Jaka Krajnc in Janez Škrlj

Poročilo spletnega pajka

Seminarska naloga pri predmetu iskanje in ekstrakcija podatkov s spleta

MENTOR: doc. dr. Slavko Žitnik

V poročilu smo predstavili razvoj spletnega pajka, ki pridobiva strani iz domene .gov.si. Predstavili smo uporabljena orodja za razvoj in opisali glavne funkcionalnosti pajka ter način njegovega delovanja. Predstavili smo rezultate in vizualizacijo pridobljenih podatkov

1. UVOD

Spletni pajek je internetni bot, ki sistematično pregleduje svetovni splet, običajno za namen spletnega indeksiranja. Spletni iskalniki in nekatera druga spletna mesta uporabljajo programsko opremo za pajkanje po spletu za posodobitev spletne vsebine ali indeksov drugih spletnih mest.

Pajki pogosto obiščejo spletna mesta brez odobritve. Obstajajo mehanizmi za javna spletna mesta, ki omogočajo, da se neke spletne strani naj ne bi obiskovali s pomočjo določenega agenta. Datoteka robots.txt lahko na primer določa, do česa lahko določen agent dostopa in do česa ne sme.

Število spletnih strani je zelo veliko; celo največji pajki nimajo popolnega indeksa. Zato so iskalniki v prvih letih svetovnega spleta pred letom 2000 težko iskali ustrezne rezultate iskanja. Danes so pomembni rezultati podani skoraj takoj [1].

V poročilo bomo predstavili razvoj spletnega pajka, ki bo pridobival podatke iz vladnih strani. Pajek bo shranjeval vse strani, slike ter dokumente, ki jih bo pridobil na spletnih mestih, prav tako bo hranil pot, ki jo je opravil ter upošteval omejitve, ki bodo podane v datoteki robots.txt

2. Implementacija

Spletnega pajka smo implementirali v programskem jeziku Python. Glavni deli programa so sestavljeni iz vrste (frontier), delavcev (workers), vozlišča (node) ter modula za shranjevanje podatkov v podatkovno bazo. Delavci vzporedno jemljejo vozlišča iz vrste in jih sprocesirajo, analizirajo ter shranijo v podatkovno bazo.

2.1 Vrsta (Frontier)

Vrsta hrani podatke o straneh, ki jih še moramo obiskati. Vrsta je implementirana tako v podatkovni bazi kot v pomnilniku programa. Da zagotovimo BFS procesiranje, iz podatkovne baze prenašamo vozlišča v vrsto po zaporednih številkah dodajanja. Da vsakemu delavcu ni potrebno v podatkovno bazo po naslednje vozlišče imamo v programu implementirano večprocesno vrsto iz katere delavci pobirajo vozlišča. Omenjena vrsta se ob padcu velikosti pod določen nivo ponovno napolni iz podatkovne baze, dokler v njej ne zmanjka vozlišč, ki so označena kot frontier. V primeru, da se na kakšnem iz vozlišč zgodi napaka, v podatkovni bazi hranimo tudi število procesiranj, ko to doseže določeno mejo (v našem primer pet neuspešnih procesiranj), potem se to vozlišče več ne prenaša nazaj v vrsto.

2.2 Detekcija duplikatov

Ob dodajanju novih zapisov v frontier vedno preverimo, če isti zapis že obstaja, če obstaja ga ne dodamo. Vse urlje razčlenimo s pomočjo pythonove knjižnice urlparse, da zagotovimo kanonizirano obliko urlja in preprečimo morebitno podvajanje strani. Prav tako za vsako stran preverimo če njena zgoščena vrednost že obstaja. V kolikor obstaja jo dodamo v podatkovno bazo kot duplikat.

Prid dodajanju urljev v frontier ignoriramo urlje, ki so daljši od določegena števila znakov, saj ne želimo, da bi se pajek ujel v past, kjer bi se url nadaljeval zelo daleč naprej.

2.2 Procesiranje vozlišč

Delavci vozlišča pobirajo iz vrste in jih procesirajo. Za vsako vozlišče najprej preverimo, če smo za to lokacijo že pridobili podatke o vsebini datoteke robots.txt

ter vsebino datoteke sitemap. V kolikor potrebnih podatkov še nimamo, jih pridobimo, v kolikor pa so že shranjeni v podatkovni bazi, pa jih pridobimo iz nje. Nato preverimo, če lahko do dotičnega vozlišča sploh dostopamo na podlagi vsebine robots.txt, ter na podlagi zadnjega dostopa preverimo če je preteklo dovolj časa med zahtevami na dotično spletišče. V kolikor ne smemo dostopati do podane lokacije, potem vozlišče ignoriramo in ga ne vrnemo nazaj v vrsto, v kolikor pa še ni preteklo dovolj časa, pa vozlišče vrnemo v vrsto. Vsako vozlišče shranimo v podatkovno bazo, ter vse pripadajoče povezave (link) ter dodamo končna vozlišča kot nove vnose v tabelo page z oznako frontier.

2.2 Shranjevanje datotek in slik

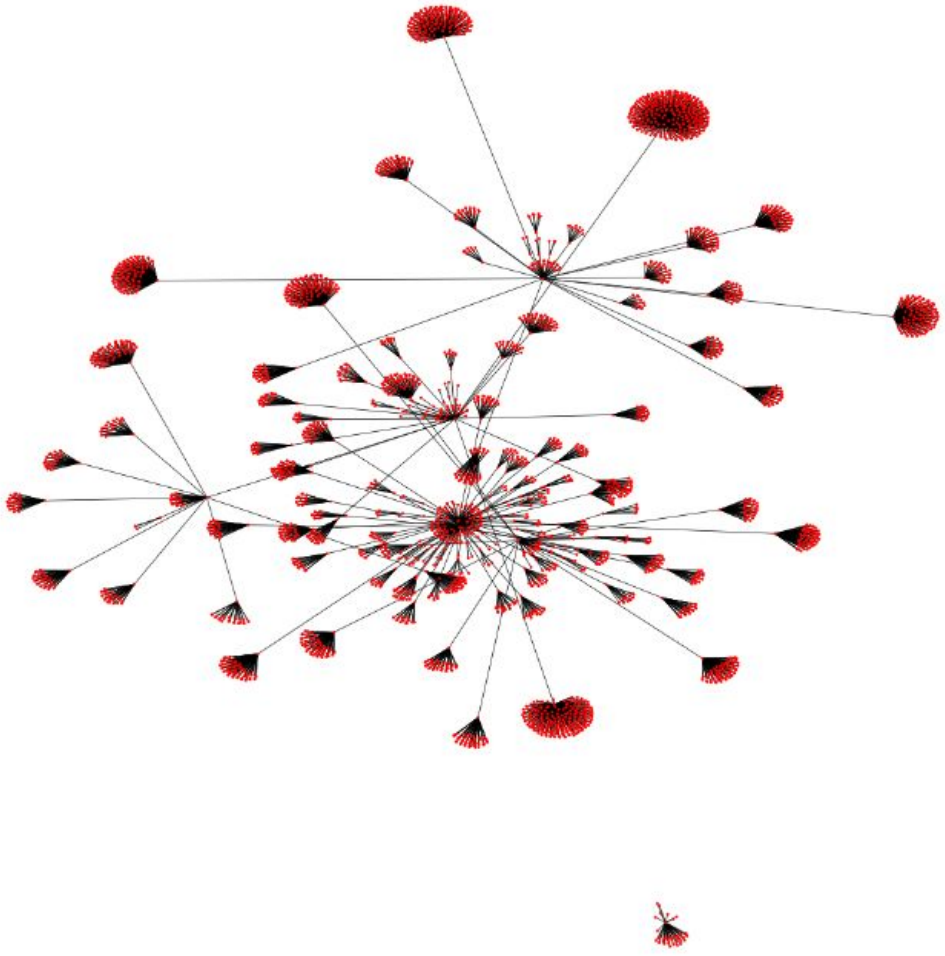
Shranjevanje binarnih datotek v bazo je potekalo z uporabo knjižnice requests, s katero smo najprej prenesli datoteke iz spleta in nato shranili njihovo binarno vsebino v samo bazo.

2.3 Podatkovna baza

V podatkovno bazo smo dodali dodatna polja v tabelo page, kjer spremljamo tudi število neuspešnih pridobitev spletne strani. Ko dosežemo pet neuspešnih poskusov potem strani več ne obiskujemo. Dodali smo tudi stolpec hex_digest, ki predstavlja zgoščeno vrednost html strani, ki smo jo pridobili iz spleta. Na podlagi te vrednosti ugotovimo, če smo isto stran že dobili preko drugega urlja

5. Rezultati in vizualizacija

Za vizualizacijo povezav smo uporabili knjižnici matplotlib in networkx. Pri izrisu grafa povezav nismo izbrali vseh vrstic iz baze, ker bi graf postal nepregleden, zato smo njihovo število omejili. Posamezni skupki predstavljajo spletna mesta na katere kaže največ povezav, oziroma ostalih spletnih mest. Vidimo, da so spletišča v večini povezana med seboj. Imamo pa tudi nek osamelec, ki ni povezan z ostalimi skupki. Ta predstavlja novo, ročno dodano stran, do katere se pajek ni dokopal sam, pač pa smo jo na začetku določili mi. Zaradi tega tudi ni povezan z ostalimi, ker na to spletno mesto ne kaže nobena povezava. Vendar pa to, da bodo vsa ročno dodana spletna mesta nepovezana, ne velja vedno. To ne velja zaradi tega, ker obstaja možnost, da eno izmed ročno dodanih spletnih mest kaže na drugo spletno mesto, ki je bilo ravno tako dodano ročno. V primeru slike 1 je skupek osamljenega spletnega mesta manjši od ostalih zaradi tega, ker nismo iz baze izbrali vseh vnosov.



Slika 1: Vizualizacija povezav

6. ZAKLJUČEK

Za razvoj spletnega pajka smo porabili veliko časa. Največ dela smo imeli z zagotavljanjem zanesljivost delovanja spletnega pajka, ker smo morali od začetka razviti program, ki se bo samostojno izvajal v več procesih, dostopal do spletnih mest o katerih ne vemo nič in pridobljene podatke zapisoval v podatkovno bazo. To nam je uspelo z uporabo programskega jezika Python in njegovih knjižnic. Razviti spletni pajek je preiskal strani, ki smo mu jih določili na začetku. Zbrane podatke pa je sproti zapisoval v podatkovno bazo. Delovanje spletnega pajka bi lahko pohitrili tako, da bi ga poganjali na računalniku, ki ima več jeder in niti ter dostop do hitrejše internetne povezave. Naslednja izboljšava bi bila, da bi bazo prestavili na oddaljen strežnik ter spletnega pajka poganjali na več različnih računalnikih. V tem primeru bi dosegli večjo količino obiskanih spletnih strani v krajšem času obenem pa bi morali paziti na ACID [2] zahteve podatkovne baze.

LITERATURA

[1] https://en.wikipedia.org/wiki/Web_crawler (Dostopano 2. 4. 2019)

[2] https://en.wikipedia.org/wiki/ACID_(computer_science) (Dostopano 2. 4. 2019)