Spletni pajek - Poročilo

1. domača naloga pri predmetu WIER

Jakob Merljak, Kristjan Panjan, Blaž Dobravec

MENTOR: asist. prof. dr. Slavko Žitnik

1 Uvod

Za prvo nalogo pri predmetu Iskanje in ekstrakcija podatkov iz spleta smo implementirali preprostega večnitnega spletnega pajka, ki se sprehaja v domeni *.gov.si*. V tem dokumentu bomo najprej predstavili implementacijo in njene značilnosti, nato pa bomo predstavili rezultate in vizualizacijo.

2 Implementacija

2.1 Strategije plazenja

Strategija plazenja pajka je določena z vrstnim redom, po katerem se obdelujejo najdene strani. Implementirali smo Breadth First Search (BFS). Struktura večine spletnih strani vodi v zaporedja najdenih povezav, ki kažejo na isto domeno, zato precejšen del delovanja pajka niti čakajo v vrsti za pošiljanje zahtevka na isto domeno. To v praksi pomeni, da velik del časa pajek obratuje samo v eni niti, kar upočasnjuje njegovo delovanje in zmanjšuje zmožnost obdelave večjega števila strani. Zato smo implementirali še strategijo rotiranja domen, kjer se vsaki niti poskuša dodeliti stran z druge domene. Pri tem se domene sortirajo po času zadnjega dostopa, za vsako domeno pa se pridobi najprej vstavljene strani.

2.2 Interakcija s podatkovno bazo

Pajek s podatkovno bazo komunicira preko podatkovne strukture *DB*, ki vsebuje povezavo na podatkovno bazo in ključavnico, s katero preprečuje izvajanje operacij več kot eni niti naenkrat. Vse operacije na podatkovni bazi, ki so potrebne v drugih gradnikih pajka, so implementirane znotraj te strukture, kar omogoča ustrezno deljenje povezave in sinhronizacijo niti.

2.3 Razvrščevalnik

Pajek upošteva časovni razmak med zahtevki za vsako domeno. Za to uporablja podatkovno strukturo *Scheduler*, ki za vsako domeno hrani časovni žig zadnjega dostopa in ključavnico. Pred pošiljanjem zahtevka na dano stran se nit postavi v čakalno vrsto na ključavnici za ustrezno domeno. Ko nit pride na vrsto in zaklene čakalno funkcijo, se ji izračuna preostali potreben čas čakanja glede na zadnji dostop in trenutni čas. Nit zaspi za izračunani čas. Po končanem spanju se osveži časovni žig zadnjega dostopa, nit zapusti funkcijo in odklene ključavnico. Nato nit pošlje zahtevek na dano stran. Poleg ključavnice in časovnega žiga razvrščevalnik shranjuje tudi podatke za domene in metodo za zapis teh podatkov v podatkovno bazo.

2.4 Frontier

V podatkovno strukturo *Frontier* pajek vstavlja najdene strani in iz nje pridobiva naslednje strani za obdelavo. *Frontier* pri vstavljanju obdela povezavo in preveri ali pravila plazenja za ustrezno domeno pajku dovolijo dostop do strani. Vstavljene strani hrani v vrsti (FIFO) pri strategiji BFS. Pri strategiji rotiranja domen pa nove strani sproti prebere iz podatkovne baze.

2.5 Plazenje po posamezni strani

Pri obdelavi strani se najprej prenesejo njene zaglavne informacije (angl. headers). S tem ugotovimo tip vsebine strani in njeno dostopnost. Če stran predstavlja dokument ali sliko, se pridobljene informacije zapiše v bazo in označi stran. V primeru napake, preusmeritve ali tipa strani, ki nas ne zanima, smo v shemo podatkovne baze dodali nove tipe strani: *ERROR*, *REDIRECT* in *OTHER*. Če se na dani strani nahaja HTML dokument, se dokument prenese s knjižnico Selenium, razčleni in vstavi povezave ter slike dokumenta v frontir.

2.6 Večnitnost

Večnitno izvajanje je implementirano s podatkovno strukturo *ThreadPoolExecutor* iz standardne knjižnice. Pri implementaciji smo naleteli na več ovir. Ni nam uspelo najti načina za ustrezno prekinitev niti, zato se niti izvajajo paketno. V neskončni zanki ustvarimo niti, vsaki podamo eno stran v obdelavo in po končanju niti zaspimo za majhno časovno enoto, kar uporabniku omogoči prekinitev programa s signalom SIGINT ali analogno metodo. Pomanjkljivost je, da se morajo niti medsebojno čakati preden lahko začnejo obdelovati naslednje strani, pri čemer se izgublja čas. Ta izguba se še dodatno poveča v primeru, da morajo niti čakati v vrsti na isto domeno. To težavo bi lahko rešili z uporabo modula *multiprocessing*, ki omogoča ustrezno odzivanje na signale, vendar bi pri tem morali zavreči trenutno zasnovo deljenja virov med nitmi in preiti na model komuniciranja med procesi preko vrst.

2.7 Detekcija duplikata

Detekcijo duplikata smo implementirali s pomočjo dodatnega parametra v *crawler.site*, in sicer "*html_hash*". Parameter je določen na podlagi HTML vsebine posamezne

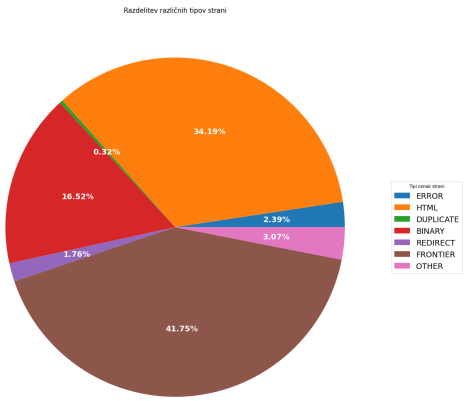
strani in služi za preverjanje enakosti vsebine. Za detekcijo podobnosti smo najprej pregledali literaturo in si izbrali članek [1], vendar nam je nato za implementacijo zmanjkalo časa.

3 Analiza rezultatov

Analiza je sestavljena iz dveh delov. Prvi del se osredotoča na število posameznih atributov, ki jih je naš spletni pajek tekom brskanja našel. Drugi del pa se nanaša na grafično analizo, in njeno interpretacijo.

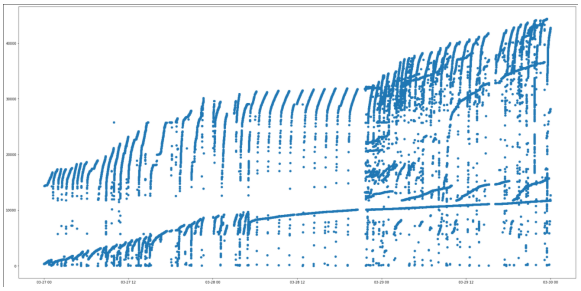
Tip dokumenta	Št. najdenih
DOC	910
PPT	7
DOCX	1390
PDF	7182
PPTX	29

V tabeli 3 opazimo, da je največ PDF datotek v preiskanih domenah. Zanimala nas je tudi razporeditev tipov posameznih pregledanih strani. Rezultati so prikazani na sliki 1.



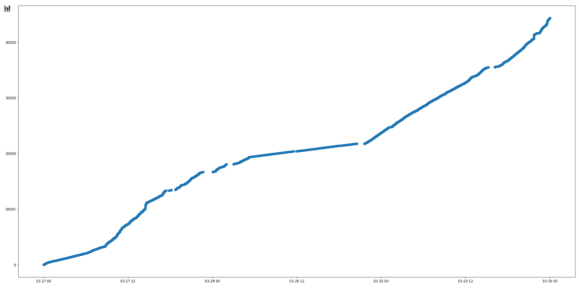
Slika 1: Pregled porazdelitve tipov pregledanih stran

Način preiskovanja strani našega pajka v času je prikazan na sliki 2, kjer je dobro videna implementacija BFS.



Slika 2: Neurejeno komulativno iskanje strani skozi čas

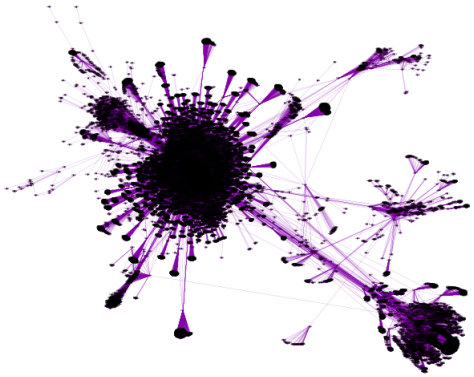
Ko pa si pogledamo urejene rezultate na sliki 4 lahko opazimo hitrost iskanja. Vmesne ”luknje” se pojavijo zaradi manualnega ustavljanja iskanja in posodobitve kode za izboljšanje.



Slika 3: Urejeno komulativno iskanje strani skozi čas

4 Vizualizacija rezultatov

Vizualizacija je ponazorjena s točkami - črnimi pikami, ki predstavljajo posamezno stran, in pa s povezavami - linki, ki so vijolične črte in predstavljajo povezave med stranmi.



Slika 4: Mrežni diagram preiskanih strani.

5 Zaključek

Uspešno smo realizirali vse zadane naloge. Implementirali smo delovanje z več nitmi in iskanje pajka v širino. Spletni pajek je sposoben zaznati duplikatne spletne strani in upoštevati datoteko robots.txt. V nadaljevanju, bi lahko kot razširitev tega projekta analizirali in primerjali našo implementaicjo z implementacijami drugih sošolcev ali z znanimi obstoječimi implementacijami spletnih pajkov.

Literatura

[1] Jonathan Koren. Near-duplicate detection. March 2015. [Online; accessed 30-March-2021].