Spletni pajek

Anja Brelih, Ana Knafelc, Eva Vidmar

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

1 Uvod

Pri prvem projektnem delu pri predmetu Iskanje in ekstrakcija podatkov s spleta je bila naša naloga implementirati večnitnega spletnega pajka z iskanjem v širino po domeni *"gov.si"*. Cilj projektnega dela je, poleg zbrati podatke iz 50.000 spletnih strani v domeni, nalogo opraviti tudi etično. Torej spletni pajek upošteva omejitve domen, ki jih dostopa, da se izognemo napadu zavrnitve storitve (*angl. Denial of Service - DoS*), oziroma distribuirani različici napada.

2 Implementacija

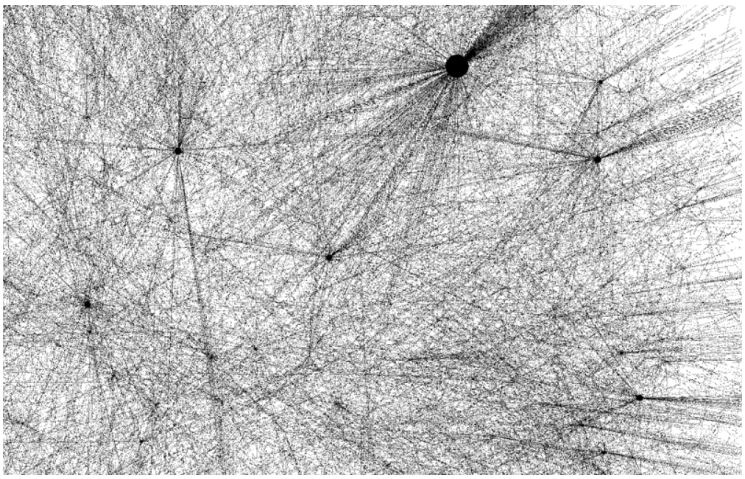
Pajek ob zagonu preveri, če obstajajo *url*-ji v *frontier*-ju ter, v kolikor jih ni, ustavi semena v podatkovno bazo z značko FRONTIER in pogleda njihovo domene in pridobi informacije iz *robots.txt* datoteke (*crawl-delay*, *sitemap* ter *disallow*) ter tudi to zabeleži v bazo. V tabeli domen v podatkovni bazi smo dodali tudi stolpec, ki zapiše zadnji čas dostopa v domeno, ki se redno posodablja, da se izognemo prepogostim klicem v domeno. Pri zadnji implementaciji je v tabeli domen dodan tudi stolpec, ki z zastavico označuje, ali do domene trenutno dostopamo z drugo nitjo. Naslednji korak je aktiviranje globalno definiranega števila niti s pomočjo objekta niti. Ko se niti zaženejo, po vrstnem redu iskanja v širino iz podatkovne baze pridobijo *url*-je. Najprej se preveri, če se glede na *crawl-delay* in čas zadnjega dostopa v domeno spletno stran že lahko dostopa, v kolikor ne, počaka primeren čas. Spletno stran najprej dostopamo z *Requests*, da pridobimo informacije glede tipa spletne strani. V kolikor je spletna stran tipa HTML po pretečenem času *crawl-delay*-a do nje dostopamo še enkrat s *Selenium*-om, da pridobimo html vsebino. Podatki spletne strani se nato zabeležijo v podatkovno bazo, html vsebina pa gre naprej v iskanje novih *url*-jev, ki jih pridobimo glede na značke *href*, *button* ter *img*. Potencialne url-lje je potrebno nato še prečistiti, kanonizirati, pregledati njihove domene ter vse skupaj zabeležiti v podatkovno bazo, kjer se hranijo le unikatni *url*-lji. Tik preden spletno stran zabeležimo v podatkovno bazo, pridobimo *hash* vsebine spletne strani z zgoščevalno funkcijo s katerim preverimo, če identična vsebina v podatkovni bazi že obstaja. V tem primeru se stran klasificira kot DUPLIKAT, ter se to primerno zabeleži v podatkovni bazi. V kolikor se po poizvedbi *Requests* spletna stran klasificira kot BINARY, se glede na kodo spletne strani pogleda tip datoteke na katero vodi *url* ter informacijo zabeležimo v podatkovno bazo. Ob tem je delo niti zaključeno in se celoten proces ponovi. Nit pridobi prvi url v *frontier*-ju, ki ima prosto domeno.

3 Pridobljeni podatki

S spletnim pajkom smo pri prvi implementaciji pridobili približno 23.000 *url*-jev, kjer se je pajek ustavil zaradi nepravilnega čiščenja *url*-jev. Odločili smo se, da popravimo napake, pobrišemo pridobljene podatke ter pajka zaženemo še enkrat od začetka, zato pajek ni dosegel načrtovanega števila vsebin spletnih strani, preiskal je 18.000 *url*-jev. Pri tretji implementaciji se je implementacijo več-nitnosti spletnega pajka še izboljšalo ter začelo zbirati podatke od začetka. Spletni pajek je preiskal preko 90.000 *url*-jev. Število pridobljenih spletnih strani, glede na zabeležen tip v podatkovni bazi, je zapisano v spodnji tabeli.

	Količina
Domene	331
Spletne strani	91.664
HTML	50.050
Error	7.302
Duplikat	2.203
PDF	14.010
DOC, DOCX	5.175
PPT, PPTX	92
Slike	8.964

V začetku smo imeli podana štiri semena. Iz semena *"www.gov.si"* smo pridobili 26.205 strani, iz *"www.evem.gov.si"* 290 strani, iz *"www.e-uprava.gov.si"* 2.015 strani ter iz semena *"www.e-prostor.gov.si"* 591 strani.



Slika 1: Vizualizacija prepletenosti *url*-jev

4 Zaključek

Največji problem pri projektu je bil vzpostaviti več-nitnost. Poskusili smo več različnih metod, vendar se je vse do zadnje implemetacije dogajalo to, da so si niti med sabo kradle spremenljivke. Pri prvi implementaciji nam je težave povzročalo preverjanje SSL certifikata tako pri *Requests*, kot pri *Selenium*, zato vsebine strani nismo prejeli, kar je eden od razlogov, zakaj je pri prvem poskusu zmanjkalo *url*-jev. Drugi problem pa je bil neprimerno čiščenje *url*-jev, na neprimerni točki smo potencialni *url* preverjali, če ta sodi v katero od domen *"gov.si"*. Pri novi izvedbi smo tudi podrobneje pregledali iskanje potencialnih *url*-jev s knjižnico *Selenium*, da upoštevamo tudi relativne *url*-je. Kar nekaj težav nam je povzročala povezava s podatkovno bazo, zato tudi sama končna implementacija vsebuje več poizvedb kot bi bilo potrebno za pridobitev želenih informacij. Dilema je bila tudi na temo tega, ali naj povezave z bazo ustvarimo globalno glede na število niti ali naj povezavo odpiramo in zapiramo pri vsakem klicu. Končna implementacija vsebuje odpiranje in zapiranje podatkovne baze ob vsakem klicu skupaj z zaklepanjem. Nekaj težav se je pojavljalo tudi pri prepoznavi tipa dokumenta, vsi dokumenti so bili na koncu zabeleženi še kot DOCX. Težava je se je pojavljala le v začetku druge implementacije. Pri tretji implementaciji je v tabelo domen ustavljen nov stolpec, ki z zastavico odklepa in zaklepa domene, tako da se hkraten klic več niti v isto domeno ne more zgoditi.

Literatura

1. Level Up Coding, Multi-Threaded Python Web Crawler for HTTPS pages. *https://levelup.gitconnected.com/multi-threaded-python-web-crawler-for-https-pages-e103f0839b71*. Dostopano: 28.3.2022
2. Selenium. *https://selenium-python.readthedocs.io/*. Dostopano: 25.3.2022
3. Psycopg. *https://www.psycopg.org/docs/*. Dostopano: 26.3.2022
4. Requests. *https://docs.python-requests.org/en/latest/*. Dostopano: 25.3.2022