Programska naloga 1 - Spletni pajek

Kim Ana Badovinac, Jakob Petek, Jovan Prodanov
Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

I. Uvod

Spletni pajki (angl. web crawlers) so programska oprema, zasnovana za sistematično brskanje in indeksiranje vsebin spletnih strani. Ti programi sledijo hiperpovezavam za odkrivanje in prenos spletnih strani, nato pa iz njih izvlečejo informacije za različne namene, kot so indeksiranje spletnih iskalnikov, rudarjenje podatkov ter analiza vsebine. V sklopu prve programske naloge je bil naš cilj zgraditi spletnega pajka, ki obiskuje različne spletne strani znotraj *.gov.si* domene in iz njih pridobiva vsebino.

II. Implementacija

Komponente spletnega pajka smo implementirali v programskem jeziku Python. Pridobljene podatke shranjujemo v podatkovno bazo PostgreSQL. Osnovnega podatkovnega modela za bazo nismo spreminjali, smo pa ga dodatno razširili s tabelo *hash* ter dodatnimi polji v tabeli *site*.

Arhitektura implementiranega pajka je paralelna in uporablja večnitno delovanje. Za paralelno pridobivanje različnih spletnih strani uporabljamo modul threading [1]. S tem modulom na začetku izvajanja ustvarimo toliko niti, kot veleva parameter *NUMBER_OF_WORKERS*, podan v konfiguracijski datoteki *ProjectConfig.py*. Niti si delijo objekt *DatabaseController* za upravljanje s podatkovno bazo. Vsaka nit vzame element iz frontierja in iz spletne strani izlušči vsebino, dokler obstaja element v frontierju.

A. Frontier

Frontier je seznam URL-jev, ki čakajo na razčlenjevanje. Sprva smo za shrambo uporabili lokalno tabelo, vendar smo se kasneje odločili in restrukturirali implementacijo, da se za shranjevanje trenutne frontier vrste uporabi tabelo *page* v podatkovni bazi in imajo te vnosi v tabeli v stolpcu *page_data_type* označeno *FRONTIER*. Ker se strani v frontier shranjujejo z naraščajočim *id*-jem, lahko vzamemo prvo stran označeno *FRONTIER* z najmanjšim *id*-jem, da dobimo prvi element frontierja. Tako upoštevamo strategijo izbire URL-jev - iskanje v širino. Nove strani dodajamo 'na konec' vrste (večji id), vzamemo pa iz 'začetka' (najmanjši id).

Na začetku imamo podane 4 semenske strani URL, ki jih vstavimo v podatkovno bazo kot *page* z oznako *FRONTIER*. Potem pa spletni pajek jemlje *FRONTIER* strani po vrsti po naraščajoče urejenem *id*-jem. Stran iz frontierja vzamemo z implementirano metodo *pop_frontier*, ki vrne *page_id* in *url* od najdene strani, prav tako pa odstrani oznako v stolpcu *page_type_code*. Oznako *FRONTIER* odstranimo, zato da te strani ne najdejo druge niti in jo istočasno obdelujejo.

B. Obdelava strani

Ko pajek pridobi URL, se začne metoda pridobivanja vsebine informacij z dane spletne strani. Ker izvajamo večnitno delovanje, se lahko zgodi, da več kot en pajek dostopa do različne spletne strani z enako domeno ali IP-naslovom, zato sprva z URL-ja pridobimo ravno ta dva podatka, da s tem ne kršimo pravil etike spletnih pajkov ter onemogočimo prepogosto pošiljanje zahtev na strežnik. V ta namen smo tabelo *site* razširili ter dodali polji *ip_address* in *last_accessed*. V primeru, da je bil čas zadnjega dostopa manj kot 5 sekund, počakamo dokler ta ne mine.

Z metodo nato nadaljujemo, kjer prvič pošljemo zahtevo na strežnik ter pridobimo osnovne informacije o spletni strani ter njen tip vsebine. V primeru, da spletna stran ni dosegljiva, v bazo zapišemo statusno kodo 404 ter zapustimo metodo.

Preden začnemo pridobitev podatkov ter vstavljanja le-teh v bazo pa še preverimo ali to sploh smemo. Večina domen namreč zapiše določena pravila, ki dopuščajo ali preprečujejo dostop spletnih pajkov na določene spletne strani. Te datoteki pravimo *robots.txt*, ki poleg opisanega vsebuje tudi naslov sitemapa oziroma naslov vseh spletnih strani ter drugih datotek dane domene. V skladu s pravili *robots.txt* se iz metode vrnemo, v primeru, če dani URL ni dovoljen, sicer nadaljujemo.

Na tem mestu drugič pošljemo zahtevo na strežnik, tokrat za pridobitev vsebine z uporabo web driver-ja, kjer glede na vrnjeno vsebino ustrezno polnimo bazo (vstavimo site, stran, slike ali datoteke). V primeru, da strežnika ne moremo doseči, vpišemo statusno kodo 500.

Preden pričnemo s polnjenjem baze pa še preverimo, ali je dani URL duplikat. Za to zaznavo smo razširili podatkovni model z dodatno tabelo *hash* ter stolpcema *hash* in *page_id*. V dano tabelo tako shranjujemo zgoščene vsebine strani, katere uporabimo za zaznavo duplikatov. V primeru, da trenutna spletna stran ni duplikat, njen hash na koncu vstavimo v tabelo.

Na koncu metode še pridobimo ter ustrezno umestimo vse URL-je, ki obstajajo znotraj *href* ter *onclick* atributa. Če pridobljeni URL-ji niso tipa html, jih uvrstimo v bazo, drugače pa vstavimo v frontier. Pri tem pazimo, da stran, ki jo želimo vstaviti v frontier, še vedno ustreza domeni *gov.si* ter ni potencialna past za spletne pajke.

III. Statistika

Implementiran spletni pajek je v skoraj 60 urah obiskal 211 glavnih domen in obdelal 63285 strani. Ob zaključku delovanja smo imeli v frontierju prisotnih še 36817 novih URL-jev, s katerimi bi pajek lahko nadaljeval obiskovanje. Program se je izvajal v 8 nitih. Spodnje tri tabele prikazujejo statistiko

spletnih strani po vrsti, po številu glede na semenske strani ter statistiko za celotno tabelo *crawl.db*.

Vrsta strani	Število
HTML	51020
Binary	11494
Duplicate	771
Frontier	36817
Skupaj	109824

Tabela I
ŠTEVILO STRANI PO VRSTI.

	Število
Domene	4
Spletne strani	14934
Duplikati	185
PDF	17.16
DOC, DOCX	12.06
PPT, PPTX	3.4
Slike	4.15

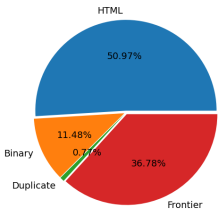
Tabela II
STATISTIKA ZA SEMENSKE STRANI.

	Število	Povprečno št. na domeno
Domene	211	x
Spletne strani	63121	241.73
Duplikati	771	3.65
PDF	51246	242.87
DOC, DOCX	23627	111.97
PPT, PPTX	351	61.46
Slike	147769	700.33

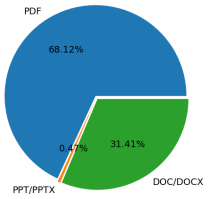
Tabela III
STATISTIKA ZA CELOTEN CRAWLDB.

IV. VIZUALIZACIJA

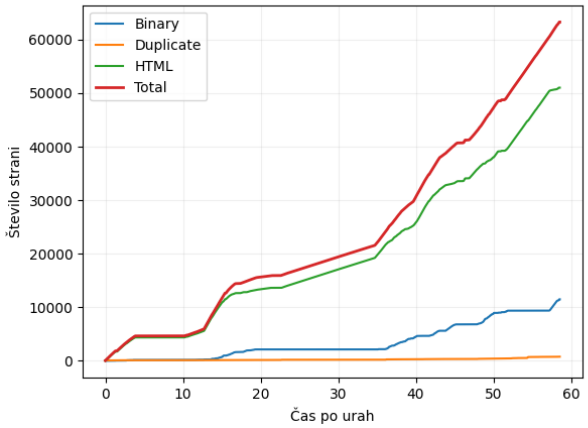
Poleg podane statistike smo na koncu še s pomočjo odprtokodnega programa Gephi [4] izrisali shemo pridobljenih spletnih strani (Slika 4) ter obenem izrisali dva tortna prikaza, ki prikazujeta delež vrst spletnih strani (Slika 1) ter vrst datotek (Slika 2). Dodali smo tudi graf števila pridobitve spletnih strani čez čas, glede na njihov tip.



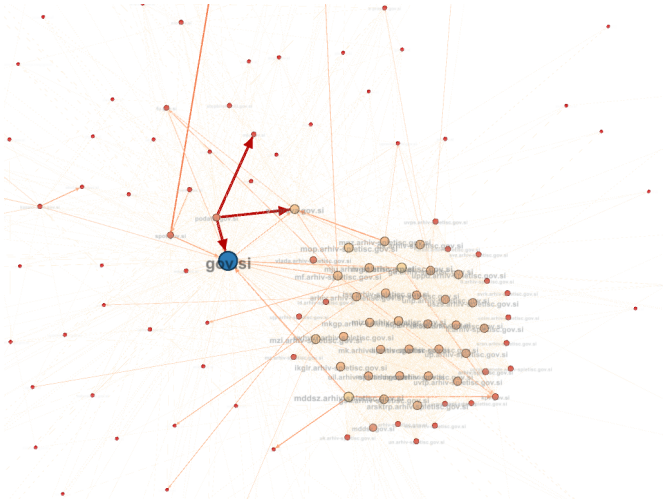
Slika 1. Vrste strani



Slika 2. Vrste datotek



Slika 3. Spletne strani skozi čas



Slika 4. Shema Gephi z vsemi povezavami

V. ZAKLJUČEK

Z implementacijo spletnega pajka smo dosegli zahtevane rezultate. V približno 60 urah smo uspeli obdelati 50 tisoč spletnih strani, toda ne brez manjših zapletov. Ena stvar, ki bi jo morali opozoriti, je bila, da na začetku izvajanja, kot lahko vidimo na sliki 3, eno orodje za normalizacijo url zruši naše niti zaradi preveč znakov. Zaradi tega smo ustvarili sistem za ponovni zagon omenjenih niti. Prav tako nam je izvajanje upočasnjevalo število strani v bazi. Ker smo za shrambo frontierja uporabili podatkovno bazo, smo morali prvi element v frontier vrsti v bazi tudi poiskati. To je vzelo čedalje več časa z večjim številom strani v bazi. Prednost hrambe frontierja v bazi pa je bilo, da se ti podatki niso izgubili ob prekinitvi izvajanja.

LITERATURA

[1] Python threading https://docs.python.org/3/library/threading.html
[2] Selenium https://www.selenium.dev
[3] Postgreql https://www.postgresql.org
[4] Gephi https://gephi.org/