Žiga Kleine, Tim Križnik in Luka Železnik

Spletni pajek

Prva seminarska naloga, Iskanje in ekstrakcija podatkov s spleta, marec 2020

Mentor: doc. dr. Slavko Žitnik, prof. dr. Marko Bajec

V tem poročilu je predstavljeno delo opravljeno pri prvi seminarski nalogi predmeta Iskanje in ekstrakcija podatkov s spleta in izdelava spletnega pajka, ki avtomatizirano poizveduje po spletnih straneh z domenami *gov.si*. V nadaljevanju bo opisana strategija, tehnična in programska izvedba, statistična analiza, nepričakovani problemi in rešitve pri implementaciji takšnega spletnega pajka.

1. UVOD

Spletni pajki so programi, ki izkoriščajo strukturo spleta na način, da iz spletne strani izluščijo nadaljnje hiperpovezave spletnih strani, ki bodo obiskane. So osnova sodobnih spletnih iskalnikov, dobro orodje v poslovni inteligenci in nudijo pospešitev pri analizi spleta. Spletni pajki s spletnih strani zberejo HTML dokumente (ang. hyper text markup language), slike, besedilo, itd. [1] Ena izmed osnovnih delitev pajkov je na: fokusirane pajke, inkrementalne pajke, skrite pajke in porazdeljene pajke [2]. Osnovne naloge, ki jih izvajajo, so pridobivanje strani (ang. fetching), razčlenjevanje (ang. parsing), shranjevanje v datotečni sistem (ang. repository) in sočasnost procesiranja.

Na rezultat spletnega pajka lahko vpliva več dejavnikov, ki jih je potrebno vključiti v načrtovanje in izvedbo; strategija, dinamičnost spleta, razširljivost, podvajanje, etika, itd. Vse te dejavnike smo v okviru te naloge skušali čim bolje raziskati in jim sproti prilagajati program z namenom, da izboljšamo robustnost, kakovost in varnost pajka.

2. IMPLEMENTACIJA

Pajek je programiran v programskem jeziku Python 3. Za zagon programa se izvede skripta main.py, večji del logike pa se nahaja v razredu Crawler v datoteki cralwer.py. (Github repozitorij: https://github.com/LukaZeleznik/wieramemo_vase)

Datotečni sistem

Za shranjevanje podatkov v bazi spletnega pajka (ang. repository) je bila uporabljena PostgreSQL baza, po možnosti v kombinaciji z vmesnikom pgAdmin 4. Za kreiranje sheme je bila uporabljena priložena SQL skripta z dodanim poljem v tabeli *page* za zgoščeno vsebino strani. Niz v polju *page_type_code* v tabeli *page* opisuje spletno stran kot: HTML – prebrana spletna vsebina, DUPLICATE – podvojeno, neshranjeno, FRONTIER – v čakalni vrsti, BINARY – vsebina tipa .doc,.pdf,.ppt. Za operacije vnosa, branja, posodabljanja in brisanja iz baze so na voljo pomožne funkcije z deli SQL poizvedb.

Čakalna vrsta

Čakalna vrsta (ang. Frontier) je izvedena po principu iskanja po širini (ang. breadth-first strategy) pri katerem se novo odkrite povezave, dodajajo na konec čakalne vrste. Tako se najprej obiščejo vse spletne strani, ki so na enakem nivoju. Za vrsto našega pajka je uporabljena tabela *page* v bazi, ki je inicializirana s štirimi URL povezavami (ang. seed URLs) in sicer: *'http://gov.si'*, *'http://evem.gov.si'*,

'http://e-uprava.gov.si' in 'http://e-prostor.gov.si'. Funkcija *insert_seed_urls_into_db()* poskrbi, da se izvedejo prvi štirje zapisi v podatkovno bazo v tabelo *site* in *page*, kjer zapisi dobijo atribut *PAGE_TYPE_CODES* nastavljen na niz *»FRONTIER«*. Pajek mora nato pridobiti URL povezavo, ki bo ustrezni kandidat, da jo obišče, za kar poskrbi funkcija *get_page_to_crawl()*. Ko pajek pridobi URL povezave z obiskane strani se le te dodajo na konec baze s kodo »FRONTIER«, koda trenutne strani pa se posodobi.

Pridobivanje spletnih strani

Za pridobivanje ali fetching se uporablja knjižnica Seleniumwire s spletnim gonilnikom brskalnika Chrome, ki v glavi HTTP odgovora vrne vrsto vsebine, kot tudi samo HTML vsebino, ki jo očisti knjižnica BeautifulSoup. Z omenjeno knjižnico izluščimo še elemente tipa <a>, elemente s funkcijo *onclick* in <img>. Pridobivanje relevantnih elementov (relativne URL povezave, vire slik) izvaja funkcija *gather_links().*

Zaznava duplikatov

Za preprečevanje podvojenih URL naslovov pred vpisom v frontier smo uporabili knjižnico urlcanon (dostopno na: https://github.com/iipc/urlcanon, 30.3.2021) s katero smo najdene naslove normalizirali – lahko se namreč zgodi, da nekateri parametri v url naslovu vodijo na isto spletišče. Dodatno smo razširili *page* tabelo v bazi in ji dodali polje *hash_content*, ki je zgoščena verzija HTML vsebine, zračunana s *hashlib* funkcijo *sha256*. Če v bazi že obstaja stran z enakim podpisom se ta stran označi za podvojeno.

Etika

Spletni pajek na vsakemu spletišču preveri, če obstaja datoteka */robots.txt*, ki podaja dovoljenja za plazenje ne tem spletišču in jo shrani v bazo. Pred vsakim zapisom URL povezave v frontier se preveri, če je lahko le ta sploh dodana, in sicer s funkcijo *.can_fetch()* razreda *RobotFileParser* v knjižnici *urllib.*

Varovalo, da pajek ne izvaja napad za zavrnitev storitve (ang. DOS attack) smo realizirali s slovarjem, ki si za vsako domeno shranjuje čas zadnjega dostopa in v primeru, da ni minilo vsaj 5s (vhodni parameter pajka) ne obišče strani na tej domeni.

Večnitnost

Ob zagonu programa se ustvari vneseno število objektov *Crawler*, ki jim je podan razred *Lock()* knjižnice *threading*. S tem dosežemo paralelno izvajanje več pajkov hkrati, pri čemer moramo biti pozorni na dostopanje skupnih virov – lahko z blokiranjem ključavnic ali z reševanjem konfliktov v bazi. Odločili smo se za slednje.

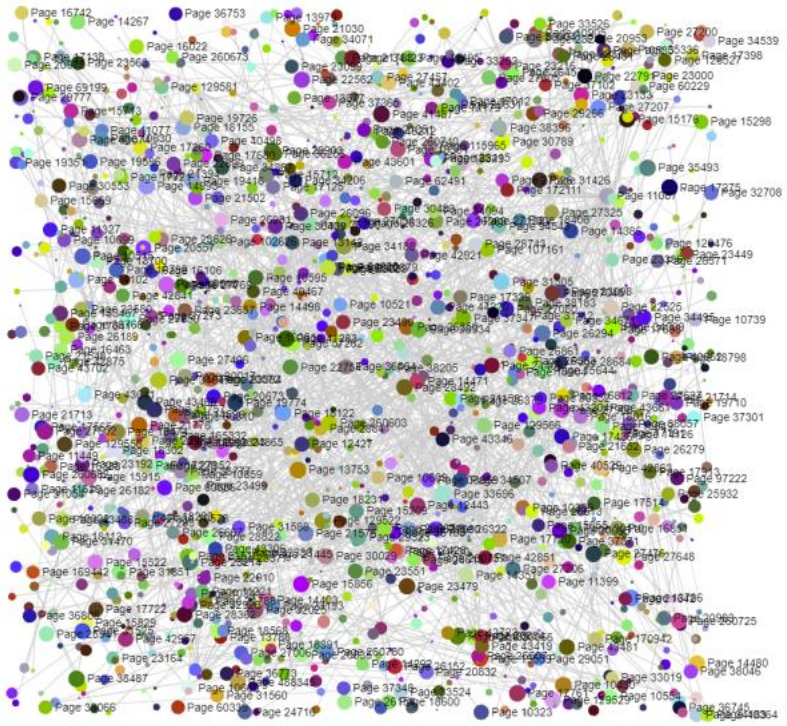
Statistika

Pajek je deloval približno efekt. 30 ur. Okvirna statistika podanih strani in celotne zgodovine v tem času znaša:

www.gov.si:	**evem.gov.si:**	**e-uprava.gov.si:**	**e-prostor.gov.si:**	**CELOTNA ZGODOVINA**
7617 pages 6434 frontier **885 HTML** 257 duplicates 41 binary 1797 images 2,03 images/page 2 DOCX 0 DOC 56 PDF 0 PPTX 0 PPT	**298 pages** 221 frontier **25 HTML** 43 duplicates 9 binary 128 images 5,12 images/page 0 DOCX 0 DOC 10 PDF 0 PPTX 0 PPT	**35251 pages** 31336 frontier **3830 HTML** 23 duplicates 32 binary 4580 images 1,20 images/page 0 DOCX 0 DOC 36 PDF 0 PPTX 0 PPT	**809 pages** 340 frontier **211 HTML** 201 duplicates 57 binary 182 images 0,86 images/page 1 DOCX 8 DOC 40 PDF 0 PPTX 0 PPT	**Number of sites:** 222 **Number of pages**: 61321 51094 frontier **9029 HTML** 805 DUPLICATE 393 BINARY 34768 images 3.85 images/page 35 DOCX 26 DOC 407 PDF 0 PPT 0 PPTX

Vizualizacija

Vizualizacija spletnih strani njihovih povezav in pripadnost domeni.



3. Težave in zaključek

S trudom smo poskušali spletnega pajka narediti čim bolj robustnega, pri tem smo ugotavljali kako zares veliko strani je med sabo povezanih in kako so hkrati lahko zelo nestandardizirane. Ocenjujemo, da smo proces ekstrakcije pričeli nekoliko prepozno, saj bi morali našega pajka še pohitriti. Med izvajanjem programa smo našli še kakšno programsko napako, ki je povzročila napačno vnašanje strani v bazo ali napačno delovanje zaklepanja niti. Vmes smo pomotoma tudi izbrisali podatkovno bazo.

Problem je bila še visoka poraba procesne moči – okvirno poraba 9 GB RAM pomnilnika, zaradi česar je prišlo tudi do blokad. Splačalo bi se iskati še rešitve za pohitritev kode na nekaterih mestih, predvsem iskanju in preverjanju dokumentov v bazi. Ampak kljub vsemu, smo dobili boljši vpogled v avtomatizirano ekstrakcijo dokumentov s spleta, predvsem na omejitve takšnih programov, odpravljanje njihovih napak, paralelno izvajanje, neenotnost spleta, etično poizvedovanje, itd.

LITERATURA

[1] C. Saini and V. Arora, "Information retrieval in web crawling: A survey," *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, 2016, pp. 2635-2643, doi: 10.1109/ICACCI.2016.7732456.

[2] Chaitra P.G., Deepthi V., Vidyashree K.P., Rajini S. (2020) A Study on Different Types of Web Crawlers. In: Choudhury S., Mishra R., Mishra R., Kumar A. (eds) Intelligent Communication, Control and Devices. Advances in Intelligent Systems and Computing, vol 989. Springer, Singapore. https://doi.org/10.1007/978-981-13-8618-3_80