Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Aljaž Srša in Gregor Sušnik
Implementacija spletnega pajka
Poročilo pri predmetu Iskanje in ekstrakcija podatkov s spleta
Asistent: asist. prof. dr. Slavko Žitnik

Povzetek

Spletni pajek [1] imenovan tudi spletni robot je program, ki na strukturiran način avtomatično preiskuje spletišče z namenom indeksiranja vsebine. V seminarski nalogi poskusiva implementirati preprostega spletnega pajka. Pri implementaciji ne uporabljava knjižnic, ki delno že implementirajo delovanje spletnega pajka. Posledično sva naletela na številne težave, ki sva jih morala reševati sistematično in premišljeno. Na koncu preizkusiva izdelanega spletnega pajka na v naprej določenih spletnih mestih in pridobljene rezultate vizualno predstaviva in pokomentirava.

1. Uvod

Iskanje vsebine po spletišču je za uporabnika nadvse preprosto. V priljubljen iskalnik vnese iskano geslo in že v nekaj trenutkih dobi ogromno spletnih strani, ki se navezujejo na iskano geslo. V ozadju delovanja iskalnika pa se neprestano dogaja veliko več. Spletni pajek obiskuje in preiskuje ogromno količino spletnih strani in hkrati obdeluje še večjo količino podatkov z namenom, da poskuša kategorizirati (indeksirati) spletno stran. Sama arhitektura spletnega pajka je odvisna od njegovega namena in je v večini primerov kompleksna. Spletni pajek mora za dobro delovanje obiskati in obdelati čim večje število spletnih strani, to pa pomeni izkoriščanje vseh možnih strojnih virov, ki so na voljo.

V prvi seminarski nalogi smo se lotili problema implementacije spletnega pajka. Pri implementaciji smo bili omejeni pri uporabi knjižnic. Uporabili smo lahko le tiste, s katerimi si lahko nekoliko olajšamo programiranje, ne pa tistih, ki v neki meri že implementirajo spletnega pajka. Ideja seminarske naloge je v poljubnem programskem jeziku izdelati bolj preprost primerek spletnega pajka in poizkusiti njegovo delovanje na določenih spletnih mestih.

1. Implementacija pajka

1.1. Oris spletnega pajka

Problema sva se lotila sistematično. Najprej sva načrtala glavne dele spletnega pajka in razmišljala o tehnologijah, ki bodo pokrivale te dele. Za implementacijo logike spletnega pajka sva izbrala programski jezik Python. Podatkovna baza, v katero spletni pajek shranjuje podatke med svojim delovanjem, je bila že v naprej določena. Uporabiti je bilo potrebno odprtokodno rešitev PostgreSQL. Podana je bila tudi skripta, ki je v podatkovni shrambi pripravila ustrezne tabele in pravila. Seznam, v katerega se odlagajo spletne strani, nisva implementirala posebej. Odločila sva se, da bo vlogo seznama pri nama imela kar tabela.

1.2. Arhitektura pajka

Logiko spletnega pajka sva razdelila na več delov. Glavna logika, ki štarta celotnega pajka se nahaja v datoteki main.py. V njej so nahajajo trije parametri, ki se lahko nastavljajo. To so naslednji parametri:

- seedPages – predstavlja seznam vstopnih spletnih strani. Nad izbranimi stranmi spletni pajek izvaja preiskovanje,

- threads – predstavlja število spletnih pajkov naj se »ustvari« sočasno. Spletni pajki nato sočasno obiskujejo strani – sistem večnitnega delovanja spletnega pajka,
- option – predstavlja izbiro ali se dokumenti oz. datoteke prenesejo na računalnik in hkrati shranijo v binarni obliki v tabelo ali pa se ne prenesejo na računalnik in se delno zabeležijo v tabeli – shrani se podatek, da je spletni pajek naletel na dokument, vsebina dokumenta pa se ne shrani.

V glavnem main programu se nahaja logika, ki obdela seznam vstopnih spletnih strani in ustvari izbrano število niti – spletnih pajkov. Vsak spletni pajek se nato prične izvajati. Spletni pajek je predstavljen kot razred. Razred ima definirano metodo run, ki se izvaja toliko časa, dokler še obstajajo neobdelane spletne strani v frontier. Logika obdelave slikovnega objekta je implementirana v ločeni datoteki. Prav tako je v ločeni datoteki implementirana logika obdelave dokumenta. Vsa komunikacija med podatkovno bazo in programom je implementirana v ločeni datoteki in je tako za boljšo preglednost ločena od ostale programske kode. Tudi ogrodje spletnega pajka je v svoji datoteki. Implementacija ogrodja je zelo povezana z implementacijo obdelave slikovnega objekta oz. dokumenta. Ogrodje skrbi za iskanje hiperpovezav in slik ter dokumentov. Samo obdelavo pa prepusti zgoraj omenjenim datotekam.

1.3. Implementacija seznama fronte

Seznam, v katerega se odlagajo linki po principu »FIFO«, sva implementirala s pomočjo tabele Page. Ko spletni pajek na neki spletni strani obdeluje <a> značke HTML ali pa dogodek onClick, najdene hiperpovezave zabeleži v tabelo Page z oznako FRONTIER. Hiperpovezave se v tabelo pišejo takoj, ko jih spletni pajek obdela. Posledično ima tista hiperpovezava, ki se zabeleži prej manjši atribut ID v tabeli, kot pa tista, ki se zabeleži kasneje. Na podlagi tega je sistem »FIFO« že zagotovljen. Odvzemanje naslovov URL s fronte sva implementirala tako, da pajek v tabeli Page najde prvi zapis, ki ima oznako FRONTIER. Le-temu takoj spremeni oznako na null in nadaljuje z obdelavo. V tem delu kode uporabljava še ukaza threadLock.acquire() in threadLock.release(), saj morajo preostali spletni pajki, ki se izvajajo v nitih, počakati, da se zapis spremeni in šele nato lahko odvzamejo naslednji naslov url iz seznama. Tako zagotoviva, da vsaka nit obdeluje svoj url naslov in ne pride do konflikta pri odvzemanju naslovov iz seznama fronte.

1.4. Opis delovanja spletnega pajka

Podatki, ki jih pridobi spletni pajek se sproti shranjujejo v ustrezne tabele. Začetnim izbranim spletnim stranem se na začetku pregleda, ali obstaja datoteka robots.txt oz. sitemap. Vse skupaj se zabeleži v ustrezno tabelo. V primeru, da obstaja sitemap, spletni pajek iz nje izlušči vse spletne naslove in jih uvrsti na seznam fronte. Iz datoteke robots.txt se z uporabo knjižnice urllib prebere dovoljen čas pošiljanja zahtevkov na stran – t.i. crawling delay. Iz omenjene datoteke se pri procesu obdelovanja spletnih naslovov upošteva tudi to, ali lahko spletni pajek obišče določeno podstran ali ne.

Spletni pajek začne izvajati while zanko, ki se zaključi, ko v tabeli ni več spletnih strani, ki imajo oznako »FRONTIER«. Če še obstaja kaka taka spletna stran, jo obdela s pomočjo funkcije processSeed(). Omenjena funkcija je srce spletnega pajka. Skrbi za procesiranje dokumentov, slik in obdelovanja hiperpovezav. Če spletni pajek naleti na stran, ki je označena kot 'BINARY' jo s pomočjo funkcije processBinary() obdela. Funkcija processBinary() poskrbi, da se dokument pretvori v binarno obliko in se zapiše v ustrezno tabelo in dokument prenese na računalnik v mapo 'media'. Če spletni pajek naleti na sliko, jo s pomočjo funkcije processImg() obdela. Funkcija prav tako poskrbi, da se slika pretvori v binarno obliko in zapiše v ustrezno tabelo, če je parameter option nastavljen na 0. V primeru, da je option nastavljen na 1 se prenos na računalnik ne zgodi. Prav tako pa se ne zgodi pretvorba vsebine dokumenta oz. slike v binarni zapis. To pomeni, da se v tabelo poleg ostalih parametrov zapiše tudi vsebina, ki pa je nastavljena na null. Opisano velja tudi za funkcijo processBinary(). Pri obdelavi hiperlinkov spletni pajek izvede funkcijo processFrontier().

Funkcija processFrontier() pri obdelavi hiperlinka pridobi podatka statusne kode – html status code in čas dostopa do spletne strani. Spletni pajek s pomočjo knjižnice Selenium [2] in spletnega brskalnika Chrome pošlje zahtevek na spletno stran in kot rezultat dobi HTML kodo. Pri nastavljanju Seleniuma sva upoštevala zastavico »HEADLESS«. Nastavila sva tudi čas čakanja nalaganja spletne strani. Ta čas znaša 3 sekunde. Spletni pajek ob poslanem zahtevku počaka 3 sekunde, nato pa se loti nalaganja spletne strani. Ker pa sva med delovanjem pajka opazila, da se veliko spletnih strani zelo počasi odzove, sva spletnemu pajku nastavila največje dovoljeno čakanje na spletno stran. Tega sva nastavila na 20 sekund. Po 20 sekundah spletni pajek opusti trenutno stran in se loti naslednje strani iz seznama fronte. Ob uspešnem »prejemu« HTML kode se trenutni spletni strani izračuna kanonična oblika naslova, ki jo uporabiva za detekcijo duplikatov. Pri tem se odstranijo različni dodani parametri v url naslovu, kot tudi 'https' oz. 'http'. Odstrani se tudi vodilni znak na koncu url naslova'/'. Po vseh zbranih podatkih se v ustrezno tabelo doda nov zapis obdelanem naslovu URL.

1.5. Implementacija večnitne komunikacije s podatkovno zbirko

Za komunikacijo med spletnim pajkom in podatkovno bazo skrbi poseben objekt ThreadedConnectionPool, ki je del knjižnice psycopg2 [3]. Objekt zagotavlja večnitno podporo in orkestrira komunikacijo med aplikacijo in podatkovno zbirko. Objekt se ob zagonu programa inicializira z naslednjimi parametri:

- uporabniško ime in geslo s katerim dostopa do podatkovne baze,
- naslov in vrata na katerem je dostopna podatkovna baza,
- ime podatkovne baze na katero se objekt poveže in
- minimalno in maksimalno število objektov s katerimi se vzpostavi komunikacija med aplikacijo in podatkovno bazo.

Omenjeni objekt posrbi, da je na voljo vedno minimalno število komunikacijskih objektov, vendar jih vseeno ustvari maksimalno podano število. Pri implementaciji spletnega pajka sva nastavila minimalno število komunikacijskih objektov na 5 in maksimalno število komunikacijskih objektov na 10. Komunikacijski objekti, ki niso v uporabi so v stanju mirovanja. Preden spletni pajek shrani nek zapis v podatkovno zbirko, iz objekta ThreadedConnectionPool s pomočjo metode getconn() pridobi komunikacijski objekt. Nato izvede shranjevanje oz. branje in ob zaključku vrne komunikacijski objekt nazaj v ThreadedConnectionPool objekt. V primeru, da se spletni pajek poganja na štirih nitih, omenjen postopek zagotovi, da vsaka nit pridobi komunikacijski objekt s katerim izvaja transakcijo nad podatkovno zbirko. Pomembno pri procesu je proženje metode rollback() v primeru, ko se zgodi napaka med komunikacijo.

1.6. Izzivi pri implementaciji spletnega pajka

Med delovanjem spletnega pajka, sva naletela na raznolike hiperpovezave. Te so bili podani v neki meri pravilno. Bili pa so podani tudi relativno. Takšne linke sva morala obdelati. Lotila sva se jih tako, da sva odstranila znake ‘www’ in s pomočjo knjižnice urllib uporabila metodo urlparse(), ki spletni naslov razbije na posamezne segmente. Tako sva lahko sestavila ustrezen link in ga po potrebi še malo obdelala – odstranila znake ‘/’ in podobno. Zelo podoben postopek sva uporabila pri slikah in dokumentih. Tudi tam so se pojavile težave s podanimi relativnimi oz. absolutnimi potmi in kopico dodatnih parametrov, ki niso potrebni.

Nekoliko težav sva imela pri zajemanju hiperpovezav, ki so vezani na dogodek “onClick”. Tukaj sva izvedla “onClick” dogodka in s pomočjo Selenium objekta driver pridobila spletni naslov za trenutno spletno stran, ki je v objektu driver. Paziti sva morala tudi na to, kako dogodek odpre spletno stran. Nekateri dogodki so spletno stran odprli v novem zavihku, drugi pa v novem oknu brskalnika. Posledično sva morala implementirati logiko, ki zna obdelati “onClick” dogodek in pridobiti spletni

naslov ter zavihek oz. okno zapreti – vrnitev na stran, od koder smo prišli in nadaljevanje zajemanja hiperpovezav.

Ker sva seznam fronte implementirala kar s pomočjo tabele, sva morala zagotoviti, da niti pravilno izvajajo pobiranje elementov s fronte. Niti ne smejo pobirati istih zapisov iz fronte ampak jih morajo pobirati izmenično. Po razmisleku sva izziv rešila tako, da sva uporabila zaklepanje in odklepanje nad delom kode, ki skrbi za odvzemanje elementa s fronte. To sva zagotovila z uporabo t.i "lock objekta" in uporabo metode threading.Lock(). Ko se nit "zaklene" prebere seznam fronte in vzame prvi element. Elementu spremeni tip iz FRONTIER na null nato pa se nit "odklene" in nadaljuje proces. Ko se naslednja nit zaklene in prebere seznam fronte, prejšnjega elementa ni več v njem, saj ni več tipa FRONTIER. S takšnim sistemom zagotoviva, da niti ne obdelujejo istih naslovov URL.

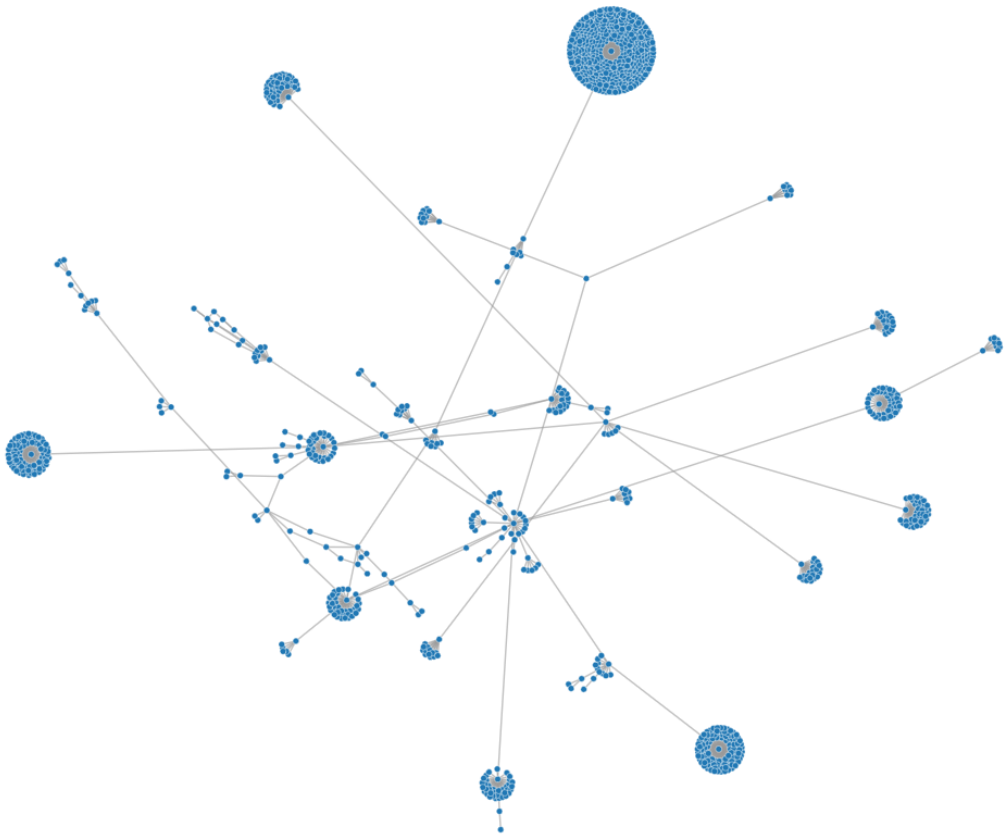
Veliko težavo nama je povzročal Chrome driver v povezavi s Selenium knjižnico. Težava je bila pri pridobivanju kode HTML, ko objekt driver pošlje zahtevek po spletni strani. Pogostokrat se je dogajalo, da je objekt potreboval več minut, da je pridobil kodo HTML. Zaradi prepočasnega odziva sva spletnemu pajku nastavila čas, v katerem mora objekt driver obdelati spletno zahtevo in vrniti kodo HTML. Tega sva nastavila na 20 sekund. Če je objekt prepočasen, se sproži izjema in nadaljuje z naslednjim naslovom URL.

Da se pajek v nedogled ne sprehaja po spletišču, je mehanizem detekcije duplikatov obvezen. Prvi poskus detekcije duplikata je bil računanje hash vrednosti nad vsebino spletne strani. Ker pa se kasneje tak način ni izkazal za najboljšega sva ga opustila in reševala problem s pomočjo kanonične oblike naslova URL. Implementacija le-tega je zahtevala kar nekaj premisleka. Na koncu pa sva dosegla, da se spletne strani v tabelo vpišejo z oznako DUPLICATE. Ideja pa je bila, da bi duplikate le zaznala s pomočjo izračuna hash vrednosti, vendar ne nad celotno vsebijo strani ampak nad nekaterimi deli spletne strani. Denimo nad vsemi značkami tipa <p> ali in podobno.

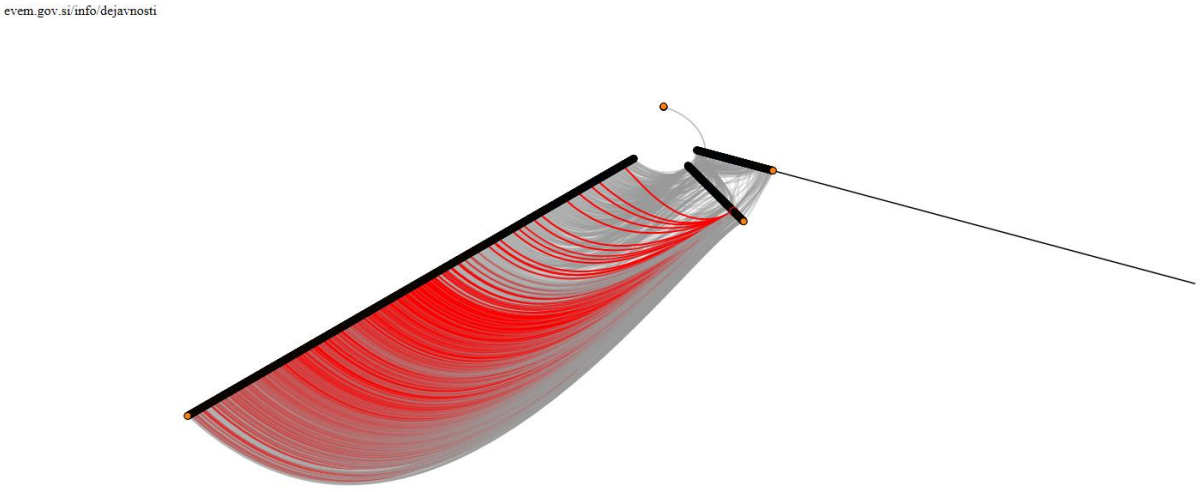
3. Rezultati

Pri vizualizaciji sva si pomagala s knjižnico D3JS [4]. Spodnji dve sliki prikazujeta graf strani in povezav za spletni domeni *e-prostor.gov.si* in *evem.gov.si*.

e-prostor.gov.si



evem.gov.si



evem-gov.si

Tabela rezultatov

Domena	Število spletnih strani – brez duplikatov	Število binarnih datotek	Število slikovnih datotek	Čas izvajanja
evem.gov.si	1222	61	46	4,5h
e-prostor.gov.si	541	225	63	
euprava.gov.si	2669	0	487	12,5h
podatki.gov.si	3305	0	306	
Izbranih 5*	4958	169	0	12h**

*izbrane začetne domene: *mgrt.gov.si*, *mz.gov.si*, *uvps.gov.si*, *mju.gov.si*, *osha.mddsz.gov.si*
**preklic delovanja zaradi memory leaka.

Literatura

[1] „Web crawler,“ [Elektronski]. Available: https://www.sciencedaily.com/terms/web_crawler.htm. [Poskus dostopa 31. 3. 2019].

[2] „Selenium,“ [Elektronski]. Available: https://selenium-python.readthedocs.io/. [Poskus dostopa 1. 4. 2019].

[3] „Psycopg dokumentacija,“ [Elektronski]. Available: http://initd.org/psycopg/docs/. [Poskus dostopa 1. 4. 2019].

[4] „Data-Driven Documents,“ [Elektronski]. Available: https://d3js.org/. [Poskus dostopa 1. 4. 2019].