University *of Ljubljana*
Faculty *of Computer and Information Science*

Naloga 1: Pajek

Marija Klemenčič, Timotej Šalamon, Krištof Zupan

1. Implementacija

Za implementacijo našega spletnega pajka smo izbrali programski jezik Python. Na začetku nastavimo lokacijo gonilnika geckodriver.exe, časovno omejitev 5 sekund, poljubno število delavcev in naslove predpisanih začetnih spletnih strani. Nastavimo, da brskalnik teče v načinu brez prikazanega uporabniškega vmesnika, uporabniškega agenta brskalnika nastavimo na "fri-wier-Skupina_G", lokacijo izvršljive datoteke Firefoxa in omogočimo brskalniku, da sprejema neveljavne ali samopodpisane SSL certifikate.

Vzpostavi se povezava s podatkovno bazo postgreSQL, implementirana pa je tudi možnost, da se pred vpisovanjem pridobljenih podatkov v bazo le-ta izprazni. Za zagotavljanje načina iskanja v širino smo za frontier uporabili vrsto, v kateri hranimo naslove spletnih strani, ki jih je treba obiskati. Da zagotovimo, da se pridibobivajo spletne strani v širino jih razvrščamo po času pridobitve spletne strani. Kar pomeni, da so najprej obdelane strani, ki so največ časa v vrsti. Niti nato po tem vrstnem redu jemljejo strani iz frontierja, preverijo robots.txt in glede na odgovor izvedejo ustrezno dejanje. Če je stran označena kot prepovedana za obisk našega pajka, je le-ta ne obišče, sicer pa pošlje zahtevo za pridobitev strani in podatke ustrezno obdela.

V primeru, da je vsebina strani html, se stran prenese in njeno vsebino in podatke o njej shrani v podatkovno bazo. V primeru, da gre za sliko, se le-to shrani v tabelo 'images'. Vsebinskim datotekam z drugimi končnicami kodo tipa strani ('page_type_code') na binary, njihove vsebine pa ne shranjujemo v podatkovno bazo. Spletni pajek na obdelanih straneh poišče nove hiperpovezave, ki jih shrani v frontier na konec vrste.

Za preprečevanje shranjevanja večih strani z isto vsebino smo v tabeli 'page' dodali stolpec 'hash_page', v katerega z uporabo knjižnice hashlib in algoritma sha256 shranimo vrednost zgoščevalne funkcije. Preverimo, če novo izračunani hash že obstaja v tabeli in glede na to ustrezno ukrepamo – stran upoštevamo kot novo in shranimo v podatkovno bazo (s 'page_type_code' 'binary') ali pa jo označimo kot duplikat ('page_type_code' nastavimo na 'duplicate').

V primeru, da delovanje niti naleti na prazen frontier, počaka in večkrat pogleda še enkrat, s čimer smo povečali robustnost delovanja sistema. Poleg tega nadziramo tudi morebitne napake. V primeru, da pride do napake, kjer koli med procesom analiziranja strani pajek počaka 5 sekund in nato še enkrat proba analizirati to spletno mesto. Če po 4 poskusih ne uspe izvesti obdelave strani jo zavrže.

Za to, da je sistem odporen na nepričakovane napake ali nepričakovano ustavljanje programa smo dodali v tabelo page polje "in-use", ki se ob začetku analiziranja strani nastavi na "true". Če se slučajno v času analiziranja kaj zalomi ta vrednost ostane enaka. Ko se ponovno zažene pajek se preveri podatkovno bazo za primere strani, ki imajo "in-use"nastavljen. Za te strani najprej odstranimo slike, ki so bile že dodeljene v tabelo images, da se podatki ne podvajajo. Nato nastavimo te spremenljivke na "false"tako, da so ob začetku izvajanja pajka ponovno analizirane. Če je pajek, že dodal določeno spletno mesto v frontier ga izpustimo.

2. Vhodni parameti, problemi in njihove rešitve

Pajek kot vhodne podatke sprjeme štiri pet parametorov:

- Timeout [TIMEOUT]: Določimo koliko sekund bo pajek čakal preden bo na spletno stran pošiljal nove zahteve.
- Število delavcev (oz. število niti) [NUM_OF_WORKERS]: Določimo koliko delavcev hkrati po iskalo po spletnih straneh. Pomembno je, da več delavcev ne dostopa do iste strani.
- Začetni url-ji [WEB_PAGE_ADDRESSES]: Določimo url-je, na katerih bo pajek začel.
- Lokacija Firefox-a [firefox_options.binary_location]: Lokacija datoteke Firefox.exe

Pri implementaciji smo naleteli na problem s hkratnim dostopom do virov in podatkovne baze, saj smo naša implementacija vsebuje več delavcev / niti, ki lahko hkrati dosopajo do podatkovne baze oz. želijo v njo pisati. Da smo ta problem rešili smo na več mestih implementirali t. i. "Lock", ki skbi, da lahko do baze dostopa le en delavec hrati. Pri tem smo pazili tudi na to, da se znotraj "Lock-a", ne poskuša zopet dostopati do novega "Lock-a", saj se v tem primeru drugi "Lock"nikoli ne mora zagnati.

Problem, ki je tudi nastal je kako analizirati kakšna vsebina je na spletni strani. Na začetku smo implementirali preprosto razčlenjevanje URL-ja in analiziranje končnic. Nato smo pa uporabili še content-type, ki se pridobi ob klicu za spletno stran. Analiza tega podatka nam je dala na voljo informacijo o vsebini spletne strani in kako nastaviti polja v podatkovni bazi.

Problem, ki pa je izhajal iz prejšnjega pristopa je pa bil to, da je v primeru, da je spletna stran na primer vsebovala .zip datoteko jo je pajek celo prenesel. Kar je enkrat vmes nastal problem, ko smo naleteli na več zaporednih povezav, kjer so bile shranjene datoteke z več gigabyti. Za to smo naredili novo rešitev, kjer ob klicu za stran preverimo, če je content-type HTML nadaljujemo prenos in v primeru, da ni ga prekinemo. Tako se izognemo nepotrebnemu prenašanju spletnih vsebin, ki se ne shranjujejo.

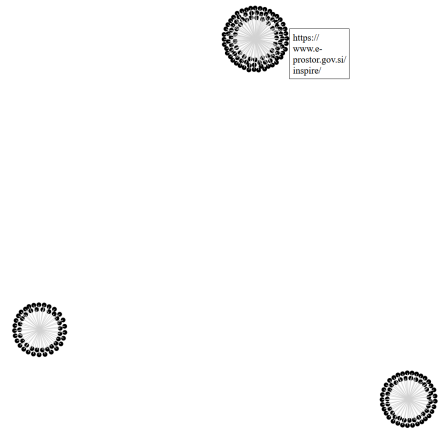
Zadnji problem, ki smo ga imeli je bil povezan z zaganjanjem pajka. Na začetku smo implementacijo testirali na 1 ali 2 nitih. Ko smo pa želeli zagnati na več nitih je pa prišlo do težave, ker je cel sistem začel delati napake. Ugotovili smo, da se preobremeni procesor in pomnilnik. Kar pomeni, da je za vsakega posameznika primerno različno število niti. Na koncu nam je uspelo na sistemu s 64 GB pomnilnika in procesorjem s 8 jedri pognati 24 vzporednih niti.

3. Statistika

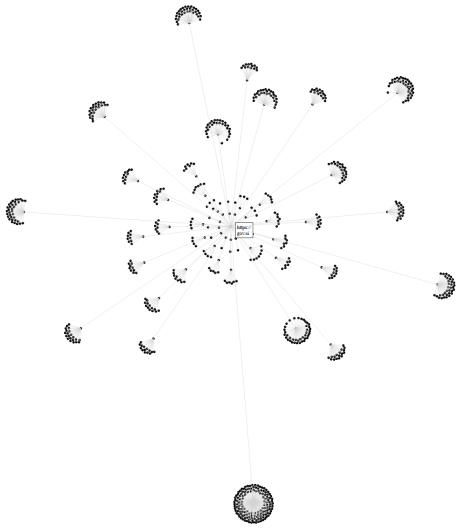
S pajkom smo zajeli 4561 strani in pridobili 716931 url-jev. Shranili smo 51639 HTML strani, zaznali in zabeležili 1638 duplikatov in 810913 slik. V kategorijo 'binary' smo uvrstili 10957 zadetkov, od tega jih je 4382 tipa .pdf, 1506 .docx, 386 .doc, 27 .pptx, 2 tipa .ppt in 1209 tipa 'other'. V povprečju smo na stran našli 15,23 slik.

4. Vizualizacija povezav

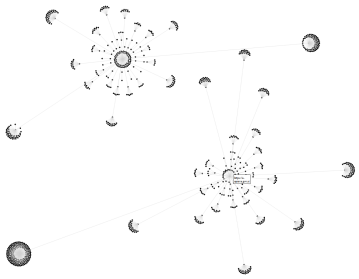
Vizualizacijo podatkov smo naredili s pomočjo knjižnice D3js v html/javascript node.js ogrodju. Kot dodatno funkcionalnost smo dodali možnost prikaza različnih števil povezav. Če uporabnik na tipkovnici vnese številko 1 bo prikazana začetna spletna stran in povezave samo do prve strani, ki vodijo iz začetne. Če uporabnik vnese številko 2 se mu prikaže drugi nivo povezav in tako naprej. Zaradi tega bomo prikazali več različnih vizualizacij.



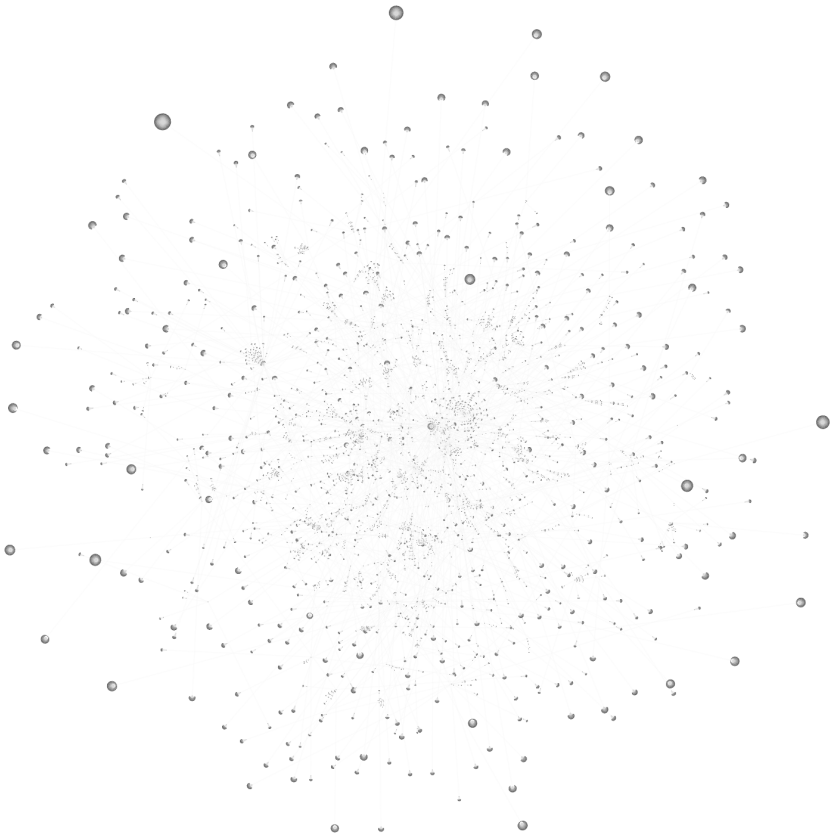
Slika 1. Začetne 3 strani gov.si, e-prostor.gov in e-uprava.gov samo z 1 povezavo.



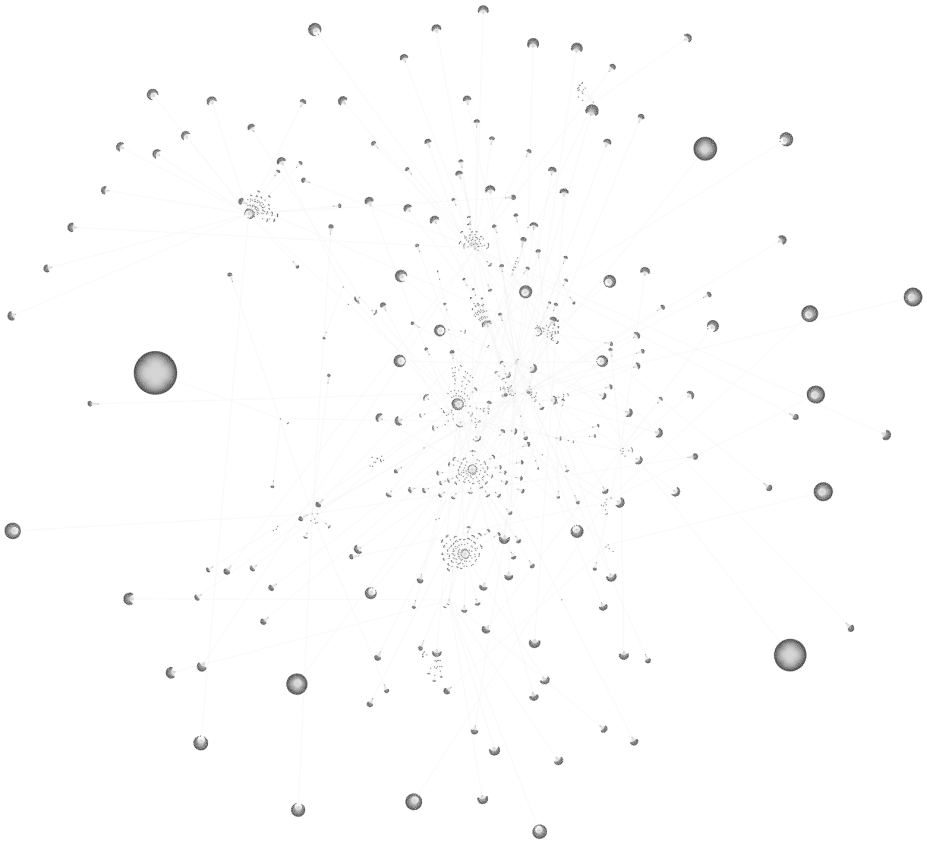
Slika 2. gov.si z 2 povezavama.



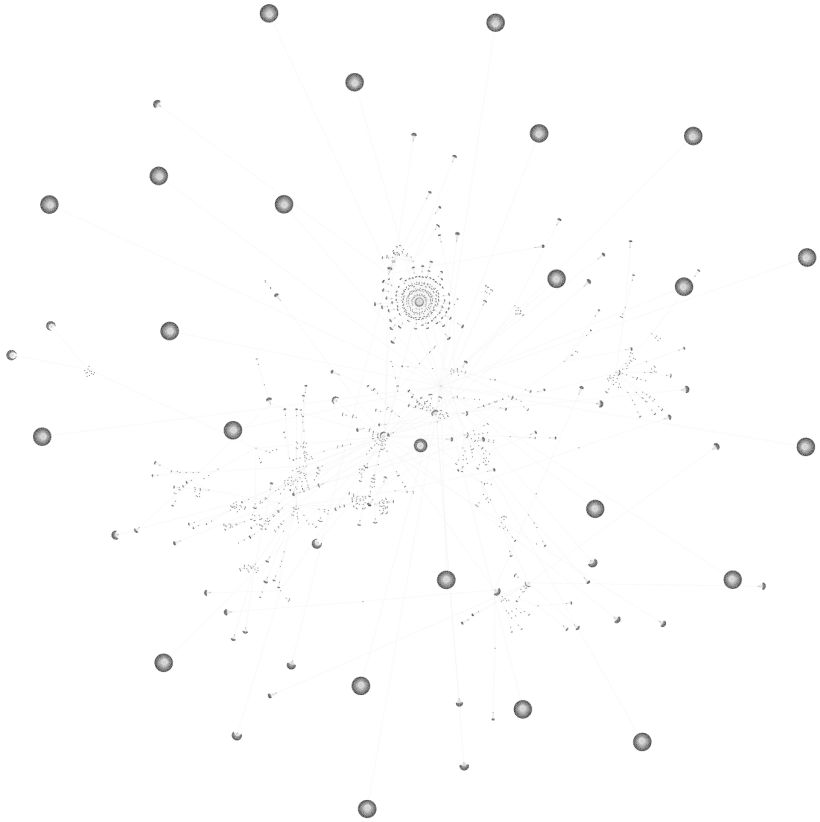
Slika 3. e-uprava.gov in e-prostor.gov z 2 povezavama (desno spodaj e-uprava)



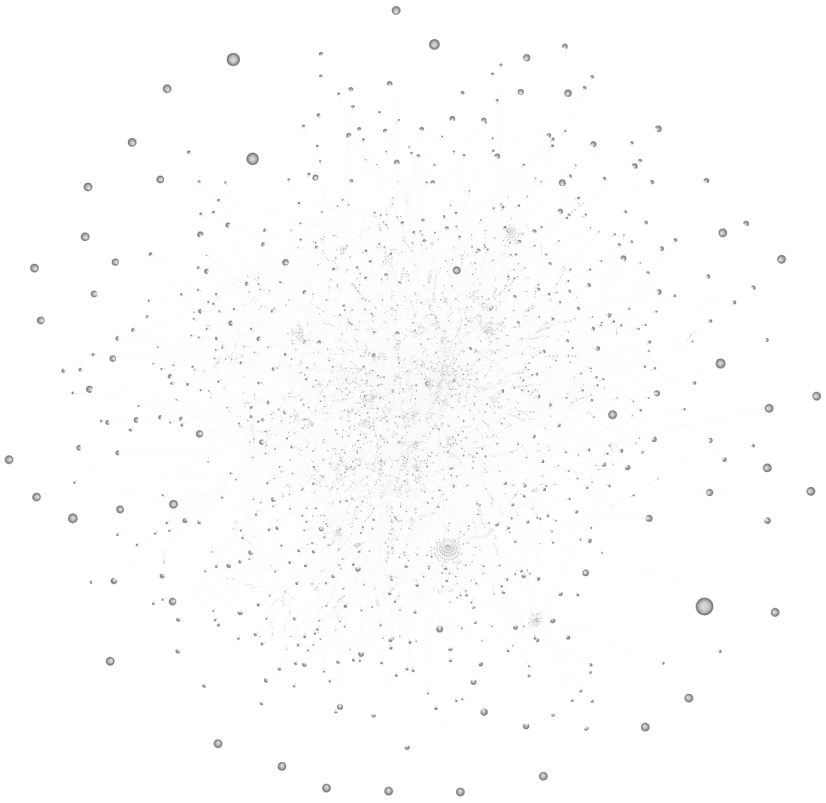
Slika 4. gov.si vse povezave



Slika 5. e-prostor.gov vse povezave



Slika 6. e-uprava.gov vse povezave



Slika 7. Vizualizacija vse spletnih strani, ki jih je obiskal pajek