Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Matevž Lapajne, Simon Klavžar in Tine Črnugelj

Ekstrakcija podatkov s spleta

Domača naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta
MENTOR: prof. dr. Marko Bajec, doc. dr. Slavko Žitnik

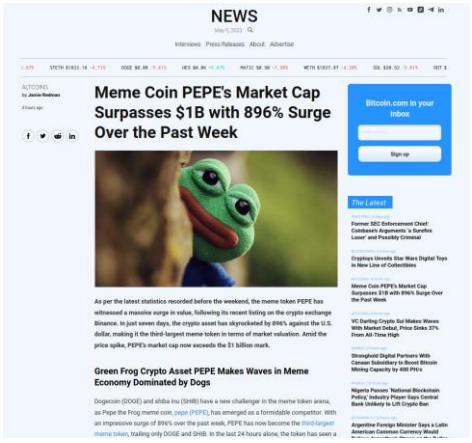
Poročilo predstavlja implementacijo ekstrakcije podatkov s tremi različnimi pristopi: z uporabo regularnih izrazov, z uporabo jezika XPath in z algoritmom RoadRunner. Uspešnost ekstrakcije smo testirali spletnih straneh iz domen rtvslo.si, overstock.com, news.bitcoin.com in zacimbe.si.

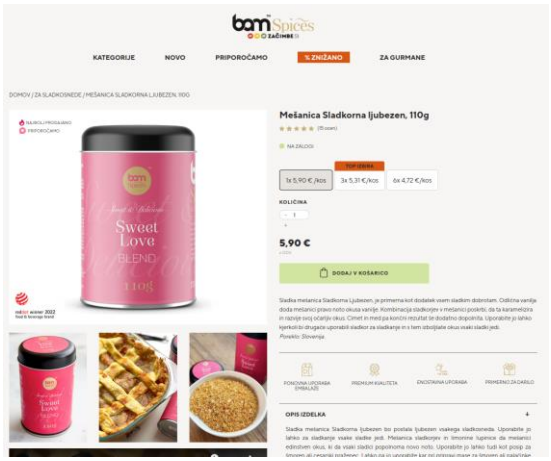
1. Uvod

Naloga je bila ustvariti program v programskem jeziku Python, ki bo uporabil XPATH, regularne izraze in algoritem Road Runner za pridobitev želenih podatkov iz spletne strani.

2. Izbira dodatnih spletnih strani

Za izdelavo naloge sta bile izbrane zraven že dodeljenih spletnih domen, stran za novice o kriptovalutah news.bitcoin.com in spletna stran zacimbe.si, ki se uporablja za prodajo začimb.





3. Implementacija

3. 1 Regularni izrazi

Regularni izrazi (angl. "regular expressions" ali "regex") so izrazi, ki se uporabljajo za iskanje in ujemanje vzorcev v besedilu. So močno orodje za obdelavo in manipulacijo nizov, ki omogoča napredno iskanje, filtriranje in zamenjavo besed ali vzorcev v besedilu. Dandanes se v namene zbiranja informacij iz spleta ne uporabljajo več tako pogosto.

Stran: rtvslo.si

```
title_reg = r"<h1>(.*)</h1>"
subtitle_reg = r'<div class="subtitle">(.*)</div>'
lead_reg = r'<p class="lead">(.*)</p>'
author_re = r'<div class="author-name">(.*)</div>'
published_time_re = r'<div class="publish-meta">\n\t\t(.*)<br>'
content_re = r'<div *?class="article-body">(.*)<div class="gallery">'
```

Stran: overstock.com

```
title_re = r'</table></td><td valign="top"> \n<a href="[^\"]*"><b>(.*?)</b>'
content_re = r'<span class="normal">([^<]*)<'
list_price_re = r'<s>([^<]*)</s>'
price_re = r'<span class="bigred"><b>([^<]*)</b></span>'
saving_re = r'<span class="littleorange">([^\s]*)\s[^<]*</span>'
saving_percent_re = r'<span class="littleorange">[^\(]*\(([^<]*)\)</span>'
```

Stran: zacimbe.si

```
title_re = r'<div class="product_content_box">\s.*<h1>([^<]*)</h1>'
content_re = r'<div class="short">\s*<p[^<]*>([^<]*)<'
opis_re = r'<div class="card-body editor_text">\s*(?:<[^>]+>)?([^<]*)<'
price_re = r'<div class="price">([^<]*)</div>'
dostava_re = r'<div class="additional_data">\s.*</h4>([^<]*)<'
saving_percent_re = r'<span>Neto masa:</span>([^<]*)<'
```

Najpogosteje uporabljeno zaporedje znakov »(.*)«, »([^<]*)« predstavljajo skupino za zajemanje, s pomočjo katerega zajamemo poljubno zaporedje znakov z izjemo prelomov vrstic (\n ali \t). Skupina znakov »*?« ali ».*«, »\s.*« med ali zunaj elementov, zamenjajo poljuben tekst in nam omogočijo izbiro brez, da bi poznali celotne lastnosti elementa. Znakovni niz »\s*(?:<[^>]+>)?([^<]*)« predstavlja izbiro vrednosti elementa in vrednosti njegovih otrok, če obstajajo. Najbolj kompleksna zveza elementov »(?!(?:<img|<strong|<em))([\s\S]*?)« predstavlja izbiro vsebino p elementa, če ta ne vsebuje katerih od zapisanih značk (<img|<strong|<em).

3.2 XPath

Predložena koda je sestavljena iz izrazov XPath, ki se uporabljajo za pridobivanje določenih informacij iz različnih spletnih mest. Oglejmo si vsako spletno mesto in ustrezne izraze XPath, da bi razumeli, kateri podatki so ciljni.

stran: rtvslo.si

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

stran: overstock.com

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

stran: zacimbe.si

```
title = '//div[@class="product_content_box"]/h1/text()'
content = '//div[@class="short"]/p[1]/text()'
description = '//div[@class="card-body editor_text"]//text()'
ingredients = '//div[@id="product_2"]/div[@class="card-body"]/div[@class="text editor_text"]/p/text()'
storage = '//div[@id="product_2"]/div[@class="card-body"]/div[@class="text editor_text"]/text()'
price = '//div[@class="price"]/text()'
delivery = '//div[@class="additional_data"]/text()'
postage = '//div[@class="additional_data"]/ul[@class="basic_list"]/li[1]/text()'
weight = '//div[@class="additional_data"]/ul[@class="basic_list"]/li[2]/text()'
dimensions = '//div[@class="additional_data"]/ul[@class="basic_list"]/li[3]/text()'
```

Simbol "@" se uporablja za označevanje atributa elementa XML. V kombinaciji z imenom elementa omogoča sklicevanje na določen atribut v tem elementu. Za izbiro določenega atributa v elementu se uporabilja simbol "@" , ki nam omogoči ozek izbor želenih elementov. Oglati oklepaji, ki se pojavljajo ob imenu elementa nam omejijo izbor glede na položaj elementa, ki se nahaja v drugem. Dve zaporedni poševnici naprej nakazujeta na možnost pojava poljubnega elementa med njima.

3.3 RoadRunner

Implementacije algoritma RoadRunner smo se lotili na dva različna načina, vendar nam nobenega od pristopov ni uspelo implementirati do točke, kjer bi dobili pravilen izhod na naših testnih straneh. Pri prvem pristopu (psevdokoda je v prilogi) smo sledili postopku, opisanem v [1], vendar smo prišli do zapletov pri implementaciji rekurzije in algoritma tako nismo dokončali. Pri lažjih primerih odsekov HTML strani iz [1] je bil algoritem sicer uspešen, pri testiranju na straneh iz našega nabora pa algoritem ni bil uspešen.

Pri drugem pristopu (psevdokoda je spodaj) implementacije algoritma RoadRunner smo si pomagali s psevdokodo avtorjev algoritma RoadRunner [2], s to izjemo, da nismo sproti gradili tabele ovojnice (ang. wrapper), ampak sproti manipulirali prvotno tabelo HTML oznak. Pri testiranju na bolj zahtevnih spletnih straneh smo ugotovili, da algoritem vrača le delno pravilne rezultate, ki so precej daljši od pravilnih izhodov.

```
function road_runner(html1, html2)
  remove attributes from both htmls
  while not at end of both arrays do
    if string mismatch then
      mark data with PCDATA
    end
    else if tag mismatch then
      find section that represents a whole element and determine in which array
      check if section is repeating
      if is repeating then
        save section in array of repeating sections
        if section not in array that represents a wrapper then
          add section to array that represents a wrapper
        end
      end
    end
    else
      check if section is optional
      if optional then
        save section in array of optional sections
        if section not in array that represents a wrapper then
          add section to array that represents a wrapper;
        end
      end
    end
  end
  join array that represents a wrapper into one string
  for sections in array of repeating sections do
    index = index of first occurrence of section in wrapper string
    remove all occurrences of a section in wrapper string
    add ”(section)+” to the wrapper string on a wrapper string at index
  end
  for sections in array of optional sections do
    index = index of first occurrence of section in wrapper string
    remove occurrence of a section in wrapper string
    add ”(section)?” to the wrapper string on a wrapper string at index
  end
  return wrapper string
```

Psevdokoda algortma RoadRunner pri drugem pristopu.

4. Zaključek

V sklopu naloge smo poskušali implementirali ekstrakcijo podatkov s spleta na tri različne načine. Uspešno smo implementirali prva dva pristopa (regularni izrazi in XPath), pri implementaciji algoritma RoadRunner pa smo naleteli na več težav in pomanjkanje časa, zato nam implementacije ni uspelo dokončati.

5. Viri

[1] V. Crescenzi in G. Mecca, „Automatic information extraction from large websites“, *J. ACM*, let. 51, št. 5, str. 731–779, sep. 2004, doi: 10.1145/1017460.1017462

[2] V. Crescenzi, G. Mecca, in P. Merialdo, „RoadRunner: Towards Automatic Data Extraction from Large Web Sites“, v *Proceedings of the 27th International Conference on Very Large Data Bases*, v VLDB ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., sep. 2001, str. 109–118.

6. Priloge

6. 1 Psevdokoda prvega pristopa

```
function road_runner(html1, html2)
  pre-process html with BeautifulSoup (remove attributes, remove html comments)
  tokenize html
  initialize wrapper = html1
  while not at end of both arrays do
    if string mismatch then
      set element in wrapper to '#PCDATA'
    end
    else if tag mismatch then
      identify terminal tag (one element before mismatch)
      search html1 forwards for occurrence of terminal tag
        if terminal tag found then
          compare potential square element by element backwards with initial square
            if string mismatch occurs then
              label both elements as '#PCDATA'
            if tag mismatch occurs then
              end comparison
          if no tag mismatch is found then
            search wrapper backwards and forwards for all contiguous instances of the square found
            replace area of iteration in wrapper with regular expression for iteration
            end search
      search html2 forwards for occurrence of terminal tag
        if terminal tag found then
          compare potential square element by element backwards with initial square
            if string mismatch occurs then
              label both elements as '#PCDATA'
            if tag mismatch occurs then
              end comparison
          if no tag mismatch is found then
            search wrapper backwards and forwards for all contiguous instances of the square found
            replace area of iteration in wrapper with regular expression for iteration
            end search
      if no iterators found then
        search forwards both html1 and html2 for re-occurrence of mismatching tags
          if mismatching tag of html2 is found on html1 then
            replace content in wrapper from current to re-occurrence with regular expression for op-
tional
          if mismatching tag of html1 is found on html2 then
            insert regular expression of optional content into wrapper at current index

      end
```

Psevdokoda algortma RoadRunner pri prvem pristopu.

6. 2 RoadRunner izhodi

Izhodi algoritma RoadRunner se zaradi obsežnosti nahajajo v datotekah:

Prvi pristop: *pa2/implementation-extraction/outputs/roadrunner_v01_outputs*
Drugi pristop: *pa2/implementation-extraction/outputs/roadrunner_v02_outputs*