Špela Čoha, Žan Luka Nusdorfer, Neja Veternik

Implementacija spletnega pajka

Projektna naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta

Cilj prve projektne naloge je bil razvoj spletnega pajka, ki bo sposoben raziskati spletne strani znotraj izbrane domene, izvleči relevantne podatke (kot so slike, povezave, dokumenti) in te podatke shraniti v vnaprej strukturirano bazo podatkov. Pri implementaciji smo upoštevali določene pristope in pravila, kot je za primer spoštovanje pravil, določenih v datoteki robots.txt in implementacija strategije preprečevanja podvajanja, da se izognemo ponovnemu raziskovanju istih strani. Prav tako je bila implementirana strategija prednostnega raziskovanja, ki omogoča prioritetno obravnavo bolj relevantnih povezav. V tem poročilu so opisane nekatere odločitve, sprejete med razvojem pajka, določeni izzivi in rešitve ter povzetek rezultatov končnega preiskovanja.

1. Implementacija spletnega pajka

Za implementacijo spletnega pajka smo se odločili za programski jezik Python in PostgreSQL podatkovno bazo. Slednjo smo zaradi enostavnosti gostovali na platformi Aiven, ki nam je omogočila spletno gostovanje in upravljanje baze. Kot vmesnik pri komunikaciji z bazo smo uporabili ORM SQLAlchemy, ki je poenostavil vse operacije s podatkovno bazo. Za prenašanje spletnih strani, ki so dinamično generirane z JavaScriptom, pa smo uporabili knjižnico Selenium. Ta omogoča zajemanje celotne vsebine, ki je sicer ne bi bilo mogoče pridobiti z osnovnimi HTTP zahtevki. Za ekstrakcijo podatkov s spletnih strani smo uporabili knjižnico za razčlenjevanje HTML kode BeautifulSoup. Ta je omogočila enostavno navigacijo HTML strukture in pridobivanje vseh pomembnih podatkov, kot so povezave (<a href>) in slike (<img src>).

1.1 Implementirana strategija iskanja duplikatov

Za detekcijo podvojenih strani, smo implementirali strategijo preverjanja podvajanj, ki temelji na izračunu SHA-256 hash funkcije celotne HTML vsebine strani. Pred izračunom hasha HTML smo vsebino očistili, pri čemer smo odstranili vse <script>, <style> in <meta> elemente. Tako očiščeno vsebino za bolj zanesljivo primerjavo še formatirali. Za potrebe preverjanja, smo tabelo page v bazi razširili s stolpcem page_hash. S primerjavo že obstoječih hashov v bazi, smo na enostaven način preverjali obstoj strani.

1.2 Implementirana strategija prednostnega pajkanja

Za preferenčno pajkanje strani smo implementirali strategijo, s katero se ocenjuje pomembnost povezav glede na domene in ključne besede v URL-ju. Za vsak URL se preveri, ali ta pripada določeni domeni (v našem primeru je to slo-tech.com), kar poviša relevantnost povezave za eno točko. Dodatno se dodeli višja relevantnost, če URL vsebuje specifične ključne besede, kot so "*novice*", "*forum*" ali "*članki*", saj te besede nakazujejo, da je stran lahko bolj informativna. Če je bila povezava najdena v zemljevidu strani (sitemap), se ji avtomatsko dodeli najvišja pomembnost, saj domnevamo, da je spletna stran sama označila to povezavo kot pomembno.

Za shranjevanje frontierja smo uporabili tabelo page, kjer smo polju page_type_code nastavili vrednost FRONTIER. Dodatno smo tabeli dodali polje relevance, v katerega shranjujemo numerično vrednost pomembnosti posameznega URL-ja. Pri izbiri strani za nadaljnje pajkanje smo vedno izbrali tisto, ki je imela najvišjo vrednost relevance. Takoj ko smo stran začeli

obdelovati, smo njeno polje page_type_code posodobili na CRAWLING, s čimer smo preprečili, da bi bila istočasno izbrana za obdelavo še v kakšnem drugem procesu.

2. Parametri in nastavitve iskalnika

Med glavnimi parametri, ki jih je mogoče nastaviti, so začetni URL (seed URL), ki je bil v našem primeru https://slo-tech.com/, število delavcev (worker), ki lahko delujejo sočasno, ter največje število spletnih strani, ki jih iskalnik naj obišče (privzeto 25000). Poleg tega se upošteva tudi nastavitev zamika pri iskanju, ki je pridobljena iz datoteke robots.txt posameznih spletnih mest. Če ta ni na voljo, je privzeta zakasnitev 5 sekund. Nastavitev uporabniškega agenta je bila prilagojena na fri-wier-skupina_n, da je bila zagotovljena skladnost z zahtevami.

3. Izzivi implementacije

Med razvojem se je pojavilo več izzivov, ki so zahtevali naprednejše iskanje rešitev. Eden izmed izzivov je bila za primer zaznava povezav, ki so druge vsebine kot HTML. Torej pri povezavah s content-type parametrom, drugačnim od text/html. Posebno težavo nam je predstavljajo tudi to, da je bila vsebina s končnico .pdf večkrat ozačena s content-type=text/html in s tem zaznana kot običajna HTML stran, namesto pravilne vrednosti application/pdf. Tako smo za takšne primere uvedli posebno preverjanje končnice, da so lahko bile te povezave pravilno označene kot tip BINARY.

4. Statistika pridobljenih rezultatov v bazi podatkov (Crawldb)

Po končani raziskavi izbrane spletne domene, smo s pomočjo poizvedb zbrali naslednje statistike iz podatkovne baze podatkov:

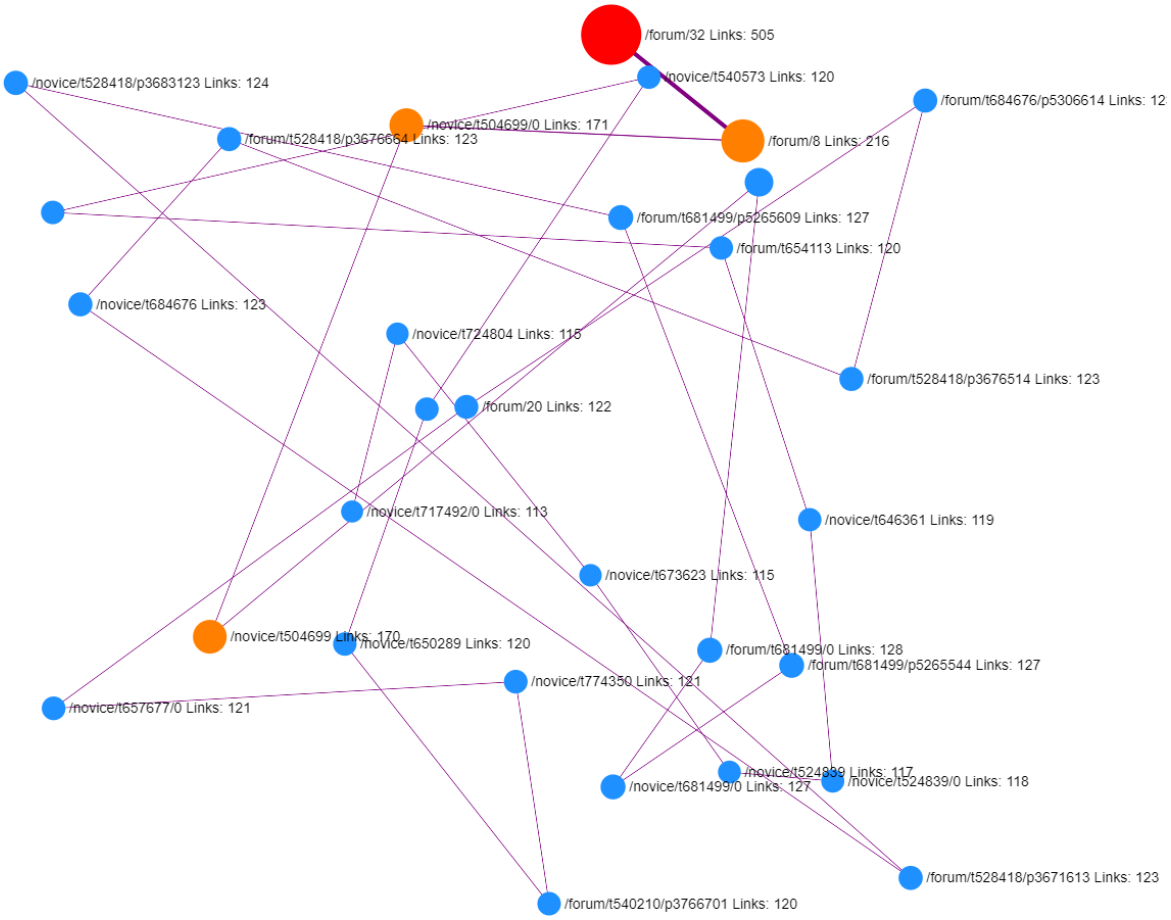
Vrsta zajete strani	**Število**
HTML	6119
Frontier	61602
Binary	17
Skupaj	**67759**

Tabela 1: Statistika zajetih strani

Najdeno je bilo le eno spletno mesto (slo-tech.com).

5. Povezave strani

Skupno število najdenih povezav na vseh spletnih straneh je bilo 450250. V povprečju pa je bilo najdenih 73,7 na stran. S pomočjo orodja Sigma smo vizualizirali prvih nekaj strani z največjim številom povezav. S slike 1 je razvidno, da jih je vodilna stran (označena z rdečim vozliščem) imela kar 505. Sledijo tri strani s številom strani, večjim od 150 (označene z oranžnim vozliščem) in preostale.



Slika 1 Vizualizacija strani z največ povezavami na spletnem mestu slo-tech.com