Jure Arsovič, Matic Klopčič in Karla Gliha

# SPLETNI PAJEK

Prva naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta

MENTOR: prof. dr. Marko Bajec in asistent Timotej Knez

## 1 UVOD

Prva domača naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta je predstavljala implementacijo in razumevanje strukture spletnega pajka. S pajkom smo se sprehajali po domeni gov.si in iz nje želeli pridobiti čim več različnih url-jev, ob upoštevanju etičnega vidika naloge.

## 2 RAZVOJ

Spletnega pajka smo se odločili razviti s pomočjo programskega jezika Python, ki nam z velikim številom raznovrstnih knjižnic omogoča razvoj pajka. S knjižnicama Selenium [1] in Requests [2] nam je omogočen dostop do spletnih strani s katerih lahko nato izberemo podatke, ki jih želimo hraniti. Pomemben vir informacij je datoteka robots.txt, ki je unikatna za vsako domeno. Iz nje smo pridobili navodila o crawl-delay, sitemaps ter omejitve dovoljenega brskanja (allow) in prepovedi (disallow), ki so bila temelj za nadaljnji razvoj in spoštovanje spletnih standardov.

### 2.1 IMPLEMENTACIJA

Začetna faza našega pajka je vključevala branje internetnih naslovov iz baze, označenih s tipom "FRONTIER". To pomeni, da so bili naslovi še neobdelani in je bilo potrebno postaviti temelje za njihovo nadaljnje razvrščanje v čakalno vrsto. Vrsta je bila urejena glede na časovni žig dodajanja posameznega URL naslova v bazo, kar je zagotovilo, da smo naslove obdelali po principu iskanja v širino. Sledilo je čiščenje oziroma obdelava teh naslovov, kjer smo odstranili vse nerelevantne segmente, na primer dele, ki sledijo znaku "#". S tem smo zagotovili, da so bili naslovi, ki so šli v nadaljno obdelavo, čisti in brez nepotrebnih parametrov. Nato smo s pomočjo datoteke robots.txt preverili dostopnost strani ter morebitne zakasnitve med zaporednimi zahtevami (crawl-delay). Če nam je bila obdelava dovoljena, smo nadaljevali, prvo s preverjanjem statusne kode. Če je bila statusna koda strani izven območja med 200 in 300, smo jih označili za neveljavne, "INVALID". Ta ukrep zagotavlja, da naš spletni pajek nadaljuje z obdelavo samo tistih strani, ki so uspešno dostopne in vračajo statusne kode, ki nakazujejo uspešen prenos. Strani, ki ne ustrezajo tem pogojem, so izločene iz obdelovalnega cikla, s čimer se izognemo nepotrebnemu porabljanju virov in časa na neuporabnih URL-jih. Sledi preverjanje tipa vsebine. V primerih "BINARY" je to vključevalo identifikacijo ali gre za tekstovne vrste datotek kot so PDF, DOC, DOCX, PPT, PPTX, slikovne vrste datotek (JPG, JPEG, PNG, SVG) oziroma vse ostalo "OTHER". Če pa gre za običajen HTML, preberemo in z uporabo knjižnice Selenium sprocesiramo spletno stran. Da se izognemo duplikatom v sami bazi, smo vsebino zakodirali v zgoščeno vrednost, hash, s katero tudi preverimo, ali se v bazi že nahaja duplikat njene vsebine. Če se, potem spletno stran označimo kot "DUPLICATE". Končni del našega procesa zajema uvedbo paralelizma, kjer smo za zaklep dostopa do vrste in

baze uporabili pet delovnih niti. Celotno delovanje našega spletnega pajka je trajalo približno štiri dni.

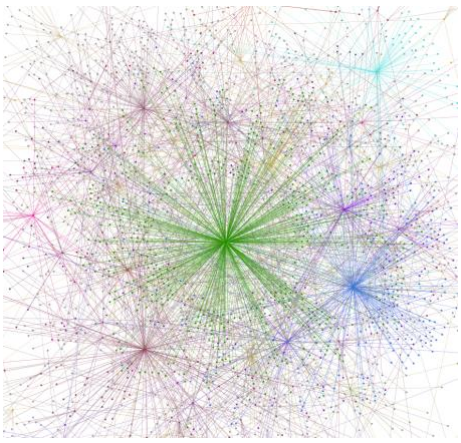
## 2.2 POSEBNOSTI KODE IN POHITRITVE

Nekaj težav so nam povzročale spletne strani, ki so zahtevale razne certifikate. To smo rešili z nastavitvijo https konteksta na ssl._create_unverified_context. Prav tako smo določili največji čas obdelave spletne strani na 10 sekund, da se izognemo čakanju strani, ki niso dosegljive.
Poleg simulacije vrste z bazo (z uporabo značke "FRONTIER" in časovnega žiga), smo med samim izvajanjem uporabljali vrsto v Pythonu. S tem smo pohitrili izvajanje, saj smo zmanjšali število dostopov do baze.
Ugotovili smo, da poleg branja html strani največ časa vzame dostopanje do spletne strani in preverjanje statusne kode. Zato smo to naredili le pri straneh, za katere smo prepričani, da jih smemo in jih tudi bomo prebrali. To je dodatno pohitrilo izvajanje.
Še ena sprememba, ki smo jo uvedli v originalno bazo, sta stolpca link_original in link_to v tabeli page. V ti dve polji smo vnesli ID strani iz katere smo prišli na določeno stran in ID-je strani, na katere smo prišli iz določene strani. Link_original je ID strani, ki nas je privedla na določeno stran, link_to pa je array ID-jev vseh linkov iz te strani. Seveda smo hkrati napolnili tudi tabelo link iz originalno baze, a ti dve polji smo implementirali kot dodatno funkcionalnost za boljši takojšen vpogled v povezave za specifično stran v tabeli page.

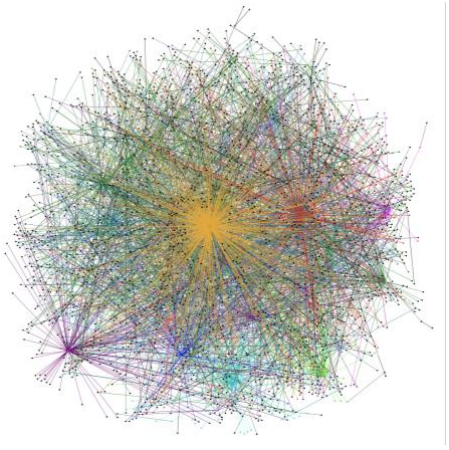
# 3 PRIDOBLJENI PODATKI

| | |
|---|---|
| FRONTIER | 59.306 |
| HTML | 52.743 |
| BINARY | 39.585 |
| DOMAINS | 257 |
| IMAGES | 6.561 |

*Tabela 1 Statistika pridobljenih podatkov*



*Slika 1 Prikaz povezav z manjšim številom spletnih strani*



*Slika 2 Prikaz povezav z večjim številom spletnih strani*

# 4 ZAKLJUČEK

Naša naloga pri razvoju spletnega pajka, ki bi raziskoval domeno gov.si, je bila uspešno zaključena. Pajek je ob koncu svojega štiridnevnega delovanja uspešno preiskal 52.743 HTML strani, identificiral 39.585 binarnih datotek in zbral podatke o 6.561 slikah, kar kaže na obsežno in raznoliko zbirko informacij. Posebno pozornost smo namenili tudi etičnim vidikom našega dela, zlasti pri spoštovanju pravil datoteke robots.txt. Vendar naše delo prav tako razkriva omejitve in izzive, povezane s spletnim pajkanjem, kot so tehnične težave s spletnimi stranmi, ki zahtevajo posebne certifikate, in potrebo po izboljšanju učinkovitosti obdelave.

# 5 LITERATURA


1. „Selenium,“ [Elektronski]. Available: https://selenium python.readthedocs.io/. [Poskus dostopa 31 Marec 2024]
2. „Requests,“ [Elektronski]. Available: https://docs.python-requests.org/en/latest/. [Poskus dostopa 31 Marec 2024].