Iskanje in ekstrakcija podatkov s spleta: Spletni pajek

Rok Zidarn (63120283), Edo Ljubijankič (63130142)

April 2, 2019

Abstract

Naša naloga je bila implementirati spletnega pajka, ki bo iskal povezave med vladnimi stranmi, poleg zgradbe samega spletišča pa bomo skušali pridobiti čim več slik in datotek določenih tipov. Za hitrejše delovanje bomo iskanje izvajali vzporedno s pomočjo niti. Na koncu bomo predstavili rezultate v obliki statistike posameznih domen in usmerjenega grafa povezav.

1 Uvod

Klasično brskanje po spletu s pomočjo grafičnega vmesnika, brskalnika je sicer zelo enostavno in uporabniku prijazno. Vendar samo v primeru iskanja točno določenega odgovora, če pa je naša želja pridobiti čim več informacij, v čim krajšem času je smiselno implementirati pajka, ki pospeši in avtomatizira naše delo. Za seminarsko nalogo smo implementirali spletnega pajka namenjenega brskanju po slovenskih vladnih straneh (**.gov.si*). Za hitrejše delovanje in boljšo izkoriščenost virov smo uporabili pristop z nitmi.

2 Implementacija

Programirali smo v programskem jeziku Python, delovanje z nitmi smo dosegli s paketom *ThreadPoolExecutor*, kjer se določi maksimalno število niti (*workers*), ki simulirajo vzporedno delovanje, mi smo jih uporabili 12. Posamezno nit izvajamo s funkcijo *ThreadPoolExecutor.pool.submit()*, kjer se ustvari zahtevek na določen URL, nato se izvede ekstrakcija podatkov, torej naslednjih URLjev, slik in datotek. Iskanje v širino smo enostavno implementirali s pomočjo Python vgrajene vrste *Queue*, ki nam je služila kot Frontier. Namreč deluje po principu FIFO (*First In First Out*), v takšnem vrstnem redu kot strani pridejo v Frontier, kjer čakajo na obdelavo, ga tudi zapustijo. Prva stran, ki je v vrsti se v obdelavo niti preda kot posel. Sem spada iskanje novih povezav, slik in datotek tipov PDF, DOC, DOCX, PPT in PPTX. Te podatke shranimo v PostgreSQL podatkovno bazo, za povezavo in ostale operacije nad njo smo uporabili paket ORM SQLAlchemy.

Glavni del pajka predstavlja paket Selenium in gonilnik Chrome driver, ki simulira brskalnik v načinu *headless*. V primeru pasti (*spider traps*) smo uporabili nastavitev, ki onemogoča piškotke. Ker se zelo redko pojavi določen *Crawl delay*, smo nato preventivno počakali 6 sekund pred novim zahtevkom na stran s pomočjo funkcije *webdriver.Chrome.wait_ implicitly()*. Paket nam tudi omogoča dostop do HTML kode posamezne strani, za pridobivanje novih povezav pa smo uporabili funkcijo *webdriver.Chrome.find_ elements_ by_ xpath()*.

Duplikate spletnih strani smo preverjali na dva načina, preko URLja in primerjave HTML kode. Če se je v URLju pojavil znak #, pomeni, da smo le premaknjeni na drug del strani. HTML kodo pa smo primerjali s pomočjo zgoščevalne funkcije MD5, ki je olajšala in pospešila delovanje v primerjavi s *String matching-om*. Že predhodno smo odstranili povezave, ki niso imele domene .gov.si, in razne povezave na datoteke, ki niso ustreznih tipov, kot sta CLS ali ZIP. Upoštevali smo tudi priporočila v datoteki *robots.txt*, pregledali smo *Sitemap* in URLje dodali v Frontier, kakor tudi *Crawl delay*. Za agenta smo imeli nastavljeno privzeto vrednost (*), direktivi *Allow, Disallow* pa smo upoštevali s pomočjo paketa Reppy, ki vsebino datoteke *robots.txt* shrani v slovar in nato preko funkcije *Robots.allowed()* preveri dovoljenje za dostop do strani.

Med razvojem smo se srečali s številnimi težavami. Najprej je težave povzročal Firefox driver za Selenium, saj ni izvajal Javascripta na domeni evem.gov.si, zato smo uporabili Chrome driver. Tukaj je težava bila pri dostopu do statusne kode odgovora, saj tega ne podpira ta driver. Poskušali smo z PhantomJS driverjem, kjer je bilo potrebno branje log datoteke, kamor se je statusna koda

zapisala. Ampak pri večnitnem izvajanju je včasih prišlo do težav pri dostopu do te datoteke. Zato smo na koncu uporabili kar Requests paket.

Med izvajanjem smo včasih naleteli na izjemo *Stale Element Reference Exception*, ki jo sproži dostop do elementa, ki več ne obstaja v DOM drevesu. Sicer smo to izjemo, kar ignorirali, čeprav do nje naj ne bi smelo priti, saj pred tem počakamo, da se stran popolnoma naloži, saj so možne kakšne spremembe zaradi preusmeritev ali zakasnjenega izvajanja Javascripta. Naša največja težava pa je bila preveliko število povezav na bazo. Kljub temu da smo namesto *SQLAlchemy.session()*, ki ni namenjena večnitnemu izvajanju, uporabili *SQLAlchemy.scoped_session()*, ki se v takih primerih uporablja. Namesto, da se je bazen povezav zapolnil v minuti, se je to zgodilo po nekaj urah. Namreč nekatere povezave na bazo so se sproti zapirale, tako kot je treba, druge pa so ostale odprte več minut in šele nato so se zaprle. Sprva smo mislili, da je težava rešena, saj je po 30 minutah število povezav padlo iz 50 na 10. Misleč, da bo se to ponavljalo, vendar je v nadaljevanju zopet prišlo do iste težave.

3 Rezultat

Izvajanje pajka je trajalo 150 minut, v tem času smo zbrali 4 domene. Iz teh domen smo pridobili 12623 strani, na katerih je bilo 4154 slik in 375 datotek. Duplikatov nismo našli, zato smo naknadno še enkrat testirali, če primerjava vrednosti zgoščevalne funkcije deluje in je delovala. Največ datotek je bilo tipa PDF (301), DOC (44), DOCX (30), PPT in PPTX naš pajek ni našel. Pri slikah je bil najpogostejši tip PNG (3471) in JPG (86), nekaj slik je bilo še tipa GIF. Se pa je vrinila tudi kakšna slika, ki je imela nenevaden tip, zaradi neustreznega razbitja niza. Pri slikah smo prav tako izločevali duplikate, saj se je na nekaterih straneh pojavila ista slika večkrat.

Statistika posamezne domene:

1. e-prostor.gov.si
    - 555 strani, 88 slik
2. e-vem.gov.si
    - 2769 strani, 2848 slik
3. e-podatki.gov.si
    - 8225 strani, 1150 slik
4. e-prostor.gov.si
    - 1074 strani, 68 slik

Repozitorij: https://github.com/rokzidarn/DotGovCrawlerSelenium

4 Zaključek

Na koncu nam je uspelo implementirati povprečnega spletnega pajka, ki je uspešno deloval v večnitnem načinu izvajanja. Vendar nam zaradi težav s povezavo na bazo ni uspelo zbrati večje število strani, kajti v Frontierju nam je ostal še del neobdelanih strani, do katerih sploh nismo prišli. Zato bi bilo bolje izbrati kak drug ORM oziroma uporabljati ročno spisano povezavo na bazo in SQL stavke. Spodaj se nahajajo tudi tri vizualizacije, ki prikazujejo le strani in povezave domene eprostor.gov.si. Vizualizirali smo le to domeno, saj vsebuje zgolj 555 strani zaradi lepšega in bolj razvidnega prikaza. Prva prikazuje celotno spletišče domene, večje točke predstavljajo, strani ki imajo največ izhodnih povezav, druga slika je zgolj prikaz pod drobnogledom prve slike, kjer so vidni tudi naslovi strani. Tretja slika pa prikazuje katere povezave so se nahajale na izvorni strani domene eprostor.gov.si. Vizualizacija je na voljo tudi v priloženi datoteki.

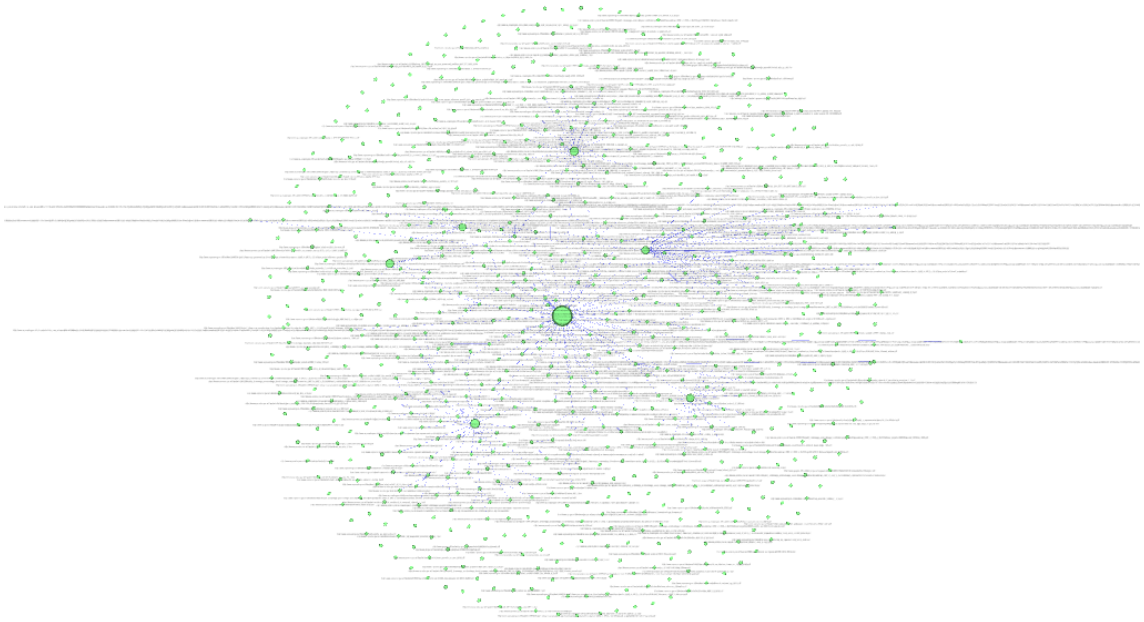

Figure 1: Celotno spletišče e-prostor.gov.si

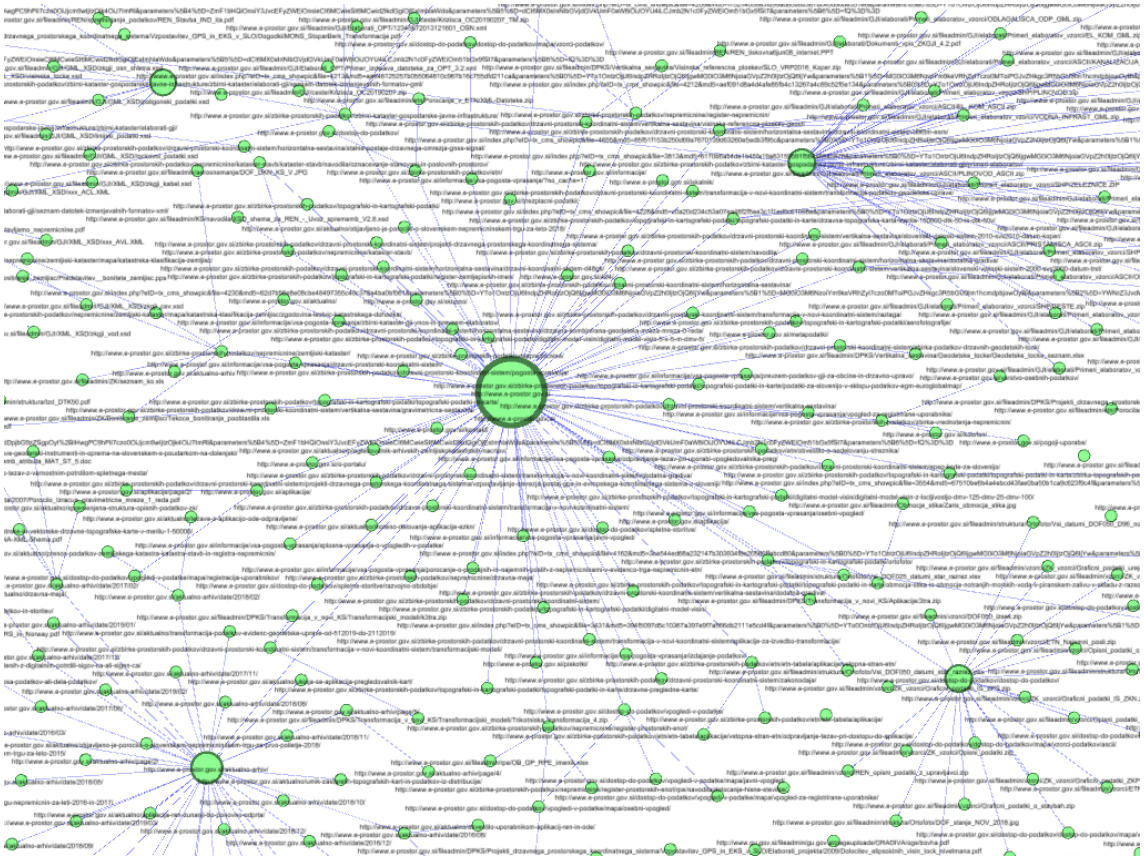

Figure 2: Del spletišča e-prostor.gov.si

Figure 3: Izvor e-prostor.gov.si