Iskanje in ekstrakcija podatkov s spleta
3. seminarska naloga

Andraž Krašovec, Luka Podgoršek in Iztok Ramovš

May 2019

1 Uvod

Cilj tokratne seminarske naloge je bila implementacija indeksa vnaprej določenih splentih strani in iskanja po le-teh. V poglavju *Implementacija* je opisana zgradba naše aplikacije in pojasnjeni so ključni elementi vsakega dela naloge. Sledi poglavje *Rezultati*, kjer predstavimo rezultate zahtevanih poizvedb in primerjamo iskanje po indeksu z naključnim iskanjem. V *Zaključku* osvetlimo dobre in slabe strani našega pristopa ter predlagamo izboljšave.

2 Implementacija

Podobno kot pri prvi nalogi, smo se odločili za uporabo ogrodja Django, ki poenostavi marsikateri korak naloge in smo z njim že dobro seznanjeni. Nalogo smo implementirali znotraj dveh Django aplikacij – *Processing* in *Retrieval*. Prva je namenjena procesiranju in indeksiranju, prav tako pa vsebuje preslikavo podatkovnega modela, medtem ko druga vsebuje implementacijo obeh pristopov iskanja.

2.1 Procesiranje

Cilj procesiranja je pretvorba html dokumenta v seznam besed, ki jih nato uporabimo pri indeksiranju in iskanju. Glavni del implementacije smo razdelili tako, da lahko kasneje pridobimo tako očiščeno, kot tudi neočiščeno besedilo.

Najprej smo izvedli ekstrakcijo neočiščenega besedila s html dokumenta. Tu smo si pomagali s knjižnico bs4, tako da smo jo inicializirali s html parserjem. Nato smo odstranili vse *script* in textitstyle tag-e, sicer bi nam vsebina le-teh pokvarila končni rezultat. Zatem smo uporabili funkcijo *get_text*, ki izvleče besedilo s html dokumenta. Na koncu smo odstranili še vse odvečne presledke in skoke v nove vrstice ter s tem dobili vsebino dokumenta kot niz, kjer so besede ločene s presledkom. Tako besedilo nam bo prišlo prav kasneje, ko bomo pridobivali odrezke iz besedila.

Ker pa tako obdelanih besed ne moremo zapisati v tabelo indeksov, smo pridobljeno besedilo dodatno očistili, da je bilo pripravljeno za vnos v podatkovno bazo. Najprej smo vse črke pretvorili v male, da preprečimo pojavitve istih besed, pisanih v različnih kombinacijah velikih in malih črk. Primarni načrt je bil, da v naslednjem koraku besede korenimo, vendar smo to na koncu opustili, saj so bili tako pridobljeni rezultati precej čudni – najverjentneje, ker smo korenili slovenske in ne angleških besedil. Nato smo besede še tokenizirali in odstranili stopword-e. S tem smo besedilo razbili na posamezne besede, ostranili ločila in pogosto rabljene besede v slovenskem jeziku, ki pa ne nosijo veliko informacije. Rezultat te funkcije je seznam besed, pripravljenih za vnos v podatkovno bazo.

2.2 Indeksiranje

Indeksiranje je tesno povezano s samim procesiranjem in zato tudi implementirano v isti Django aplikaciji. Pri indeksiranju se premikamo dokument po dokument in najprej uporabimo zgoraj omenjene funkcije, ki nam vrnejo besedilo tako očiščene, kot tudi neočiščene vsebine spletne strani vsebine. Slednja nam služi za ekstrakcijo pravilnih indeksov z besedila in jo ponovno uporabimo tudi kasneje - pri sestavljanju odrezkov rezultatov. Prva pa služi kot filter besed, ki jih želimo indeksirati – s tem preprčimo indeksiranje ločil in stopword-ov.

Zatem vsako besedo, ki je nismo zavrgli, vstavimo v podatkovno bazo, skupaj s podatki o številu pojavitev, mestih, kjer se beseda pojavi in pripadajočega dokumenta. V našem primeru je indeksiranje vseh pripravljenih spletnih strani trajalo približno eno uro. Zanj smo implementirali posebno manage.py komando *python manage.py filldatabase*.

2.3 Iskanje

S polno podatkovno bazo, se lahko osredotočimo na iskanje zadetkov s podajanjem ključnih besed. Le-to sicer izkorišča samo prvi pristop, medtem ko drugi, kot pove že ime, ne uporablja nikakršnih naprednih metod in vsakič preišče celoten prostor dokumentov in smo ga implementirali zavoljo primerjave s prvim pristopom.

2.3.1 Iskanje po indeksu

Iskanje po indeksu je implementirano v aplikaciji retrieval, natančneje v razredu *IndexRetrieval*. Za lažji zagon programa, smo implementirali manage.py funkcijo po meri – za zagon poženemo komando *python manage.py query 'želena poizvedba'*. Potek iskanja lahko povzamemo v štirih korakih:

- Procesiranje vhoda,
- pridobivanje rezultatov,
- pridobivanje odrezkov,
- izpis rezultatov.

Vsak izmed korakov je implementiran v svoji funkciji, znotraj prej omenjenega razreda.

Prvi korak naše implementacija iskanja po indeksu je, tako kot pri večini iskanj, procesiranje uporabnikovega vnosa. Predpostavili smo, da nimamo opravka z zlonamernimi uporabniki in da so potencialne poizvedbe veljavne, kar pomeni, da smo v tem koraku zgolj pretvorili vse črke v male in samo poizvedbo razdelili na posamezne besede.

Pri pridobivanju rezultatov smo za vsako besedo poizvedbe poiskali vse dokumente, v katerih je prisotna in prišteli frekvenco trenutne besede v danem dokumnetu v skupno frekvenco posameznega dokumenta. Zavoljo jedrnatosti izpisa rezultatov, smo se odločili, da bomo prikazali samo 5 najbolj relevantih rezultatov, t.j. 5 rezultatov z najvišjo skupno frekvenco iskanih besed. Zadnji korak te funkcije, je da dobljenih 5 rezultatov še razvrstimo v padajočem vrstnem redu skupne frekvence.

Ključni element pridobivanja odrezkov (snippet-ov) je v indeksih besed iz poizvedbe, ki smo jih shranili poleg skupnih frekvenc in imena vsakega dokumenta. Le-ti nam omogočijo, da iz neočiščene vsebine izbranih spletnih strani izluščimo iskano besedo, vključno s tremi besedami pred in za izbrano besedo. Na tem mestu naj omenimo, da nestična ločila (poimšljaj, tri pike, ipd.) štejejo kot samostojne besede, saj smo neočiščeno besedilo ločili na besede s presledki. Vsa ostala ločila se ne štejejo kot samostojne besede, ker se v neočiščenem besedilu držijo ostalih besed. Tako pridobljene izrezke dodamo prej pridobljenim rezultatom in tako zaključimo izvajanje te funkcije.

Zadnji korak iskanja po indeksu je izpis rezultatov. Zaradi preglednosti smo se omejili na 5 najbolj relevantih rezultatov in izpis odrezkov omejili na 200 znakov, kar v večini primerov zadošča za izpis vsaj treh odrezkov v celoti. Sam izpis je drugače trivialen – zbrane rezultate s prejšnjih funkcij zgolj izpišemo v predpisani obliki.

2.3.2 Naivno iskanje

Naivno iskanje je implementirano v razredu *Sequential* aplikacije *retreival*. Naivno iskanje zaganjamo v *main* metodi omenjenega razreda. Kot parametre razredu podamo iskalno geslo in pa pot do mape, v kateri se nahajajo datoteke v katerih iščemo. Postopek naivnega iskanja je podoben postopku iskanja po indeksu, s ključno razliko v tem, da iskanje izvajamo tako, da sproti ročno iščemo po izvornih datotekah.

Korak, ki se v primerjavi z iskanjem po indeksu, razlikuje je *pridobivanje rezultatov*. Ostali koraki (procesiranje vhoda, izpis rezultatov in pridobivanje odrezkov) so narejeni tako kot pri iskanju po indeksu.

Pri pridobivanju rezultatov smo vsako datoteko odprli in jo očistili z zgoraj omenjenimi funkcijami. Po predprocesiranju smo nato za vsako besedo iskali ujemanja in tako določili frekvenco besed za dano datoteko. Tako kot pri iskanju po indeksu, smo obdržali samo pet rezultatov z največjim rangom.

poizvedba	naivno iskanje	iskanje z indeksom
predelovalne dejavnosti	39s 122ms	136ms
trgovina	37s 270ms	520ms
social services	34s 453ms	115ms
programiranje računalnik	35s 976ms	95ms
samostojni podjetnik	36s 627ms	614ms
uprava ministrstvo	37s 265ms	676ms

Tabela 1: Čas trajanja poizvedbe s posamezno metodo iskanja.

3 Rezultati

Za primerjavo hitrosti izbranih metod iskanja, smo pognali šest različnih poizvedb:

- predelovalne dejavnosti
- trgovina
- social services
- programiranje računalnik
- samostojni podjetnik
- uprava ministrstvo

Razlika je opazna in se meri v nekaj velikostnih razredih. Seveda se iskanje z indeksom izkaže za bistveno hitrejše. V tabeli 1 je prikazan čas posamezne poizvedbe za obe metodi iskanja. Izpis iskanja poizvedb je priložen v mapi results.

Podatkovna baza, v katero smo vpisovali indekse, je na koncu vsebovala 32.129 indeksiranih besed (IndexWord) in 312.454 pojavitev besed v dokumentih (Posting). Besede z največjo frekvenco v posameznem dokumentu so: skupnost (807), krajevna (754) in ministrstvo (543).

4 Zaključek

Spoznali smo ključne elemente indeksiranja in iskanja. Opazili smo lahko bistvene razlike v učinkovitosti iskanja po indeksu v primerjavi z naivnim iskanjem. Iz tega smo se lahko naučili, kako pomembno je indeksiranje za učinkovitejšo porabo sredstev.