Implementacija preprostega indeksa in iskanje podatkov po indeksu

Skupina **DOMACI-NJOKI**: Niki Bizjak, Bojan Vrangeloski, Uroš Škrjanc

***Povzetek*—V tretji seminarski nalogi pri predmetu Iskanje in ekstrakcija podatkov s spleta, smo implementirali preprost indeks, ki, glede na našo poizvedbo, vrne spisek dokumentov, v katerih se nahajajo besede iz naše poizvedbe, ter frekvenco besed poizvedbe v vsakem dokumentu posebej. Poizvedbe smo izvajali nad HTML dokumenti, podatki pa so bili shranjeni v bazo po principu obrnjenega indeksa (inverted index). Za primerjavo hitrosti smo implementirali tudi direktno iskanje poizvedbe po HTML dokumentih.**

I. Uvod

Princip obrnjenega indeksa (inverted index) nam omogoča, da na ekonomičen način v bazo shranimo posamezne besede, ki se pojavijo v nam zanimivih dokumentih. Namesto da bi za vsak dokument vodili evidenco, ali se posamezna beseda iz slovarja v dokumentu pojavi ali ne (tabela velikosti števila dokumentov pomnoženo s številom besed v slovarju), najprej naredimo ali posodobimo slovar besed, ki se nahajajo v dokumentih, potem pa v ločeni tabeli v posamezen zapis zapišemo indeks dokumenta, indeks besede, frekvenco pojavljanja besed v določenem dokumentu in pozicije besede v dokumentu. Tako ne samo dosežemo prostorsko optimizacijo podatkov, tudi sama poizvedba se izvede hitreje, ker je potrebno manjkrat posegati na trdi disk, ki je še vedno ozko grlo informacijskih sistemov.

II. Implementacija procesiranja in indeksiranja podatkov

Vsak HTML dokument program odpre, prebere vsebino datoteke, vsebino očisti HTML kode, očiščen tekst potem razdeli na posamezne dele teksta (token). Program potem izloči nepotrebne tokene. Odločili smo se, da poleg nepotrebnih besed (stop-words), odstranimo tudi besede krajše od treh znakov, datume, številke, tako cela števila kot tudi števila z decimalno vejico ali decimalno piko. S tem se znebimo vseh podatkov, ki po našem mnenju niso relevantni in dosežemo, da je baza manjša in hitrejša pri iskanju podatkov. Za čiščenje HTML oznak smo uporabili knjižnico BeautifulSoup, za tokeniziranje teksta pa funkcijo word_tokenize iz knjižnice NLTK.

Ko program dobi unikatne tokene iz dokumenta, poišče še frekvence pojavljanja in pozicije posameznega tokena v dokumentu ter delčke dokumenta (snippet), kjer se posamezen token pojavi. Dobljene podatke zapiše v dve ločeni tabeli v bazi – frekvence in pozicije v tabelo v eno in snippete v drugo tabelo. Za potrebe naloge smo shemo podatkovne baze razširili z dodatno tabelo Snip, ki vsebuje podatke o dokumentu, tokenu, poziciji tokena v dokumentu ter snippet. Za to smo se odločili zaradi hitrejšega iskanja, saj nam tako ni treba posegati v datoteke na disku.

V tej fazi posebnih težav ni bilo. Morda je še največ težav povzročala lastnost sqlite3 knjižnice, s katero ni mogoče ugotoviti, ali baza na disku že dejansko obstaja ali ne, kar lahko včasih povzroči kako manjšo težavo.

III. Implementacija pridobivanja podatkov iz baze

Pri iskanju po bazi program iz poizvedbe ravno tako najprej izloči stop-words, kratke tokene, datume in številke. Te tokene potem z SQL poizvedbo pošlje v bazo, kot rezultat pa dobi tabelo zapisov, kjer so v vsakemu izmed zapisov podatki o dokumentu, iskanem tokenu, frekvenca tokena v dokumentu, pozicija tokena v dokumentu ter snippet, ki pripada tokenu. Pri izpisu združi tokene, ki se nahajajo v istem dokumentu v eno vrstico v izpisu.

Snippete združi tako, da najprej izpiše tiste snippete, ki vsebujejo največ besed iz poizvedbe. Zaradi dolžine izpisa program izpiše samo prvih 100 znakov združenega snippeta.

Največji izziv pri tej fazi je bilo pravilno formulirati SQL poizvedbo, sploh pri tako veliki količini podatkov, kjer je nemogoče preveriti, če poizvedba deluje na vseh primerih pravilno. Druga težava, s katero smo se ukvarjali nekaj več časa, je bil izpis rezultatov, predvsem kako pravilno izpisati snippete. Tudi za to velja, da je nemogoče preveriti vse možnosti, tako da smo delali na manjši količini dokumentov in upali, da deluje tudi na celoti.

IV. Implemetacija pridobivanja podatkov iz HTML datotek

Progam najprej poizvedbo obdela tako, kot pri iskanju po bazi. Potem odpre vsako datoteko posebej, jo na podoben način, kot smo počeli pri procesiranju in indeksiranju podatkov, tokenizira, pogleda, če se beseda nahaja v tekstu in če se, potem zapiše podatke v enako strukturirano tabelo v pomnilniku, kot smo jo dobili pri iskanju podatkov iz podatkovne baze. Za izpis podatkov potem uporabimo isto funkcijo kot pri poizvedbi iz baze, tako da je izpis enak.

V. Struktura in podatki o bazi

Za potrebe naloge smo shemo podatkovne baze razširili z dodatno tabelo Snip, ki vsebuje podatke o dokumentu, tokenu, poziciji tokena v dokumentu ter snippet. Struktura tabele je podana z naslednjo kodo:

```
CREATE TABLE Snip (
  documentName TEXT NOT NULL,
  word TEXT NOT NULL,
  posIndex INTEGER NOT NULL,
  snippet TEXT NOT NULL,
```

```
  PRIMARY KEY(documentName,word,posIndex),
  FOREIGN KEY (word) REFERENCES IndexWord(word)
);
```

Za kreiranje baze je program potreboval okrog 12 minut za vse dane HTML dokumente, njena velikost pa je 165 MB. V bazo je bilo po postopku, opisanem v drugem delu, vpisano 36.111 različnih tokenov. Tabela Posting vsebuje 359.852 zapisov, tabela Snip vsebuje 763.739 zapisov.

V prvi tabeli navajamo tokene z največjimi frekvencami v dokumentih, v drugi tabeli pa dokumente, ki vsebujejo največ različnih tokenov

token	frekvenca	št. dokumentov
podatkov	11089	864
slovenije	10507	1363
republike	8583	1165
zakon	6093	754
podatki	5809	734

dokument	št. tokenov
evem.gov.si.371.html	12384
podatki.gov.si.340.html	6489
e-prostor.gov.si.57.html	1701
evem.gov.si.398.html	1559
evem.gov.si.651.html	1277

VI. Rezultati poizvedb

Program smo testirali na naslednjih poizvedbah:

- predelovalne dejavnosti
- trgovina
- social services
- zakon
- delovno dovoljenje
- ministrstvo za javno upravo

Kot je bilo za pričakovati, so poizvedbe, sploh če so snippeti že zapisani v bazo, bistveno hitrejše kot je iskanje besed po HTML dokumentih. V spodnji tabeli so navedeni časi, potrebni za posamezne poizvedbe.

Iskanje	po bazi	po dokumentih
predelovalne dejavnosti	43 ms	57981 ms
trgovina	17 ms	55098 ms
social services	22 ms	55866 ms
zakon	10 ms	53738 ms
delovno dovoljenje	22ms	58991 ms
ministrstvo za javno upravo	88 ms	51932 ms

Če samo povzamemo kolikokrat se posamezne iskane besede in njihove kombinacije pojavljajo v dokumentih:

- **predelovalne dejavnosti** v 755 dokumentov, od tega **predelovalne+dejavnosti** v 3 dokumentih, **predelovalne** v 0 dokumentih in **dejavnosti** v 752 dokumentih
- **trgovina** v 127 dokumentih
- **social services** v 4 dokumentih, od tega **social+services** v 2 dokumentih, **social** v 0 dokumentih in **services** v 2 dokumentih
- **zakon** v 624 dokumentih
- **delovno dovoljenje** v 257 dokumentih, od tega **delovno+dovoljenje** v 19 dokumentih, **delovno** v 32 dokumentih in **dovoljenje** v 206 dokumentih
- **ministrstvo za javno upravo** v 1023 dokumentih, od tega **ministrstvo+javno+upravo** v 793 dokumentih, **ministrstvo+javno** v 12 dokumentih, **ministrstvo+upravo** v 1 dokumentu, **javno+upravo** v 11 dokumentih, **ministrstvo** v 160 dokumentih, **javno** v 22 dokumentih in **upravo** v 24 dokumentih

Izpisi vseh poizvedb se nahajajo v mapi results v naslednjih datotekah:

- poizvedba_PredelovalneDejavnosti.txt
- poizvedba_Trgovina.txt
- poizvedba_SocialServices.txt
- poizvedba_Zakon.txt
- poizvedba_DelovnoDovoljenje.txt
- poizvedba_MinistrstvoZaJavnoUpravo.txt

VII. Zaključek

Gradnja preprostega indeksa ni ravno zapletena, se pa izkaže, tako glede časovne komponente, kot tudi preprostosti, uporaba podatkovne baze kot bistveno bolj učinkovita rešitev. Gradnja baze sicer traja nekaj časa, vendar preprost izračuna pokaže, da se obrestuje že pri 20 poizvedbah.