1 Implementacija pajka

Spletnega pajka smo implementirali v programskem jeziku Python, pri tem pa smo si pomagali z nekaterimi knjižnicami:
`urllib` iz katere uporabimo pod-knjižnico `robotparser`, ki pomaga pri branju vsebine datoteke robots.txt,
`urlcanon`, ki na podlagi v naprej definiranih in lastnih pravil kanonizira URL povezavo in
`selenium-wire`, ki omogoča odpiranje spletnih strani v spletnem brskalniku. Uporaba tega je potrebna zato, da se spletne strani pravilno prikažejo (npr. z uporabo CSS, poganjanjem JS skript ipd.).

Aplikacija je strukturirana v več datotekah ter napah, in sicer:
mapa `database` vsebuje vse datoteke, ki so potrebne za komunikacijo s podatkovno bazo (v nadaljevanju okrajšano kot PB). V njej so definirane funkcije, kot npr. `get()`, `update()` itd., ki na podlagi prejetih argumentov sestavijo SQL poizvedbo, jo izvedejo ter shranijo podatke v bazi,
datoteka `_crawler.py`, ki predstavlja našo implementacijo ene instance spletnega pajka in
datoteka `process.py`, ki ob poganjanju ustvari določeno število instanc pajka, ki se izvajajo sočasno.

V tabelo `page` dodali stolpec `html_hash`, ki vsebuje zgoščeno vrednost HTML strani, v `data_type` ter `page_type` smo dodali nekaj dodatnih vrednosti, dodali pa smo tudi tabelo `site_ipaddr` s stolpci `ip_addr`, v katerem hranimo unikatne IP naslove domen, `last_access`, v katerem hranimo čas zadnjega dostopa kateregakoli pajka na to domeno, ter `delay`, v katerem hranimo minimalen čas čakanja med poizvedbami.

1.1 Delovanje naše implementacije

Proces poženemo z ukazom `python process.py --threads n`, kjer n predstavlja maksimalno število niti. Pred ustvarjanjem pajkov se ustvarita še dve ključavnici, s katerimi preprečujemo, da več pajkov hkrati bere ali piše iz baze – ena zaklepa dostop do naslednjega URL-ja v vrsti (*frontier*), druga pa zagotavlja upoštevanje časa čakanja, skladno z zahtevo iz datoteke robots.txt oz. vsaj 5 sekund.
Delovanje posameznega pajka je sledeče: iz vrste se vzame naslednji URL ter se mu spremeni stanje v `PROCESSING`, v PB se preveri ali prvič dostopamo do te domene, ter glede na rezultat prenesemo vsebino robots.txt iz PB ali iz spletne strani ter jo ustrezno razčlenimo.
Pajek nato prenese vsebino, ki se nahaja na URL-ju (po potrebi tudi počaka, da mine minimalni čas čakanja med poizvedbami). V kolikor gre za binarno datoteko, se v PB ustrezno shranijo podatki vezani nanjo, sicer pa pregledamo stran za morebitne povezave (v značkah `<a href="...">`, `<img src="...">` ter atributih `onclick`). Če v povezavi zaznamo končnico, ki bi nakazovala binarno vsebino, jo kot takšno takoj shranimo v bazo, da ne dodatno obremenjujemo pajka kasneje; v tem primeru ne zabeležimo časa dostopa, saj vsebine nismo prenašali. Ko je vsebina URL-ja pregledana, se izvajanje pajka ponovi od začetka.
V kolikor pride med izvajanjem do kritične napake ali pa stran ne vrne statusa 200, zapišemo takšen URL kot `TRASH` v PB.

1.2 Težave med razvojem

Največ težav med razvojem smo imeli pri testiranju pravilnega delovanja, saj smo le-to lahko pravzaprav izvajali komaj, ko smo proces iskanja pognali na veliki količini strani. V tem postopku smo ugotovili, da način preverjanja vsebine URL povezave (tj. ali gre za datoteko ali za HTML vsebino) ne deluje pravilno, ter smo ga morali spremeniti. Prav tako smo ugotovili, da se določeni deli kode niso izvajali, saj nismo preverili kako se parametri iz Selenium-Wire knjižnice shranjujejo. Ugotovili smo tudi, da vgrajena knjižnica `robotparser` nepravilno razčlenjuje nekatera pravila omejevanja dostopa, kar smo rešili z uporabo lastnih regularnih izrazov. Občasno smo tudi naleteli na manjše težave pri izvajanju SQL skript oziroma pomanjkanje določenih funkcionalnosti (npr. sortiranje).
Kljub pravilnemu izvajanju naše implementacije spletnega pajka, smo opazili, da se strani na domeni `arhiv-spletisc.gov.si` v večini primerov niso nalagale (prejemali smo napako, da se DNS zapis ni razrešil, tudi če smo stran obiskali v običajnem brskalniku). Domnevamo, da smo bili na domeni blokirani, zaradi česar smo zapise v bazi z navedenimi URL-ji umaknili iz frontier-a. Zato so nekateri zapisi obdelani pravilno, nekateri so bili samodejno premaknjeni v `TRASH`, nekatere pa smo ročno umaknili iz vrste, ter jim dodelili tip `WAIT`.

2 Statistike

V bazi imamo **155.677 zapisov**, od tega: 50.6k tipa HTML, 48.5k FRONTIER, 47.5k BINARY, 4k WAIT, ∼3k DUPLICATE in ∼2k TRASH. Najpogostejše zapise po domenah najdemo na e-uprava.gov.si, ki ima 69.727 zapisov (od tega 17.504 obdelanih HTML zapisov), www.gov.si z 53.5k zapisi (od tega 13.6k obdelanih) ter podatki.gov.si z 18k zapisi (obdelanih je ∼8k strani). Najmanj obdelanih strani imamo iz domen *.arhiv-spletisc.gov.si. Ugotovili smo, da najdenih 228 domen gostuje na 69 različnih IP naslovih, po večini 84.39.*.*.
Tudi **datotek** smo našli precej, največ jih je tipa PDF in sicer 18k. Sledita DOC in DOCX, ki smo jih našli ∼6k, nato XLS in XLSX, ki jih je ∼2k. Najmanj smo našli PowerPoint predstavitev, ki jih je le 117. V največji meri se datoteke nahajajo za domeno www.gov.si, kjer jih gostujejo nekaj več kot 20k, nato sledi e-uprava.gov.si, kjer jih najdemo približno 4k. Povprečno se na vsaki strani nahajajo po 3 datoteke.
Med binarnimi datotekami se nahajajo tudi **slike**. Največ jih je tipa JPG oz. JPEG, in sicer 14.411, nato sledi PNG s skoraj 2.5k zapisi, nato še GIF s slabimi 750 zapisi. Najredkejše so datoteke tipa AI ter BMP, ki jih je skupaj le 10. Tudi tukaj je največ zapisov na domeni www.gov.si, kjer je približno 10k slik, sledita pa kazalci.arso.gov.si s 3.2k slikami in www.fu.gov.si, kjer je pa le dobrih 450 slik. Povprečno se na vsaki strani nahaja 4,6 slik.
Največ zavrženih zapisov (tj. zapisov tipa `TRASH`) je vrnilo kodo 404 - teh je bilo 486. Sledi 378 strani, ki ni vrnilo statusa; ve največji meri se je to zgodilo, ko je stran želela od uporabnika pridobiti certifikat. Če strani ločimo po vrnjeni statusni kodi jih je 795 prejelo kodo 4XX, 386 jih ima status 50X, nato pa še 186 s kodo 30X. Stranem, ki se niso naložile, smo dodelili statusno kodo 0, katerih je skupno 55.
Ugotovili smo tudi, da precej strani vsebuje povezavo, ki kaže na isto stran (tj. samo nase). Takšnih strani je kar 81,53%.

2.1 Vizualizacija povezav med domenami

Na spodnji sliki je prikazana vizualizacija povezav med 31 najbolj povezanimi stranmi. Debelina črte prikazuje količino povezav, velikost kroga pa količino dohodnih povezav. Na sliki precej povezav niti ni prikazanih, saj je razmerje med najtanjšo in najdebelejšo črto 1:275. Iz tega lahko ugotovimo, da večji krogi brez narisane črte, vsebujejo precej povezav, ki pa so porazdeljene na veliko domen. Obratno velja pri velikih krogih z debelo črto, saj se pri njih najpogosteje znajdejo povezave le iz ene ali dveh domen.