Poročilo prvega projekta pri IEPS

Jan Novak Domijan, Urša Kumelj, Manca Drašček
UL FRI

POVZETEK

V tem poročilu predstavljamo implementacijo spletnega pajka, katerega glavni cilj je bil zajeti podatke iz spletne strani https://www.kulinarika.net/recepti/sladice. Spletna stran vsebuje široko paleto receptov za sladice, skupaj s pripadajočimi slikami. Cilj projekta je bil ekstrahirati najmanj 5000 url povezav iz te strani, nato pa napolniti podatkovno bazo z vsebino posameznih strani (tj. vsebina receptov) in slikami, ki so na njih objavljene.

Pajek je bil razvit v programskem jeziku *Python* z uporabo knjižnic, kot so `BeautifulSoup`, `Selenium` in `threading`. Poleg zajema podatkov smo se osredotočili tudi na reševanje težav, kot so deduplikacija strani, prioritetno iskanje po ključnih besedah, kanonizirane url povezave in zajem dinamično naloženih slik. V poročilu bomo podrobneje opisali metodologijo, rezultate in izzive, s katerimi smo se srečali med razvojem.

1 UVOD

Spletno pajkanje je proces avtomatiziranega zajemanja podatkov s spletnih strani in se pogosto uporablja v različnih aplikacijah.

V tem projektu smo razvili spletnega pajka, ki je specializiran za zajemanje receptov in pripadajočih slik. Zajeti podatki so lahko uporabni za analizo trendov pri peki sladic, avtomatsko kategorizacijo receptov ali izboljšanje priporočilnih sistemov.

V nadaljevanju bomo podrobneje opisali metodologijo razvoja pajka, izzive pri implementaciji in optimizaciji ter predstavili rezultate zajema podatkov. Pri tem smo uporabili tehnike, kot so preferenčno pajkanje na podlagi ključnih besed, deduplikacija url-jev in zajemanje dinamično naloženih slik, kar omogoča bolj ciljno usmerjeno in učinkovito pridobivanje podatkov.

2 METODOLOGIJA

2.1 Implementacija pajka

V tej implementaciji spletnega pajka je uporabljen t.i. *preferential crawling*, kar pomeni, da pajek prednostno sledi povezavam, ki vsebujejo določeno ključno besedo ali izpolnjujejo druge kriterije. Pajek začne na vnaprej določeni začetni strani (angl. seed), kjer analizira vse povezave in jih razvrsti glede na njihovo relevantnost. V našem primeru vsi url-ji vsebujejo ključno besedo 'sladice', zato pajek prednostno obravnava te povezave, kar je smiselno glede na cilje iskanja. S tem zagotovimo, da pajek ne bo naletel na nepovezane strani, kot so tiste, ki vsebujejo recepte za npr. mesne kroglice ali puranji zrezki v smetanovi omaki in podobno. Poleg tega pa pajek neprioritizira povezav s ključnimi besedami 'forumi', 'tema', 'fotoalbumi', 'iskanje', 'besede', saj bi z njimi zajeli strani, ki ne vsebujejo receptov o sladicah. Za obdelavo povezav uporablja čakalno vrsto s prednostjo (angl. priority queue), kar pomeni, da najprej obišče strani, ki so bolj relevantne, in šele nato tiste, ki so manj pomembne. Pri vsakem obisku spletne strani pajek normalizira url, preveri, ali je bila stran že obiskana, in poišče kanonični url, če je na voljo.

Normalizacija url-jev pomeni preoblikovanje url-ja v standardizirano obliko, da se preprečijo podvojeni vnosi in zagotovi, da pajek ne obišče iste strani večkrat pod različnimi naslovi. V tej implementaciji se normalizacija izvaja na več načinov: odstranjevanje zaključnega poševnika /, saj strani *primer.si/stran/* in *primer.si/stran* predstavljata isto vsebino; pretvorba domene v male črke, da se *Primer.si/Stran* in *primer.si/stran* ne obravnavata kot različni strani; ter urejanje parametrov v url-ju, kjer se t.i. query string (*?param1=value1&param2=value2*) razdeli, razvrsti in ponovno sestavi v urejenem vrstnem redu, s čimer se preprečijo podvojitve zaradi različnega vrstnega reda parametrov.

Kanonični url (angl. canonical url) je uradna različica strani, ki jo določi lastnik spletnega mesta s pomočjo oznake *<link rel='canonical' href='url'>*. Pajek preveri, ali ta oznaka obstaja v *<head>* strani, in jo uporabi, če je na voljo, s čimer se prepreči indeksiranje podvojenih strani, kot sta *primer.si/stran?id=123* in *primer.si/stran*. Če pajek najde kanonični url, preveri še, ali gre za relativno ali absolutno pot. Če se začne s /, ga pretvori v absolutni url, pri čemer ohrani glavno domeno. Na koncu se tudi kanonični url normalizira, da se odpravi morebitne variacije. Če kanonični url ni določen, se uporabi originalni url strani.

Poleg besedilnih podatkov pajek s pomočjo knjižnice `Selenium` zajema tudi slike receptov, saj ta omogoča pridobivanje dinamično naloženih vsebin, ki jih klasično zajemanje HTML kode ne bi zaznalo. Ko pajek uspešno prenese vsebino strani, jo shrani v bazo podatkov. Za shranjevanje pridobljenih podatkov v bazo podatkov uporabljamo knjižnico `psycopg2`, ki omogoča enostavno komunikacijo s PostgreSQL bazo.

Za hitrejšo obdelavo podatkov uporabimo več niti naenkrat, saj nam to omogoča hkratno branje strani, shranjevanje v bazo in analizo povezav. Čeprav pajek zajema le eno domeno, ta pristop zmanjšuje čakanje in optimizira zajem relevantnih vsebin. Za večnitno izvedbo je ključno, da se pisanje v prednostno vrsto in branje iz prednostne vrste izvedeta atomarno, kar smo zagotovili z razredom `PriorityQueue` iz knjižnice `queue`. Niti med seboj koordinirajo tudi čas med zahtevami z uporabo ključavnice in spanjem niti.

2.2 Strategija za deduplikacijo

Najprej smo se odločili za iskanje duplikatov z algoritmom MinHash. Ta nam omogoča izračun približka Jacardove podobnosti med stranmi in hitrejšo analizo podobnosti strani.

Naša implementacija MinHash algoritma naredi naslednje:

(1) izlušči vsebino strani s knjižnico BeautifulSoup,
(2) zbere v seznam vsa podzaporedja besed velikosti k, kjer je k vnaprej podan parameter,
(3) vsako podzaporedje pretvori v celo število z algoritmom CRC-32,
(4) za m vnaprej izbranih zgoščevalnih funkcij izračuna njihovo minimalno vrednost na množici podzaporedij,

(5) in minimume funkcij združi v eno samo vrednost, ki jo potem uporabimo za izračun podobnosti.

Od teh korak (3) ni potreben za delovanje algoritma in naredi cenilko podobnosti pristransko, vendar nam omogoča, da od zgoraj omejimo velikost končne vrednosti na $32m$.

Sledilo je testiranje algoritma na naši spletni strani. Pognali smo spletni pajek na omejenem številu strani in opazili, da podobnost med stranmi ni presegala 0.6. Vendar je bila bolj pomembna opazka, da strani, ki jih mi smatramo za duplikate, algoritem ni zaznal. Razlog zato je, da strani v domeni *kulinarika.net* ne vsebujejo veliko vsebine in velik del te vsebine je dinamično generirane vsakič, ko obiščemo stran.

Glede na rezultate testiranja smo se odločili, da duplikatov ne bomo iskali.

2.3 Zajemanje slik

Za ekstrakcijo slik uporabljamo knjižnico `Selenium`, saj omogoča dinamično pridobivanje vsebine, ki jo nalaga `JavaScript`. Zajemamo izključno slike receptov, saj so druge slike za naš namen nepomembne. Postopek deluje tako, da najprej na spletni strani poiščemo element * *, nato pa znotraj njega pridobimo vse značke *<img>*. Če posamezna značko vsebuje atribut data-default, iz njega pridobimo url slike. V nasprotnem primeru uporabimo atribut data-srcset, ki vsebuje več povezav do slike v različnih velikostih. Iz tega se nato izbere povezava, ki vsebuje končnico *.webp* in ne vsebuje oznak za različne velikosti (ta je vedno zadnja v seznamu). Te povezave so oblike *https://www.kulinarika.net/slikerecepti/1773/2.webp*. Za avtomatizirano pridobivanje teh podatkov uporabimo `Selenium WebDriver`. Ker želimo postopek izvajati v ozadju brez prikaza brskalnika, inicializiramo WebDriver s parametrom `–headless`. Po koncu postopka driver vedno zapremo z *driver.quit()*, da sprostimo sistemske vire.

3 ODLOČITVE

Pri implementaciji spletnega pajka smo sprejeli nekaj ključnih odločitev, s katerimi smo optimizirali njegovo delovanje:

- Ker nas zanimajo izključno recepti in pripadajoče slike, smo se osredotočili le na slike, ki so del opisa posameznega recepta, ter izločili vse druge slike na spletni strani.
- Strani z recepti vključujejo tudi komentarje uporabnikov. Ker se lahko pri priljubljenih receptih nabere veliko komentarjev, so ti razdeljeni na več podstrani. Na primer, recept na naslovu https://www.kulinarika.net/recepti/sladice/pecivo/cokoladno-mascarponejeve-kocke/12619/ ima tretjo stran komentarjev na naslovu https://www.kulinarika.net/recepti/sladice/pecivo/cokoladno-mascarponejeve-kocke/12619/?offset=20#mnenja. Obe povezavi vodita do iste vsebine recepta in imata enak kanonični URL. Ker komentarji v tej implementaciji niso relevantni, smo upoštevali le glavno stran recepta, kjer zajamemo le prvo stran s komentarji. Podoben sklep smo uporabili za ostale ključne besede za neprednostno iskanje. Te so namreč vračale strani, nepovezane z recepti.
- Datoteka *robots.txt* ne vsebuje posebnih omejitev, zato smo sami določili pravilo, da med posameznimi zahtevami upoštevamo 5 sekundni premor.
- Pred zagonom pajka je potrebno določiti nekaj parametrov, ki so shranjeni v datoteki *config.ini* za lažje urejanje. Mednje sodijo:
  - ključne besede,
  - neprioritizirane ključne besede,
  - timeout,
  - maksimalno število preiskanih strani,
  - brskalnik, v katerem `Selenium` obiskuje naše strani,
  - število niti,
  - nastavitve za povezavo s podatkovno bazo in
  - nastavitve za računanje hash funkcije (velikost podzaporedij besed k, število zgoščevalnih funkcij m).

4 REZULTATI

Pajek smo pognali za preiskovanje maksimalno 8000 url strani, z uporabo štirih niti. Po ekstrakciji smo v bazi zabeležili naslednje rezultate, ki so napisani v angleščini:

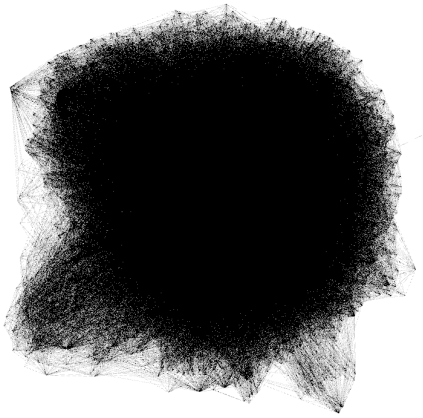
- number of sites: 1
- number of web pages: 8000
- number of duplicates: 0
- number of binary documents by type: 0
- number of images: 18087
- average number of images per web page: 2.3
- crawling time for 8000 pages: 74282s
- average crawling time per web page: 9.3s

Skozi celotno ekstrakcijo smo naleteli le na dva duplikata in sicer: https://www.kulinarika.net/recepti/seznam/sladice/, https://www.kulinarika.net/recepti/seznam/sladice/?offset=0 in https://www.kulinarika.net/recepti/seznam/sladice/pecivo https://www.kulinarika.net/recepti/seznam/sladice/pecivo?offset=0. Zaradi naključno generiranega stranskega menija imenovanega *Kaj se danes kuha* sta imeli strani različen hash in ju zato nismo zaznali kot duplikata. Za prvi primer namreč vrednost Jaccard podobnosti znaša 0.84, za drugo pa 0.82. V primeru, da bi bili strani prava duplikata bi Jaccardova podobnost znašala 1.

Prav tako na spletni strani ni bilo nobenih binarnih dokumentov.

5 VIZUALIZACIJA

Url povezave smo vizulaizirali s pomočjo programa `Gephi`.



Slika 1: Vizualizacija povezav