# 3. seminarska naloga

Delfina Bariša, Žiga Simončič, Nejc Smrkolj Koželj

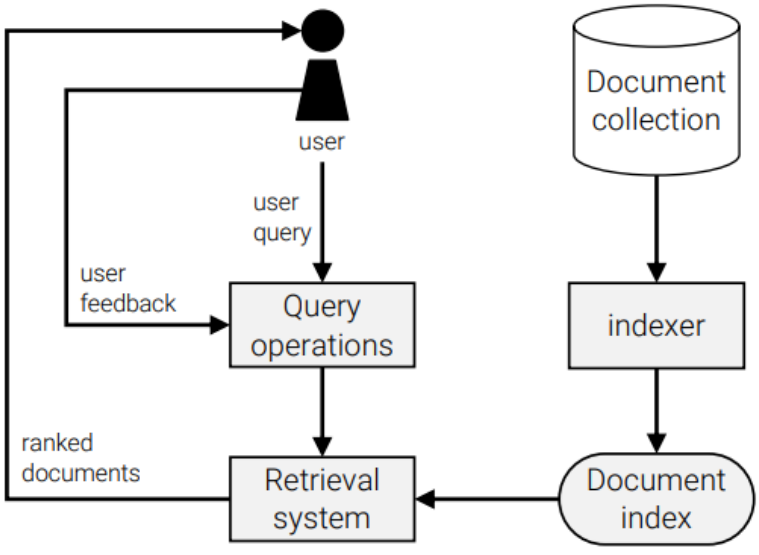
27. maj 2019

## 1 Uvod

V prvi seminarski nalogi smo se naučili kako pridobiti podatke iz spleta. V drugi kako iz spletne strani pridobiti podatke ali identificirati dele, ki nas zanimajo. V tej seminarski nalogi pa se bomo ukvarjali z informacijskim poizvedovanjem.

Arhitektura informacijskega poizvedovanje je prikazana na sliki 1 in je sestavljena iz naslednjih delov:

- poizvedbe,
- obdelave poizvedbe,
- indeksirnika in
- iskalnika.




Slika 1: Arhitektura informacijskega poizvedovanja

V seminarski nalogi smo najprej implementirali indeksirnik, ki je indeksiral podane spletne strani. Nato pa smo implementirali še iskalnik, ki je preko predhodno ustvarjenih indeksov

poiskal elemente, ki so najbolje ustrezali obdelanim podanim poizvedbam. Podrobnosti implementacije, bodo opisane v nadaljevanju.

# 2 Splošen pregled

Podanih smo imeli 1416 spletnih strani iz štirih različnih domen:

- "*e-prostor.gov.si*",
- "*e-uprava.gov.si*",
- "*evem.gov.si*"in
- "*podatki.gov.si*".

Naša naloga je bila, da iz podanih HTML datotek pridobimo besedilno vsebino, jo predprocesiramo in shranimo v indeks. V seminarski nagi smo implementirali inverzni indeks, ki je najbolj znan indeks v informacijskem poizvedovanju. Nato smo zgrajeni indeks uporabili za poizvedovanje. Implementacijo smo preizkusili s podanimi poizvedbami:

- "*predelovalne dejavnosti*",
- "*trgovina*",
- "*social services*"in
- tremi dodatnimi poizvedbami, ki vsebujemo od ene do pet besed in vsaj enim najdenim rezultatom.

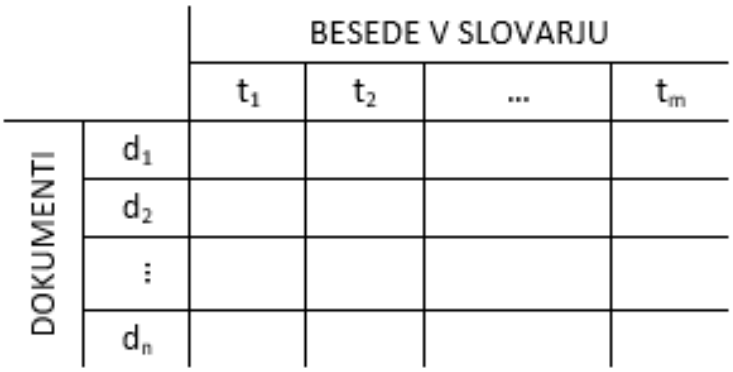
Za implementacijo indeksa so uporabili SQLite podatkovno bazo. Več o tem bomo povedali v poglavju 3.2.1. Podrobnosti implementacije indeksa bomo opisali v poglavju 3.

Da bi videli, kako indeksiranje pohitri iskanje, smo implementirati še iskanje brez indeksa in SQLite podatkovne baze (naivni pristop). Podrobna implementacija takšnega iskanja smo opisali v poglavju 4.

Primerjave porabljenega čas za iskanje z indeksom in iskanje z naivnim pristopom za zgoraj naštete poizvedbe smo opisali v poglavju 5.

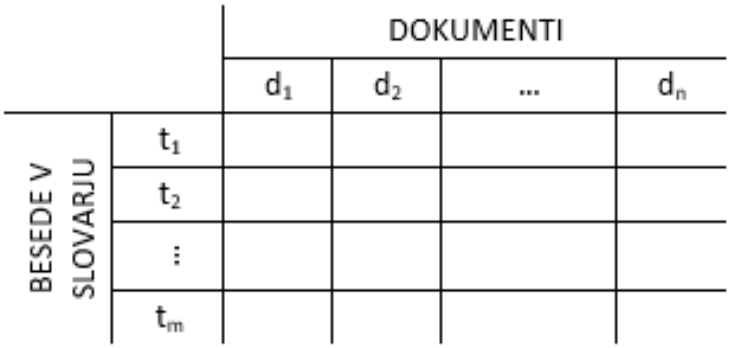
# 3 Implementacija z inverznim indeksom

Preden pojasnimo implementacijo, bomo razložili teoretične osnove. Indeksiranje uporabljamo za označevanje v katerih dokumentih se nahajajo katere besede iz slovarja vseh besed. Slika 2 prikazuje indeksiranje po navadnem indeksu.




Slika 2: Navadni indeks

Torej pri navadnem indeksu vemo, katere besede se nahajajo v posameznem dokumentu. Da bi poenostavili iskanje obrnemo tabelo in tako vemo v katerih dokumentih se pojavi posamezna beseda. Zaradi obrata takšno indeksiranje, imenujemo indeksiranje z inverznim indeksom. Slednje prikazuje slika 3.

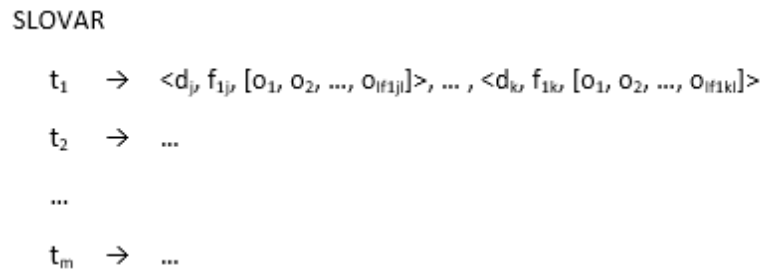



Slika 3: Inverzni indeks

Da se iskanje še pohitrili, v slovarju vsako besedo povežemo samo s tistimi dokumenti, v katerih se beseda nahaja in dodali še nekaj podatkov, kar skupaj imenujemo metapodatki. Tako za vsako povezavo hranimo sledeče metapodatke:

- $d_j$ : indeks dokumenta v katerem se nahaja beseda,
- $f_{i_j}$ : frekvenca ponovitve besede $t_i$ v dokumentu $d_j$ in
- $o_1$, $o_2$, ... , $o_{|f_{ij}|}$ : seznam indeksov na katerih se nahaja beseda $t_i$ v dokumentu $d_j$.

Tako dobimo slovar, ki vsebuje samo dokumente, v katerih se beseda $t_i$ nahaja. Takšen slovar prikazuje slika 4.




Slika 4: Slovar pri indeksiranju z inverznim indeksom

Sedaj ko poznamo teoretične osnove, lahko preidemo na implementacijo seminarske naloge. Implementacijo smo kot je bilo zahtevano razdelili na tri korake:

- procesiranje podatkov (poglavje 3.1),
- indeksiranje podatkov (poglavje 3.2) in
- iskanje podatkov (poglavje 3.3).

Vsak korak bo podrobneje pojasnjen v podpoglavjih.

### 3.1 Procesiranje podatkov

Pred samim indeksiranjem je potrebno iz vsake HTML datoteke pridobiti besedilne podatke. Slednje smo storili s pomočjo knjižnice *BeautifulSoup*, s katero smo pridobili besedilni del telesa HTML datoteke. Posebej smo morali obravnavati html oznako script, ki smo jo s pomočjo funkcij *findAll()* in *extract()* odstranili.

Ko smo pridobili podatkovni del HTML strani smo ustvarili dva seznama. Prvega poimenujemo *S1*, drugega pa *S2*.

Seznam *S1* je namenjen izračunu frekvenc pojavitve in indeksov pojavitve posamezne besede. V tem delu smo si veliko pomagali z knjižnico *string*. Ustvarili smo ga tako, da smo najprej s pomočjo metodo *strip()*, *lower()* in *space* odstranili presledke, spremenil vse črke v male tiskane in trenutno besedilo, ki je predstavljalo en dolgi niz spremenili v seznam besed oziroma nizov. Nato smo se še sprehodili skozi seznam *S1* in s pomočjo metode *replace*, odstranili vsa ločila.

Seznam *S2* bo vseboval smo "pomembne"besede. Ustvarili smo ga tako, da smo najprej z metodo *strip()* iz besedila odstranili vse nepotrebne presledke in z metodo *lower()* vse črke spremenili v male tiskane. Nato smo s pomočjo knjižnice *NLTK* in njegove meto *word_tokenize* trenutno besedilo, spremenili v seznam nizov. Sledilo je odstranjevanje ločil tako imenovanih štopword-ov. Štopword-i so besede, ki so pogoste, a so odveč in jih je potrebno odstraniti. Te so običajno vezniki, besede krajše od treh črk in ostale besede kateri pomen ni ključen. Na koncu smo se sprehodili skoz nastali seznam *S2* in odstranili prazne nize.

Ko smo dokončali seznama *S1* in *S2* smo prešli na indeksiranje podatkov.

### 3.2 Indeksiranje podatkov

Po tem, ko smo ustvarili oba seznama, smo seznam *S2* spremenili v množico. Poimenujmo jo *M2*. Nato smo se sprehodili skozi vsako besedo iz množice *M2* in zanjo naredili:

1. Pridobili frekvenco besede v HTML dokumentu.
   To smo storili tako, da s pomočjo metode *count()* preštejemo število pojavitev besede v seznamu *S1*.

2. Poiskali indekse na katerih se pojavi beseda v HTML dokumentu in ustvarili seznam iz njih.
   To smo storili tako, da smo se sprehodili skozi seznam *S1* in preverili ali se niz iz seznama ujema z našo besedo ter sproti ustvarjali seznam.

3. Ustvari terko z besedo, potjo do datoteke in metapodatki:
   (beseda, pot, frekvenca, seznam indeksov).

4. Terko z metapodatki dodali v podatkovno bazo.

Za ustvarjanje indeksa, smo potrebovali približno eno uro. Več o podatkovni bazi in ustvarjenem indeksom bomo povedli v naslednjem podpoglavju. Sedaj, ko smo ustvarili indeks, lahko začnemo s poizvedovanjem.

#### 3.2.1 Opis podatkovne baze

Na sliki 5 je prikazana shema, ki smo jo uporabili kot podatkovno strukturo za inverzni indeks.

```
CREATE TABLE IndexWord (
  word TEXT PRIMARY KEY
);

CREATE TABLE Posting (
  word TEXT NOT NULL,
  documentName TEXT NOT NULL,
  frequency INTEGER NOT NULL,
  indexes TEXT NOT NULL,
  PRIMARY KEY(word, documentName),
  FOREIGN KEY (word) REFERENCES IndexWord(word)
);
```

Slika 5: Primer izpisa

Po končanem indeksiranju je bilo:

- v tabeli *IndexWord* 48701 indeksiranih besed in v
- v tabeli *Posting* pa 392504 vrstic.

Tabela 1 prikazuje 10 besed in dokumentov z najvišjo frekvenco v tabeli *Posting*:

Tabela 1: Besede in dokumenti z najvišjo frekvenco

| Beseda | Dokument | frekvenca |
|---|---|---|
| ”proizvodnja” | data/evem.gov.si/evem.gov.si.371.html | 2266 |
| ”gl” | data/evem.gov.si/evem.gov.si.371.html | 1668 |
| špada” | data/evem.gov.si/evem.gov.si.371.html | 1336 |
| ”dejavnosti” | data/evem.gov.si/evem.gov.si.371.html | 1283 |
| ”doo” | data/podatki.gov.si/podatki.gov.si.340.html | 1056 |
| škupnost” | data/podatki.gov.si/podatki.gov.si.340.html | 809 |
| ”krajevna” | data/podatki.gov.si/podatki.gov.si.340.html | 754 |
| ”ministrstvo” | data/evem.gov.si/evem.gov.si.371.html | 589 |
| ”šola” | data/podatki.gov.si/podatki.gov.si.340.html | 582 |
| ”ipd” | data/evem.gov.si/evem.gov.si.371.html | 562 |

Iz tabele je razvidno, da se v datotekah *data/evem.gov.si/evem.gov.si.371.html* in *data/podatki.gov.si/podatki.gov.si.340.html* pojavi veliko enakih besed. Posledično je ravno pri njiju največja frekvenca besed, ki so prikazane v tabeli 1.

### 3.3 Iskanje podatkov

V tretjem koraku implementacije smo morali uporabiti že prej zgrajen indeks in vrniti rezultat iskanja za podano poizvedbo. Za večbesedne poizvedbe, smo morali združiti rezultate iskanja. Rezultat smo morali vrniti v padajočem vrstnem redu po vsoti frekvenc. Ob frekvenci pojavitve in dokumenta v katerem se nahajajo besede iz poizvedbe, smo morali izpisati še del besedila okrog iskanih besede iz poizvedbe. Primer izhoda prikazuje slika 6.

```
Frequencies Document                            Snippet
----------- ----------------------------------- ------------------------------------------------------------
4           evem.gov.si/evem.gov.si.666.html    Sistem SPOT je eden boljši ... dosedanje delovanje SPOT ni zadovoljivo za ... je bila zaključena. Sistem ni deloval dobro ...
1           e-uprava.gov.si/e-uprava.gov.si.42.html  ... ministrstvo je nadgradilo sistem za učinkovitejšo uporabo.
```

Slika 6: Primer izpisa

Opišimo podrobno implementacijo. Najprej smo ustvarili seznam L v katerega smo shranjevali vse rezultate. Za vsako besedo iz poizvedbe smo storili naslednje:

1. Odstranili vsa ločila iz besede.

2. Ustvarili poizvedbo na bazo, da smo dobili vse rezultate iz tabele *Posting*, ki vsebujejo besedo iz poizvedbe.

3. Ustvarili prazen seznam S, v katerega smo shranili rezultate

4. Vsak zadetek iz baze je terka oblike:
   (beseda, HTML datoteka, frekvenca, seznam indeksov)
   Za vsak zadetek iz baze smo storili:

   (a) Odprli HTML datoteko in pridobili besedilni del telesa HTML datoteke.

   (b) Besedilo smo s pomočjo metod *strip()* in *split()* odstranili presledke in ustvarili seznam besed.

   (c) Za vsak indeks i iz seznama indeksov smo storili:

      i. Pridobimo besedo in določeno število besede okoli nje ter odspredaj dodamo "... ".

      ii. Dodamo nov rezultat v seznam S.

      iii. Ko obdelamo vse indekse vrnemo frekvenco, dokument in seznam S

   (d) Če je v L že obstajalo polje za dokument, ki smo ga vrnili, smo v to polje kot seznam dodali [frekvenco, dokument, seznam s], sicer pa smo ustvarili novo polje v tabeli L.

5. Nato smo ustvarili nov seznam M in se sprehodili skozi seznam L in za vsak notranji seznam storili:

   (a) Sešteli vse frekvence pojavitve besed v dokumentu.

   (b) Združili vse besede in dele okoli njih.

   (c) V seznam M dodali terko (frekvenca, dokument, besede z okolico)

6. Na koncu smo seznam M sortirali po frekvenci in ga izpisali na standardni izhod.

# 4 Implementacija brez inverznega indeksa

Implementaciji brez inverznega indeksa, pravimo tudi naivna implementacija, saj ima veliko časovno zahtevnost. Od prejšnje implementacije se razlikuje v tem, da ne uporablja podatkovne baze in indeksa, ampak za vsako poizvedbo:

1. Ustvarimo seznam L v katerega bomo shranjevali vse rezultate.

2. Odpremo vse HTML datoteke in pridobimo besedilni del telesa vsebine ter odstranimo HTML oznako *script*.

3. Iz pridobljenega besedilnega dela ustvarimo dva seznama S1 in S2:

   (a) Seznam S1 ustvarimo, tako da iz pridobljenega besedilnega dela odstranimo presledke, črke spremenimo v male tiskane in ustvarimo seznam besede ter v v njemu odstranimo ločila.

(b) Seznam S2 ustvarimo tako, da iz pridobljenega besedilnega dela odstranimo presledke, črke spremenimo v male tiskane in ustvarimo seznam besede ter v v njemu odstranimo ločila, štopword-e in prazne besede.

4. Za vsako besedo v seznamu:

   (a) Preverimo, ali je enaka besedi iz poizvedbe. Če je:

      i. Poiščemo indekse drugih pojavitve besede v sezamu.

      ii. Za vsako pojavitev besede pridobimo besedo in določeno število besede okoli nje ter odspredaj dodamo "... ".

      iii. Če v L že obstaja polje za dokument, ki smo ga vrnili, v to polje kot seznam dodamo [frekvenco, dokument, seznam s], sicer ustvarimo novo polje v tabeli L.

5. Nato ustvarimo nov seznam M in se sprehodimo skozi seznam L ter za vsak notranji seznam storimo:

   (a) Seštejemo vse frekvence pojavitve besed v dokumentu.

   (b) Združimo vse besede in dele okoli njih.

   (c) V seznam M dodamo terko (frekvenca, dokument, besede z okolico)

6. Na koncu seznam M sortiramo po frekvenci in ga izpišemo na standardni izhod.

Sedaj, ko smo opisali obe implementaciji, sledi še njuna primerjava.

# 5 Primerjava implementacije z in brez indeksa

Za že podane poizvedbe, je primerjava časov izvajanja prikazana v tabeli 2.

Tabela 2: Primerjava implementacij z indeksiranjem in brez

| Poizvedba | Implementacija z indeksom | Naivna implementacija |
|---|---|---|
| "*predelovalne dejavnosti*" | 17.105801582336426 s | 102.0218288898468 s |
| "*trgovina*" | 4.199073314666748 s | 52.325023889541626 s |
| "*social services*" | 1.0172810554504395 s | 52.43954372406006 s |

Iz tabele 2 je razvidno, da je časovna zahtevnost naivnega pristopa tudi do več kot 10 krat večja od implementacije z indeksom. Sedaj pa preverimo, kako je s časovno zahtevnostjo, če bi bile v poizvedbi več kot dve besedi.

Izbrali naslednje poizvedbe:

- "*vodenje poslovnih knjig*"
- "*geodetska uprava rs*"

- "*statistični urad republike slovenije*"

Primerjavo časov izvajanja za zgornje poizvedbe, z in brez indeksa, prikazuje tabela 3.

Tabela 3: Primerjava implementacij z indeksiranjem in brez za poljubne poizvedbe

| Poizvedba | Implementacija z indeksom | Naivna implementacija |
|---|---|---|
| "*vodenje poslovnih knjig*" | 39.87649750709534 s | 102.48178553581238 s |
| "*geodetska uprava rs*" | 29.186506748199463 s | 101.03641557693481 s |
| "*statistični urad republike slovenije*" | 96.77972841262817 s | 158.2153856754303 s |

Opazimo, da se časa izvajanja pri daljših poizvedbah malo manj razlikujeta, a je razlika med njima še vedno velika. Sklepamo, da je relativno visok čas pri iskanju z indeksom zaradi pogosto uporabljenih besed v dokumentih (npr. "republika slovenija" se verjetno pojavi v večini dokumentov, lahko tudi večkrat, saj gre za vladne strani). Pokazali smo, da je časovna zahtevnost naivne impelemnatcije dosti večja od tiste z inverznim indeksom.

# 6 Možne izboljšave

Trenutna implementacija z inverznim indeksom pri večbesednih poizvedbah preverja vsako besedo poizvedbe posebej. Pri združevanju rezultatov pa ne preverja, ali se besede v dokumentu skupaj in v enakem vrstnem redu. To izboljšavo bi lahko implementirali tako, da bi enako kot sedaj preverili vsako besedo posebej. Nato pa bi pogledali v katerih dokumentih se nahajajo vse besede iz poizvedbe in s pomočjo tabele indeksov preverili, ali se nahajajo v dokumentu skupaj in v enakem vrstnem redu, kot v poizvedbi.

# 7 Zaključek

V poročilu smo opisali našo implementacijo informacijskega poizvedovanja z in brez indeksa. Naredili smo splošen pregled funkcionalnosti nato pa smo opisali podrobnosti. Poročilo smo zaključili s predstavitvijo in primerjavo rezultatov ter možnimi izboljšavami.