f|d]+\">[\S\s]*?
<b>List Price:<\/b>[\S\s]*?<s>(.*?)<\/s>
```

- price:

```
<tr bgcolor=\"#[f|d]+\">[\S\s]*?
<b>Price:<\/b>[\S\s]*?<b>(.*?)<\/b>
```

- saving:

```
<tr bgcolor=\"#[f|d]+\">[\S\s]*?<b>
You Save:<\/b>[\S\s]*?<span class=
\"littleorange\">(.*?)\(.*?\)<\/span>
```

- savingPercent:

```
<tr bgcolor=\"#[f|d]+\">[\S\s]*
<b>You Save:<\/b>[\S\s]*?<span class=
\"littleorange\">.*? \((.*?)\)<\/span>
```

- content:
  `<tr bgcolor=\"#[f|d]+\">[\S\s]*?<a href=\"http://www\.overstock\.com/cgi-bin/d2\.cgi\?PAGE=PROFRAME&PROD_ID=[0-9]+?\">[\S\s]*?<span class=\"normal\">([\S\s]*?)<br>`

3) ceneje.si:

- category:
  `<div class=\"topContentBox\">[\s]+?<h1>(.*?)<\/h1>`
- numberOfListings:
  `<div class=\"numPro\">[\s]+?<b>(.*?)<\/b>`
- title:
  `<div class=\"content\">[\s\S]*?<h3>[\s]*?<a onclick=\"GaTrackEvent.*?>(.*?)<\/a>`
- price:
  `<p class=\"priceInfo\">[\s\S]*?<a onclick=\"GaTrackEvent.*?>[\s\S]*?<b>(.*?)<\/b>`
- store:
  `<p class=\"priceInfo\">[\s\S]*?<\/a>[\s]*?<a href=.* class=\"qtySellers\">[\s]*?<b>\s*?(\w [\S| ]*)\s*?<\/b>`

B. XPath

1) Rtvslo.si:

- 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]`
- 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:
  `//*[@id="main-container"]/div[3]/div/div[2]/article/p/text()`
  Vse odstavke nato združimo v en niz vsebine.

2) overstock.com:

- title:
  `/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td[2]/a/b/text()`
- listPrice:
  `/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td[2]/table/tbody/tr/td[1]/table/tbody/tr[1]/td[2]/s/text()`
- price:
  `/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td[2]/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/span/b/text()`
- saving in savingPercent:
  `/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td[2]/table/tbody/tr/td[1]/table/tbody/tr[3]/td[2]/span/text()`
  Nato uporabimo še regularni izraz tako, da za savingPercent dobimo samo procent brez oklepajev:
  `(?<=\().+?(?=\))`
- content:
  `/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td[2]/table/tbody/tr/td[2]/span/text()`

3) ceneje.si:

- category:
  `//*[@id="mainContent"]/div[3]/h1/text()`
- numberOfListings:
  `//*[@id="mainContent"]/div[6]/div[2]/div[1]/div[1]/b/text()`
- title:
  `//*[@id="productGrid"]/div/div/div[2]/h3/a/text()`
- price:
  `//*[@id="productGrid"]/div/div/div[2]/p/a[1]/b/text()`
- store:
  `//*[@id="productGrid"]/div/div/div[2]/p/a[2]/b/text()`
  Trgovine še dodatno filtriramo z uporabo metod v pythonu za delo z nizi.

C. Automatic Web extraction - RoadRunner

Najprej smo iz html datotek pobrisali vse, kar nam ni bilo v pomoč pri poravnavi dveh strani (skripte, stile, komentarje, nepotrebne atribute ...).

Nato smo vzeli dve datoteki iz iste strani in poravnali njune značke (tage) s pomočjo global sequence alignment, bolj natančno z Needleman-Wunsch algoritmom. Uporabili smo privzeto točkovno tabelo, ki za ujemajoča znaka vrne +1, za neujemajoča in prazno polje pa -1. Dobili smo dva enodimenzionalna seznama, v vsakem polju pa je bila ena značka oz. prazno polje. Čez oba seznama smo se sprehodili (od prve značke do konca) tako, da smo primerjali cele elemente. Če elementa nista bila enaka, smo rekurzivno klicali njegove podelemente in sproti zapisovali značke v rezultat. Če sta elementa bila enaka, ga nismo zapisali v rezultat. Sproti smo še preverjali, če je eden od elementov imel sama prazna polja, potem smo glede na mesto, kjer se je nahajal, dodali '+' oziroma '?'.

Pseudocode:

```
# pocistimo html datoteki
h1,h2 = clean(html1,html2)
# uporabimo sequence alignment
align1, align2 = nw_align(h1, h2)
# pricnemo grajenje ovojnice
RoadRunner(align1,align2)

def RoadRunner(html1,html2):
    # sprehodimo se cez vse
    # elemente na istem levelu DOM drevesa
  for e in elements:

    e1 <- naslednji el. iz html1
    e2 <- naslednji el. iz html2

    # elementa sta enaka, ju
    # ne dodamo v ovojnico
    if e1 == e2:
      return
    else:
      if noChilds(e1) or noChilds(e2)
        AddToResult
      else:
        RoadRunner(childs(e1),childs(e2))
```

Algoritem je dobro prepoznal elemente, ki so v eni datoteki manjkali[1]. Imel je nekaj težav s prepoznavanjem značk in <p>. Prav tako je v določenih primerih blok razdelil v manjše manjkajoče elemente, namesto da bi posamezen blok cel označil kot manjkajoč.

[1]Zaradi lažje evalvacije ovojnice, nekaterih neujemajočih nizov nismo zamenjali s text.

Fig. 2. Primer manjkajočega bloka strani pri primerjavi dveh strani iz rtvslo.si.

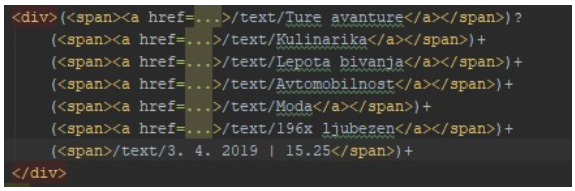

Fig. 3. Prikaz pomanjkljivosti našega algoritma - cel blok iz slike 2 bi lahko označil kot en '?' element

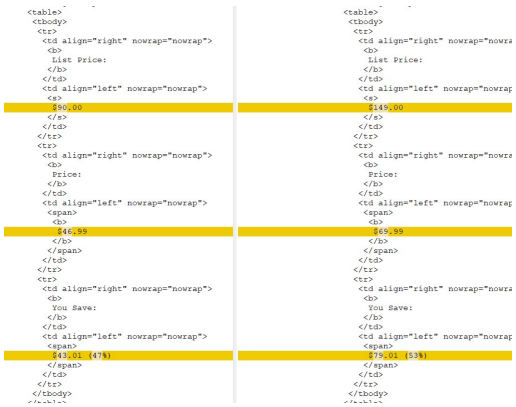

Fig. 4. Primerjava dveh strani iz overstock. Razlike samo v ceni.

Fig. 5. Algoritem dobro prepozna posamezne razlike (primer iz slike 4)

IV. ZAKLJUČEK

Krajši regularni izrazi so se izkazali za bolj robustne, saj z njimi lahko pridobimo enake podatke tudi pri dinamičnih spletnih straneh (kjer se struktura strani pogosto spreminja). Slabost regularnih izrazov pa je zelo zamudno pisanje le-teh.

Pristop z XPath se je izkazal za zelo nezanesljivega, saj ne omogoča dinamičnosti in moramo za vsako spletno stran ustvariti nove poizvedbe.

Če bi želeli uporabiti algoritem RoadRunner za ekstrakcijo podatkov iz spleta, bi morali odpraviti pomanjkljivosti in napake omenjene v poglavju III-C.

REFERENCES

[1] Github repozitorij. Dosegljivo: https://github.com/van123helsing/pachong-11. [Dostopano: 3. 4. 2020].