Ekstrakcija strukturiranih podatkov s spleta

Druga naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta

Žiga Kleine
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Email: zk5343@student.uni-lj.si

Tim Križnik
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Email: tk0730@student.uni-lj.si

Luka Železnik
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Email: lz6356@student.uni-lj.si

I. Povzetek

V tem poročilu bomo predstavili implementacijo in rezultate druge naloge pri predmetu Iskanje in ekstrakcija podatkov s spleta. Naloga vključuje implementacijo treh različnih metod za ekstrakcijo strukturiranih podatkov s spleta, in sicer ekstrakcijo s pomočjo regularnih izrazov, ekstrakcijo s pomočjo XPath izrazov, in uporabo algoritma za avtomatsko ekstrakcijo RoadRuner. Pri vseh treh metodah bomo opisali njihovo delovanje in predstavili njihove izhode.

II. Uvod

Ekstrakcija strukturiranih podatkov s spleta je področje računalništva, ki se ukvarja s izluščevanjem uporabnih podatkov s spletnih strani. Ker so spletne strani na internetu v prvi vrsti narejene z namenom, da jih bodo brali ljudje, se pri strojnem branju teh strani pogosto pojavijo težave. Zaradi velike razširjenosti in obsega modernega spleta, spletne strani povečini ne ponujajo strukture podatkov, ki bi bila standardizirana, in tako strojno berljiva. Zato za ekstrakcijo potrebujemo spletne ovojnice. Spletna ovojnica definira postopek, kako iz določene strani pridobimo uporabne podatke. Do najučinkovitejše ekstrakcije pridemo, če takšne ovojnice ročno generiramo, a so taki postopki pogosto precej zamudni, zato poskušamo generacijo ovojnic avtomatizirati z uporabo algoritmov za avtomatsko generacijo ovojnic. V tem poročilu bomo predstavili ekstrakciijo podatkov s tremi različnimi metodami, pri katerih dve uporabljata ročno generirane ovojnice, ena pa avtomatsko generirane.

III. Implementacija

Celotna naloga je implementirana v programskem jeziku Python, dostopna pa je na naslovu https://github.com/LukaZeleznik/wieramemo_vase/tree/main/pa2.

Podatke smo s tremi različnimi algoritmi ekstrahirali iz šestih različnih spletnih strani, po dve strani sta iz vsake od treh različnih spletnih domen, https://www.rtvslo.si/, https://www.overstock.com/ in https://www.imdb.com/. Spletni strani iz slednje domene smo dodali sami, medtem ko sta bili rtvslo in overstock dani v gradivu za nalogo.

Z vsake spletne strani smo izluščili enega ali več ”data recordov”, ki so vsebovali ”data iteme”. Data record predstavlja neke vrste objekt, ki vsebuje več atributov ali data itemov. Spletne strani lahko glede na ekstrahirane podatke delimo na strani seznamov in strani detajlov. Ekstrahirane podatke s strani seznamov lahko tako prikažemo kot seznam data recordov z enakimi data itemi, strani detajlov pa kot en sam data record. Pri straneh z domene rtvslo gre za detajlne strani, pri overstock pa za strani seznamov. Pri podatkih s strani z domene imdb gre za kombinacijo obeh prej omenjenih oblik.

Iz spletnih strani na rtvslo smo izluščili en data record z data itemi ”Author”, ”Title”, ”PublishedTime”, ”SubTitle”, ”Lead” in ”Content”, ki smo jih nato strukturirali v json datoteko 1. Z overstock strani smo izluščili seznam data recordov z atributi ”Title”, ”ListPrice”, ”Price”, ”Saving”, ”SavingPercent” in ”Content” 2. Pri straneh na domeni imdb smo najprej ekstrahirali atributa ”ListTitle” in ”ListSubtitle”, nato pa še seznam objektov ”List” z atributi ”Rank”, ”Title”, ”Year” in ”Rating”.


```
{
    "Author": "Miha Merljak",
    "Title": "Audi A6 50 TDI quattro: nemir v
      premijskem razredu",
    "PublishedTime": "28. december 2018 ob 08:51",
    "SubTitle": "Test nove generacije",
    "Lead": "To je novi audi A6. V razred najdražjih
      in najbolj premijskih žrebcev je vnesel nemir,
      še preden je sploh zapeljal na parkirni prostor
      , rezerviran za izvršnega direktorja. ",
    "Content": "Samo poglejte njegovo masko – to
      ogromno satovje z radarji na takem položaju, da
      se ti na avtocesti tudi pri 120 km/h vsi
      spoštljivo umikajo, saj so prepričani, da gre
      za Pahorjev ali Šarčev avto. Seveda, novi A6
      lahko cesto in promet skenira s kar petimi
      radarji, petimi kamerami..."
}
```


Fig. 1. Json struktura podatkov, izluščenih iz strani na portalu rtvslo.

A. RegEx implementacija

Za ekstrakcijo s regularnimi izrazi smo uporabili Pythonovo knjižnico **re**. Bolj specifično metodo **findall**. Ta nam vrne vse rezultate, ki se ujemajo s podanim regularnim izrazom. Uporabili smo tudi zastavico **re.DOTALL**, ki nam je omogočila iskanje po vseh vrsticah in podobnih posebnih znakih. V funkcijo smo v oklepaje vnesli regex niz, ki nam je vrnil vse znake med dvema nizoma znakov. Če je bilo potrebno smo dobljen niz še naprej očistili s uporabo Pythonove **replace** funkcije. Pridobljene nize smo nato shranili v svoje objekte.

Seznam uporabljenih RegEx izrazov:

```
[
    {
        "Title": "10-kt. Seven Diamond Ladies Heart
          Ring (0.08 TW)",
        "ListPrice": "$149.00",
        "Price": "$69.99",
        "Saving": "$79.01",
        "SavingPercent": "53%",
        "Content": "This ladies fashion ring dazzles
          with hearts and diamonds. The gold band is
          crafted into delicate, open hearts. Seven
          brilliant-cut diamonds add a bit of sparkle
          . Click here to purchase."
    },
    {
        "Title": "10-Kt. Diamond Ring (.25 TW)",
        "ListPrice": "$250.00",
        "Price": "$74.90",
        "Saving": "$175.10",
        "SavingPercent": "70%",
        "Content": "Nineteen round diamonds accent
          this 10-karat yellow gold ring with
          filigree accents.Click here to purchase."
    }
    // ...
]
```


Fig. 2. Json struktura podatkov, izluščenih iz strani na portalu overstock


```
{
    "ListTitle": "Top Rated Movies",
    "ListSubitle": "Top 250 as rated by IMDb Users",
    "List": [
        {
            "Rank": "1.",
            "Title": "The Shawshank Redemption",
            "Year": "1994",
            "Rating": "9.2"
        },
        {
            "Rank": "2.",
            "Title": "The Godfather",
            "Year": "1972",
            "Rating": "9.1"
        },
        {
            "Rank": "3.",
            "Title": "The Godfather: Part II",
            "Year": "1974",
            "Rating": "9.0"
        }
        // ...
    ]
}
```


Fig. 3. Json struktura podatkov, izluščenih iz strani na portalu imdb.

```
#overstock:
    title1 = 'PROD_ID=([0-9]+?)"><b>(.+?)</b>',
    listPrice1 = '''List Price:</b></td>\n*\s*<td align="left" nowrap="nowrap"><s>(.+?)</s>'''
    price1 = '''class="bigred"><b>(.+?)</b></span></td>'''
    priceSaving1 = '''<td align="left" nowrap="nowrap"><span class="littleorange">(.+?)<{1}|\n{1}'''
    content1 = '''<td valign="top"><span class="normal">(.+?)</b></span>'''

#rtv:
    title1 = '<title>(.+?)</title>'
    subtitle1 = '<div class="subtitle">(.+?)</div>'
    lead1 = '<p class="lead">(.+?)</p>'
    author1 = '<div class="author-name">(.+?)</div>'
    publishedTime1 = '<div class="publish-meta">(.+?)<br>'
    content1 = '<article class="article">.*?(<p.*>(.*?)<\/p>).*?<div class="gallery">'

#imdb:
    rank1 = '<td class="titleColumn">(.+?)<a'
    title1 = '''class="titleColumn">(.+?)" title="(.+?)">(.+?)</a>'''
    year1 = '<span class="secondaryInfo">[(](.+?)[)]</span>'
    rating1 = ' user ratings">(.+?)</strong>'
```

Fig. 4. Seznam uporabljenih RegEx izrazov.

B. XPath implementacija

Pri metodi ekstrakcije z metodo XPath smo podatke iz HTML dokumentov pridobivali s pomočjo jezika XPath, ki je bil sprva razvit za potrebe poizvedb po podatkih v XML dokumentih. Ker gre pri HTML in XML dokumentih, podobno kot pri datotečnih sistemih, za drevesno strukturo, je sintaksa XPath podobna sintaksi datotečnih direktorijev. Po Xpath direktoriju namreč iščemo tako, da podamo pot imen HTML elementov, ločenih z poševnicami, ki opiše podatke, ki jih želimo ekstrahirati. Pri tem nam ni treba začeti v korenu dokumenta, prav tako pa lahko iščemo elemente z specifičnimi atributi. Spodaj lahko vidimo seznam XPath izrazov, ki smo jih uporabili 5.

```
rtvslo:

    author: '//div[@class="author-timestamp"]/strong/text()'
    title: '//header[@class="article-header"]/h1/text()'
    published_time: '//div[@class="author-timestamp"]/text()'
    sub_title: '//div[@class="subtitle"]/text()'
    lead: '//p[@class="lead"]/text()'
    content: '//article[@class="article"]/p'

overstock:

    titles: '/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()'
    list_prices: '/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/
        tr[*]/td[2]/table/tbody/tr/td[1]/table/tbody/tr[1]/td[2]/s/text()'
    prices: '/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()'
    savings_and_savings_percents: '/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()'
    contents: '/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]')

imdb:

    list_titles: '//*[@id="main"]/div/span/div/div/h1/text()'
    list_subtitle: '//*[@id="main"]/div/span/div/div/div[2]/text()'

    ranks: '//tbody[@class="lister-list"]/tr[*]/td[2]/text()'
    titles: '//tbody[@class="lister-list"]/tr[*]/td[2]/a/text()'
    years: '//span[@class="secondaryInfo"]/text()'
    ratings: '//tbody[@class="lister-list"]/tr[*]/td[3]/strong/text()'
```

Fig. 5. Seznam uporabljenih XPath izrazov.

C. RoadRunner implementacija

Pred algoritmom smo HTML kodo pretvorili v strukturiran objekt knjižnice *BeautifulSoup* in nad njim izvedli operacijo *prettify* s katero se zadovoljivo očisti nepravilnosti v kodi (zapre nezaprte oznake, ipd.). Z metodo *decompose* smo odstranili še vse oznake tipa 'script', v nadaljnjem delu z oznakami pa smo izpustili še ostale nevsebinske oznake kot so 'b', 'br', 'em', 'hr'... Čeprav drevesna struktura *soup* objekta omogoča dobro indeksiranje po DOM, smo dobljen objekt z rekurzivno funkcijo spremenili v tabelo vseh oznak - za stran wrapper in comparison. Ker nas zanima vsebina strani je vhodni elemet vanjo *body*. V tej funkciji *tag.children* vrne seznam potomcev dobimo, iz katerega filtriramo objekte tipa NavigableString (t.j. razni prazni prostori v kodi). Seznama smo nato primerjali, pri čemer je bil cilj poiskati oznake in besedilo, ki se razlikujejo. Enaki deli predstavljajo ovojnico. Če se v sprehodu skozi seznama oznaki ujemata, se pregleda še njun tekst (*.findAll(text=True)*), odstrani whitespace in če sta različna se to označi v ovojnici. Slabost zaporedne iteracije, ki smo jo reševali, je dohitevanje ponavljajočih oznak (npr. različno dolge tabele) ter ostale neskladnosti, ki hitro zapletejo ta način primerjanja.

Fig. 6. Roadrunner ovojnica.

IV. Rezultati

Čeprav smo imeli pri RegEx implementaciji naprej malce problemov, smo na koncu uspeli izluščiti pomembne podatke iz vseh podanih spletnih strani. Največ problemov smo imeli z pridobitvijo "Content" polja iz rtv.si, saj smo morali uporabiti ugnezden iskalni niz, katerega smo nato v večih korakih očistili nepotrebnih tekstovnih polj.

JSON datoteke, ki smo jih pridobili kot izhode, smo priložili v mapi output_jsons_regex in output_jsons_xpath, ki se nahajata poleg poročila.

V. Zaključek

V nalogi smo pokazali, da je ekstrakcija podatkov iz spletnih strani možna na več načinov. Vsak od njih ima svoje prednosti in slabosti, težavnost implementacije pa variira glede na izbrana orodja in predznanje. Pri RegEx in XPath implementaciji je bil cilj naloge dosežen, za izboljšave na RoadRunner algoritmu pa nam je žal zmanjkalo časa. Vseeno smo dobili občutek kakšne težave nastopajo pri avtomatizirani ekstrakciji in kako reševati kompleksnost strukture spletnih strani z določenimi hevristikami.