

# Poročilo druge seminarske naloge

Mark Bogataj in Jakob Maležič

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Ljubljana, Slovenija

Mentor: asist. prof. dr. Slavko Žitnik

## I. OPIS IZBRANIH SPLETNIH STRANI

Za dodatni spletni strani sva si izbrala dve strani iz novičarske spletne strani [Žurnal24](#). Pri izbiranju strani sva se odločila za dve, ki opisujeta avtomobile, podobno kot strani iz domene rtvsl, saj naju je zanimala razlika med stranema, ki sicer izgledata zelo podobno, ampak po sami strukturi pa se precej razlikujeta. Primer strani je prikazan na sliki 1.

Iz spletnih strani domene zurnal24 sva izluščila naslednje podatke:

- author - ime avtorja, ki je napisal novico
- publishedTime - datum in čas objave novice
- title - naslov novice
- viewCount - število ogledov novice
- lead - uvod
- content - samo tekstovna vsebina

## II. EKSTRAKCIJA Z UPORABO REGULARNIH IZRAZOV

### A. Overstock

Regularni izraz za ekstrakcijo podatkov iz spletnih strani Overstock je sledeč:

```
<b>([0-9].+)</b>.*\n.*\n.*<b>(.+)</b>.*<s>([\$0-9,.]+)(.*\n.*<b>(.+)</b>.*<b>([\$0-9.]+))?(.*\n.*<b>(.+)</b>.*<span class="littleorange">([\$0-9,.]+)\s.([0-9.]+))?(.*\n.*<span class="normal">(.*)\n?(.*)\n?(.*)<br><a>
```

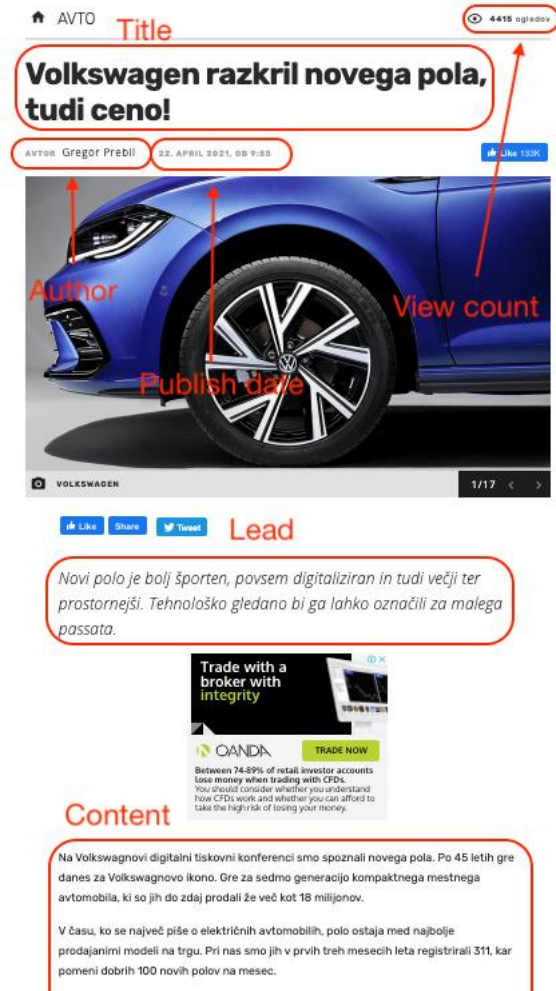
Večina delov regularnega izraza je zaviti še v oklepaj z vprašajem na koncu, kar pomeni, da je lahko ta del opcijski. S tem regularni izraz izlušči vsebino tudi, če katera od cen ni navedena. To sva tudi testirala tako, da sva iz html kode odstranila značko, ki vsebuje ceno.

### B. Rtvsl

Regularni izraz za ekstrakcijo podatkov iz spletnih strani Rtvsl je sledeč:

```
<h1>(.)</h1>\n<div class="subtitle">(.)</div>[\s\S]*<p class="lead">(.)</p>[\s\S]*<div class="author">[\s\S]*<div class="author-name">(.)</div>[\s\S]*<div class="publish-meta">\n[W]*(.)<br>[\s\S]*</figure>[\n]*<p(?:[\s\r]+[^\>]*)?>(.)</p>
```

Na koncu sva dobljeno vsebino počistila na ta način, da sva odstranila vse html značke, ki so ostale v vsebini. To sva naredila z uporabo regularnega izraza `<[^\>]*>`.



Slika 1: Primer strani z označenimi podatki, ki smo jih izluščili

### C. Žurnal24

Regularni izraz za ekstrakcijo podatkov iz spletnih strani Žurnal24 je sledeč:

```
<span class="article__views">[\n]*<i>.*</i>[\n]*<strong>(.)</strong>[\s\S]*<h1 class="article__title">(.)</h1>[\n]*<div class="article__authors">[\s\S]*<a
```

```
href=.*>(.*)</a>[\n]*</div>[\n]*<time
class="article__time">(.*)</time>[\s\S]*<div
class="article__leadtext">[\n]*(.*)[\n]*</div>[\s\S]*<div
class="article__content no_page_break cf"
.*>[\n([\s\S]*)</div>[\n]*<div class="text-center
fold_article__bellow_content">
```

Na koncu sva dobljeno vsebino ponovno počistila in odstranila vse html značke, ki so ostale v vsebini. To sva naredila z uporabo regularnega izraza: `<[^\>]*>`.

### III. EKSTRAKCIJA Z UPORABO XPATH

#### A. Overstock

Najprej z xpath izrazom izberemo vse kartice, ki imajo ozadje določene barve in imajo 2 `<td>` otroka z atributom `@valign='top'`: `cards = treeContent.xpath("//tbody/tr[contains(@bgcolor, '#ffffff') or contains(@bgcolor, '#dddddd')] and count(td[@valign='top'])=2]").` Nato iz vsakega element iz seznama kartic izluščimo informacije:

- `card.xpath("td[@valign='top']/a/b/text()")[0]`
- `card.xpath("td[@valign='top']/table/tbody/tr/td[1]/table/tbody/tr[1]/td[2]/s/text()")[0]`
- `card.xpath("td[@valign='top']/table/tbody/tr/td[1]/table/tbod/tr[2]/td[2]/span/b/text()")[0]`
- `card.xpath("substring-before(td[@valign='top']/table/tbody/tr/td[1]/table/tbody/tr[3]/td[2]/span/text(), ' (')")`
- `card.xpath("substring-before(substring-after(td[@valign='top']/table/tbody/tr/td[1]/table/tbody/tr[3]/td[2]/span/text(), ' (', ')")`
- `card.xpath("td[@valign='top']/table/tbody/tr/td[2]/span/text()")[0]`

Na koncu vsebino pridobljeno z zadnjim izrazom (opis) uredimo tako, da znak za novo vrstico zamenjamo s predledkom.

#### B. Rtvlo

Xpath izrazi za ekstrakcijo podatkov iz strani rtvslo so:

- `treeContent.xpath('//*[ @id="main-container"]/div[3]/div/header/h1/text()')[0]`
- `treeContent.xpath('//*[ @id="main-container"]/div[3]/div/header/div[2]/text()')[0]`
- `treeContent.xpath('//*[ @id="main-container"]/div[3]/div/div[1]/div[1]/div/text()')[0]`
- `treeContent.xpath('normalize-space(//*[ @id="main-container"]/div[3]/div/div[1]/div[2]/text())')`
- `treeContent.xpath('//*[ @id="main-container"]/div[3]/div/header/p/text()')[0]`
- `treeContent.xpath('//*[ @id="main-container"]/div[3]/div/div[2]/article/p/text() | //*[ @id="main-container"]/div[3]/div/div[2]/article//strong/text()')`

#### C. Žurnal24

Pri ekstrakciji podatkov iz strani Žurnal24 najprej izluščimo glavo članka iz katerega bom razbrali število ogledov, naslov, avtorja in čas objave:

- `treeContent.xpath("//article/header")[0]`
- `articleHeader.xpath('div[1]/span/strong/text()')[0]`
- `articleHeader.xpath('h1/text()')[0]`
- `articleHeader.xpath('div[2]/a/text()')[0]`
- `articleHeader.xpath('time/text()')[0]`

Uvod in vsebino pa izluščimo z naslednjima izrazoma:

- `treeContent.xpath("//article/div/div[2]/div/div[3]/text()')[0]`
- `treeContent.xpath("//article/div/div[2]/div/div[5]//text() [normalize-space(.)')]`

### IV. IMPLEMENTACIJA AVTOMATIZIRANE EKSTRAKCIJE

Implementirala sva algoritem Road Runner, ki avtomatizirano ekstrahira ovojnico za ekstrakcijo podatkov.

#### A. Priprava besedil

Iz besedil najprej vzameva zgolj glavni del – *body*, odstraniva skoke v novo vrstico, izbriševa komentarje in nato odstraniva značke: *script*, *noscript*, *link*, *nav*, *footer*, *input*, *form*, *style*, *iframe*, *button*, *svg*, *figure* in *br*. Med temi značkami ni pomembnih podatkov, ki bi nas zanimali in s tem skušamo čimbolj zmanjšati razliko med besediloma. Potem iz besedil z uporabo regularnega izraza izluščiva vse značke in jih shraniva v seznam za vsako od strani.

#### B. Poravnava besedil

Čeprav naj bi bili besedili, ki ju uporabimo za ekstrakcijo ovojnice podobni, ju moramo najprej poravnati tako, da se karseda ujemata po značkah. To narediva tako, da iterirava po značkah iz obeh strani hkrati in primerjava dve znački naenkrat in preveriva vse možne kombinacije značk:

- **Enaki:** znački sta enaki kadar se njuno ime ujema, sta na isti globini<sup>1</sup> in, če sta odpirajoči, vsebujeta približno enako število značk. To število je določeno glede na začetno razliko v številu značk besedil. V tem primeru ju samo dodamo ali odstranimo iz sklada.
- **Samo-zapirajoči:** obe znački sta samo-zapirajoči, vendar se razlikujeta po imenu.
- **Samo-zapirajoča in odpirajoča:** v seznam pred odpirajočo dodava None.
- **Samo-zapirajoča in zapirajoča:** v seznam pred zapirajočo dodava None.
- **Odpirajoča in zapirajoča:** v seznam pred zapirajočo dodava None.
- **Zapirajoči:** pogledava kateri sklad je manjši, kar pomeni da je tista značka na manjši globini in zato pred tisto zapirajočo značko dodava None.
- **Odpirajoči:** pogledava katera odpirajoča značka ima več otrok in v seznam s to značko dodava toliko objektov None, kolikor otrok ima soležna odpirajoča značka.

Obenem pa ves čas skrbiva za pravilno stanje skladov, ki vsebujeta, še ne zaprte, odpirajoče značke.

<sup>1</sup> To načeloma ni potrebno, saj kadarkoli, ko primerjamo dve znački morata biti na enaki globini, kar pa implementacija že zagotavlja.

### C. Ekstrakcija ovojnice

Za gradnjo ovojnice sva pripravila razred Tag, ki ga uporablja za opis značk in izpis ovojnice. Ovojnico sedaj zgradimo tako, da ponovno iteriramo čez oba seznama značk, jih ponovno paroma primerjamo in kreiramo nov seznam objektov Tag, iz katerega bomo nato zgradila tekstovno obliko ovojnice:

- **Enaki in nista None:** primerjamo znački še po vsebini oziroma tekstu in če ta ni enak, nastavimo vsebino na *#TEXT* ter shranimo eno izmed značk v nov seznam
- **Ena izmed značk je None:** znački, ki ni None, nastavimo atribut *optional* na *True* in jo shranimo v seznam

Hkrati shranjujeva tudi globino značke, ki nama nato omogoča lepši izpis s pravilnimi zamiki.

### D. Detekcija ponovljenih blokov

Po ekstrakciji ovojnice imava samo še en seznam objektov tipa Tag. To nama olajša detekcijo duplikatov. Za detekcijo duplikatov iterirava čez novo ustvarjen seznam in primerjava trenutno in naslednjo značko. Če je trenutna značka enaka zapirajoči znački naslednje, potem preveriva ali gre za ponovljen blok značk. To narediva tako, da najdeva začetek trenutnega bloka in trenutni ter naslednji blok primerjava po značkah in vsebini. Med primerjanjem zgradija začasni seznam s posodobljenimi značkami in vsebino. Če se bloka popolnoma ujameta v značkah, značke v glavnem seznamu s pomočjo začasnega seznama posodobiva oziroma odvečne značke odstraniva.

### E. Kreiranje ovojnice

Kreiranje končne ovojnice je sedaj zelo preprosto. Narediva zgolj eno iteracijo čez seznam značk, ki so ostale in uporablja podatke shranjene v objektu Tag za izpis ustreznega zamika, števnosti, pripone, predpone in opcijske besede.

- **multiple** pomeni da se značka pojavi enkrat ali večkrat
- **optional** pomeni da je značka opcijska
- **multiple optional** pomeni, da se značka pojavi ničkrat ali večkrat

Ponovljeni bloki so označeni z oklepajem, zaklepajem in zvezdico.

### F. Izživi

TODO Jakob

### G. Rezultati

Izmerila sva delež značk, ki so bile pri poravnavi besedila prepoznane kot enake. Ostale značke so bili označene kot opcijske, torej naj bi se pojavile zgolj na eni izmed strani. Glede na ta kriterij, se je poravnava besedila najboljše izkazala pri straneh iz *Rtvslo*. Predvidevava da zato, ker je bila razlika med številom značk med stranema po čiščenju besedila majhna.

Overstock	1962 : 1238 (63.10 %)	1214 : 1987 (61.10 %)
Rtvslo	1300 : 1316 (98.78 %)	1246 : 1371 (90.88 %)
Žurnal	3005 : 4345 (69.16 %)	2688 : 4663 (57.65 %)

Avtomatsko ekstrahirana ovojnica za strani *TestA* in *TestB*:

```
<HTML>
  <BODY>
    Books of:
      <B>
        #TEXT
      </B>
    <BASE/> optional
    <IMG/> multiple optional
    <OPEN> optional
      <B> optional
        Hello
      </B> optional
    </OPEN> optional
    <UL>
      (
        <LI>
          <I>
            Title:
          </I>
          #TEXT
        </LI>
      ) *
    </UL>
    <P> optional
      Hello
    </P> optional
  </BODY>
</HTML>
```

	Razlika v številu značk na začetku	Delež enakih značk po poravnavi
Test	21 : 28 (75.00 %)	18 : 32 (56.25 %)