Iskanje in ekstrakcija podatkov s spleta
1. seminarska naloga

Andraž Krašovec, Luka Podgoršek in Iztok Ramovš

April 2019

1 Uvod

Web crawlerji ali lepše slovensko spletni pajki, so programi, namenjeni sistematični in avtomatski obdelavi spletnih strani. Najbolj znan primer uporabe je zagotovo indeksiranje spletnih strani, na kateri temeljijo spletni iskalniki. Osnovni elementi vsakega spletnega pajka:

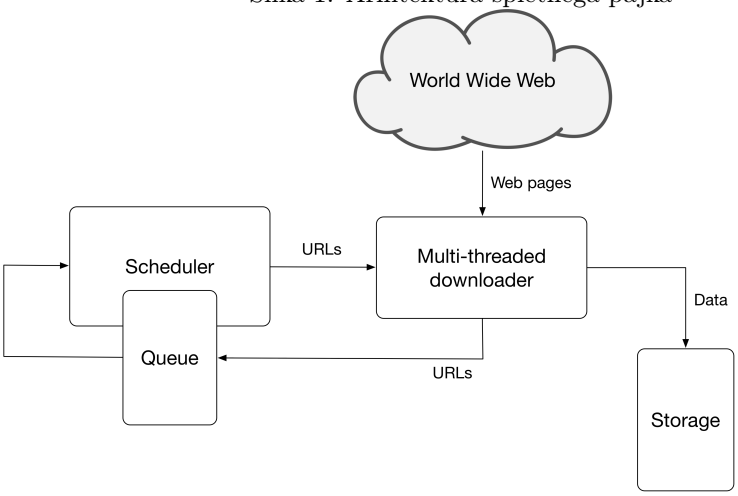
- HTTP prenašalec vsebin (downloader),
- HTTP upodabljalnik (renderer),
- zaznavalec duplikatov,
- podatkovni razčlenjevalnik,
- URL frontier,
- podatkovna baza.

Prav implementacija teh gradnikov je cilj naše seminarske naloge. Osnovna arhitektura spletnega pajka je predstavljena na sliki 1.

2 Implementacija pajka

Pred začetkom implementacije smo si postavili razvojno okolje in se znotraj ekipe dogovorili katere tehnologije bomo uporabili. Kot programski jezik smo izbrali *python* s programskim ogrodjem *Django*. Nato smo uporabili pripravljeno SQL skripto in povezali podatkovno bazo z Django-m. Sledila je implementacija posameznih modulov spletnega pajka. Kot prvi modul smo razvili modul za prenašanje vsebine s spleta. Uporabili smo knjižnico *requests*. Modul nam je omogočal izvajanje HEAD in GET zahtevkov, prenašanje vsebine spletne strani, prenašanje vsebine sitemap-ov in teksovnih datotek robots.txt.

Slika 1: Arhitektura spletnega pajka



2.1 Modul Extractor

Drugi modul, ki smo ga implementirali smo poimenovali Extractor. Ta modul je namenjen ekstrakciji podatkov s prenešenih vsebin. Omogočal nam je čiščenje HTML-ja s čimer smo zagotovili pravilno delovanje razčlenjevalnika. Najpomembnejši del modula predstavila funkcija za ekstrakcijo URL-jev s prenesenega HTML-ja. Za iskanje značk smo uporabili knjižnico Beautifulsoup4. S pomočjo te knjižnice smo poiskali vse html <a> elemente in shranili vsebino zapisano v href atributu. To smo nadgradili še z uporabo regularnih izrazov. Ker se na modernih spletnih straneh uporablja vse več JavaScript-a, tudi navigacija med spletnimi stranmi ne deluje več samo preko <a> elementov. Napisali smo regularni izraz, ki poišče URL-je zapisane v funkcijah iz javascript location api-ja.

Uporabljeni regularni izraz:

```
regex = r"
    location.assign\((.*)\)|
    location.replace\((.*)\)|
    location\.href\="(.*)"|
    location\.href\='(.*)'
  "
```

Extractor modul vsebuje še funkcije za iskanje slik, dodatnih datotek na spletni strani (iz nabora podanega v navodilih) in funkcijo za razčlenjevanje sitemap-a. Iz sitemap.xml datoteke ekstrahira vse URL-je. Tukaj smo naleteli na problem, saj se uporabljata 2 različna standarda za strukturo datoteke sitemap.xml. Ker nismo želeli spuščati URL-jev, ki bi jih lahko na enostaven način pridobili, smo podprli oba formata datotek sitemap.xml. Modul vedno vrača seznam URL-jev, vendar so lahko URL-ji na spletni strani nepopolni. Ne-

katere povezave so relativne, druge so lahko v popolnoma neuporabe. Npr <a href=""/>

2.2 Modul UrlUtils

Ker nismo želeli v frontir shranjevati URL-jev, ki ne obstajajo oz. niso v pravem formati, smo napisali modul, ki poskrbi za preverjanje in popravljanje URL-jev. Uporabili smo standardno python knjižnico *urllib*. S tem modulom smo zagotovili, da bo modul, ki skrbi za prenos datotek vedno dobil url, ki ga lahko zahteva. Za primere, ko ekstrahiran url ne obstaja smo poskrbeli v modulu za prenos vsebin.

2.3 Modul Frontier

Ko smo imeli pripravljene in stestirane osnovne komponente spletnega pajka, smo začeli delati na modulu frontier. Ta modul je predstavljal frontir spletnega pajka. Uporabili smo knjižnico queue implementirano kot FIFO vrsto. Ta modul predstavlja enega izmed najpomembnejših elementov spletnega pajka. Prva zahteva, ki smo jo morali podpreti s tem modulom je bila, da zagotavlja varno in konsistentno delovanje med večimi nitmi. Nato smo implementirali funkcijo, kateri smo podali nov URL ta pa je url preslikala v bazo kot Site ali Page. Če smo neke domeno našli prvič, smo domeno urlja shranili v Site. Poleg tega pa smo prenesli vsebino sitemapa in robots.txt ter zapisali vsebini v bazo. Na koncu pa še poskrbeli za pravilen zapis v Link tabelo. V primeru, da je šlo za URL, katera domena je že obstajala v Site smo ta url zapisali kot Page in označili kot frontier. Na koncu pa smo Page objekt zapisali še v FIFO vrsto. Zaradi načina implementacije smo imeli podvojen frontier. En frontier je bil zapisan v bazo, drugi v vrsto med delovanjem samega programa. Ker smo imeli stanje frontira shranjen tudi v bazi, smo dopisali funkcijo, ki je znala napolniti vrsto na podlagi informacij v bazi. Tako smo lahko kadarkoli ustavili delovanje pajka in ga nato ponovno pognali, pajek pa je nadaljeval s preiskovanjem tam kjer je končal. To obnašanje pajka lahko nadziramo z zagonskimi parametri.

2.4 Modul Crawler

Nato smo se lotili implementacije modula, ki smo ga poimenovali Crawler. Ta modul predstavlja dejanskega pajka. Glavna funkcija pajka sledi naslednjim korakom:

- Pridobi element iz frontierja
- Na podlagi pridobljenega elementa nastavi trenutni url, domeno in crawl delay
- Počakaj na crawl delay
- Prenesi vsebino spletne strani

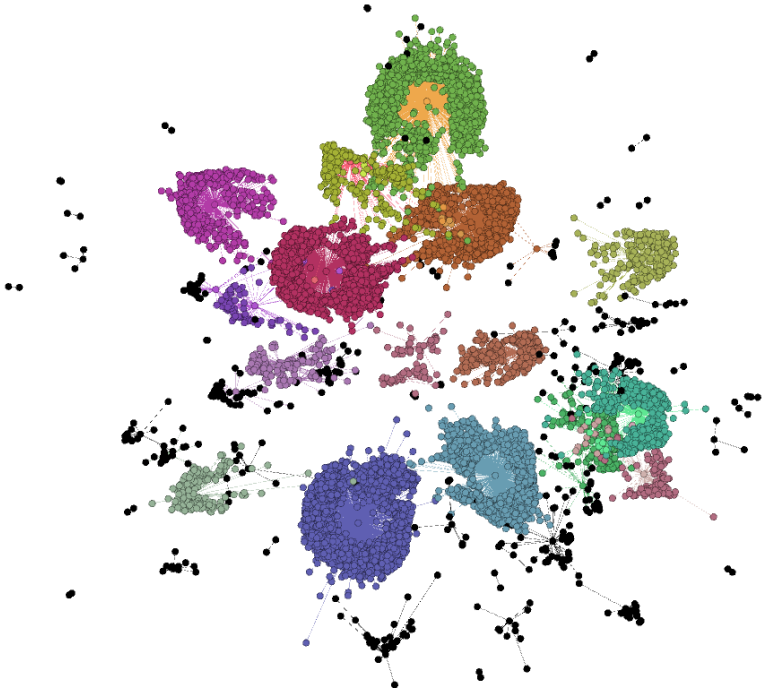
- Prečisti pridobljen html
- Preparsaj in ekstrahiraj urlje, urlje slik in urlje dokumentov
- Preparsaj in ekstrahiraj urlje, urlje slik in urlje dokumentov
- Preveri če obstaja robots.txt
- Če obstaja jo podaj RobotParser modulu
- Pridobi podatke o crawl delay
- Poišči urlje sitemapov v robots.txt
- Prenesi nove sitemape ter shranjen sitemap iz baze in ekstrahiraj urlje
- Prečisti vse URLje in odstrani tiste, ki niso dovoljeni v robots.txt
- Zakleni dostop do baze
  - Shrani urlje v frontier
  - Prenesi slike iz dovoljenih domen
  - Prenesi dokumente iz dovoljenih domen
- Sprosti dostop do baze

Pri implementaciji pajka smo se srečali s problemom crawl delay-a. To smo rešili tako, da smo v page tabelo shranili crawl_delay atribut. Ta atribut smo nato uporabili v pajku, ki je čakal ustrezno število sekund, saj nismo želeli izvesti DDOS napada na strežnik. Privzeto vrednost za crawl delay smo nastavili na 4s. Poleg crawl delay-a smo morali filtrirati nedovoljene urlje. Zato smo napisali modul RobotsParser. RobotsParser modul uporablja knjižnico urllib.robotparser. S pomočjo te knjižnice smo prebrali robots.txt datoteko in odstranili urlje do katerih pajek ne sme dostopati. Nato smo se lotili testiranja delovanja celotne rešitve. V vsak modul smo dopisali funkcije za logiranje delovanja, log datoteko pa shranili na disk. Zaradi uporabe loggov smo točno vedeli, kako deluje pajek in če kje pride do težav. Med testiranjem smo odstranili nekaj manjših hroščev in se lotili implementacije Scedulerja. Primer izpisa programa v log (glej 5.1).

2.5 Modul ThreadManager

Modul scheduler smo si pustili za konec, saj smo želeli najprej implementirati in stestirati celotno rešitev na eni niti. Modul zažene toliko niti kolikor mu jih podamo v zagonskih parametrih. Poleg zagona niti preverja, če vse niti spijo. Če vse niti spijo pomeni, da je frontier prazen in da smo končali s crawlanjem. Dodali smo tudi preverjanje, če pride do napake na posamezni niti kar povzroči, da nit ne deluje več. V takšnem primeru sceduler ponovno zažene instanco pajka na tisti niti. Tako smo zagotovili, da paralelno teče več pajkov tudi, če pride do napake.

Slika 2: Vizualizacija prvih 10.000 prenesenih strani.



3 Analiza delovanja

Naš spletni pajek je deloval **17,5 ur**, nato pa smo ga ugasnili, saj je prenesel dovolj veliko število vsebin. Pajek je prenesel **156.170** spletnih strani, v frontirju pa je ostalo še **1.616.699** unikatnih vnosov. V povprečju je pajek obdelal ***8.800*** *spletnih strani na uro*, kar je primerljivo z Apache Nutch.

Ozko grlo delovanja je predstavljala vrednost crawl delayja in dodajanje novih linkov v frontier. Procesor, pomnilniški dostopi in internetni prenos namreč pri zagonu z osmimi nitmi niso bili popolnoma izkoriščeni.

4 Parametri za zagon pajka

```
# Zazeni pajka s 16 nitmi
python manage.py startcrawler 16
# Obnovi stanje iz baze in zazeni pajka s 16 nitmi
python manage.py startcrawler 16 --restore
```

5 Zaključek

Uspešno smo implementirali spletnega pajka brez uporabe namenskih knjižnic. Uporaba programskega jezika Python v navezi z ogrodjem Django se je izkazala za uspešno, saj smo uspeli implementirati spletnega pajka, po hitrosti primerljivega z že uveljavljenimi rešitavmi. Potencialne izboljšave vidimo v naprednejši implementaciji ključavnic, saj trenutna implementacija prepogosto in za predolgo zaklene dostopanje do baze, kar pomeni, da ostale niti prebijejo izdatno količino časa v čakajočem stanju. Naslednja možnost pohitritve našega pajka je prenašanje datotek v ločenih nitih, kar pomeni, da ob prenosu večjih datotek, pajek zopet ne čaka po nepotrebnem.

5.1 Izpis ene iteracije pajka

```
[INFO]: Running crawler
[INFO]: Getting element from frontier
[INFO]: Got obj from frontier
http://www.projekt.e-prostor.gov.si/
[INFO]: Domain http://www.projekt.e-prostor.gov.si
[INFO]: Waiting crawl delay 4 for
http://www.projekt.e-prostor.gov.si/
[INFO]: [HTML]
[INFO]: Downloading page HTML
[INFO]: Get page body for URL:
http://www.projekt.e-prostor.gov.si/
[INFO]: Status code: 200
[INFO]: Status code: [200]
[INFO]: Cleaning HTML
[INFO]: Cleaning html
[INFO]: Parsing URLs
[INFO]: Parsing urls from <a/>
[INFO]: Parsing urls from location api
[INFO]: Found 165
[INFO]: [IMAGE]
[INFO]: Parsing urls from <img>
[INFO]: Found 56 img urls
[INFO]: [FILES]
[INFO]: Parsing urls for files
[INFO]: Found 15 documents
[INFO]: [ROBOTS]
[INFO]: Handle robots.txt
[INFO]: Setting new content
[INFO]: Setting url
http://www.projekt.e-prostor.gov.si/robots.txt for robot parser
[INFO]: Reading robots file
[INFO]: Content saved in robot parser
[INFO]: parsing robots.txt content for sitemap urls
[INFO]: List of sitemaps in robots
['http://www.e-prostor.gov.si/?eID=dd\_googlesitemap']
[INFO]: Crawl delay is None
[INFO]: Request rate is None
[INFO]: [SITEMAP]
[INFO]: Downloading sitemap for
http://www.e-prostor.gov.si/?eID=dd\_googlesitemap
[INFO]: GET sitemap.xml for
http://www.e-prostor.gov.si/?eID=dd\_googlesitemap
[INFO]: FOUND sitemap.xml file
[INFO]: The number of sitemaps are 76
```

```
[INFO]: Appending sitemap urls to all urls 76
[INFO]: Sitemap not found
[INFO]: [URL]
[INFO]: Extracted 241 urls
[INFO]: Removing disallowed URLs
[INFO]: Acquiring lock
[INFO]: Lock acquired
[INFO]: [DATABASE]
[INFO]: saving page
[INFO]: Saving page http://www.projekt.e-prostor.gov.si/...
[INFO]: Paged http://www.projekt.e-prostor.gov.si/ saved.
[INFO]: Adding items to frontier
[INFO]: GET sitemap.xml for http://www.mop.gov.si/sitemap.xml
[INFO]: NO sitemap.xml file
[INFO]: GET robots.txt for http://www.mop.gov.si/robots.txt
[INFO]: FOUND robots.txt file
[INFO]: GET sitemap.xml for http://www.svrk.gov.si/sitemap.xml
[INFO]: NO sitemap.xml file
[INFO]: GET robots.txt for http://www.svrk.gov.si/robots.txt
[INFO]: FOUND robots.txt file
[INFO]: Frontier updated
[INFO]: Saving images
[INFO]: Done
[INFO]: Saving files
[INFO]: Done
[INFO]: Releasing lock
[INFO]: Lock released
```