Seminarska naloga 1 - Spletni pajek

Gal Bumbar, Jan Lampič, Maja Umek

April 2019

1 Uvod

V času, ko nam medmrežje omogoča dostop do skoraj neomejene količine podatkov, so metode za zajem in obdelavo le teh zelo koristne in zaželene. Primer metode za zajem podatkov je spletni pajek, ki v kratkem času po neki vnaprej predpisani strategiji obišče čim več čim bolj relevantnih spletišč.

V poročilu je predstavljen program, ki išče le spletne strani z domeno *gov.si*. Program je implementiran tako, da vzporedno deluje več pajkov, ki pridobivajo različne spletne strani po princpu iskanja v širino. Število pajkov je podano kot vhodni parameter programa.

2 Implementacija

2.1 Hiter pregled

Pred začetkom preiskovanja s seznamom začetnih strani (*angl. seed pages*) in s podatki o bazi inicaliziramo frontier. Ob zagonu programu podamo podatek o želenem številu pajkov. Na podlagi tega odpre bazen povezav do baze (*angl. connection pool*), iz katerega se vsakemu pajku dodeli ena povezava. Dokler ne velja zaustavitveni pogoj, vsak pajek dela sledeče:

1. V frontierju poišče naslednjega kandidata za obdelavo. Pri izboru kandidata upošteva prepovedane strani in zamik (*angl. delay*) med zaporednimi HTTP klici na neko domeno iz `robots.txt`, ter aktivnost ostalih pajov.

2. Ko pridobi primerno stran, jo označi kot zasedeno in prične z obdelavo. Pri tem doda nove povezave v seznam URLjev, prenese slike ter datoteke. Med obdelavo preveri ali je stran vsebinski duplikat kakšne že pregledane strani in ta podatek upošteva.

3. Ko je obdelava zaključena, stran odstrani iz frontierja in jo v bazi označi kot obdelano.

2.2 Inicializacija

V kolikor je ob zagonu frontier prazen, se vanj vstavijo URLji, ki so definirani v spremenljivki `seed_urls` v konfiguracijski datoteki. Če pa ob zagonu programa uporabljena baza ni prazna, nadaljuje z že prej začetim delom. Obstaja tudi možnost, da so bile ob prekinitvi prejšnjega delovanja nekatere strani sredi obdelave, zato zasedenim stranem v frontierju odstrani oznako, da bodo lahko odšle v ponovno obdelavo.

2.3 Frontier

Frontier smo implementirali kot ločeno tabelo `crawldb.frontier`, ki je povezana s tabelo `crawldb.page` preko kjuča `page_id`. Sestavlajo jo trije stoplci: `page_id`, `time_added` (čas, ko je bila stran dodana v frontier) ter `occupied` (logična vrednost, ki nam pove, če stran že obdeluje nek pajek).

Poizvedba po novem URLju iz frontierja deluje tako, da poiščemo stran z najnižjim `time_added`, katerega domena ima kasnejši timestamp `next_access` od trenutnega časa. Opomba: `crawldb.site` smo v ta namen razširil s stoplcem `next_access`, za katerega skrbi komponenta našega programa `robots`.

2.4 Vzporedni pajki

Za implementacijo vzporednih pajkov smo uporabili knjižnico threading (1), s katero lahko posameznega pajka definiraš kot funkcijo, ki se lahko izvaja vzporedno z drugimi enakimi funkcijami za hitrejše dosežen skupni cilj in jih tudi zaženeš vsakega kot svojo nit. Vsak pajek kot argument sprejme povezavo iz bazena povezav, katero kasneje uporablja za poizvedbe in manipulacije z bazo.

Pajek obratuje dokler ne velja zaustavitveni pogoj, za katerega najpreprostejša rešitev bi bila prazen frontier. Ker se ob večih pajkih lahko zgodi, da je frontier samo začasno prazen, damo ob praznem frontierju pajka v spanje, po katerem ponovno poizkusi pridobiti novo stran, če je po določenem številu ponovnih poizkusov frontier še vedno prazen, preneha z delovanjem.

Ob pridobitvi novega URLja iz frontierja mora pajek preveriti, če sme dano stran obiskati, to preveri s fukncijo iz komponente `robots`. Če je ne sme, jo shrani v bazo z `page_status_code` 'FORBIDDEN'. Zakasnitev (*angl. delay*) upošteva že ob tem, ko pridobiva URL, saj je v sami bazi že označeno, kdaj naslednjič lahko dostopamo do spletne strani (več o tem ponovno pri `robots`). Ko po preverjanih konča s stranjo, katero sme obdelati, jo označi kot zasedeno in prične z obdelavo. Po končani obdelavi posodobi tabelo spletne strani in jo izbriše iz frontierja. Program se ustavi, ko se ustavjo vsi pajki.

2.5 Obdelava strani

Pred pridobivanjem vsebine strani, program najprej pošlje HTTP HEAD zahtevo na dani URL, da preveri dosegljivost posamezne strani. V kolikor stran vrne neuspešen odgovor, se stran prestavi na konec frontierja. Če ista stran tudi

ob naslednji poizvedbi vrne neuspešen odgovor, ji nastavi neuspešno statusno kodo in jo odstani iz frontierja.

Nato program s knjižnico `Selenium`(5) pridobi HTML strani. Iz njega izlušči vse URLje strani, ki se nahajajo v `href` ali `onclick` atributih in jih doda v frontier. Poišče tudi vse URLje, ki kažejo na binarne datoteke tipov pdf, doc, docx, ppt in pptx, ter URLje slik, ki se nahajajo v značkah img in imajo eno izmed končnic `png`, `img`, `jpg`, `jpeg`. Slike in preostale omenjene tipe datotek nato pridobi in shrani v podatkovno bazo. Pri pridobivanju datotek smo iz performančnih razlogov omejili količino prenosa na 10MB na posamezno datoteko. Vse URLje se takoj po pridobitvi tudi kanonizira. Na koncu obdelave podatke o strani hrani še v bazo v tabelo `crawldb.page`.

2.6 Detekcija duplikatov

Detekcija duplikatov je ločena na dva dela, detekcija URL duplikatov in detekcija vsebinskih duplikatov.

Pri detekciji URL duplikatov zgolj opravimo poizvedbo po tabeli `crawldb.page`. Če dobimo neprazen seznam strani, ki pripadajo danemu URLju smatramo, da je duplikat. To preverjanje izvajamo pred dodajanjem novih URLjev v frontier.

Pri vsebinskih duplikatih storimo podobno, le da primerjamo `md5 hash` (4) vsebine. To preverjanje izvajamo med obdelavo strani.

2.7 Upoštevanje vsebine robots.txt in sitemap

Ob pridobitvi novega URLja iz frontierja mora pajek najprej preveriti, če dano stran sme obiskati. Pri tem mora dodatno upoštevati vsebino `robots.txt`, če obstaja, ter zamik (*delay*) domene, ukaze User-agent, Allow, Disallow. Vse potrebne informacije pajek pridobi s funkcijo `can_fetch_page` iz komponente robots.

Funkcija kot vhodni argument prejeme URL strani, vrne pa informacijo, če se stran lahko obišče. To stori tako, da za pripadajočo domeno preveri, če ima že definirano vsebino `robots.txt`. V primeru, da vsebine ni, se to prenese in obdela. Iz nje se razbere, če se pripadajočo stran lahko obišče, sitemap domene, zamik domene (če ni določen, se privzame vrednost 4 sekund) ter določi čas naslednjega možnega obiska strani te domene `next_acces`. Funkcija prav tako doda vse strani iz sitemapa v frontier. Ko pa ima domena že določen `robots.txt`, pa funkcija preveri, če se stran lahko obišče ter posodobi čas naslednjega možnega obiska.

3 Težave pri implementaciji

Največ problemov nam je povzročala paralelizacija. Problem z izmenjavo informacij dobro reši uporaba baze, vendar se je tu pojavil problem hkratnih poizvedb. Zaradi teh je prihajalo do večkratne obdelave istih strani. Ker bazen povezav ne preprečuje hkratnih poizvedb, je več pajkov kljub polju occupied v

tabeli crawldb.frontier iz nje vzelo isto stran. Problem smo rešili z zvito uporabo sleep funkcije ter dodatnega preverjanja za duplikate, kot dodatno zagotovilo.

Nekaj težav se je pojavilo tudi pri kanonizaciji URLjev, saj smo pridobili precej nepričakovanih nizov (npr. e-poštni naslovi, nepričakovani nabori znakov, itd.). S problemom smo se delno spoprijeli z regularnimi izrazi in dodatnim dekodiranjem, deloma pa krivdo vseeno pripisujemo ljudem, ki uporabljajo šumnike, ć ter druge nenavadne znake v povezavah.

Imeli smo tudi problem, kjer program ni našel primernih strani v frontierju, čeprav je ob vpogledu v bazo delovalo, kot da bi jih moralo biti veliko. Po dolgotrajnem razhroščevanju smo ugotovili, da sta bili baza in knjižnica datetime na različnih časovnih pasovih. To nam je dalo vedeti, da še nimamo dovolj izkušenj.

4 Rezultati in statistika

Program za preiskovanje poljubnih spletnih strani na celotni domeni gov.si, se je izvajal 36 ur. Pri tem je preiskal 22.594 spletnih strani, kar je povprečno približno 627 strani na uro. Program se je poganjal z različnim številom pajkov, in sicer od 1 do 6 pajkov hkrati. Program smo med zaporednimi poganjanji večkrat popravili, vmes smo tudi stestirali, kako hitro deluje, če ne pobiramo binarnih datotek. V tem primeru je pajek dosegel hitrost okoli 3000 strani na uro. Dodatne statistike o podatkovni bazi spletnega pajka so predstavljene v tabeli 1.

	Količina	**Povprečno količina na domeno**	**Povprečno količina na spletno stran**
Domene	361	-	**-**
Spletne strani	22.594	62,59	**-**
Duplikati	1.774	4,91	**-**
PDF Dokumenti	41.771	115,71	1,85
PPT Dokumenti	217	0,60	0,01
PPTX Dokumenti	197	0,55	0,01
DOC Dokumenti	9721	26,93	0,43
DOCX Dokumenti	3002	8,32	0,13
Slike	122.436	339,16	5,43

Tabela 1: Tabela statistik za crawlanje po vseh .gov domenah.

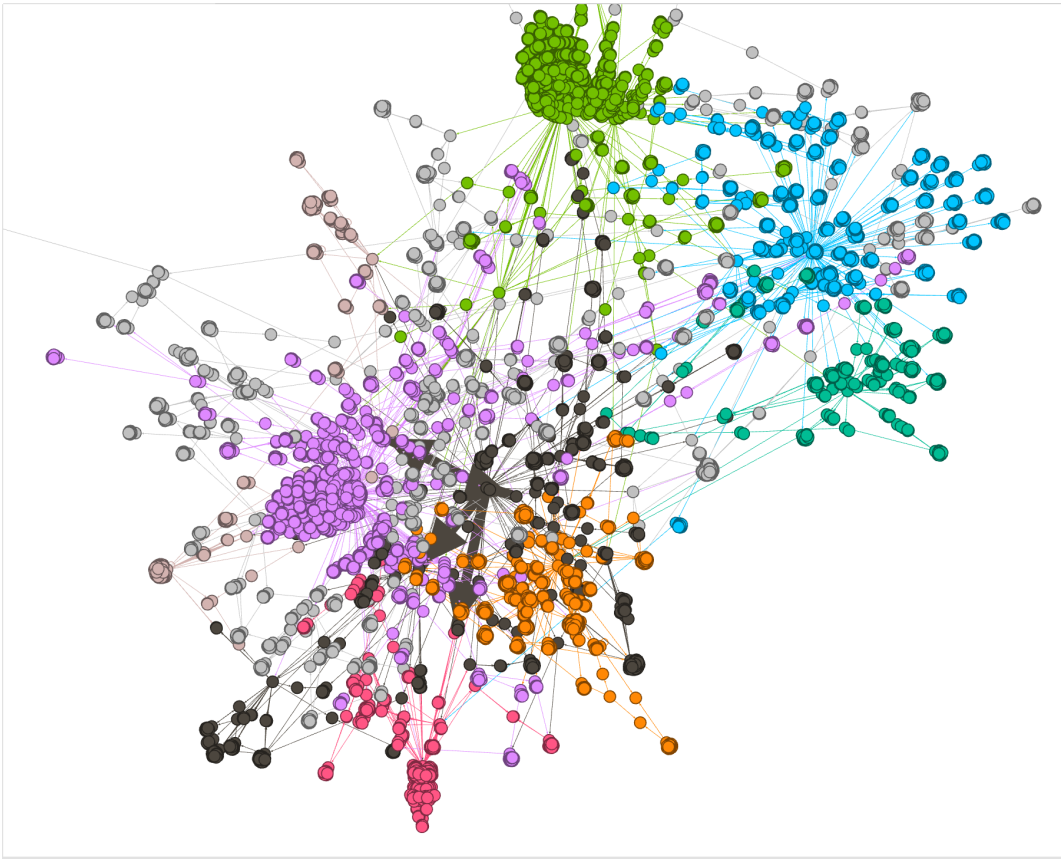
V tabeli 2 pa so predstavljene statistike za preiskovanje spletnih strani po domenah *e-vem.gov.si* in *e-prostor.gov.si*. Ker smo se pri tem procesiranju strani omejili na le dve domeni, je bila hitrost procesiranja zaradi upoštevanja privzete zakasnitve med pridobivanjem strani iste domene precej manjša kot pa pri crawlanju vseh gov.si domen. Program se je po 1112 preiskanih straneh ustavil. Vse skupaj je crawlal 3 ure in pol, torej je na 1 uro pregledal povprečno skoraj 320 strani.

	Količina	Povprečno količina na domeno	Povprečno količina na spletno stran
Domene	4	-	**-**
Spletne strani	1.112	278,00	**-**
Duplikati	152	38,00	**-**
PDF Dokumenti	481	120,25	0,43
PPT Dokumenti	1	0,25	0,00
PPTX Dokumenti	0	0,00	0,00
DOC Dokumenti	64	16,00	0,06
DOCX Dokumenti	36	8,00	0,03
Slike	1.336	334,00	1,20

Tabela 2: Tabela statistik za crawlanje po domenah e-vem.gov.si in e-prostor.gov.si.

5 Vizualizacija

Vizualizacija 1 je narejena s pomočjo orodja Gephi(7) z nastavitvijo za layout ForceAtlas 2.
Uporabljenih je bilo 10 000 povezav iz drugega crawlanja (celotna gov.si domena). Vozlišča grafa predstavljajo spletne strani, povezave pa povezave med temi stranmi. Barve predstavljajo optimalne gruče glede na modularnost.



Slika 1: Vizualizacija prvih 10000 povezav iz drugega crawlanja (celotna .gov domena).

Literatura

[1] https://docs.python.org/3/library/threading.html

[2] https://pypi.org/project/psycopg2/

[3] http://docs.python-requests.org/en/master/

[4] https://docs.python.org/3/library/hashlib.html

[5] https://selenium-python.readthedocs.io

[6] https://pypi.org/project/urltools/

[7] https://gephi.org