Jaka Krajnc in Janez Škrlj

Poročilo grajenja indeksa in poizvedb

Seminarska naloga pri predmetu iskanje in ekstrakcija podatkov s spleta

MENTOR: doc. dr. Slavko Žitnik

V poročilu bomo predstavili grajenje preprostega indeksa besed in dokumentov ter poizvedovanje z uporabo tega indeksa.

1. Uvod

Na spletu se nahaja velika količina podatkov. Da lahko po vsej tej vsebini hitro in pregledno poizvedujemo, moramo zgraditi podatkovno strukturo imenovano indeks. Eden takih indeksov je invertni indeks. V njem hranimo besede kot ključe in pa mesta, kjer se besede pojavljajo, kot vrednosti. Na tak način lahko hitro izvemo, kje vse se določena beseda nahaja in izpišemo vsebino. Edina cena, ki jo moramo za to plačati je predprocesiranje besedila preden ga vstavimo v podatkovno bazo, oziroma indeks. Invertni indeks je najbolj razširjena podatkovna struktura iskalnikov [1]. V naši seminarski nalogi bomo iz večjega števila HTML dokumentov izvlekli njihovo vsebino, jo predprocesirali in shranili v indeks. Nato bomo zgrajeni indeks uporabili za poizvedovanje, oziroma iskanje želenih besed.

2. Procesiranje podatkov

Pred indeksiranjem smo morali pridobiti besedilo in spletnih strani. To smo dobili z uporabo funkcije handle knjižnice html2text, ki nam pretvori html dokument v čisti ascii tekst. Določili smo tudi naj ne upošteva povezav.

```
def extract_text_from_html(html):
    h = html2text.HTML2Text()
    h.ignore_links = True
    return h.handle(html)
```

Slika 1: Funkcija za pridobivanje besedila iz html dokumentov.

S funkcijo get_page_words(filepath) odpremo html dokument, izluščimo tekst in damo glavne besede v seznam.

```
def get_page_words(filepath):
    page_data = read_data(filepath)
    html_text = extract_text_from_html(page_data)
    all_tokens = word_tokenize(html_text)
    return all_tokens
```

Slika 2: Funkcija za izluščevanje glavnih besed.

S funkcijo normalize_word(word) iz besede (word) odstranimo ločila in jo pretvorimo v male črke. To funkcijo uporabljamo tako pri grajenju indeksa, kot tudi pri obdelavi iskane besede v primeru iskanja.

```
def normalize_word(word):
    return word.strip(string.punctuation).lower()
```

Slika 3: Funkcija za normalizacijo besed.

3. Indeksiranje podatkov

Za indeksiranje podatkov smo napisali funkcijo index_files. Najprej s funkcijo get_document_paths() pridobimo seznam vseh dokumentov, ki jih moramo poindeksirati. Za boljšo preglednost smo nastavili tudi spremenljivko processed, s katero bomo spremljali število že indeksiranih dokumentov. Nad vsakim dokumentom pokličemo funkcijo get_page_words, ki nam vrne seznam izluščenih besed dokumenta. Nad seznamom teh pa pokličemo funkcijo build_word_indexes, ki nam zgradi indekse.

```
def index_files(display_progress=False):
    to_index = get_document_paths()
    processed = 0
    all = len(to_index)
    for file_name in to_index:
        processed += 1
        if display_progress:
            print('In progress: {0} -> ({1} of {2})'.format(file_name, processed, all))
        relative_p = '{0}/{1}'.format(DOCUMENT_ROOT, file_name)
        words = get_page_words(relative_p)
        indexes = build_word_indexes(words)
        store_indexes(file_name, indexes)
```

Slika 4: Funkcija za indeksiranje podatkov, oziroma datotek.

Funkcija build_word_indexes najprej ustvari slovar. Vsako besedo podanega seznama besed najprej normaliziramo (odstranimo ločila in pretvorimo v male črke). Nato preverimo, če normalizirana beseda ni slučajo stopword ali pa ločilo. Če ni, jo dodamo v slovar. Ta slovar na koncu tudi vrnemo kot indeks.

```
def build_word_indexes(word_list):
    index = defaultdict(list)
    for i in range(len(word_list)):
        word = word_list[i]
        normalized = normalize_word(word)
        if normalized not in stop_words_slovene and normalized not in string.punctuation:
            index[normalized].append(i)
    return index
```

Slika 5: Funkcija za grajenje indeksov besed.

Vrnjen slovar besed (indeks) za posamezen dokument shranimo v bazo s funkcijo store_indexes. Za vstavljanje v bazo pokličemo funkcijo db.insert_posting.

```
def store_indexes(document_name, indexes):
    for key in indexes.keys():
        db.insert_posting(key, document_name, len(indexes[key]), ",".join([str(x) for x in indexes[key]]))
```

Slika 6: Funkcija za shranjevanje indeksov

Indeks vstavimo v bazo s pomočjo funkcije inser_word, ki v tabelo IndexWord vstavi besedo, če ta že ne obstaja v njej, in pa funkcije insert_posting, ki v tabelo Posting vstavi besedo, ime dokumenta, število ponovitev in pa indeks, kjer se beseda nahaja.

```
def insert_word(word, commit=False):
    try:
        res = cursor.execute("SELECT * FROM IndexWord WHERE  word='{0}'".format(word))
        # Do not insert if already exist
        if res.fetchone() is not None:
            return word
        cursor.execute("INSERT INTO IndexWord VALUES ('{0}')".format(word))
        if commit:
            conn.commit()
        return word
    except Exception as e:
        print('Exception (insert_word): ')
        print(e)


def insert_posting(word, document_name, frequency, indexes):
    try:
        insert_word(word)
        cursor.execute("INSERT INTO Posting VALUES ('{0}','{1}',{2},'{3}')".format(word, document_name, frequency, indexes))
        conn.commit()
    except Exception as e:
        print('Exception (insert_posting): ')
        print(e)
```

Slika 7: Funkciji za vstavljanje podatkov v bazo.

4. Poizvedovanje podatkov

Z invertnim indeksom

Za iskanje besed po invertnem indeksu smo uporabljali naslednjo funkcijo. Iskali smo osnovne besede:: “Predelovalne dejavnosti”, “trgovina”, “social services”. In pa dodatne: “Inšpekcijskem”, “preiskovalna komisija”, “komercialni”. Iskano besedo najprej normaliziramo, nato pa jo poiščemo v tabeli Posting.

```
def search(expression):
    # Start measuring time
    start = time.time()

    parts = [normalize_word(x) for x in expression.split(' ')]
    documents = db.search_words_in_index(parts)
    results = []
    for doc in documents:
        document = doc[1]
        freq = doc[2]
        text = build_search_snippet(document, doc[3].split(','))
        results.append([freq, document, text])

    print('Result for query {0}: \n'.format(expression))
    print('Result found in {0}ms'.format(round(time.time() - start, 2)))
    print(tabulate(results, headers=['Frequencies', 'Document', 'Snippet']))
```

Slika 8: Funkcija za iskanje besed po invertnem indeksu.

```
search("predelovalne dejavnosti")
search("trgovina")
search("social services")
#additional
search("inšpekcijskem")
search("preiskovalna komisija")
search("komercialni")
```

Slika 9: Besede za katerimi smo poizvedovali.

Opažanja

Poizvedovanje za besedami je bilo z invertnim indeksom zelo hitro, skorajda instantno. Kar pa je vzelo čas, pa je bilo grajenje snippetov, ker je bilo potrebno najdene dokumente še enkrat odpreti, prebrati in izpisati tri sosednje besede. V primeru, da smo izklopili izpis snippetov je bilo iskanje instantno, če pa so bili snippeti vklopljeni pa je vse skupaj trajalo dlje. To bi lahko rešili tako, da bi ob grajenju indeksa v tabelo vstavili nov stolpec, v katerega bi shranjevali snippete za posamezne besede, ali pa da nebi izpisovali snippetov za vsako pojavitev ampak le za nekaj na začetku. Ena od rešitev bi lahko bila tudi, da bi imeli ves čas dokumente prebrane v ramu, vendar bi za to potrebovali veliko rama, ki ga pa nimamo.

Brez invertnega indeksa

Zaporedno iskanje po dokumentih brez invertnega indeksa se nam je zdelo brezupno, saj moramo vedno odpreti in prebrati vse dokumente, kar pa traja celo večnost. Že pri iskanju z invertnim indeksom, je branje dokumentov in izpisovanje snippetov vzelo največ časa, pa smo bili zelo daleč od števila vseh dokumentov, za razliko od iskanja brez invertnega indeksa, kjer pa moramo preiskati vse. Tako iskanje bi bilo hitro, če bi imeli vse dokumente prebrane v ramu, vendar toliko rama nimamo in to bi potem tudi izničilo pomen invertnega indeksa.

5. Rezultati

5.1 Opis podatkovne baze

Število indeksiranih besed	48318
Besede in dokument z največjimi frekvencami	proizvodnja (2266), gl(1666), spada(1338) (evem.gov.si/evem.gov.si.371.html)

6. Zaključek

V tej seminarski nalogi smo se navadili izvleči vsebino iz večjega števila dokumentov. Videli smo, da lahko kljub velikemu številu dokumentov, še vedno hitro iščemo med njimi, če pred tem vsebino predprocesiramo in shranimo v pravo podatkovno strukturo. V našem primeru je bil to invertni indeks besed in dokumentov. Tega smo nato še uporabili za iskanje treh obveznih besednih zvez in še dodatnih treh po izbiri. Ker smo se želeli prepričati, da je iskanje z invertnim indeksom res hitrejše od iskanje brez njega, smo poizkusili po dokumentih iskati še zaporedno, kar pa je trajalo predolgo.

7. Literatura

[1] https://en.wikipedia.org/wiki/Inverted_index (Dostopano 28.5.2019)