Poročilo prve seminarske naloge

Mark Bogataj in Jakob Maležič
Fakulteta za računalništvo in informatiko, Univerza v Ljubljani
Ljubljana, Slovenija
Mentor: asist. prof. dr. Slavko Žitnik

I. Uvod

V tem poročilu je predstavljena implementacija pajka, katerega cilj je preiskati čim več strani znotraj domene *gov.si* in pri tem shraniti podatke kot so slike, vsebina in druge binarne datoteke. Pajek ima podprto razpoznavanje strani s podvojeno vsebino in deluje na več nitih. Predstavljen pa je tudi algoritem za razpoznavanje podvojenih strani, ki temelji na metodi najmanjše zgoščene vrednosti (ang. MinHash).

II. Implementacija spletnega pajka

Za implementacijo spletnega pajka sva se odločila da bova uporabila jezik Java in ogrodje Spring Boot. Ogrodje nama je olajšalo delo z bazo in omogočilo vstavljanje odvisnosti v razrede. Za pridobivanje podatkov s spletnih strani, pa sva uporabila knjižnico Selenium.

Najprej sva pripravila entitete, ki predstavljajo bazne objekte po specifikaciji iz navodil, pripadajoče bazne repozitorije in servise za dostop do baze. Spring Boot v ozadju uporablja odprto kodni ORM, zato nama ni bilo potrebno pripravljati skript za gradnjo bazne sheme, saj se ta zgradi sama ob zagonu programa.

Logika za preiskovanje in poganjanje več preiskovanj na enkrat je sestavljena iz treh večjih razredov.

A. CrawlManagerService

Glavni razred, ki skrbi za poganjanje vseh pajkov in njihovo pravilno delovanje. Ob kreiranju razreda se ustvarijo primerki gonilnikov Selenium in njihove nastavitve. Število primerkov predstavlja tudi število hkratnih preiskovanj in je odvisno od podanega števila na vhodu programa. Glavna zanka, ki dodeljuje strani in zaganja razrede `PageCrawl` v različnih nitih, se samodejno zažene, ko so vsi razredi pripravljeni. Strani za preiskovanje pridobiva iz razreda `FrontierService`.

B. PageCrawl

Razred, ki implementira vmesnik `Callable` in vsebuje logiko za pridobivanje podatkov iz posamezne strani. Kot argument prejme entiteto `Page` z naslovom spletne strani, ki jo želimo preiskati. Preiskovanje deluje po naslednjih korakih:

- Najprej se preveri končnica spletnega naslova. V primeru da končnica predstavlja enega izmed binarnih podatkovnih formatov, se ta stran izbriše iz baze in preiskovanje prekine. To se naredi za vsak slučaj, saj takšnih naslovov ne bi smelo biti sploh v fronti, temveč bi morali biti shranjeni v tabeli `PageData`[1]. V kolikor spletni naslov predstavlja spletno stran, se preiskovanje nadaljuje.
- Nato se iz naslova izlušči domena. Če je pridobivanje domene neuspešno, je ta stran prav tako zavržena, saj gre verjetno za neveljaven spletni naslov. Pomembno je, da so takšne strani odstranjene iz fronte, saj se drugače lahko ob ponovnem zagonu spet pojavijo.
- V kolikor je domena uspešno izluščena, se v bazi preveri, če že obstaja entiteta `Site` s takšno domeno. Če še ne obstaja, se pridobijo pravila preiskovanja in zemljevid strani za to domeno in shrani nova entiteta `Site`. Če se je zemljevid strani uspešno pridobil, se doda v fronto.
- Če pridobljena entiteta `Site` vsebuje pravila preiskovanja, se spletni naslov preveri ali ga lahko obiščemo. Če pravila tega ne dopuščajo, se stran odstrani iz baze in prekine preiskovanje te strani. Drugače pa se naredi zahtevek za pridobitev vsebine.
- Najprej se vsebina poskusi pridobiti z gonilnikom Selenium. Če pride do napake, se stran zavrže in preiskovanje prekine, če pa je vsebina uspešno pridobljena, se iz Seleniumovega beležnika razbere koda http statusa. V kolikor je to neuspešno, se naredi še en zahtevek na isto spletno stran s knjižnico JSoup. Na ta način se zagotovo pridobi koda http statusa, pri čemer se upoštevajo pravila in zakasnitve.
- Nato se preveri, če stran s takšno vsebino že obstaja v bazi. V entiteti `Page` se dodatno hrani tudi zgoščeno vrednost vsebine. To omogoča, da se preprosto izračuna zgoščeno vrednost trenutne strani in preveri ali takšna zgoščena vrednost v bazi še ne obstaja. V primeru da obstaja, se stran shrani kot duplikat, drugače pa se stran preišče in iz nje izlušči vse potrebne podatke.
- Povezave se preprosto izluščijo s pomočjo Seleniuma, medtem ko se naslovi slik preberejo iz Seleniumovega beležnika, saj ta vsebuje vse prenesene slike in nudi tudi tip vsebine. Slike se shranijo, medtem ko se vsak pridobljen naslov preveri ali gre za binarno podatkovno datoteko, neveljaven naslov ali duplikat in se zavrže, ustrezno shrani ali pa doda v fronto.

C. FrontierService

To je razred, ki skrbi za polnitev fronte, izbiro strani za preiskovanje in pravilno zakasnitev pri pridobivanju vsebine.

[1] Verjetno bi se lahko ta naslov shranil tudi v tem delu, vendar se žal tega nisva prej spomnila.

Ob pripravi razreda se najprej ustvari vrsta, ki predstavlja fronto in deluje po principu FIFO. Za vrsto sva uporabila razred, ki omogoča branje in pisanje v vrsto iz več niti hkrati. Na ta način lahko vse niti hkrati dodajajo nove strani v vrsto. Ob pripravi se prav tako v vrsto in v bazo vstavijo začetne strani, ki so bile podane v navodilih. Pripravi pa se tudi slovar domen in časa njihovih zadnji dostopov. Razred vsebuje še dve pomembni metodi:

- `getNextPage`: Funkcija vrne naslednjo stran za preiskovanje. To naredi tako, da iz vrste pridobi prvo stran, izlušči domeno iz spletnega naslova in preveri če lahko domeno že obiščemo, glede na časovno zakasnitev. Če jo lahko obiščemo, stran vrne, drugače pa stran doda na konec vrste in poskusi z naslednjo v vrsti. Na ta način, pospešimo delovanje pajka.
- `makeRequest`: Funkcija s podanim gonilnikom Selenium pridobi vsebino strani z upoštevanjem pravilne zakasnitve domene.

D. Locality-sensitive hashing

Dodatno sva implementirala tudi lasten sistem za zaznavanje podvojene vsebine pri različnih spletnih naslovih z uporabo lokalno občutljivega razprševanja (angl. locality-sensitive hashing). Bolj natančno sva implementirala in uporabila MinHash locality-sensitive hashing metodo, katere prednost je hitro ocenjevanje podobnosti dveh setov. Dobljena seta vrednosti na koncu primerjava in izračunava Jaccard-ovo razdaljo. Manjša kot je razdalja bolj sta si seta podobna. Pri testiranju metode sva naletela na zapleten problem in sicer, kako določiti mejo, ki bo učinkovito določala ali je stran duplikat ali ne. Najlažje duplikate je zaznati pri spletnih straneh, kjer je jaccard-ova razdalja enaka 0. To pomeni, da sta vsebini strani popolnoma enaki, med tem, ko se spletna naslova med sabo razlikujeta. Primer takih parov strani so: stran 1 in stran 2, stran 3 in stran 4 ter stran 5 in stran 6. Problem pri določanju meje pa se pojavi pri parih strani kot sta naslednja dva: stran 7 in stran 8, ter stran 9 in stran 10. Strani 7 in 8 imata vsebino povsem enako, vendar se strani med sabo razlikujeta po sami zgradbi. Ena izmed strani ima na vrhu še orodno vrstico, spodaj pa strani prikazujeta različne sorodne članke. Njuna jaccard-ova razdalja je 0,13. Na drugi strani pa imamo strani 9 in 10, ki sta po zgradbi povsem enaki, vendar se razlikuje njuna vsebina. Ena govori o globini, druga pa o širini. Njuna jaccard-ova razdalja pa je 0,017. Ravno zaradi takih primerov strani kot so našteti tu, je težko določiti dobro mejo, ki bo ločevala stran od duplikata. Na koncu sva za mejo vzela vrednost 0.05. Za lažje testiranje implementacije locality-sensitive hashing je pripravljen test, kjer sami podamo dva spletna naslova in počakamo na rezultat.

III. Izzivi

A. Izluščitev domene in kanonikalizacija spletnega naslova

Z pridobivanjem domene spletnega naslova nisva imela veliko težav. Iz spletnega naslova sva zgolj kreirala nov primerek razreda `URI`, poklicala metodo `getHost()` in odstranila morebitni *www* na začetku naslova.

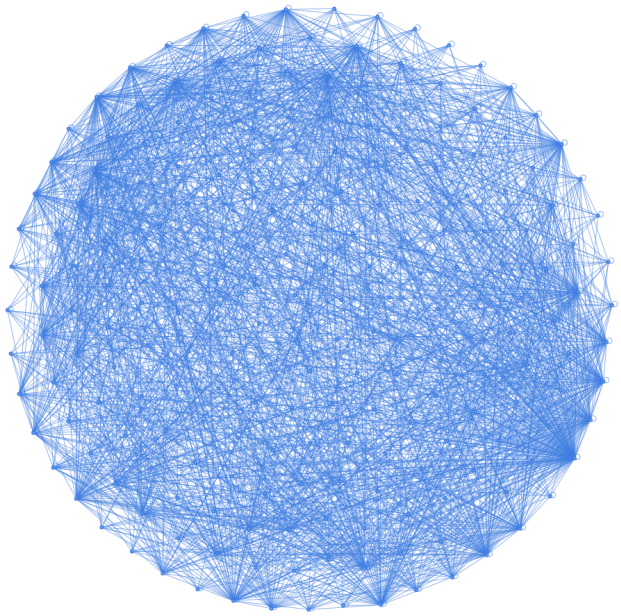

Fig. 1. Povezave med vsemi obiskanimi domenami.

S kreiranjem kanoničnega spletnega naslova pa sva imela nekaj več težav. Najprej sva spletni naslov prav tako pretvorila v primerek razreda `URI`, nato pa sva uporabila še knjižnico *urlcannon* in spletni naslov skrajšala na največ 3000 znakov.

B. Shranjevanje in večnitnost

Da bi karseda pospešila delovanje najinega pajka, sva shranjevanje baznih entitet naredila v razredu `PageCrawl`, ki se izvaja v več različnih nitih naenkrat. Ogrodje sicer podpira takšno shranjevanje, vendar se lahko zgodi, da dve različni niti želita shraniti entiteto z enakim unikatnim atributom. Zato sva morala namesto paketnega shranjevanja, shraniti vsako stran posebej ter ob napaki, stran shraniti kot duplikat. To je zelo upočasnilo celotno delovanje pajka.

C. Ustavljanje pajka

Ker pajek načeloma dela v skoraj neskončni zanki (dokler ni prazna fronta), je njegovo pravilno zaustavljanje precej oteženo. Največji problem predstavljajo gonilnik Selenium, ki se ob nenadni zaustavitvi programa ne konča. To sva rešila tako, da ob zaustavitvi prisilno ustaviva vse brskalnike Chrome ter gonilnike Selenium.

IV. Rezultati

Najinega pajka sva pustila prižganega približno tri dni, z vmesnimi ustavitvami. Naenkrat sva uporabljala 10 niti, kar nama je omogočalo preiskovanje desetih strani naenkrat. Vsega skupaj sva zajela nekaj čez 100000 strani. Od tega jih je kar 54000 duplikatov. Zajela pa sva tudi veliko slik in binarnih datotek, ki so predstavljeni v tabeli I.

TABLE I
STATISTIKA OBISKANIH STRANI.

Domena	Število domen	Število strani	Število duplikatov	Število slik
gov.si	1	11787	8312	34340
evem.gov.si	1	551	201	3143
e-uprava.gov.si	1	1783	5015	10536
e-prostor.gov.si	1	2099	351	7713
Skupaj	353	52267	54674	428059

TABLE II
STATISTIKA PRIDOBLJENIH BINARNIH DATOTEK PO TIPU.

Domena	EXE	DOC	DOCX	MP3	MP4	PDF	ZIP	PPT	PPTX	XLS	XLSX
gov.si	0	875	1048	2	19	6539	27	3	12	35	212
evem.gov.si	1	81	26	0	0	15	0	0	0	0	10523
e-uprava.gov.si	0	4	0	0	0	15	0	0	0	0	2
e-prostor.gov.si	0	242	34	0	0	1450	86	9	0	3	2
Skupaj	2	5131	5163	23	39	35893	4577	151	137	596	1048

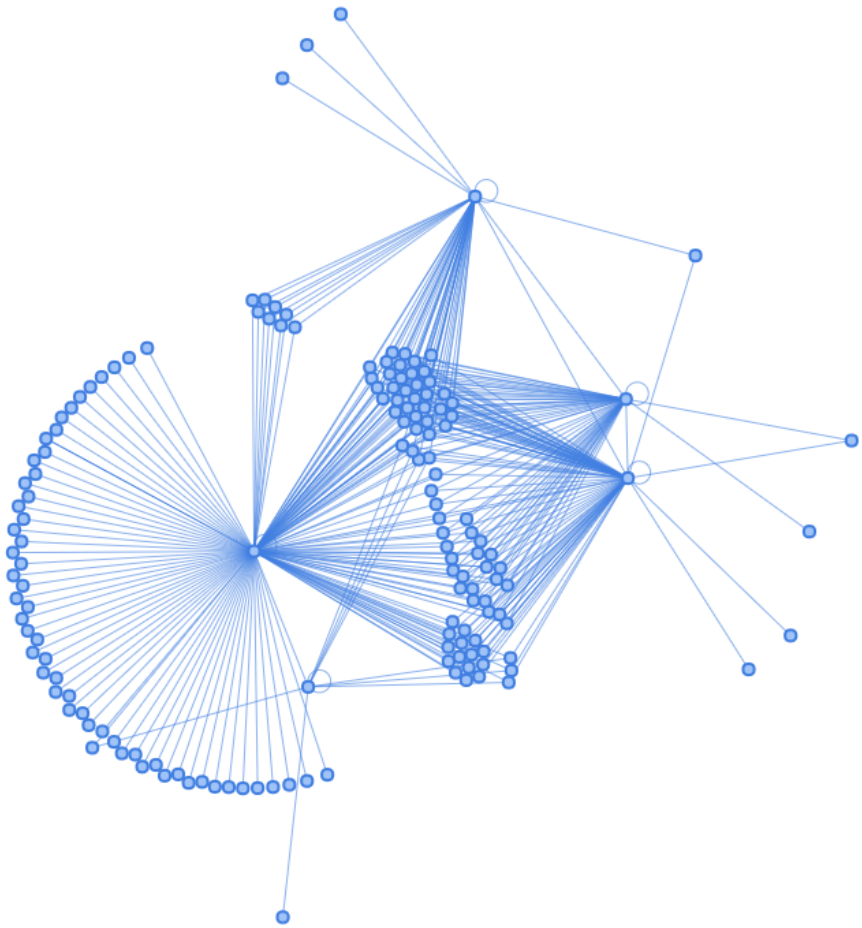

Fig. 2. Povezave iz začetnih štirih domen do drugih domen.