

Implementacija spletnega pajka

Adam Prestor, Lojze Žust

ap2408@student.uni-lj.si, lojze.zust@student.uni-lj.si

1 Uvod

Spletni pajek je program, ki se sistematično premika po svetovnem spletu. Običajno se uporablja z namenom izgradnje indeksa spletnih strani, ki omogočajo spletnim brskalnikom, kot so Google, da bolj učinkovito preiskujejo splet. Eden izmed bolj znanih spletnih pajkov je Googlebot - Googleov spletni pajek.

V tej seminarski nalogi smo implementirali spletnega pajka, ki se sprehaja po spletišču *gov.si*. Pri implementaciji smo se držali podanih navodil za implementacijo, ki jih bomo podrobneje opisali v naslednjih poglavjih.

2 Metodologija

Za implementacijo smo uporabili programski jezik Python in postgresql bazo.

2.1 Razširitev baze

Osnovni strukturi baze smo dodali nekaj polj, za lažje izvajanje pajka. Tabeli *site* smo dodali polji *delay* in *next_allowed_time*, ki se uporabljata pri omejevanju števila zahtevkov na posamezno domeno, tabeli *page* pa smo dodali polje *html_content_hash*, ki se uporablja pri zaznavi duplikatov.

2.2 Inicializacija

Plazenje vedno začne glavni proces in sicer z dodajanjem začetnih strani v frontir. Poleg tega pogleda v bazo, če obstajajo strani brez oznake – te strani so namreč obtičale sredi obdelave ali pa je prišlo pri njih do napake – ter jih prestavi nazaj v frontir. Ko konča, pogleda koliko procesov naj uporabi. Slednjo informacijo dobi iz argumenta klica funkcija *-n*. Prenastavljena vrednost je 8. Vsak proces posebej potem predstavlja svojega delavca, ki vzema povezave iz frontira in jih ustrezno obdela.

2.3 Plazenje

Plazenje se dogaja znotraj novo nastalih procesov. Vsak delavec zažene Chrome gonilnik, ki ga potrebuje knjižnica Selenium za izločanje vsebine strani. V neskončni zanki delavec pobira strani iz frontirja in jih zajema z gonilnikom. Ker tu lahko pride do primera, da bi lahko več delavcev dobilo isto povezavo, smo uporabili ključavnico. Tako do frontirja lahko dostopa le en delavec na enkrat.

2.3.1 Frontir

Frontir deluje po principu FIFO (first-in-first-out), kar pomeni, da pajek deluje po principu *breadth first search*. To dosežemo tako, da dovoljene povezave v frontirju uredi po id polju, saj imajo povezave, ki so bile dodane prej, nižji id.

Kaj pomeni dovoljena povezava? Vsaka povezava v bazi je povezana z domeno, na kateri jo najdemo. Vsaka domena ima shranjeno, kdaj bomo lahko naslednjič dostopali do nje. Stran je dovoljena, če lahko dostopamo do domene na kateri je stran. Ko stran pridobimo iz frontirja posodobimo tudi domeno in s tem spremenimo dovoljeni čas naslednjega obiska na domeni. Ta se izračuna tako, da trenutnemu času prištejemo zakasnitev domene (*delay*).

Če v frontirju ni dovoljene strani, frontir pa še vedno ni prazen, vrnemo izjemo *FrontierNotAvailableException*. V tem primeru delavec počaka 2 sekundi, da se domene sprostijo, ter poskusi znova. Če pa je frontir prazen, vrnemo *EmptyFrontierException*, v katerem primeru delavec počaka 10 sekund, da se morebitne strani v ostalih delavcih obdelajo. Če je frontir še vedno prazen, potem lahko sklepamo, da smo prišli do konca plazenja.

2.3.2 Domena

Ko delavec pridobi povezave strani, za vsako najprej izlušči domeno in preveri, če je že zapisana v bazi. V nasprotnem primeru jo doda. Pri tem po-

skusi pridobiti tudi *robots.txt* in *sitemap*, če na domeni obstajata. Iz *robots.txt* prebere tudi željeno zakasnitev obiskov, ter jo shrani v polje *delay*. Če slednjega ne more prebrati, potem nastavi privzeto vrednost 5 sekund.

2.3.3 Obdelava vsebine strani

Preden sploh začnemo obdelovati vsebino strani, se prepričamo, če gre sploh za HTML vsebino in če do nje lahko dostopamo. Zato najprej pošljemo head zahtevek s knjižnico *requests* na stran in iz njega izluščimo status in tip. Če gre za napako, potem strani spremenimo tip v bazi na *null* in nadaljujemo s plazenjem naprej. Če stran ne vsebuje headerja *content-type* z vsebino *text/html* potem stran ne vsebuje HTML vsebine in zapišemo, da gre za binarno stran.

Če smo naleteli na navadno HTML stran brez napake, potem uporabimo Selenium gonilnik, da izluščimo vsebino strani. Ta proces omogoča generiranje strani kot bi jo videl uporabnik v brskalniku, vključno z izvajanjem kode JavaScript. Nalaganje lahko traja nekaj časa, zato po zahtevku za stran počakamo 5 sekund preden pričnemo z branjem vsebine. Iz HTML vsebine strani nato izluščimo povezave iz href atributa a elementov in onclick funkcij (s posebnim regexom). Te povezave ločimo na običajne in binarne, binarne se končajo s končnicami, ki so zapisane v tabeli *data_type*. Prav tako izluščimo slike iz slikovnih elementov vsebine, kjer preberemo vsebino atributa src. Slike izločimo tudi iz navadnih povezav (a href in onclick) na podlagi končnic. Vse povezave (navadne, slike in datoteke) kanonikaliziramo z uporabo knjižnice *urltools*. Odstranimo tudi določila fragmentov, ki kažejo na različne dele iste strani.

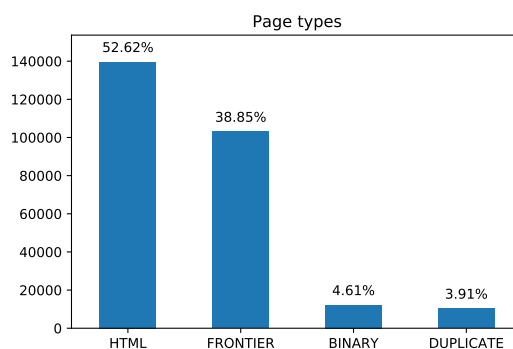
2.3.4 Preverjanje duplikatov

Vsebinsko, ki smo jo dobili z obiskane strani, potem z zgoščevalno funkcijo md5 stisnemo in pogledamo, če je enaka kateri drugi strani, ki je že v bazi. Za hitrejšo iskanje duplikatov smo v bazi nad stolpcem z md5 vsebino strani naredili indeks. Če najdemo ujemanje, potem stran označimo kot duplikat, vsebino pustimo prazno in nadaljujemo s plazenjem.

2.3.5 Dodajanje vsebine in strani

Vsebinsko strani shranimo v bazo, skupaj s statusom in tipom vsebine.

Binarne povezave in povezave do slik shranimo v bazo direktno. Edina stvar, ki jo moramo najti



Slika 1: Porazdelitev števila strani glede na tip.

je tip, ki ga dobimo iz končnice. V primerih, ko v povezavi ni prisotna končnica, tipa ne določimo. Končnico dobimo z razrezom povezave po pikah, kjer zadnji del predstavlja končnico. Če je ta predolga, potem končnice nismo uspeli pridobiti in ponastavimo vrednost tipa na prazen niz. Tip slike bi lahko pridobili tudi z zahtevkom HEAD, vendar se nam zaradi velike količine zahtevkov to ni zdelo smiselno.

Pri dodajanju povezav na nove strani, pa pred dodajanjem v bazo preverimo, če prihaja iz *gov.si* domene. Stran potem poskusimo dodati v frontir, v bazi se preveri, če je url že zapisan ali če *robots.txt* ne dovoljuje dostopa. V teh dveh primerih strani ne dodamo v frontir. V bazo zapišemo tudi povezavo med stranema.

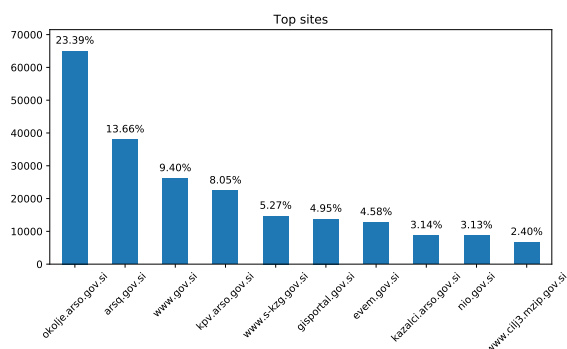
To je bil še zadnji korak obdelave strani. Sedaj je delavec pripravljen na obdelavo nove strani in cikel se ponovi.

3 Rezultati

Spletni pajek se je izvajal skupno 62 ur z 8 vzporednimi procesi. Toliko časa smo ga pustili, ker smo mislili, da bo v tem času lahko preiskal celotno spletišče na domeni *gov.si*, vendar temu ni bilo tako, zato smo ga bili na koncu primorani prisilno ustaviti.

V podatkovni bazi je bilo ob zaustavitvi skupaj 277 954 strani. Od tega so velik del še vedno predstavljale strani v frontirju (skoraj 40%). Prikaz porazdelitve strani po različnih kategorijah je prikazan na Sliki 1. Pajek je detektiral in označil 10 385 strani, ki predstavljajo duplikate in 12 243 strani, ki v resnici predstavljajo binarne podatke. V nadaljevanju analize opazujemo strani, katerih vsebina je bila uspešno zajeta in označena kot HTML.

Med plazenjem je pajek zaznal 354 različnih



Slika 2: Porazdelitev števila strani po najpogostejših 10 domenah.

poddomen. Na Sliki 2 je prikazano število strani, ki pripadajo desetim najpogostejšim domenam. Le dve domeni sta imeli v robots.txt definiran crawl-time drugačen od 5.

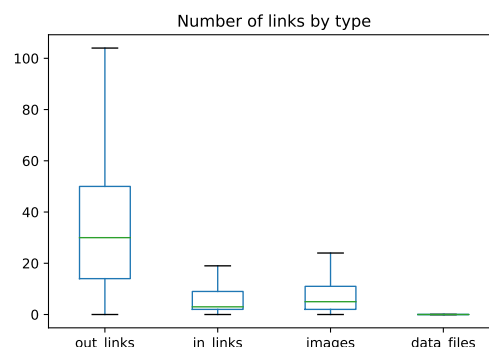
Analizirali smo tudi statistiko povezav. V Tabeli 1 in na Sliki 3 so prikazane osnovne statistike števila vhodnih in izhodnih povezav, slik in datotek. Poiskali smo strani z največjim številom posameznih povezav. Največje število izhodnih povezav vsebuje stran <http://evem.gov.si/info/podpogoji/>. Največ srtani kaže na stran <http://www.arso.gov.si/>. Največ slik vsebuje stran http://meteo.arso.gov.si/met/en/watercycle/maps/growa_si, ki vsebuje povezave na ogromno število slik modelov. Največje število datotek vsebuje stran <https://www.gov.si/drzavni-organi/ministrstva/ministrstvo-za-kmetijstvo-gozdarstvo-in-prehrano/o-ministrstvu/seznami-certificiranih-proizvajalcev/>.

	AVG	MAX
Št. vhodnih povezav	61.57	43660
Št. izhodnih povezav	64.36	2309
Št. slik	6.74	412
Št. datotek	0.67	786

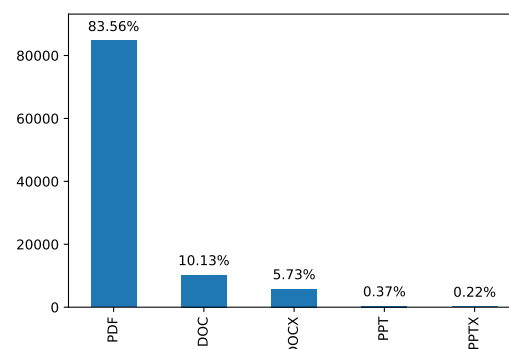
Tabela 1: Povprečje in maksimalna vrednost različnih tipov povezav posameznih zajetih strani.

Podrobneje smo pogledali tudi porazdelitev števila datotek po tipih (glej Sliko 4). Daleč največ datotek je tipa PDF, sledijo Wordove datoteke, zelo majhen delež pa je PowerPoint datotek.

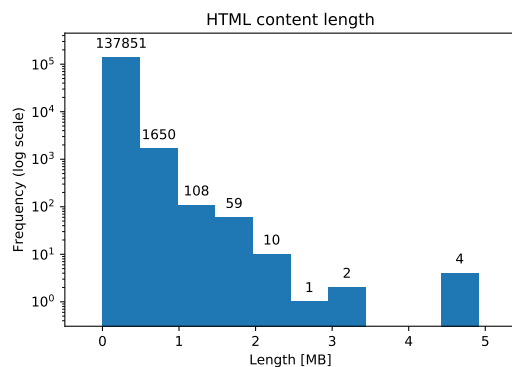
Zaradi ogromne velikosti končne baze strani, smo se odločili raziskati tudi velikost posameznih strani glede na vsebino HTML (glej Sliki 5 in 6). Velikost največje strani v bazi je skoraj 5 MB. V



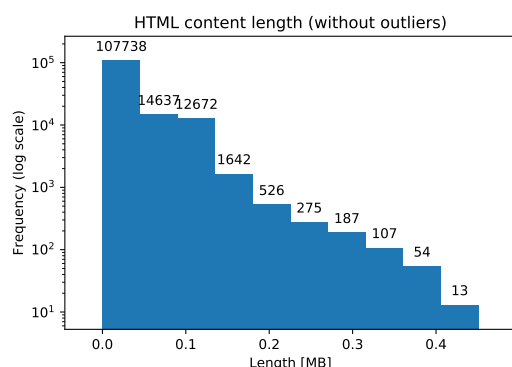
Slika 3: Število različnih povezav in vsebin s prikazom škatle z brki. Osamelci niso prikazani. Povprečna stran vsebuje več izhodnih povezav, kot vhodnih. Vendar obstaja majhen del strani z ogromnim številom vhodnih povezav, ki uravnovesijo povprečno število vhodnih in izhodnih povezav.



Slika 4: Porazdelitev različnih tipov datotek. Daleč največji delež predstavljajo datoteke PDF.



Slika 5: Histogram velikosti HTML vsebine strani v logaritemski skali. Porazdelitev je močno nagnjena proti majhnim stranem.



Slika 6: Histogram velikosti HTML vsebine strani v logaritemski skali. Prikaz brez osamelcev. Prikazane so le strani manjše od 0.5 MB.

splošnem je strani večjih od 1 MB v bazi zelo malo. Veliko večino strani predstavljajo majhne datoteke. Kljub temu pa velike datoteke prispevajo velik del k skupni velikosti. Največjih 20 000 strani prispeva skoraj tretjino skupne velikosti strani.

4 Zaključek

V tej seminarski nalogi smo razvili učinkovitega pajka, ki je sposoben paralelnega zajema strani, upoštevanja etičnih navodil spletnih strani (robots.txt), zaznavanja duplikatov. Z njim smo zajeli veliko število strani z vladnih spletnih strani in jih analizirali. Pajek je deloval robustno in brez prekinitev, zajete strani pa so bile smiselne. V prihodnje bi ga bilo mogoče nadgraditi z delnim preverjanjem duplikatov in ponovnim preverjanjem neuspešno zajetih strani.