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 rtvslo, 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

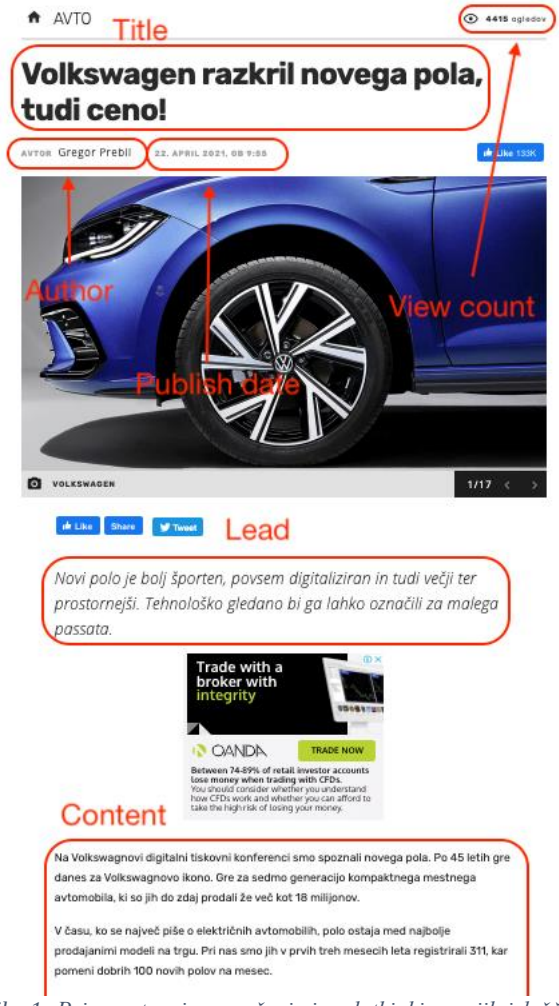
A. *Overstock*

Regularni izraz za ekstrakcijo podatkov iz spletnih strani Overstock je sledeč:
([0-9].+)<\/b>.\n.*\n.*(.+).*<s>([\$0-9,.]+)(.*\n.*(.+).*([\$0-9.]+))?(.*\n.*(.+).*<span class=\"littleorange\">([\$0-9.,]+)\s.([0-9]+.))?(.*\n.*\n.*<span class=\"normal\">(.*\n?(.*)\n?(.*)\n?(.*))
<a)?*
Večina delov regularnega izraza je zavitih 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. *Rtvslo*

Regularni izraz za ekstrakcijo podatkov iz spletnih strani Rtvslo je sledeč:
<h1>(.)</h1>\n (.*) [\s\S]* (.*) [\s\S]* [\s\S]* (.*) [\s\S]* \n[\W]*(.*)
[\s\S]*</figure>[\n]*<p(?:[\s\r]+[^>]*)?>(.*) *

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:
[\n]<i.*>.*</i>[\n]*(.*)[\s\S]*<h1 class="article__title">(.*)</h1>[\n]* [\s\S]*<a*

.

href=.>(.*)[\n]* [\n]*<time class="article__time">(.*)</time>[\s\S]* [\n]*(.*)[\n]* [\s\S]*<div class="article__content no_page_break cf" .*>\n([\s\S]*) [\n]* *

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. Rtvslo

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 ter preveriva vse možne kombinacije značk:

- **Enaki**: znački sta enaki kadar se njuno ime ujema, sta na isti globini[1] in, če sta odpirajoči, vsebujeta približno enako število značk. To število je določeno glede na začetno razliko v številu značk. V tem primeru ju samo dodamo ali odstranimo iz sklada.
- **Samo-zapirajoči**: obe znački sta samo-zapirajoči, vendar se razlikujeta po imenu. V seznama izmenično dodava None. Tako veva, da sta obe znački opcijski.
- **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.

[1] 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 uporabiva za opis značk in izpis ovojnice. Ovojnico nato zgradiva tako, da ponovno iterirava čez oba seznama značk, jih ponovno paroma primerjava in kreirava nov seznam objektov Tag, iz katerega bova nato zgradila tekstovno obliko ovojnice. Glede na znački narediva naslednje:

- **Enaki in nista None**: primerjava znački še po vsebini oziroma tekstu in če ta ni enak, nastaviva vsebino na *#TEXT* ter shraniva eno izmed značk v nov seznam
- **Ena izmed značk je None**: znački, ki ni None, nastaviva atribut *optional* na *True* in jo shraniva 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 zgradiva 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 uporabiva podatke shranjene v objektu Tag za izpis ustreznega zamika, številnosti, pripone, predpone in opcijske besede. Oznake številnosti so:

- **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. Izzivi

Največji izziv nama je predstavljala poravnava besedil. Ko primerjava dve odpirajoči znački, imava dve problematični situaciji:

- **Odpirajoči znački sta različni**: pri tem ne moremo zagotovo vedeti, katera nastopa prej in katera kasneje. Kadar se napačno odločiva, pride do posledic tudi pri kasnejših poravnavah – zazna značke kot opcijske, čeprav ne bi smel, ker niso pravilno poravnane.
- **Imeni odpirajočih značk sta enaki, vendar gre za različni znački**: to poskušava rešiti tako, da primerjava tudi število elementov v bloku odpirajočih značk, vendar se to število lahko razlikuje zaradi opcijski elementov.

Velik izziv pa nama je predstavljala tudi detekcija duplikatov, ki sva jo poenostavila zgolj na popolno ujemanje dveh blokov. To ni popolnoma pravilno, saj ima lahko en blok kakšen opcijski element pa gre še vedno za enak blok. Čeprav do neke mere upoštevava tudi opcijske elemente, saj le te najprej poiščeva nato pa poiščeva še duplikate.

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 v enem izmed besedil. 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 že po čiščenju besedila majhna.

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

Tabela 1: Rezultati

Zaradi oblike in predstavljenih izzivov, ki sva jih mogoče slabo rešila so najine ovojnice precej večje od A4 formata, zato jih nisva dodala v poročilo, vendar se nahajajo v mapi *output_extraction*. Sledi zgolj ovojnica za testni 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>
```

H. Pseudokoda

```
def road_runner(file1, file2):
    clean_html(file1)
    clean_html(file2)

    tags_a = find_tags(file1)
    tags_b = find_tags(file2)

    # poravnava
    while tags_a and tags_b:
      if tag_a equals tag_b:
        if tag_a is opening:
            add tag_a to stack
            add tag_b to stack
        else if tag_a is closing:
            pop stack_a
            pop stack_b
      else if tag_a and tag_b are selfclosing:
        add None at current index to tags_b
        add None at current index + 1 to tags_a
      else if one is self-closing and other opening:
        insert None to opposite one:
      else if one is self closing and other closing:
        insert None to opposite one:
      else if one is opening and other closing:
        insert None to opposite one
      else if both are closing:
        insert None to bigger one
      else if both are opening:
        find smaller and fill other with None

    #ekstrakcija ovojnice
    while tags_a and tags_b:
      tag_a = Tag(tag_a)
      tag_b = Tag(tag_b)

      if tag_a equals tag_b:
        if tag_a.text not equals tag_b.text:
          tag_a.text = "#TEXT"
        save tag_a
      elif one is none:
        save other as optional

    #detekcija ponovljenih blokov
    while tags:
      if current_tag closes next_tag:
        iterate over current_tag.children and next_tag.children:
          if current_tag.child equals next_tag.child:
            add current_tag to update_list
        if all children match:
          update tags with update_list

    #kreiranje ovojnice
    while tags:
      wrapper += tag.to_string()

    return wrapper
```