Seminarska naloga 2 - Ekstrakcija podatkov iz spleta

Gal Bumbar, Jan Lampič, Maja Umek

Maj 2019

1 Uvod

Hiter napredek pri tehnikah pridobivanja podatkov je povzročil, da se količina podatkov iz dneva v dan eksponentno veča. Ocenjuje se, da je 80% svetovnih podatkov shranjenih v nestrukturiranem besedilu. Tekstovno rudarjenje je tako postalo zanimivo raziskovalno področje, saj poskuša odkriti dragocene informacije iz nestrukturiranih besedil.

V poročilu so predstavljeni trije pristopi za ekstrakcijo strukturiranih podatkov iz spleta. V prvem delu poročila sta implementirani funkciji, ki za vhod prejmeta HTML datoteko, nato pa s pomočjo regularnih izrazov oziroma XPatha vrneta želene podatke v strukturirani obliki. V drugem delu naloge pa je predstavljena implementacija RoadRunner algoritma, ki za vhod prejme dve spletni strani istega tipa ter vrne regularni izraz, ki se ga lahko uporabi za pridobivanje podatkov.

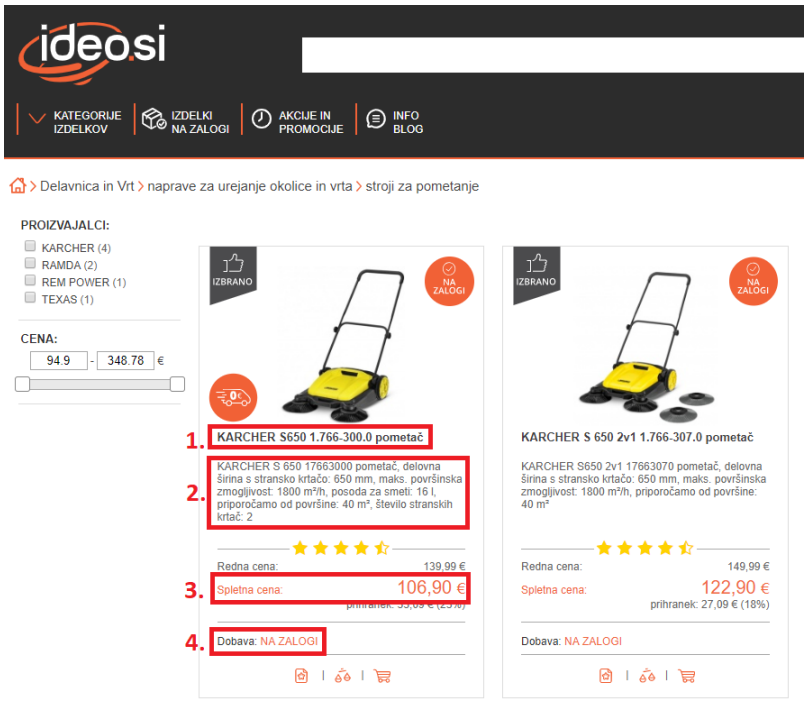
2 Implementacija regularnih izrazov in XPath-a

V okviru prvega dela seminarske naloge smo iz danih spletnih strani (strani iz domen Overstock in Rtvslo.si) ter iz še enega izbranega para strani ekstrahirali želene podatke.

Za dodaten primer smo si izbrali strani iz spletne trgovine dosegljive na `www.ideo.si`. Vsaka izbrana stran vsebuje seznam prodajnih izdelkov različnih dolžin. Naš cilj je bil, da za vsak izdelek pridobimo njegovo ime, opis, ceno ter zalogo oz. dobavni čas. Želeni podatki so označeni na spodnji sliki 1.

2.1 Ekstrakcija z regularnimi izrazi

Za vsak par strani je implementirana svoja funkcija, ki z enim regularnim izrazom pridobi vse podatke enega želenega polja (npr. za vse naslove izdelkov je uporabljen en regularni izraz). Funkcije sprejmejo vsebino HTML v obliki niza, na katerem se nato izvajajo funkcije knjižnice *re* za ekstrakcijo z našimi regularnimi izrazi. V primeru elementa *Content* spletne strani Rtvslo.si je z dodatnim



Slika 1: Ciljni podatki spletne strani. 1 - ime izdelka, 2 - opis izdelka, 3 - cena izdelka in 4 - dobava/zaloga.

regularnim izrazom vsebina še očiščena odvečnih značk. Dobljeni podatki se shranijo v slovar, ki se ga nato pretvori v obliko JSON in izpiše na standardni izhod. Izhodi posameznih strani se nahajajo v mapi `outputs/regex`.

2.2 Ekstrakcija z uporabo XPatha

Enako kot pri ekstrakciji z regularnimi izrazi, smo za vsak par strani definirali svojo funkcijo, ki za vsak ciljni podatek pridobi z uporaba enega XPatha. Funkcije sprejmejo vsebino HTML v obliki niza, nato pa se s pomočjo knjižnice *lxml* in podanih XPathov izluščijo želene podatke. Nekateri podatki so nato dodatno obdelani za lep in pravilen izpis (npr. ko se več želenih podatkov nahaja v vsebini iste značke). Urejene podatke se pred izpisom ponovno shrani v slovar in pretvori v JSON obliko. Izhodi posameznih strani se nahajajo v mapi `outputs/xpath`.

3 Implementacija RoadRunner-like programa

Cilj programa je da, na podlagi primerjave dveh HTML strani istega tipa vrne regularni izraz, ki se lahko uporabi za ekstrakcijo podatkov. Pri implementaciji

smo si pomagali s člankom (1), kjer je predstavljena ideja samega RoadRunnerja.

3.1 Predprocesiranje HTML strani

Pred začetkom primerjave spletnih strani, program najprej uredi vhodna HTML niza. To stori s pomočjo funkcije `clean_up`, ki iz HTMLja odstrani skripte, javascript, komentarje ter atribute značk. Funkcija z dodajanjem manjkajočih značk poskrbi tudi za njihovo ujemanje. Nato spletni strani pretvori v seznam, kjer je vsak element seznama bodisi značka bodisi niz.

3.2 Primerjava HTML strani

Za primerjavo HTML strani in konstrukcijo regularnega izraza smo definirali funkcijo `compare_html`. Ta za vhod prejme dva seznama HTML strani, imenovana `wrapper_list` in `sample_list`. Na koncu vrne seznam, iz katerega se kasneje zgradi regularni izraz.

Funkcija gradi seznam regularnih izrazov tako, da se premika po seznamih strani in primerja elemente. Pri primerjavi elementov upošteva naslednja pravila. Če se trenutna elementa ujemata (sta enaka niza ali enaki znački), funkcija element doda v seznam regularnih izrazov, nato pa se premakne na naslednja elementa seznamov.

Pri neujemanju elementov pa funkcija obravnava več primerov:

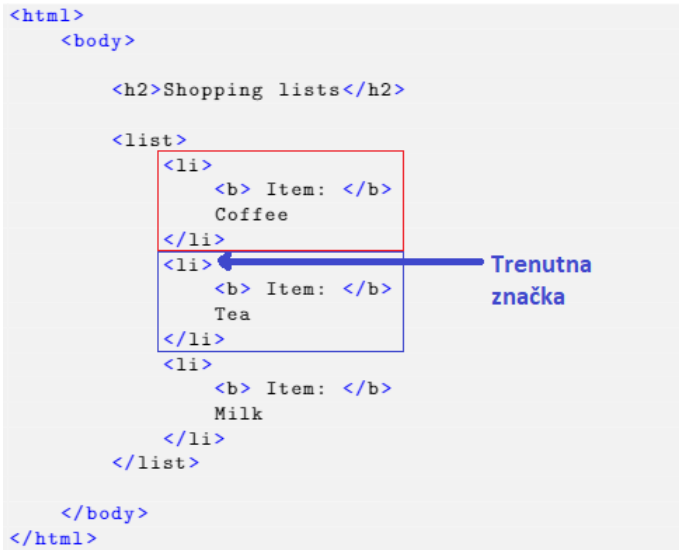
- **Neujemanje niz in niz**:
  V primeru neujemanja dveh nizov, se v seznam regularnih izrazov doda `(.*?)` in se na obeh seznamih, ki jih primerjamo premakne naprej.

- **Neujemanje značka (v `wrapper_list`) in niz (v `sample_list`)**:
  V tem primeru funkcija v seznam regularnih izrazov doda `(.*?)` in se v seznamu `sample_list` premakne naprej.

- **Neujemanje niz (v `wrapper_list`) in značka (v `sample_list`)**:
  V tem primeru funkcija prav tako v seznam regularnih izrazov doda `(.*?)` in se nato premakne naprej po seznamu `wrapper_list`.

- **Neujemanje značka in značka**:
  V takem primeru funkcija najprej pridobi predhodni znački obeh seznamov. Ti algoritmu pomagata pri iskanju iteratorjev. Če se ime predhodne značke ujema s trenutno in je trenutna značka začetna značka, predhoda pa končna značka, se izvede preverjanje iteratorjev. Tu se najprej poišče bloke, ki se jih nato primerja. En blok predstavlja celoten del, ki ga vsebuje trenutna značka, drugi pa del, ki ga vsebuje predhodna značka. Primer za omenjena bloka je prikazan na sliki 2.

  Če sta bloka enaka oz. prihaja le do neujemanj nizov, je to iterator. V regexu ga označimo kot `(<regex_iteratorja>`$\backslash s^*)^*$ in iz obstoječega regexa odstranimo vse predhodne bloke enake iteratorju. V nasprotnem



Slika 2: Z modro je obarvan blok, ki ga vsebuje trenutna značka, z rdečo pa blok predhodne značke.

primeru pa za oba primerjana seznama poišče naslednjo značko, ki se nahaja izven bloka trenutne značke. Nato se izvede tako imenovan *cross-matching*, pri katerem se pogleda naslednjo značko nasprotnega seznama. Če se znački ujemata, se blok, ki ga definira trenutna značka nasprotnega seznama označi kot opcijski (v regexu je označen kot `(<blok>)?`).

3.3 Konstrukcija regularnega izraza

Iz seznama, ki ga vrne primerjava strani, se ustvari regularni izraz s konkatenacijo njegovih elementov. Pri tem se vsak znački doda še regularni izraz za morebitne atribute (npr. značko `<list>` pretvorimo v `<list.*?>`).

3.4 Izvajanje programa

Implementirani program smo preizkusili na primeru iz predavanj, kjer program vrne ustrezen regularni izraz. Pri preizkusih na dejanskih spletnih straneh (Overstock, Rtvslo, Ideo), pa smo zaradi odločitve, da namesto drevesa uporabimo seznam, naleteli na težave pri večkratnem gnezdenju. Program tudi na teh straneh vrne regularni izraz, ki sicer zgleda obetaven in tudi deloma deluje, a vendarle ne vrne podatkov v obliki, ki smo jo hoteli doseči. Izhodi programa se nahajajo v mapi `outputs/road_runner`.

4 Psevdokoda RoadRunner algoritma

Algoritem 1 RoadRunner

VHOD: HTML niza, ki ju želimo primerjati.
IZHOD: Regularni izraz za pridobitev podatkov iz spletnih strani.

```
function get_wrapper(html1, html2):
  clean_up(html1, html2)
  wrapper_list = html2list(html1)
  sample_list = html2list(html2)
  regex_list = compare_html(wrapper_list, sample_list)
  convert regex_list to regex_string
  & add whitespace and tag attributes matching
  return regex_string

function compare_html(wrapper_list, sample_list):
  regex_list = [ ]
  while (some unchecked elements left):
    if(current items match):
      regex_list.add(current_item)
      continue
    if (tag mismatch):
      get previous tag names
      for wrapper, sample check:
        if (is start tag & matches previous tag):
          get upper_square, current_square
          if iterator(upper_square, current_square):
            regex_list.add((<current_square_regex>)*)
            regex_list.remove(other same iterators)
          else:
            cross-match for optionals
            if (optional):
              regex_list.add((<optional_square>)?)
    elif (string mismatch):
      regex_list.add((.*?))
    elif (string-tag mismatch):
      regex_list.add((.*?))
      advance only for the list with string
return regex_list
```

Literatura

[1] Crescenzi V., Mecca G., Merialdo P. RoadRunner: Towards Automatic Data Extraction from Large Web .[online]. Na voljo na spletu: http://vldb.org/conf/2001/P109.pdf?fbclid=IwAR2Mag08vAML6sPBz7qA7h-peca2MDBcZObpK76gQ4dC2GS3vNW2kSWoWUc