Aljoša Omejc in Robert Košir
https://github.com/roalj/Seminar1-Web-Crawling

Iskanje in ekstrakcija podatkov iz spleta - poročilo

1. Uvod

V naslednjih nekaj poglavjih bomo na kratko opisali projekt pri predmetu Iskanje in ekstrakcija podatkov s spleta. Sam projekt se ukvarja z zajemom domen gov.si. Sam program bo v bazi shranjeval vse slike in njihove metapodatke, podatke o dokumentih na določeni strani in relacijo med povezavami na strani.

2. Delovanje

Za delovanje programa, je na začetku potrebno določiti število niti, ki bodo brskali po določenih domenah. Začetne strani so že vnaprej določene, tako lahko našim nitim določimo stran, na kateri bodo začele brskati. Naša osnovna struktura povezav je zasnovana po naslednjih principih:

a) Dodaj vse začetne povezave v globalni frontier
b) Vsak prost delavec vzame domeno iz globalnega frontierja in jo preizkuje
c) Če najde nove povezave iz iste domene jih doda v lokalni frontier, ostale pa v globalnega:
   preiskuje dokler ni lokalni frontier prazen
d) Vrni se na b, dokler ni globalni frontier prazen in vsi delavci nehajo preiskovati

Pred postopkom razbiranja strani, program najprej pogleda bazo, ali je bila določena stran že pregledana in shranjena v bazo. Sočasno pa se pregleda tudi dokument robots.txt, ki se prenaša po povezavah iste domene, ali jo je sploh mogoče pregledati. To naredimo z uporabo knjižnice ***urllib.robotparser,*** ki nam omogoča uporabo slednje metode ***can_fetch('*', url)***. Zadnji korak, ki je potreben pred iskanjem slik in povezav je iskanje duplikatov v bazi. Veliko je strani, ki so si enake vendar njihova povezave različne (www.gov.si in www.gov.si/si) . Z uporabo hash stolpca v bazi, kjer hranimo has določene strani, pregledamo bazo za možne duplikate in če le-ta obstaja pregled strani zaključimo. V primeru, da

duplikate na najdemo, shrani novo stran v bazo (preverimo še, ali je ‘content-type’ html) in začnemo z iskanjem vseh povezav in slik. Vsaka povezava je najprej preurejena v normalno obliko (primer iz /podrocja/ v https://www.gov.si/podrocja/). Nato pregledamo ali povezava vsebuje kakšno končnico (PDF, DOC, DOCX itd.) in shranimo v tabelo page_data. Če pa povezava ne vsebuje končnice pa jo shranimo v začasni list (od tu naprej poimenovam new_urls). Nato pregledamo celoten new_urls list in dodamo v lokalni frontier povezave, ki imajo enako domeno in še niso bile obiskane. Če pa povezava nima enake domene, pa jo dodamo v globalni frontier le če ta stran (site) še ni bila obiskana. Nato postopek ponavljamo toliko časa, dokler lokalni frontier ni prazen.

3. Problemi

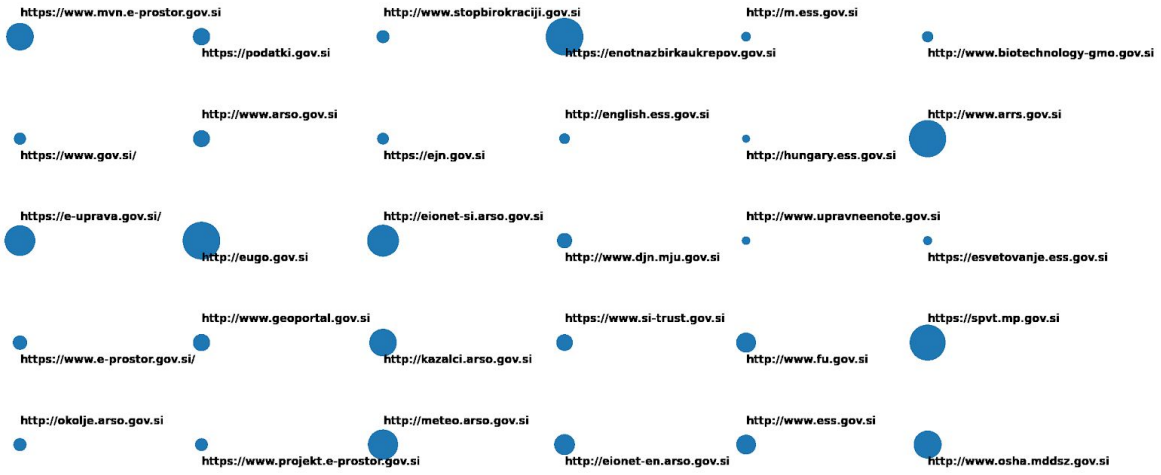
Največ problemov nama je povzročala implementacija večnitnosti, ki sva jo rešila na način, ki je opisan v prejšnjem odstavku. Problem pa sva imela tudi z uporabo Selenium-a, saj je po ekstrakciji nekaj tisoč strani začel delovati zelo počasi. Tako sva izpustila vse povezave, ki so generirane z uporabo javascripta. Sama implementacija zgoraj omenjenega problema ni težka, imava ga implementiranega v master branchu, ki ga bi bilo treba razširiti le še z iskanjem javascript linkov, vendar zaradi težav z Seleniumom sva z delom nadaljevala brez njega v branchu ‘bs4’ (https://github.com/roalj/Seminar1-Web-Crawling/tree/bs4). Ob pisanju poročila, pa sva prišla do spoznanja, da moramo v bazi hraniti vse povezave med stranmi. Z sedanjo rešitvijo v bazi hraniva le povezave med stranmi, ki so določeno povezavo našli prvi.

Problem, ki pa sva ugotovila dan pred oddajo, pa je bil povezan z samo ekstrakcijo podatkov. Saj se nama je program “zaciklal” in tako sva v bazi imela shranjenih preko 30 tisoč duplikatov ene strani. Ko sva spoznala, da nekaj ni prav, sva hitro ugotovila, da morava duplikate le shraniti v bazo in ne ponovno preiskati.

4. Rezultati

Kot je omenjeno v prejšnjem odstavku, zaradi napačnega razumevanja povezav, sva za predstavitev izbrala nekoliko drugačen način. Namesto povezav med stranmi sva napisala program, ki za določeno bazo izriše vse strani, ki imajo vsaj 50 podstrani. Seveda je ta vrednost nastavljena po naših željah in se zlahka zamenja. Na spodnji sliki vidimo le del slike

zaradi večje preglednosti. Boljšo predstavitev slik lahko najdete na najinemu githubu, ki je naveden na začetku poročila. Imena so navedena po naslednjem principu, 20.png pomeni, da so prikaze vse strani z 20 podstranmi.



Število shranjenih linkov	HTML page	DUPLICATE page	BINARY page	Čas
30300	17561	9280	3459	7 ur

5. Zaključek

Sama seminarska naloga je bila zelo zanimiva, saj dejansko predstavlja nekaj, kar bomo (naj bi) v življenju rabili. Vesela sva, da sva se lahko spoznala z tovrstnimi stvarmi in jih bolj podrobno obdelala. Opravičujeva pa se zaradi premalega števila strani, saj sva napako ugotovila šele dan pred samo oddajo poročila.