Spletni pajek

Martin Arsovski, Maja Nikoloska, Emil Batakliev
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Email:
ma6068@student.uni-lj.si
mn2921@student.uni-lj.si
eb4361@student.uni-lj.si

I. Uvod

V tem poročilu je opisana implementacija našega pajka, probleme s katerimi smo se srečali, naše odločitve pri implementaciji in končne rezultate, ki smo jih dobili. Glavna naloga je bila izdelati spletnega pajka, ki se bo premikal čez strani ki so v domeni "gov.si" in bo shranjeval njihove podatke (html vsebino, slike, dokumente itd.). Pajek je trebal bit sestavljen iz komponente

- Prenosnik in upodabljalnik HTTP: za pridobivanje in upodabljanje spletne strani,
- Funkcije za pridobivanje slik in hiperpovezav iz html vsebino,
- Detektor duplikatov
- Url frontier: seznam URL-jev, ki čakajo v vrsti da bi bili obdelani

II. Implementacija

Naš projekt smo izdelali v programskem jeziku Python, ker nam je bil že znan. Ta programski jezik tudi vsebuje veliko knjižnic ki nam pomagajo pri dostopu spletne strani in obdelavo podatkov iz nje.

A. Podatkovna baza

Na začetku smo se ukvarjali z namestitev podatkovne baze. Izbrali smo da bomo uporabljali PgAdmin4. Shemo začetne podatkovene baze smo imeli podano in smo jo naložili. Raširili smo je z enim atributom ki je pripadal tabeli "page". Namen tega atributa je bil da v njega shranimo html vsebimo spletni strani v hash verziji. To polje smo potem uporabljali da bi primerjali html vsebino spletnih stran in smo se na ta način izognili duplikatov. Program db.py je bil sestavljen iz več funkcij z katerimi smo uspostavili komunikacijo z podatkovno bazo. Implementirali smo funkcije za konektiranje in diskonektiranje baze, kreiranje celotne sheme z vsemi tabelami, funkcije za dodajanje spletne strani, slike itd. "insertSite", "insertPage", "insertPageData", "insertImage" v podatkovno bazo in povezavo med spletnimi strani ("insertLink"). Implementirali smo tudi funkcije za pridobivanje podatkov ("getSiteByDomain", "getPageByHash", "getPageByUrl"), ki smo jih rabili za preverjanje ali nekateri "site" / "page" že obstaja.

B. Frontir

Frontira smo se odločili implementirati ločeno od podatkovne baze. Naš frontir je lista. Na začetku smo v njega hranili samo url-je spletnih stran. Ko smo začeli implentirati delovanje pajka, smo sklepali da moramo na nek način hraniti tudi kdo je oče te strani. Zato smo frontira proširili in smo naredili listo v kateri je vsak element bil lista od dva elementa. En je bil url spletne stran ki čaka da pride na vrsto za da bi bila obdelana, drugi je pa bil id, ki je predstavljal id očeta te spletne strani. Za delo z frontirja smo imeli funkcije za dodajanje ("addUrl") in pridobitev ("getUrl") elementa iz frontirja.

C. Crawler

V našem programu "crawler.py" je bila celotna obdelava spletne strani. Na začetku imamo nekatera preverjanja. Če smo prišli do url-ja ki je že v bazi pomeni da smo to spletno stran obdelali. Zato to spletno stran ne obdelujemo ponovno in nadaljujemo na naslednjo. Drugo preverjanje je če smo prišli do url ki je tipa ".zip". Ker takšno obliko ni treba posebi obdelati nadaljujemo na naslednjo stran. Potem poskušamo odpreti spletno stran ki je prišla na vrsti za obdelavo. To naredimo samo v primeru če je od zadnjega poskusa za dostop do strani poteklo 5 sekund. Lahko se zgodi da to ne moremo in imamo "status code" ki ni 200. To pomeni da je prišlo do neke napake (stran ne obstaja ali imamo napako na strežniku). V takšnih primerih dodamo zapis v bazo. Vnesemo novo spletno stran, za katero dodelimo samo "status code" in url. Izvajanje programa nadaljujemo z novo spletno stran, ki je prva na vrsti v frontier. Ignoriramo strani ki niso na domeni "gov.si" ali pa se že nahajajo v listo v katero hranimo strani ki jih ne smemo obiskati(pridobljene iz robots.txt). Pregledamo če je treba dodati nov "site" preko atributa "domain". Če domain naše strani, ki je trebamo obdelati ne najdemo v bazi, dodamo nov "site". Za vsako spletno strano preverjamo, če je tipa HTML, BINARY ali DUPLICATE. Če je tipa BINARY, to pomeni da spletna strana vsebuje binarnih vsebin in pri shranjevanje strani v bazo, označimo da je strana tipa BINARY in binarnih vsebin hranimo v tabeli page_data. Za vsako binarno vsebino preverjamo če je tipa *PDF*, *DOC*, *DOCX*, *PPT* ali *PPTX*, če ni nobena od teh tipov, jo ne shranjujemo v bazi. Odločili smo se da ne je ne shranujemo ker bi morali imeti dodatne primerjave za da bi določili tip dokumenta. Če je strana tipa HTML, vzamemo njeno vsebino in jo hashiramo. Potem

preverjamo, če je strana duplikat. Označevanje da je strana duplikat poteka na ta način, da najprej preverimo, če ta hash vsebina obstaja v bazi.Če obstaja jo označimo kot duplikat (spremenimo page_type_code v DUPLICATE), jo povežemo z originalno (dodamo zapis id-jev obeh straneh v tabeli link), in jo shranimo v bazi. Če hash vsebina ne obstaja v bazi, jo dodamo kot nov zapis. Vsaka povezava, katero hranimo v bazi je kanonična, združujemo *base URL* in *relative URL* in ga nato normaliziramo. To pomeni da so vse strani shranjene na enak način in tako se izognemo nepotrebnih duplikatov (na primer *www.gov.si* in *gov.si*, bi bil duplikat, če ne bi uporabljali kanonično obliko povezave za shranjevanje v bazi). Naš pajek preverja če v vsebino spletne strani najde tudi slike, ki jih izvlečemo iz *img* oznak in povezave, ki jih izvlečemo iz *a* oznak. Najdene slike hranimo v tabeli image, medtem ko najdene povezave dodamo v frontier. Vse povezave do slike, ki se ne odprejo, preskočimo.

III. Rezultati

Program smo štartali z enega pajka in je tekel 2 dni, 6 ur, 50 minut in 48 sekund. Dobili smo naslednje podatke (na levi strani je opis, na desni strani pa je prikazano število ki ustreza opisu):

Site	
Skupno število	91
”Sites” ki imajo robot.txt	36
”Sites” ki imajo sitemap	3

Pages	
Skupno število	29530
HTML page	21880
Binary page	7022
Duplikat	185
Status code 200	29087
Status code 404	225
Status code 500	43
Status code 302	25
Status code 307	2
Status code 403	30
Status code 410	83
Status code 412	1
Status code 502	8
Status code 503	18

Image	
Skupno število	61922
SVG	21192
GIF	8781
PNG	16826
JPG	6210
JPEG	81

Page data	
Skupno število	5740
PDF	4167
DOC	677
DOCX	866
PPT	4
PPTX	26

IV. Zaključek

Naloga je bila zanimiv izziv in smo se veliko naučili. Videli smo da pajek pobere veliko informacije, kot so povezave med strani, kakšne strani vsebuje ena domena..

References

[1] https://www.python.org/
https://www.pgadmin.org/