Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Seminar 3

Anja Hrovatič, Jan Fekonja in Nina Vehovec

27. Maj, 2019

1 Uvod

V sklopu seminarske naloge [1] smo implementirali predprocesiranje, indeksiranje in pridobivanje podatkov. Iz podanih HTML datotek smo pridobili celotno besedilo iz spletne strani ter nad le-tem izvedli predprocesiranje in hrambo podatkov v indeks. Pri predprocesiranju smo odstranili vsebinsko nepomembne besede *angl. stopwords*, ločila in velike črke spremenili v male. Nato smo implementirali še pridobitev podatkov s pomočjo poizvedb z uporabo invertnega indeksa ter z naivnim načinom pridobitve podatkov(sekvenčno branje datotek). Učinkovitost pridobivanja podatkov (naših metod) smo preverili na šestih različnih poizvedbah (čas izvajanja).

2 Predprocesiranje podatkov

Podatke smo prebrali iz HTML datotek in celotno vsebino iz strani pridobili s pomočjo knjižnice BeautifulSoup [2]. Nato smo iz pridobljene vsebine odstranili elemente tipa "script" in "style," ter zatem s funkcijo "get_text()" iz vsebine izluščili besedilo. Iz besedilo smo nato razčlenili v vrstice, odstranili vse odvečne presledke in skoke v novo vrstico.

Za tokenizacijo pridobljenega teksta smo uporabili nltk.tokenize paket [3]. Iz tokeniziranega besedila smo nato odstranili še besede brez vsebinskega pomena *(ang. stop words)*. Z funkcijo "stopwords.words("slovenian")" smo pridobili besede iz dokumenta [4] in jih uporabili nad pridobljenim besedilom iz HTML strani. Iz tokeniziranega seznama besed smo odstranili tudi različne znake kot so: ".,!?-><:;)(–,–/|“»«" in vse besede pretvorili v majhne črke.

Na zgoraj opisan način smo torej pridobili besedilo nad katerim smo potem izvedli indeksiranje podatkov.

3 Indeksiranje podatkov

V tem sklopu smo se ukvarjali z indeksiranjem podatkov. Za indeksiranje smo simulirali invertiran indeks z uporabo SQLite podatkovne baze s shemo, kot je podana v navodilih [1].

Postopek indeksiranja smo izvedli tako, da smo za vsak posamezen HTML dokument pridobili besedilo, ga obdelali kot je opisano v prejšnem razdelku ter za vsako besedo v obdelanem besedilu poiskali indekse pojavitev le-te v originalnem besedilu. Določili smo tudi frekvenco besede, torej število pojavitev. Nato smo besedo shranili v podatkovno bazo v tabelo *IndexWord*. Dodali smo samo besede, ki še niso bile prisotne v tabeli. Nato smo v bazo dodali tudi Posting, ki je vključeval besedo, ime HTML dokumenta, frekvenco pojavitev ter indekse, kjer se beseda nahaja v originalnem besedilu.

4 Pridobivanje podatkov

Pridobivanje podatkov smo implementirali na dva različna načina. Prvi način je bil s pomočjo invertnega indeksa, kjer smo podazke pridobivali iz baze. Pri drugem načinu pa nismo uporabljali baze in indeksov, zato smo vsak dokument posebej odprli, ga sprocesirali in v njem poiskali podatke, ki smo jih želeli pridobiti. Torej na oba načina smo pridobili podatke za poizvedbe: "predelovalne dejavnosti", "trgovina" in "social services". Poleg teh treh poizvedb, pa smo definirali še tri svoje poizvedbe, ki so bile: "statistični urad RS", "otroški dodatki in državna štipendija" in poizvedba "fakulteta".

4.1 Pridobivanje podatkov z uporabo indeksa

Preden smo se lotili pridobitve podatkov, smo poizvedbo še predprocesirali, tako kot smo pred tem predprocesirali besedilo iz html dokumenta. V prvem koraku smo poizvedbo tekonizirali in nato iz nje odstranili vse besede brez vsebinskega pomena *angl. stopwords*, ločila ter vse velike črke spremenili v majhne. Tokenizirano poizvedbo lahko vidimo v spodnjem prikazu kode 1.

Torej pri pridobivanju podatkov s pomočjo invertnega indeksa imamo v bazi shranjene vse indekse in v tabeli *Posting* tudi podatke, kjer se ti indeksi nahajajo v posameznih HTML dokumentih oziroma pridobljenih besedilih iz leteh. Na podlagi tega smo definirali poizvedbo na tabeli *Posting*. Poizvedba, ki je prikazana spodaj, nam vrne vse dokumente, ki vsebujejo besede iz predprocesirane poizvedbe. Ti dokumenti so pogrupirani in razvrščeni po padajoči frekvenci. Frekvenca, pa je ubistvu seštevek pojavitev posameznih besed iz poizvedbe v določenem dokumentu. Poizvedba nam vrne tudi vse pozicije(indekse) iskanih besed, torej pozicije, kje se iskana beseda nahaja v originalnem besedilu(dokumentu).

```
1
2       def get_querry_from_posting(self, word1, word2, word3,
            word4, word5):
```

```
3           self.curr.execute('SELECT p.documentName AS docName,
                SUM(frequency) AS freq, GROUP_CONCAT(indexes) AS
                idxs '+
4                             'FROM Posting p WHERE p.word IN (?,
                                  ?, ?, ?, ?) '+
5                             'GROUP BY p.documentName '+
6                             'ORDER BY freq DESC;',(word1, word2,
                                  word3, word4, word5))
7           data = self.curr.fetchall()
8           print("Data from querry: ", data)
9           return data
10
11      querried_data = database.get_querry_from_posting('
            predelovalne', 'dejavnosti', '', '', '')
```

Listing 1: Poizvedba za inverted indeks

Ko imamo vse te podatke, se samo sprehodimo čez vse dokumente, ki nam jih je vrnila poizvedba, za vsakega izpišemo njegov url in frekvenco, ter za največ (prvih) pet pozicij iskanih besed v dokumentu izpišemo še izrezke *angl. snippet*. V izrezku so ločila prepoznana kot svoj člen in pred vsakim ločilom je presledek, ker smo uporabili člene iz razčljenjenega besedila. Izrezek vsebuje tri člene pred iskano besedo in tri člene za iskano besedo. V kodi tudi preverjamo, če je iskana beseda na začetku ali koncu dokumenta, v tekih primerih izpišemo manj ali nič členov pred ali za iskano besedo, odvisno od pozicije besede v dokumentu.

4.2 Pridobivanje podatkov na naiven način

4.3 Primerjava metod za pridobivanje podatkov

Primerjava obeh metod za pridobivanje podatkov je pokazala, da je metoda, kjer uporabljamo SQLite bazo z invertiranim indeksom veliko hitrejša od metode s sekvenčnim poizvedovanjem in združevanjem podatkov. Metoda s sekvenčnim branjem datoteke (naiven način) vrne rezultate v nekaj 10 sekundah, načeloma pa je to malo odvisno od poizvedbe, ampak pri nas je bilo to v povprečju približno v pol minute oz. malo več. Metoda, ki uporablja invertni indeks, pa poizvedbo naredi v nekaj milisekundah. Ponovno je čas zopet precej odvisen od poizvedbe oziroma natančneje števila rezultatov v poizvedbi. Z gotovostjo pa lahko rečemo, da invertni indeks pohitri poizvedbo vsaj nekaj 10 tisočkrat (v našem primeru približno 30 000 krat), kar lahko vidimo v tabeli 1.

5 Podatkovna baza

Spodaj so prikazani rezultati našega indeksiranja. Vidimo lahko koliko besed smo indeksirali, število zapisov v tabeli Posting, besede in dokumente z najvišjimi frekvencami. V tabeli *IndexWord* so prisotne tudi besede, ki nimajo

poizvedba	invertni indeks	sekvenčno branje datoteke
"trgovina"	4 ms	36 s
"social services"	< 1 ms	37 s
"predelovalne dejavnosti"	22 ms	45 s
"statistični urad RS"	17 ms	36 s
"otroški dodatki in državna štipendija"	2 ms	36 s
"fakulteta"	< 1 ms	40 s

Table 1: Primerjava časov izvajnja poizvedb za obe metodi.

pomena, kot so "/xsd", "lt", "gt", datumi, številke in podobno, kar narekuje na še bolj podrobno predprocesiranje besedila, da bi indeksirali res samo pomenske besede.

- Število indeksiranih besed:
  - 49231
- Besede z najvišjimi frekvencami:
  - proizvodnja (2266)
  - spada (1338)
  - dejavnosti (1284)
  - skupnost (809)
  - krajevna (754)
  - ministrstvo (589)
  - šola (582)
  - dejavnost (544)
  - vir (526)
  - družina (475)
- Dokumenti z najvišjimi frekvencami:
  - evem.gov.si/evem.gov.si.371.html (83583)
  - podatki.gov.si/podatki.gov.si.340.html (27742)
  - e-prostor.gov.si/e-prostor.gov.si.166.html (11199)
  - podatki.gov.si/podatki.gov.si.511.html (5178)
  - e-prostor.gov.si/e-prostor.gov.si.218.html (4500)
  - evem.gov.si/evem.gov.si.398.html (4396)
  - e-prostor.gov.si/e-prostor.gov.si.147.html (4395)
  - e-prostor.gov.si/e-prostor.gov.si.57.html (3764)
  - podatki.gov.si/podatki.gov.si.327.html (3369)

  - e-prostor.gov.si/e-prostor.gov.si.150.html (3107)

- Število zapisov v tabeli Posting:
  - 394904

6 Zaključek

Implementirali smo pridobitev in predprocesiranje besedila iz HTML dokumentov ter indeksiranje besed z uporabo invertnega indeksa. Na podlagi pridobljenega besedila smo nato implementirali tudi pridobitev podatkov, in sicer na dva načina: z uporabo invertnega indeksa ter naiven pristop k pridobitvi podatkov.

Predprocesiranje smo izvedli z uporabo nltk.tokenize paketa in iz besedila odstranili tudi vse besede brez pomena, ločila ter druge znake. Invertiran indeks pa smo simulirali z uporabo SQLite podatkovne baze. Indeksirali smo 49231 besed, marsikatere od teh pa so tudi besede brez pomena, številke, datumi in znaki, ki bi jih lahko odstranili z še bolj podrobnim predprocesiranjem besedila. Predprocesiranje je pomembno, ker je od tega odvisno število indeksov, ki jih imamo in s tem velikost baze. Manjša ko tje baza, hitreje bomo iskali po njej.

Pridobivanje podatkov smo implementirali na dva načina, torej z uporabo invertnega indeksa ter naiven pristop. Primerjava obeh pristopov je pokazala, da je uporaba invertnega indeksa precej hitrejša od naivnega pristopa - za poizvedbe z malo rezulatati, kot so npr. poizvedbe: "trgovina","social services", "fakulteta" in "otroški dodatki in državna štipendija" traja pridobiev podatkov le okoli 1ms, za poizvedbe z več rezultati pa v našem primeru tudi do 22ms. Seveda, če bi poizvedbe vsebovale še več rezultatov, bi bil čas še malo večji. Medtem ko je z naivnim pristopom ta čas precej daljši od 36s, tako za poizvedbe z malo rezultati, kot tudi za poizvedbe z velikim številom retultatov, kar pomeni,da je poizvedba s pomočjo invertenga indeksa pri naših primerih vsaj 30 000 krat hitrejša.

Literatura

[1] Slavko Žitnik, "Programming assignment 2," [Dostopano: 26. 5. 2019]. [Online]. Available: http://zitnik.si/teaching/wier/PA3.html

[2] BeautifulSoup, "Beautiful soup documentation," [Dostopano: 20. 5. 2019]. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[3] Nltk, "Nltk tokenize package," [Dostopano: 20. 5. 2019]. [Online]. Available: https://www.nltk.org/api/nltk.tokenize.html

[4] ——, "Stopwords for slovene," [Dostopano: 20. 5. 2019]. [Online]. Available: https://github.com/nltk/nltk_data/issues/54