Data processing and indexing

Edo Ljubijankić

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani,Večna pot 113, Slovenija
el0978@student.uni-lj.si

Abstract. Naloga je sestavljena iz treh delov. Pri del je indeksiranje podatkov, drugi del je izvedba sql poizvedb in hitro iskanje podatkov po bazi z uporabo indeksov, tretji pa je podoben drugemu delu le da se ne uporablja baza. Drugi in tretji del naloge sta vrnila enake rezultate, le časovno izvajanje je bilo drugačno. Opis opravljenega dela in rezultati časovno primerjave so predstavljeni v tem poročilu.

1 Uvod

Cilj te naloge je bilo podrobneje spoznati kako hitro in učinkovito je najti podatke znotraj baze s pomočjo indeksa. V našem primeru smo zgradili inverted index, s katerim smo preslikali vsebino (besede) html datotek v bazo. Vsaka od besed je bila indeksirana tj. imela lokacijo v tabeli. Namen obrnjenega indeksa je omogočiti hitro iskanje po celotnem besedilu. Inverted index se uporablja v sistemih za pridobivanje dokumentov, v različnih brskalnikih itd.

2 Data processing and indexing

Za začetek smo imeli na voljo množico html datotek iz 4 različnih domen. Ustvarili smo tabeli IndexWord in Posting, ki predstavljata našo bazo. S for zanko smo prebrali vsako html datoteko in shranili iz nje samo tekst brez zna?k in skript. Uporabili smo get_text() funkcijo iz knjižnice BeautifulSoup v kombinaciji s html parserjem. Vsebina oz. tekst vsake datoteke je pred indeksiranjem morala biti predprocesirana.

Pri predpocesiranju smo velike črke spremenili v male, kar je veljalo za celoten tekst. Nato smo odstranili ločila. Zaradi tega je morda snippet malo grši ni pa to ključnega pomena. Nobenemu uporabniku ni namen s poizvedbo iskati iskati ločila, zato so bila odstranjena. Opazil sem, da regex ni odstranil znaka ¨ ; kar je neposrečeno, a rešitve za to nismo našli v primernem času. Nato smo tekst tokenizirali (torej razdelili na besede). Iz teh besed (tokeni) smo izločili tako imenovane stopworde. Stopwordi so v našem primeru samo slovenske besede.

Za vsako od teh besed (token) smo izračunali frekvenco pojavitve in hkrati indekse pojavitev. Nato smo sestavili zapise (tuple) z besedo, datoteko besede, frekvenco pojavitve besede in indekse pojavitve besed. Indekse pojavitve smo kot stringe pripeli skupaj z vejico in zadnjo vejico odstranili. Te zapise smo vstavili v bazo tabel in sestavili inverted index bazo.

3 Data retrieval with inverted index

Z SQL poizvedbo (query) smo iz baze hoteli najti nekatere od indeksiranih besed. Če smo s poizvedbo iskali z eno besedo smo zelo hitro locirali, kje se beseda nahaja v bazo zaradi indeksov. Izvedba je tudi vrnila rezultate sortirane po frekvencah. To besedo smo pridobili ponavadi pod sekundo in izpisali na standardni izhod v katerih vse datotekah se nahaja, frekvenco pojavitve v datoteteki in snippet (okolica besed v tekstu).

Če je poizvedba zahtevala dve ali več besed smo te besede razdelili in iskali vsako besedo posebej tako kot je opisano na zgornji način. Recimo za dve besedi, smo dobili seznam, kjer smo nato pogledali kateri dve besedi sta znotraj iste datoteke. Take besede, ki so se našle so bile združene, pravtako se je združila njihova frekvenca in njihovi indeksi. Nato smo seznam sortirali po frekvencah pojavitev od največje do najmanjše in izpisali tako kot je opisano na zgornji način.

4 Data retrieval without inverted index

Za primerjavo kako poteka iskanje brez inverted indeksa smo implementirali funkcijo na naslednji način. Sprehodili smo se s for zanko čez vse html datoteke, prebrali tekst, predprocesirali in tokenizirali. V vsaki iteraciji for loopa smo imeli en dokument iz katerega smo takoj na mestu računali frekvenco iskane besede, stevilo pojavitev in njene indekse. Nato smo z dodatno funkcijo ustvarili zapis v vsaki iteraciji, kjer je bila iskana beseda prisotna. Ko se je celotna for loop zanka končala smo seznam zapisov sortirali po frekvencah pojavitev in izpisali v formatiranem izpisu na standardni izhod. Čas iskanja po vsakem dokumentu posebej je bil vedno nad minuto.

5 Baza

Baza je ogromna po indeksiranju. Tabela IndexWord ima 33299 vstic zapisov oz. toliko indeksiranih besed, tabela Posting pa 360798 vrstic. Največ frekvenc ima beseda *proizvodnja*, ki je v datoteki *evem.gov.si.371.html*. Naslednji dve besedi po frekvencah sta prav tako v tej datoteki. In še:

- podatki.gov.si.340.html: skupnost
- e-uprava.gov.si.36.html: območje

6 Ugotovitve

Čas iskanja po vsakem dokumentu, ki ni indeksiran je izrazito večji od inverted index iskanja. Merjenje s štoparico je v primeru inverted index potekala tako, da je začela meriti čas takoj ko se je začela poizvedba po bazi in koncala po združitvi (merganju) zapisov. Torej merjenje izpisa na standardni vhod ni vključeno. V

primeru brez inverted indexa je merjenje tudi začeto na začetku pred branjem datotek in končano pred izpisom na standardni izhod.

Primeri izhodov so shranjeni v mapi outputs v projektu. Recimo poizvedba po besedi *arhiv* vrne za inverted index iskanje 0.006093 sekund, medtem ko za naivno iskanje porabi 71.074457 sekund. Primer časov je prikazan na tabeli 1.

	arhiv	davek
z indeksom	0.006093 s	0.000631 s
brez indeksa	71.074457 s	72.466413 s

Table 1. Primerjava časov.