Spletni Pajek - 1. IEPS poročilo

Klobučar Rok, Mihelič Mohor Luka, Ostovršnik Anja

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Abstract. Spletni pajek je namenjen učinkovitem in fleksibilnem premikanju po spletnih mestih. V tem poročilu bomo opisali, kako smo razvili pajek, ki se sprehaja po domenah *si.gov*. Da spletnemu pajku omogočimo lažje in hitrejše sprehajanje čez celotni nabor spletnih mest in njihovih hiperpovezav, moramo implementirati mehanizem preprečevanja duplikatov, večnitnega izvajanja in uporabe Selenium gonilnika za brskalnike. Potrebno je ustvariti in vzpostaviti povezavo s podatkovno bazo, ki bo hranila metapodatke obiskanih spletnih strani in vsebin ter nosila seznam že obiskanih spletnih mest [3]. Rezultate sprehajanja pajka smo tudi prikazali na grafu s pomočjo knjižnice za vizualizacijo struktur omrežnih povezav.

Keywords: spletni pajek · ekstrakcija podatkov · brskalnik · svetovni splet

1 Uvod

V poročilu bomo predstavili našo idejo in postopke implementacije pri ustvarjanju lastnega spletnega pajka. Seznanili smo se z že obstoječimi projekti ter njihovo uporabo, ter ubrali standardni pristop k ustvarjanju PostgreSQL podatkovne baze znotraj Docker zabojnika. V naslednjih poglavjih in podpoglavjih bomo opisali proces oblikovanja in izdelave spletnega pajka.

Če si želimo implementirati Python skripto, ki se je zmožna sprehajati po spletu in pošiljati zahtevke preko protokola HTTP, si to lahko zagotovimo z uporabo knjižnice `requests`. Prav tako si želimo shranjevati povezave obiskanih spletnih strani in dostopane dokumente, da zmanjšamo čas izvajanja ter upoštevamo spletne strani s podobno poddomeno ter izognemo zančnem ponavljanju obiskovanja spletnih strani. Zato je potrebno pajka povezati s podatkovno bazo, kamor bo shranjeval prehojena spletna mesta. V ta namen bomo uporabili knjižnico `psycopg2`, s katero lahko izvajamo ukaze nad podatkovno bazo v PostgreSQL, ustvarjeno po shemi na spletni učilnici. Da poskrbimo za pravilno obdelavo spletnih mestih z dinamičnimi spletnimi vsebinami, ki se generirajo kot posledica asinhronih JavaScript klicev ali pogojne logike znotraj dokumentov DOM, uporabimo Selenium gonilnik za brskalnik Chrome. Ko s pajkom obiščemo spletno stran, moramo tudi upoštevati datoteko *robots.txt*, kjer so navedena navodila o katerem delu spletne strani smemo dostopati.

2 Implementacija

V glavni datoteki *crawler.py* so deklarirani razredi `Frontier`, `Page` in `Site`, s katerimi si shranjujemo dostopane spletne strani. V razredu Frontier vzpostavimo čakalno vrsto za shranjevanje spletnih objektov, ustvarimo slovar za ločevanje strani po domenah in števec za obiske spletnih strani ter nastavimo privzeti čas za čakanje.

2.1 Frontier

Za hitrejše in bolj organizirano izvedbo sprehajanja pajkov smo uporabili prej omenjen *frontier*, ki je shranjeval URL dobljenih strani na dani domeni, ter nenehno preiskuje strani za *pages*. To počne, dokler jih mu ne zmanjka. Tako so pajki najprej preiskali stran in vse povezave shranili v frontier. Vsak naslednji pajek se je sprehodil čez strani, ki so v frontier-ju. Ko je bila stran shranjena v podatkovno bazo, smo jo odstranili iz frontier-ja.

2.2 Duplikati

Ustavrili smo funkcijo za preverbo duplikatov. Ta je preverila elemente v že obstoječi podatkovni bazi. Ustvari se nabor, kjer vsaka vrstica predstavlja ujemajočo povezavo. V primeru, da je nabor daljši od ena, oziroma da povezava že obstaja v podatkovni bazi, jo zazna kot duplikat in je ne vpiše v podatkovno bazo. V nasprotnem primeru preveri tudi zgoščeno vrednost povezav. To stori z uporabo MD5 hash algoritma [1]. V kolikor se povezava tudi tu ne zazna kot duplikat, jo pajek doda v podatkovno bazo.

2.3 Niti in sočasno sprehajanje

En sam pajek se izjemno počasi sprehaja skozi strani. Cilj je, da se sprehodi skozi 50.000 strani. Proces sprehajanja skozi tako veliko število strani je dolgotrajno za enega pajka, zato smo se poslužili uporabe niti [2]. Te so nam omogočile sočasno delovanja večih pajkov. Pajki ne delujejo vzporedno, oziroma ne delujejo vsi hkrati. Razdelijo si časovni cikel tako, da ima vsak pajek en izsek cikla, ki ga porabi za sprehod skozi stran. Vsi skupaj si delijo isto podatkovno bazo, kjer shranjujejo podatke, vendar si ne smejo deliti istih strani. Dva različna pajka se ne smeta sprehajati po isti strani ter si deliti virov. To je bil razlog, da smo uvedli ključavnico, ki to preprečuje. Z uporabo nitenja (ang. threading) smo dosegli hitrejše sprehajanje. Posledično smo potrebovali bistveno manj časa za doseg cilja.

2.4 Obdelava

2.5 Težave

Problem se je pojavil tudi v nestabilni internetni povezavi. Za poganjanje spletnega pajka je potrebna stabilna internetna povezava, ki ni bila vedno na voljo.

To je rezultiralo v zgolj 6480 prehojenih strani. Pogost hrošč v začetnih različicah kode je bil, da se pajek ni sprehodil skozi stran, temveč jo je zgolj vrnil v frontier. Težave smo imeli tudi z dostopom do spletne strani, saj začetna stran evem.gov.si ni delovala. Odločili smo se, da fiksiramo zgolj domeno gov.si in obiskujemo strani v tej domeni.

3 Rezultati

Pajka smo poganjali 15 ur. V tem času se je sprehodil skozi 6480 strani. Od tega je zaznal 539 duplikatov. Cilj je bil doseči 50.000 strani. Kratek časovni okvir in nestabilna internetna povezava sta nam žal to onemogočila. V razpredelnici smo predstavili rezultate poizvedb spletnega pajka.

Table 1. Rezultati poizvedb.

	Site	Page	Binary	Duplicates	Image	Images per Webpage	Files
gov.si		159	3	74	392	2.47	5
spot.gov.si		271	1	0	1104	4.07	0
e-uprava.gov.si		330	3	83	334	0.99	0
e-prostor.gov.si		53	2	13	551	10.4	4
vse	187	6480	793	539	149952	23.14	460

S pomočjo knjižnice *networkx* smo izrisali graf delovanja pajka, ki ga lahko vidimo na Figure 1. Ponovni obiski istih strani se prikažejo v obliki kroga. To se zgodi dvakrat, kar ni prepogosto.

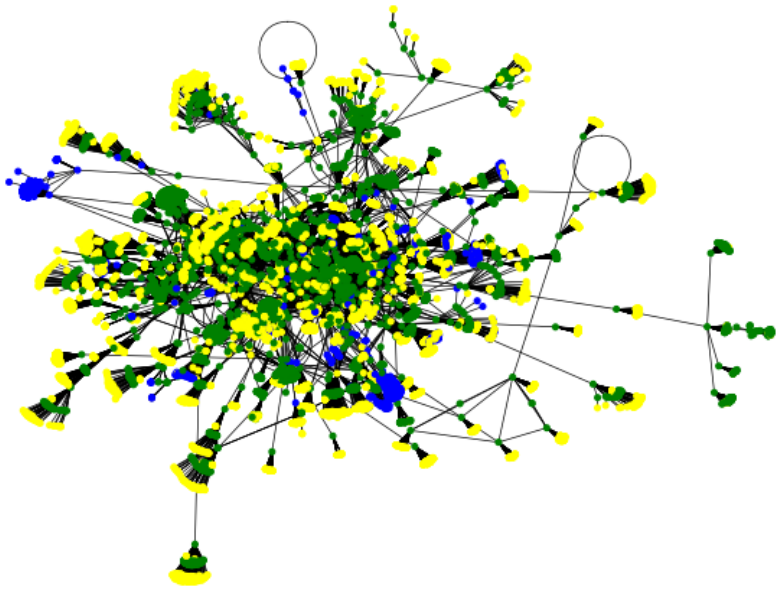

Fig. 1. Graf delovanja pajka

4 Literatura

References

1. hashlib — Secure hashes and message digests — Python 3.12.2 documentation, https://docs.python.org/3/library/hashlib.html, (obiskano 5.04.2024)
2. dr. Tadej Tuma, Programming embedded systems, Univerza v Ljubljani, Februar, 2023
3. C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge UP, 2008
4. Ricardo Baeza-Yates, Berthier Ribeiro-Neto Modern Information Retrieval: The Concepts and Technology behind Search, 2nd Edition, ACM Press Books, 2010
5. Bing Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Data-Centric Systems and Applications, 2nd edition, Springer, 2013