

Procesiranje, indeksiranje in iskanje po podatkih

Tretja naloga pri predmetu Iskanje in ekstrakcija podatkov s spleta

Žiga Kleine
Fakulteta za računalništvo in
informatiko
Univerza v Ljubljani
Email: zk5343@student.uni-lj.si

Tim Križnik
Fakulteta za računalništvo in
informatiko
Univerza v Ljubljani
Email: tk0730@student.uni-lj.si

Luka Železnik
Fakulteta za računalništvo in
informatiko
Univerza v Ljubljani
Email: lz6356@student.uni-lj.si

I. POVZETEK

V tem poročilu bomo predstavili še zadnjo nalogo pri predmetu Iskanje in ekstrakcija podatkov s spleta. Pri tej nalogi smo tekst iz vnaprej podanih 1416 spletnih strani najprej predhodno procesirali, nato zgradili invertiran indeks s pomočjo SQLite podatkovne baze, na koncu pa še implementirali iskanje po našem invertiranem indeksu. Da smo lahko ovrednotili učinkovitost invertiranega indeksa, smo implementirali še iskanje po dokumentih brez uporabe invertiranega indeksa. Rezultate poizvedb na obeh implementacijah iskanja bomo ustrezno primerjali in predstavili.

II. UVOD

Invertirani indeks je podatkovna struktura s pomočjo katere lahko pospešimo iskanje po besedilu v večjem korpusu dokumentov. Deluje tako, da mapira besede v točne lokacije teh besed v izvornih besedilih.

Da lahko z invertiranim indeksom uspešno indeksiramo večje korpuse dokumentov, jih moramo ustrezno predhodno procesirati. Metode predhodnega procesiranja, uporabljene za potrebe naše implementacije invertiranega indeksa so pretvorba besedil v majhne črke, odstranjevanje nerelevantnih besed (angleško *stopwords*) in tokenizacija besedila.

III. IMPLEMENTACIJA

A. Predhodno procesiranje in indeksiranje

Datoteke tipa HTML program *indexing.py* prebere eno za drugo in jo s razščenjenalno knjižnico *BeautifulSoup* pretvori v objekt *soup*. Elementi tipa *script* in *style* so takoj na začetku s klicem metode *extract()* izločeni iz tega objekta. Hkrati tudi elementi tipa *Comment*, saj ne želimo indeksirati komentarjev v kodi. Iz objekta *soup* pridobimo ves tekst z ukazom: *soup.body.get_text(separator=' ')*. Ta vrne ločeno besedilo iz različnih HTML oznak, ki je pripravljeno na predprocesiranje.

Prvi korak predprocesiranja je tokenizacija pridobljenega besedila s funkcijo *word_tokenize()* knjižnice za obdelavo naravnega jezika *nlTK.tokenize*. Rezultat je tabela blokov (ang. *tokens*) - med sabo ločene besede, števila in simboli na trenutno obravnavani strani. Nize v tej tabeli pretvorimo v nize z malimi črkami s *string.lower()* - obdelava besed poteka le na nizih z malimi črkami.

Pomemben korak na tem mestu je dodelitev indeksa besede v tej tabeli. Podatkovna struktura se spremeni v tabelo Python setov, npr. [(*'ta'*, 0), (*'stran'*, 1), (*'uporablja'*, 2), (*'piškotke'*, 3) ...].

Ker želimo ohraniti pomensko bogate besede seznam še dodatno filtriramo z izločitvijo nerelevantnih slovenskih besed, ki se nahajajo v besednem korpusu paketa *nlTK.corpus*. Korpus smo podaljšali še s seznamom besed v priloženi datoteki *stopwords.py* in sprotimi abnormalijami (npr. niz *'x'*). Z regularnim izrazom izločimo še nize, ki vsebujejo števke (*!bool(re.search(r'', niz))*) in nize simbolov (*bool(re.match('([a-zA-Zčšžćčšžć]+)', niz))*). Ostanejo le pomensko relevantne besede skupaj z njihovimi indeksi.

Skozi podatkovno strukturo besed se pred pisanjem v bazo sprehodimo z namenom, da preštujemo frekvence ponovitev. Seznam indeksov, kjer se beseda ponovi ustrezno daljšamo in struktura dobi obliko, npr. [(*'uporablja'*, [2, 45]), (*'piškotke'*, [3]), ...].

```
e-prostor.gov.si.12.html
[('uporablja', [2, '511', '540', '574', '595', '628']), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.120.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.121.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.122.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.123.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.124.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.125.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])],
e-prostor.gov.si.126.html
[('uporablja', [2]), ('piškotke', [3]), ('nadaljevanje', [6]), ('uporabe', [7, '117', '155', '227', '305', '483', '511', '540', '574', '595', '628']), ('strani', [9, '650', '749']), ('sloglaži', [10])]
```

Fig. 1. Primeri podatkovne strukture z besedami in indeksi ponovitev.

Podatkovno bazo SQLite (s knjižnico *sqlite3*) napolnimo z zbranimi podatki. Vstavljanje v *IndexWord* poteka s poizvedbo *'INSERT OR IGNORE INTO IndexWord(word) VALUES (?)'*, ki ustavi vpisovanje istih besed najdenih na različnih straneh. *IndexWord* je torej seznam vseh indeksiranih besed (ang. *vocabulary*). Tabela *Posting* ima zapis nanašujoč se na specifične strani: (*beseda*, *nazivDokumenta*, *frekvenca*, *indeksi*).

B. Iskanje po podatkih z invertiranim indeksom

Iskanje z invertiranim indeksom poteka z obrnjenim seznamom - s seznama terminov, ki so v slovarju, poiščemo dokumente v katerih se nahaja in dodatne metapodatke/indekske.

Program *run-sqlite-search.py* kot vhod sprejme niz, ki ga predobdela, tako kot pridobljen tekst s spletnih strani v fazi predhodnega procesiranja - tokenizacija, pretvorba v male črke in izločitev nerelevantnih besed. Tuple nizov iz poizvedbe posredujemo metodi, ki dostopa do zapisa *Posting* v podatkovni bazi (fig 2).

```
def search_Postings(query):
    c = conn.cursor()
    sql_query = '''
    SELECT p.documentName AS docName, SUM(frequency) AS freq, GROUP_CONCAT(indexes) AS idxs
    FROM Posting p
    WHERE
        p.word IN ((seqe))
    GROUP BY p.documentName
    ORDER BY freq DESC;
    '''
    seqe = ','.join(['?'] * len(query))
    data_to_insert = query

    try:
        res = c.execute(sql_query, data_to_insert)
        conn.commit()
        return res
    except sqlite3.Error as error:
        print("Error Search: " + error.args[0])
    return
```

Fig. 2. Metoda za iskanje v SQLite podatkovni bazi.

Iz vrnjenih zadetkov v rezultatu preberemo naziv dokumenta, št. pojavitev in indekse (položaj pojavitev). Metodi *get_snippet()* posredujemo te podatke, da odpre dokument zapisan v *Posting-u*, in ga obdela popolnoma enako kot v fazi predprocesiranja: odstrani oznake *script*, *style*, *Comment*, pridobi besedilo z *soup.get_text* in tokeniziraj z enako metodo. Indeksi so posledično enaki, kot so bili pridobljeni v fazi predprocesiranja in se uporabijo za izpis okolice besede (ang. snippet) - okolica je pridobljena iz pravkar ustvarjene tabele. Rezultati večbesednih poizvedb so združeni po dokumentih.

C. Iskanje po podatkih brez invertiranega indeksa

Za implementacijo iskalnika brez invertiranega indeksa smo morali spremeniti in dodelati nekatere elemente prejšnjih dveh implementacij. Ker v tem primeru ne uporabljamo baze, smo lahko ta del povsem izpustili. Namesto v bazo, smo podatke shranjevali v lokalno spremenljivko za vsako spletno stran. Ta del je zato pohitрил shranjevanje predprocesiranja in indeksiranja podatkov, ampak posledično deluje le za eno stran naenkrat.

IV. REZULTATI

A. Podatkovna baza

Podatkovna baza ima v tabeli *IndexWord* 33370 unikatnih zapisov / besed. V tabeli *Posting* pa 346846 zapisov. Najpogostejše besede so: *proizvodnja*, *gl*, *spada*, *dejavnosti*, *d.o.o.*, *skupnost*, *krajevna*, *ministrstvo*, *šola*, *dejavnost*, *vir* (fig 3)... Najpogostejši dokumenti so: *evem.gov.si.371.html*, *podatki.gov.si.340.html*, *e-prostor.gov.si.57.html*... (fig 4).

B. Poizvedbe po podatkih z invertiranim indeksom

V priloženih posnetkih zaslona (fig. 5- 10) so rezultati poizvedb za besede "predelovalne dejavnosti", "trgovina", "social services", "hackaton", "izredni študent" in "vzdrževanje stiskalnic za iverne plošče". Z uporabo invertiranega indeksa iskanje ene besede traja ponavadi nekaj milisekund (maks.

	word	documentName	frequency
Filter	Filter	Filter	
1	proizvodnja	evem.gov.si/evem.gov.si.371.html	2266
2	gl	evem.gov.si/evem.gov.si.371.html	1668
3	spada	evem.gov.si/evem.gov.si.371.html	1338
4	dejavnosti	evem.gov.si/evem.gov.si.371.html	1287
5	d.o.o.	podatki.gov.si/podatki.gov.si.340.html	967
6	xsd	e-prostor.gov.si/e-prostor.gov.si....	927
7	skupnost	podatki.gov.si/podatki.gov.si.340.html	809
8	krajevna	podatki.gov.si/podatki.gov.si.340.html	754
9	ministrstvo	evem.gov.si/evem.gov.si.371.html	589
10	šola	podatki.gov.si/podatki.gov.si.340.html	582
11	dejavnost	evem.gov.si/evem.gov.si.371.html	545
12	vir	evem.gov.si/evem.gov.si.371.html	527
13	om	e-prostor.gov.si/e-prostor.gov.si....	511
14	ipd	evem.gov.si/evem.gov.si.371.html	488
15	družina	podatki.gov.si/podatki.gov.si.340.html	475

Fig. 3. Najpogostejše besede v podatkovni bazi.

```
1 SELECT      `documentName`,
2             COUNT(`documentName`) AS `freq`
3 FROM        `Posting`
4 GROUP BY    `documentName`
5 ORDER BY    `freq` DESC
6 LIMIT       10;
```

	documentName	freq
1	evem.gov.si/evem.gov.si.371.html	12245
2	podatki.gov.si/podatki.gov.si.340.html	6507
3	e-prostor.gov.si/e-prostor.gov.si....	1620
4	evem.gov.si/evem.gov.si.398.html	1481
5	evem.gov.si/evem.gov.si.651.html	1265
6	e-uprava.gov.si/e-uprava.gov.si....	1159
7	evem.gov.si/evem.gov.si.653.html	1007
8	e-uprava.gov.si/e-uprava.gov.si....	964
9	evem.gov.si/evem.gov.si.377.html	925
10	e-prostor.gov.si/e-prostor.gov.si....	901

Fig. 4. Najpogostejši dokumenti v podatkovni bazi.

pribl. 10ms). V redkih primerih se tudi zgodi da zaradi zagona baze ali inicializiranja, iskanje doseže okoli 100ms. Iskanje ene besede v vseh datotekah se meri v milisekundah in ne sekundah, kot je opisano v nadaljevanju v *razdelku C*. Izpis programa se za iskani niz izpiše združeno po dokumentih in s frekvenco vseh besed v poizvedbi. Poizvedbe z besedami, ki se tematsko nanašajo na spletne strani prikažejo zelo veliko rezultatov (npr. beseda *dejavnosti* se pojavi v zelo veliko dokumentih). Dodatno bi lahko izboljšali še izpis okolice iskanih besed (+3/-3 besede).

C. Poizvedbe po podatkih brez invertiranega indeksa

Na naši napravi je indeskiranje in iskanje določene ene besede vzelo od 100 do 300 milisekund. Večje število iskanih besed tudi vpliva na ta čas, najbolj pa je pomembna dolžina teksta strani in posledično število različnih besed. Iskanje ene besede v vseh datotekah je skupno vzelo okoli 260 sekund.

