

Upravljanje digitalnim dokumentima

Dragan Ivanović
Branko Milosavljević

Copyright ©2014 by Dragan Ivanović and Branko
Milosavljević

All rights reserved.

ISBN ...

Predgovor

Živimo u digitalnom dobu. Dobu u kojem se u većini slučaja ne postavlja pitanje da li u digitalnom obliku postoji korisni materijal, nego kako takav materijal pronaći u mnoštvu digitalnih dokumenata. Razvojem informaciono-komunikacionih tehnologija i razvojem veba stvorili su se uslovi da značajan deo svetske populacije ima pristup potrebnom hardveru i softveru za kreiranje i distribuiranje digitalnih dokumenata. Pojavljuju se i novi termini u srpskom jeziku koji su nastali kao posledica digitalnog doba u kojem živimo. Da li ste čuli za termine guglovanje, lajkovanje, tvitovanje, šerovanje, selfi? Ako ste čuli za ove termine onda koristite tehnologije digitalnog doba.

Ova knjiga se bavi upravljanjem digitalnim dokumentima. Na početku knjige u prvom poglavlju je obrađen pojam digitalnog dokumenta i njemu pridruženih metapodataka. Drugo i treće poglavlje opisuju sisteme koji skladište digitalne dokumente i funkcionalnosti koje ovi sistemi treba da implementiraju, kao i standardizovane formate i protokole koji se koriste u ovim sistemima. Jedna od neizostavnih funkcionalnosti ovih sistema je pretraga kolekcije dokumenata i ova funkcionalnost je centralna tema ove knjige. Osnovni pojmovi discipline pronalaženja informacija su opisani u četvrtom poglavlju ove knjige. Peto poglavlje se bavi pretragama kolekcije tekstualnih dokumenata, šesto poglavlje pretragama veba, a sedmo poglavlje pretragama kolekcija multimedijalnih dokumenata. Evaluacija performanse sistema za pretragu kolekcije i unapređenje sistema za pretragu su teme osmog poglavlja.

Najvažniji pojmovi u ovoj knjizi su istaknuti crvenom bojom i javljaju se u indeksu pojmove na kraju knjige. Plavom bojom je istaknut tekst koji je značajan za razumevanje sadržaja knjige i primeri koji se navode u knjizi.

Knjiga je namenjena studentima Fakulteta tehničkih nauka u Novom Sadu kao osnovni udžbenik za predmet "Upravljanje digitalnim dokumentima" na master akademskim studijama. Takođe, knjiga sadrži i potrebnu materiju za predmet "Tehnologije i platforme za upravljanje elektronskim sadržajima i dokumentima" na osnovnim strukovnim studijama. Naravno, knjiga može biti korisna i drugim čitaocima koji su zainteresovani za oblast pronalaženja informacija.

Ova knjiga je nastala kao rezultat višegodišnjeg rada sa studentima Fakulteta tehničkih nauka, kao i višegodišnje uspešne saradnje sa kolegama sa naučnih institucija u zemlji i inostranstvu. Sva svoja znanja i iskustva smo nadamo se uspeli da pretočimo na jednostavan, prijemčiv, razumljiv, zanimljiv način u ovu knjigu čijem se nastanku radujemo.

Ovo je drugo izdanje ove knjige u kojem su ispravljeni nedostaci uočeni u prvom izdanju. Zahvaljujemo se kolegi Milanu Stojkovu na uočenim nedostacima.

U Novom Sadu,
2015. godine

Autori

Sadržaj

1	Digitalni dokument	1
1.1	Metapodaci	4
1.1.1	Veza dokumenata i metapodataka	5
1.2	Životni ciklus	7
2	Sistemi za upravljanje dokumentima	17
2.1	Funkcionalnosti sistema	19
2.2	Primeri sistema	23
2.2.1	Poslovni sistem za upravljanje sadržajem .	24
2.2.2	Digitalne biblioteke	25
2.2.3	Institucionalni repozitorijumi	27
3	Standardi u sistemima digitalnih dokumenata	31
3.1	Formati metapodataka	33
3.1.1	MARC 21	34
3.1.2	Dublin core	36
3.1.3	ETD-MS	39
3.2	Protokoli	41
3.2.1	Razmena podataka	41
3.2.2	Pretraživanje udaljene kolekcije	44
4	Pronalaženje informacija	49

4.1	Istorija razvoja	51
4.2	Razvoj sistema za pretragu	54
5	Pretraga tekstualnih dokumenata	61
5.1	Pretprocesiranje teksta	63
5.2	Bulov model pretraživanja	74
5.2.1	Invertovani indeks	76
5.2.2	Procesiranje upita	83
5.2.3	Pointeri za preskakanje	86
5.2.4	Upiti fraze	87
5.3	Vektorski model pretraživanja	92
5.3.1	Ocena relevantnosti	93
5.3.2	Frekvencija terma	94
5.3.3	Frekvencija dokumenta	95
5.3.4	Frekvencija kolekcije	97
5.3.5	tf-idf	97
5.3.6	Težinska matrica	98
5.4	Pretraga strukturiranih tekstualnih dokumenata .	106
5.4.1	Pretraga po parametrima i zonama	107
5.4.2	Pretraga tekstualnih sadržaja složenih struk-tura	109
6	Pretraga veba	127
6.1	Veb pretraživači	135
6.2	Veb <i>crawling</i>	138
6.3	Analiza linkova	144
6.4	Search engine optimization	152
7	Pretraga multimedijalnih dokumenata	159
7.1	Slika	163
7.2	Zvuk	174

7.3 Video	177
8 Performanse pretraživanja	183
8.1 Relevantnost	185
8.2 Evaluacija performansi	193
8.3 Unapređenje sistema	198
8.3.1 Rezultati pretrage	199
8.3.2 Klasifikacija	202
8.3.3 Klasterovanje	204
8.3.4 <i>Relevance feedback</i>	206
8.3.5 Globalno proširenje upita	214
Bibliografija	221
Indeks korišćenih pojmova	229

Glava 1

Digitalni dokument

Ovo poglavlje ima za cilj da definiše pojam digitalnog dokumenta i da odgovori na sledeća pitanja:

- Koje su razlike između tradicionalnog papirnog dokumenta i digitalnog dokumenta i kako je razvoj informaciono-komunikacionih tehnologija uticao na količinu digitalnih sadržaja?
- Šta su to metapodataci za digitalni dokument, zašto su oni značajni i koji su mogući izvori za nastajanje metapodataka?
- Kroz koje faze jedan digitalni dokument može prolaziti u svom životnom ciklusu?

Polazna osnova za ovo poglavlje bio je prvi deo standarda za informatičko uređenje sistema za upravljanje dokumentima "IEC 82045-1:2001 Document Management: Part 1 - Principles and Methods" [1]. Takođe, deo informacija prezentovanih u ovom poglavlju su preuzeti iz izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Kada kažemo **dokument** možemo misliti na dve stvari:

1. tradicionalni papirni dokument,
2. digitalni dokument.

Glavna prednost **tradicionalnog papirnog dokumenta** je što nije potreban uređaj (eng. *hardware*) za čitanje ili menjanje ovog dokumenta. Papirni dokument čovek (obično službenik) čita, obrađuje, arhivira, itd. Najčešće ga popunjava klijent upotrebom hemijske olovke. Dakle, za rad sa tradicionalnim papirnim dokumentima nije nam nužno neophodan ni hardver, ni softver, a ni poznavanje rada na računaru ili nekom drugom uređaju od strane korisnika. Papirni dokument se može kreirati na potpuno praznom papiru upotrebom olovke ili pisaće maštine. Danas većina papirnih dokumenata koje se koriste na raznim službama, u administrativnim poslovima, u banci i na drugim mestima su zapravo inicijalno kreirani kao digitalni dokumenti pa zatim oštampani i na taj način je dobijen dokument ili formular. U formularima su ostavljene neke beline u kojima se očekuje da neko unese potrebne podatke i na taj način završi kreiranje papirnog dokumenta. Popunjeni papirni dokument se najčešće predaje službeniku koji neke (ili sve) podatke sa tog formulara upotrebom računara unosi u neki informacioni sistem, u nekim sistemima se čak i skenira popunjeni dokument i takav skenirani dokument unosi u informacioni sistem. Iako ovakav način rada ima nedostataka on će u nekim sistemima ostati još neko vreme na snazi zbog toga što ne zahteva postojanje opreme dostupne klijentu i što ne zahteva računarsku pismenost klijenta. Ako želimo da imamo neko izveštavanje i pretraživanje ovako unetih podataka neophodno je postojanje informacionog sistema, a unos podataka u ovaj sistem obično obavlja službenik koji je zaposlen baš sa tom namenom.

Digitalni dokument je zapravo računarski obrađena informacija kojom se rukuje kao osnovnom jedinicom obrade. Nepobitna je činjenica da danas živimo u digitalnom dobu u kojem smo okruženi uređajima koji stvaraju i prenose digitalne informacije. Razvoj informaciono-komunikacionih tehnologija - IKT

(eng. *information and communication technologies*) je zaslužan za ovakvo stanje stvari. Lični računari su danas dostupni većini stanovništva, a pored njih i razni drugi uređaji koji su opremljeni potrebnim hardverom i softverom koji omogućuje kreiranje i konzumiranje različitih digitalnih dokumenata kao što su:

- tekstualni digitalni dokumenti, npr. tekstualni opisi ili poruke,
- grafički dokumenti, npr. slike, crteži, dijagrami, grafikoni,
- strukturirani dokumenti, npr. HTML i XML+XLink dokumenti,
- mediji sa vremenskom dimenzijom kao što su zvuk i video,
- kompozitni multimedijalni dokumenti sastavljeni od teksta, slike, zvuka, ili videa.

Količina postojećih digitalnih dokumenata je dodatno povećana razvojem veba (WWW, eng. *World Wide Web*) i popularnih veb aplikacija kao što su Facebook, Twitter, Instagram, YouTube, itd. Razvijeni su algoritmi za digitalno potpisivanje dokumenata, kao i softveri koji podržavaju digitalno potpisivanje. U razvijenim zemljama postoji i pravna regulativa za digitalno potpisivanje dokumenata. Pored dostupnosti računara i drugih uređaja koji mogu da proizvode digitalne sadržaje, značajan broj svetske populacije je računarski pismen i zna da koristi savremene alate za kreiranje i konzumiranje digitalnih dokumenata. Sa stanovišta stepena razvijenosti IKT nismo daleko od mogućnosti da se svaki korak našeg života digitalizuje o čemu govori i naučna studija *The man with the perfect memory* [2]. U ovoj studiji digitalizovan je značajan deo života *Gordona Bell-a* od 2001. do 2005. godine upotrebljom male kamere koju je nosio na ramenu i ostalih savremenih tehnologija. Snimljeno je 1.300 video zapisa, 5.067 audio zapisa (uključujući i konverzacije), blizu 100.000 *email-ova*, 67.000 veb stranica koje je posetio, upotrebljom GPS (eng. *Global Positioning System*) čipa snimana je i njegova lokacija, snimani su i podaci vezani za njegovo zdravlje (potrošene kalorije, otkucaji srca). Procena naučnika je da je potrebno oko jednog terabajta memorije da se snimi sve osim videa za 83 godine života, a

da bi bilo potrebno oko 200 terabajta ako bismo snimali i video kompletног života jednog čoveka tokom 83 godine.

1.1 Metapodaci

Sadržaj digitalnog dokumenta predstavlja podatke koji se prenose onome ko konzumira digitalni sadržaj. Taj sadržaj može biti tekst koji je formatiran za prikaz, a može biti i strukturiran. Sa druge strane taj sadržaj može biti audio ili video zapis kojim se prenosi neka poruka onome ko konzumira sam sadržaj. Često je sam sadržaj kombinacija nekoliko medija čime se dobija multimedijalni sadržaj sa namenom da se što bolje prenese određena poruka konzumentima. Ne postoji medij koji je najbolji za prenos informacija konzumentima jer to zavisi od namene poruke koja se želi preneti i od ciljne grupe kojoj se želi preneti poruka. Kako odabrat odgovarajući digitalni medij za prenos određene poruke, kojim alatima i na koji način kreirati odgovarajući digitalni sadržaj prevazilazi namenu ove knjige i neće biti njena tema. Pored samog sadržaja digitalnog dokumenta koji predstavlja podatke, odnosno poruku koja se želi preneti konzumentima, digitalni dokument može sadržati i metapodatke. **Metapodaci** su podaci o dokumentu, odnosno podaci o podacima. Primeri metapodataka za tekstualni digitalni dokument su:

- autor,
- naslov,
- datum nastanka,
- ključne reči.

Primeri metapodataka za digitalnu fotografiju su:

- autor,
- datum i vreme fotografisanja,
- mesto fotografisanja,
- podešavanje aparata,

- objekti prikazani na slici.

Da li mogu postojati i metapodaci za metapodatke? Formalno mogu postojati ako to u određenim situacijama ima smisla. Na primer, *IEEE Learning Object Metadata* format (IEEE LOM) koji se koristi za opis sadržaja koji su namenjeni za učenje sadrži metapodatke za metapodatke: ko je uneo metapodatke, kada su uneti metapodaci, da li su menjani metapodaci, zašto su menjani, itd [3].

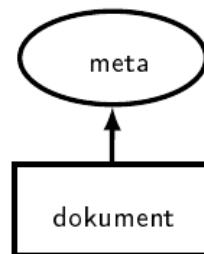
Zašto postoje metapodaci, koja je njihova namena? Metapodaci se koriste da se bolje organizuju kolekcije dokumenata, da se pregled dokumenata prilagodi korisnicima. Takođe, metapodaci mogu služiti i za klasifikaciju dokumenata. Na kraju, metapodaci mogu značajno unaprediti kvalitet i mogućnosti pretrage kolekcije digitalnih dokumenata. Mogu se koristiti i za filterisanje i sortiranje rezultata pretrage.

Mogu postojati različiti izvori iz kojih su nastali metapodaci za digitalni dokument. Sam korisnik koji je kreator digitalnog dokumenta može unositi metapodatke. Softver pomoću koga se kreira digitalni dokument ili operativni sistem unutar koga se nalazi softver za kreiranje digitalnog dokumenta mogu dodavati određene metapodatke. Na primer, ko je kreirao dokument, kada je kreiran dokument, kojim alatom je kreiran dokument, koja je rezolucija slike, itd. Poslovni proces gde se dokument koristi kao nosilac informacija između aktivnosti može biti izvor nastajanja i promena metapodataka. Takođe, može postojati neka opšta baza znanja u organizaciji u kojoj se odvija poslovni proces iz koje se preuzimaju određeni metapodaci. Na primer, naziv firme odgovorne za kreiranje dokumenta, mesto i država gde je kreiran dokument, itd. Faze životnog ciklusa kroz koji prolazi digitalni dokument mogu uzrokovati nastajanje i promene metapodataka.

1.1.1 Veza dokumenata i metapodataka

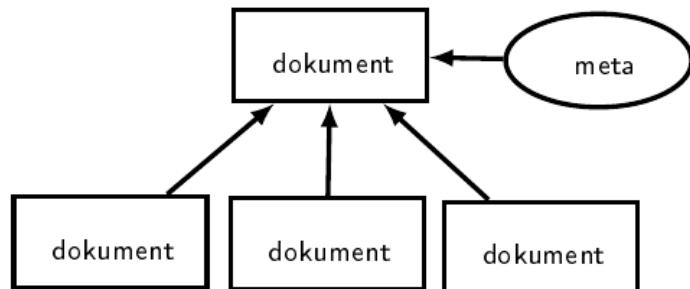
Digitalni dokumenti mogu biti različite složenosti što može uticati i na organizaciju metapodataka. Najjednostavnija je situacija kada imamo **pojedinačni dokument** kao elementarni

oblik informacija. Ovakav dokument može imati pridružene metapodatke koji opisuju njegov sadržaj ili druge karakteristike (slika 1.1).



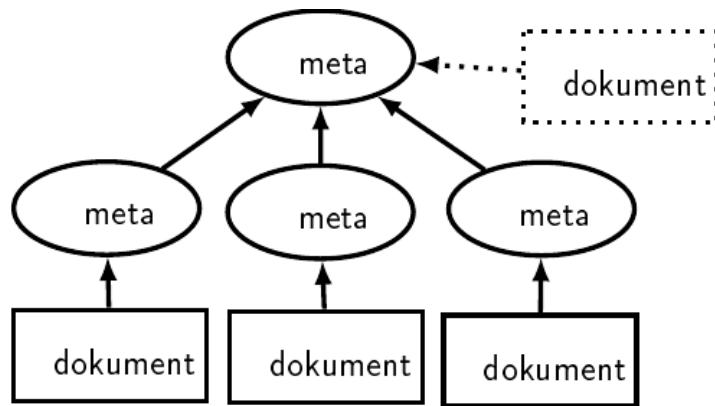
Slika 1.1: Pojedinačni dokument

Digitalni dokument može biti i **složen**, odnosno predstavljati kompoziciju više dokumenata različitih tipova. Na primer, složeni dokument može biti tehnička specifikacija koja se sastoji od tekstualnih fragmenata, crteža, i dijagrama. U ovom slučaju metapodaci se dodeljuju složenom dokumentu kao celini (slika 1.2).



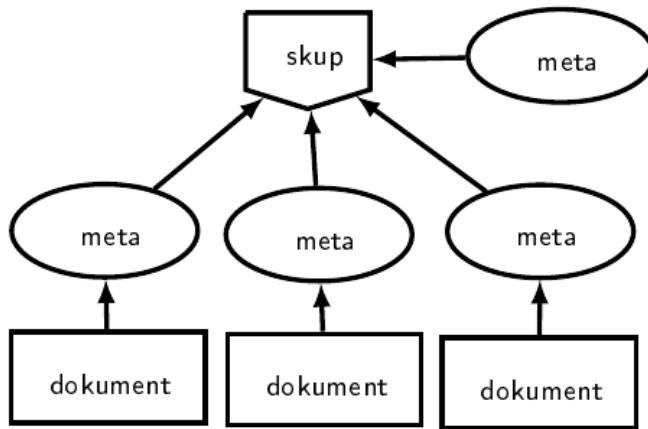
Slika 1.2: Složeni dokument

Postoje i skupovi samostalnih dokumenata od kojih svaki ima svoje metapodatke. Može se kreirati agregacija ovih samostalnih dokumenata koja ima sopstvene metapodatke. Ova **agregacija dokumenata** može, ali ne mora, da poseduje sopstveni dokument (slika 1.3).



Slika 1.3: Agregacija dokumenata

Takođe, postoje i situacije kada **skup samostalnih dokumenata** ima svoje metapodatke koji opisuju svrhu skupa, kao i sadržanih dokumenata u skupu (slika 1.4).



Slika 1.4: Skup dokumenata

1.2 Životni ciklus

Digitalni dokument može prolaziti kroz različite faze svog **životnog ciklusa**. Uzimajući u obzir današnji stepen razvoja IKT

upotreboom dostupnog hardvera i softvera korisnik može kreirati digitalni dokument i distribuirati ga putem Interneta drugim korisnicima na korišćenje. Nekada kreiranje i korišćenje digitalnog dokumenta može biti znatno složenije. Iza kreiranog dokumenta može "stajati" više odgovornih ljudi pa čak i država. Može postojati složen poslovni proces koji uključuje više ljudi i organizacija u kreiranje jednog dokumenta. Konzumenti digitalnog dokumenta mogu biti milioni ljudi. Dokument se može koristiti za regulisanje saobraćajnih propisa ili definisanje određenih zakona. Greške u ovakvim dokumentima mogu dovesti do nemogućnosti kažnjavanja prekršioca zakona ili do nekih katastrofalnih posledica koje čak mogu dovesti do ljudskih žrtava. Zamislite da postoji greška u digitalnom dokumentu koji opisuje proceduru poletanja aviona i da ta greška može dovesti do sudara dva aviona čiji su piloti uradili onako kako piše u dokumentu, tj. ispoštovali su proceduru. Postoje digitalni dokumenti koji se moraju arhivirati i čuvati određeni broj godina ili pak neograničeno. Postoje pravila i procedure za rukovanje digitalnim dokumentima u toku životnog ciklusa dokumenta. Disciplina koja se bavi ovim pravilima i procedurama zove se **upravljanje digitalnim dokumentima**. Digitalni dokumenti koji su namenjeni velikoj ciljnoj grupi i čija je važnost značajna prolaze kroz sledeće faze:

- inicijalizacija,
- priprema,
- uspostavljanje,
- korišćenje,
- revizija,
- arhiviranje,
- uklanjanje.

Inicijalizacija je prva faza u životnom ciklusu dokumenta i ona predstavlja formiranje podataka potrebnih za kasniju pripremu, ali ne obuhvata pripremu i utvrđivanje sadržaja. Rezultat ove faze je okvir u kome se dalje priprema dokument. U ovoj

fazi je potrebno odrediti identifikaciju dokumenta koja predstavlja jednoznačno određivanje dokumenta u datom kontekstu. Identifikacija dokumenta omogućava precizno referenciranje na dokument. Identifikacija treba da bude stabilna i nezavisna od načina prezentacije ili fizičke lokacije. Dokument može biti prikazan na različitim jezicima, u različitim formama: ekran, papir, itd. Digitalni dokument može izgledati različito za različite korisnike, ne mora uvek prikazivati sve informacije, naprotiv, može prikazivati samo delove digitalnog dokumenta relevantne za određenog korisnika. Ova faza dodaje identifikator dokumenta u metapodatke. Identifikator može biti: interni identifikator dokumenta u okviru organizacije, međunarodni identifikator dokumenta (ISBN, ISSN), međunarodni identifikator digitalnog dela (IDDN), itd. U ovoj fazi mogu se dodavati i metapodaci koji se odnose na klasifikaciju dokumenta. Klasifikacija dokumenta opisuje karakteristike dokumenta i pojednostavljuje pretragu dokumenta koji se bave istim ili srodnim temama. Dokumenti se mogu klasifikovati po različitim šemama: ISO/IEC 61355, ICS, interni šifarnici, ključne reči. Metapodaci o klasifikaciji mogu da obuhvate:

- funkciju dokumenta,
- jezike korišćene u dokumentu,
- identifikatore učesnika poslovnog procesa,
- datum inicijalizacije i rok za pripremu,
- prava pristupa,
- patentna i autorska prava.

Priprema je naredna faza u životnom ciklusu dokumenta, počinje nakon inicijalizacije i ona predstavlja proizvodnju sadržaja dokumenta sve do trenutka uspostavljanja. Metapodaci koji se dodaju u ovoj fazi bi mogli da sadrže:

- nivo razvoja dokumenta,
- ključne reči,

- rezime ili apstrakt,
- izvor dokumenta.

Uspostavljanje (eng. *establishment*) je naredna faza u životnom ciklusu dokumenta. Za neke dokumente je pre njihovog korišćenja neophodno dobiti odobrenje od određenih ljudi ili institucija. Ovo se radi za potrebe obezbeđivanja kvaliteta. Odobravanje se često radi ne samo pre korišćenja prve verzije dokumenta, nego i nakon kreiranja svake nove verzije dokumenta. Pravila za odobravanje se definišu na nivou poslovnog procesa, klase dokumenta ili pojedinačnog dokumenta. Metapodaci koji se dodaju u ovoj fazi su:

- ID (identifikator) zahteva za odobrenje,
- ID podnosioca,
- datum podnošenja,
- rok za dobijanje odobrenja,
- ID osobe/organizacije zadužene za proveru,
- ID osobe/organizacije zadužene za odobravanje,
- komentari vezani za proveru i odobravanje.

Korišćenje je naredna faza u životnom ciklusu dokumenta u kojoj je dokument sa metapodacima dostupan za korišćenje. Metapodaci se koriste za pretraživanje i informisanje o dokumentima i njihovim verzijama. U metapodatke se mogu dodati komentari, odnosno iskustva korisnika o korišćenju dokumenta. U ovoj fazi je aktuelno i pitanje kako distribuirati dokument korisnicima.

Distribucija, odnosno dostavljanje digitalnog dokumenta korisnicima se može izvršiti automatskim slanjem ili obaveštavanjem korisnika o dostupnom dokumentu i lokaciji putem različitih vrsta medija: novine, radio, televizija, Internet, društvene mreže, itd. Metapodaci vezani za distribuciju dokumenata su:

- distribucione liste,
- ID primalaca,

- uloge primalaca u poslovnom procesu,
- specifikacije formata distribucije,
- specifikacije formata u kojima je dokument dostupan.

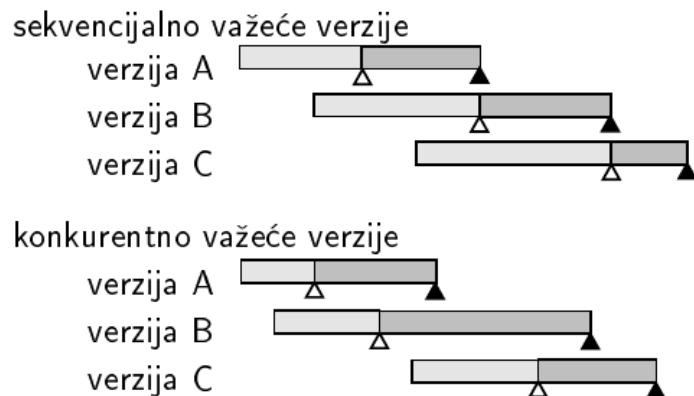
Revizija je naredna faza u životnom ciklusu dokumenta koja je opcionala i ponavljajuća, i koja predstavlja promenu sadržaja ili promenu namene dokumenta. Nakon ove faze se obično ponovo vrši uspostavljanje, odnosno odobravanje pre početka korišćenja nove verzije. U jednom trenutku može se koristiti više verzija sve dok ispunjavaju svoju namenu. Izmena dokumenta može da obuhvati:

- izmenu informacija u dokumentu i/ili
- izmenu vizuelne prezentacije informacija.

Kada dokument treba da bude sačuvan kao nova verzija? Preovlađujući stav je:

- kada se menjaju informacije - DA,
- kada se menja prezentacija - NE.

Ako dokument ima više verzija moramo imati podršku za **upravljanje verzijama**. Za svaku verziju postoji **period formiranja** kada se verzija formira i **period važenja** kada se verzija smatra važećom. Važenje verzija se može organizovati sekvencialno ili konkurentno (slika 1.5).



Slika 1.5: Važenje verzija

Sekvencijalno važenje verzija podrazumeva da je poslednja verzija dokumenta jedina važeća. Nova verzija uvek preuzima važenje od prethodne verzije i podržava sve namene svih prethodnih verzija. Nova verzija dokumenta podrazumeva ažuriranje sledećih metapodataka:

- prethodna verzija na kojoj se zasniva,
- verzije koje su zamenjene novom, ili se na njih utiče novom verzijom,
- ID osoba/organizacija koje su sprovele izmene,
- opis rezultata izmene,
- opis razloga uvođenja izmene,
- datum izmene.

Konkurentno važenje verzija podrazumeva da više različitih verzija može biti operativno u jednom trenutku. Nova verzija ne zamenjuje automatski prethodnu u smislu važenja, odnosno verzija ostaje važeća sve do eksplicitnog povlačenja verzije. Povlačenje verzije predstavlja izmene u metapodacima, ali ne i u sadržaju dokumenta. Metapodaci vezani za povlačenje verzije obuhvataju:

- veze između verzija - zamenjuje/zamenjen sa,

- verzije na koje se utiče povlačenjem,
- opis šta je učinjeno,
- opis kada je izmena načinjena.

Arhiviranje je naredna faza u životnom ciklusu dokumenta i predstavlja premeštanje dokumenta (verzije, metapodataka) u kompaktniju nepromenljivu formu. Arhiviranje se radi da se ispunе ugovorne/zakonske obaveze, na primer rok čuvanja. Potrebno je obezbediti kontrolisani pristup arhivi i mogućnost reprodukcije dokumenata. Potrebno je sprečiti mogućnost izmena dokumenata koji su arhivirani. Arhiva je baza znanja zbog čega je potrebno obezbediti pretraživanje arhive digitalnih dokumenata. U arhivama je poželjno koristiti stabilne, nepromenljive formate podataka, odnosno poželjno je izbegavati nestandardizovane formate podataka koji su podržani od strane jednog softvera i koji će vrlo verovatno u budućnosti biti promenjeni (na primer doc format). Metapodaci vezani za arhiviranje su:

- prava pristupa / nivo poverljivosti,
- korišćene hardverske i softverske komponente,
- korišćeni postupci za arhiviranje i kompresiju,
- korišćeni postupci za kriptografsku zaštitu,
- digitalni potpisi,
- vremenski ciklus osvežavanja podataka (za magnetne medije),
- istorija izmena na fizičkim nosiocima podataka,
- istorija izmena na formatu podataka,
- fizička lokacija skladišnog medija,
- fizička lokacija rezervne kopije,
- dnevnik pristupa arhiviranom dokumentu.

Uklanjanje je poslednja faza u životnom ciklusu dokumenta i dokument može ući u ovu fazu tek nakon isteka perioda za obavezno arhiviranje. Uklanjanje sadržaja i metapodataka ne mora

biti istovremeno, na primer dok se drugi dokument ili verzija referišu na dati dokument trebalo bi čuvati metapodatke. Uklanjanje dokumenata rezultuje nepovratnim gubitkom podataka, dokumenata i relacija sa drugim dokumentima.

Rezime

- Digitalni dokument je računarski obrađena informacija kojom se rukuje kao osnovnom jedinicom obrade.
- Danas živimo u digitalnom dobu u kojem smo okruženi uređajima koji stvaraju i prenose digitalne informacije za šta je zaslužan razvoj IKT, kao i razvoj veba i popularnih veb aplikacija kao što su Facebook, Twitter, Instagram, YouTube, itd.
- Sadržaj digitalnog dokumenta predstavlja podatke koji se prenose onome ko konzumira digitalni sadržaj.
- Metapodaci su podaci o dokumentu, odnosno podaci o podacima. Metapodaci se koriste da se bolje organizuju kolekcije dokumenata, da se pregled dokumenata prilagodi korisnicima, za klasifikaciju dokumenata, metapodaci mogu značajno unaprediti kvalitet i mogućnosti pretrage kolekcije digitalnih dokumenata, mogu se koristiti i za filterisanje i sortiranje rezultata pretrage.
- Digitalni dokumenti mogu biti različite složenosti što može uticati i na organizaciju metapodataka.
- Mogu postojati različiti izvori iz kojih su nastali metapodaci za digitalni dokument.
- Digitalni dokument može prolaziti kroz različite faze svog životnog ciklusa: inicijalizacija, priprema, uspostavljanje, korišćenje, revizija, arhiviranje, uklanjanje

Pitanja

1. Objasniti pojam dokumenta, papirnog dokumenta i digitalnog dokumenta.
2. Šta su to metapodaci i navesti nekoliko primera metapodataka?
3. Kakve sve veze između dokumenta i metapodataka mogu postojati?
4. Čime se bavi upravljanje digitalnim dokumentima?
5. Koje su faze životnog ciklusa dokumenta?
6. Objasniti životnu fazu dokumenta korišćenje.
7. Objasniti životnu fazu dokumenta arhiviranje.
8. Objasniti pojmove upravljanje verzijama dokumenata, sekvensijalno i konkurentno važenje verzija.

Glava 2

Sistemi za upravljanje dokumentima

Digitalni dokumenti kreirani pomoću različitih uređaja i programa za kreiranje digitalnih sadržaja mogu biti skladišteni u informacionim sistemima. Ovo poglavlje ima za cilj da predstavi sisteme koji skladište digitalne dokumente i da odgovori na sledeća pitanja:

- Koje su razlike između standardnih poslovnih aplikacija koji skladište podatke u relacionim bazama podataka i sistema koji prate i skladište nestrukturirane digitalne dokumente?
- Koje funkcionalnosti treba da implementiraju sistemi koji skladište digitalne dokumente?
- Koje sve vrste sistema koji skladište digitalne dokumente postoje?

Prilikom opisa sistema koji skladište digitalne dokumente polazna osnova bio je pregled ovih sistema prikazan u doktorskoj disertaciji “Modeliranje i implementacija digitalne biblioteke” [4]. Takođe, deo informacija

prezentovanih u ovom poglavlju su preuzeti iz izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Standardne poslovne aplikacije rukuju čvrsto strukturiranim podacima koji su najčešće smešteni u relacionim bazama podataka. Korisnički interfejs ovih aplikacija je prilagođen intenzivnom radu sa podacima (eng. *data-intensive*). Ako je potrebno obezbediti razmenu podataka ovakvih sistema obično se to radi putem XML formata. Postoje poslovne aplikacije u kojima iz ugla korisnika sistem rukuje dokumentima, a iz ugla inženjera sistem rukuje podacima organizovanim u tabelama unutar relacionih baza podataka. Ono što korisnik iz svog ugla vidi kao dokument je dovoljno dobro strukturirano da se može prikazati odnosno mapirati na podatke koji se nalaze u tabelama. U klasičnim domenima kao što je finansijsko/materijalno poslovanje čest je slučaj da korisnici aplikacije imaju utisak da rade sa dokumentima iako zapravo koriste klasičnu poslovnu aplikaciju koja čuva podatke u relacionoj bazi podataka. Primeri za to su: fatura, nalog za plaćanje, izvod iz banke, itd. Sa druge strane iz ugla korisnika dokument je i zaključak sa sednice Vlade, izveštaj komisije o izboru u nastavno zvanje, kao i drugi dokumenti koji u sebi sadrže nestrukturiran ili slabo strukturiran tekst. Ovakvih dokumenata je sve više, jer je sve više računara dostupno ljudima koji su računarski dovoljno pismeni da mogu pomoći različitim tekst procesora kao što su *Microsoft Word* ili *OpenOffice Writer* kreirati ovakve dokumente. Sa stanovišta inženjera ovakvi dokumenti koji sadrže nestrukturirane ili slabo strukturirane tekstove nisu pogodni za čuvanje u relacionim bazama podataka. Za ovakve dokumente potrebna nam je nova vrsta sistema koji su namenjeni praćenju i skladištenju digitalnih dokumenata. Iz ugla inženjera ovo je nova disciplina **upravljanje digitalnim dokumentima** i ovakvi sistemi se često zovu **sistemi za upravljanje dokumentima**.

2.1 Funkcionalnosti sistema

Funkcionalnosti koje veliki broj sistema za upravljanje dokumentima implementiraju su sledeće:

- skladištenje dokumenata,
- katalogizacija,
- pretraživanje,
- zaštita podataka,
- oporavak od katastrofe (eng. *recovery*),
- arhiviranje,
- distribucija,
- upravljanje poslovnim procesima (eng. *Workflow management*).

Skladištenje dokumenata je osnovna namena ovih sistema i postoji dva modela za organizaciju kolekcije dokumenata:

1. centralizovani,
2. distribuirani.

Centralizovani model je mnogo jednostavniji za implementaciju i brži od distribuiranog, ali je ograničen memorijskim kapacitetima jednog računara. **Distribuirani model** se implementira kao mreža tvrdih diskova na različitim računarima. Ovaj model je zavisан od mrežnog protoka, pa se zbog toga računari u distribuiranom modelu najčešće nalaze u lokalnoj mreži (LAN). Kod distribuiranog modela korisnik ne treba da bude svestan organizacije kolekcije dokumenata, treba da ima jedinstven pogled na kolekciju dokumenata i ne treba da bude opterećen informacijom na disku kog računara je skladišten neki dokument.

Katalogizacija predstavlja sistematično sređivanje liste zapisa. Ova funkcionalnost je vrlo bitna zbog mogućnosti pretrage i pregleda kolekcije dokumenata. Katalogizacija se najviše razvijala u bibliotečkim sistemima jer su se tu prvo pojavljivale velike

kolekcije zapisa koje je bilo bitno sistematično srediti. Postoje različiti standardizovani formati za katalogizaciju od kojih se neki široko prihvaćeni u bibliotečkim sistemima. Unutar definicije formata se definiše i lista metapodataka koji se mogu koristiti u tom formatu. U narednom poglavlju ove knjige (poglavlje 3) biće opisani neki od ovih formata. Katalogizaciju može obavljati jedan ili više učesnika (aktera). Učesnici mogu biti različitog stepena računarske pismenosti i različitog stepena poznavanja usvojenih formata u sistemu zbog čega se u nekim sistemima iz korisničkog interfejsa ne vidi format u kome se čuvaju metapodaci. Ako više učesnika učestvuju u procesu katalogizacije jednog zapisa sistem mora obezbediti saradnju (kolaboraciju) učesnika.

Pretraživanje je funkcionalnost sistema koja nam omogućuje pristup znanju koje se nalazi u velikim kolekcijama digitalnih dokumenata. Živimo u svetu u kojem postoje velike kolekcije digitalnih dokumenata kao što su:

- <http://archive.org>,
- <http://books.google.com>,
- <http://arxiv.org>,
- WWW (eng. *World Wide Web*) je jedan distribuirani sistem sa velikom količinom dokumenata i znanja za koji postoje sistemi za pretragu (eng. *Web Search Engine*).

Bez funkcionalnosti pretrage velikih kolekcija digitalnih dokumenata u ovim kolekcijama bilo bi gotovo nemoguće pronaći potrebne informacije. Potrebno je kreirati forme za zadavanje upita i prikaz rezultata pretrage. Interfejs je bitan, ali nije samo on bitan i u većini slučajeva interfejs nije najbitniji. Neophodno je izvršiti i pripremu podataka za pretragu. Pretragom velikih kolekcija sadržaja bavi se oblast **pronalaženje informacija** (eng. *information retrieval*) i značajan deo ove knjige je posvećen ovoj oblasti.

Zaštita podataka je takođe bitan aspekt sistema za upravljanje digitalnim dokumentima. Podaci se u ovim sistemima naлaze u digitalnim dokumentima. Kao što je u klasičnim poslovnim

aplikacijama često potrebno sprečiti neautorizovano pregledanje i izmenu podataka koji se nalaze u relacionim bazama, tako je i u sistemima za upravljanje dokumentima često potrebno sprečiti neautorizovano preuzimanje i izmenu digitalnih dokumenata. Potrebno je razviti mehanizme za autentifikaciju i autorizaciju korisnika. Često je potrebno obezbediti mogućnosti kao što su: definisanje grupa korisnika, uloga, veza između korisnika i grupa korisnika, veza između dostupnih funkcionalnosti i rola, kao i veza između grupa korisnika i uloga. Ako su neki dokumenti otvorenog pristupa (eng. *open access*) moraju biti rešena autorska i druga pravna pitanja. Često je u sistemu potrebno obezbediti da se za dokument može vezati licenca koja definiše prava korišćenja tog dokumenta.

Sistemi za upravljanje dokumentima bi trebali da imaju i mehanizme za **oporavak od katastrofe**. Želeli mi to ili ne, katastrofe u kojima se uništavaju sadržaji na tvrdim diskovima se dešavaju: požari, poplave, električni udari, fizička oštećenja uređaja, itd. Takođe, elektronika na diskovima ima vek trajanja. Pošto nam se u digitalnim dokumentima nalaze podaci koji mogu biti značajni, ove katastrofe mogu izazvati velike štete ako sistem nije za njih bio pripremljen. Pored fizičke zaštite uređaja na kojima su digitalni dokumenti, neophodno je preduzeti i niz radnji koje će umanjiti štetu ako se dogodi neka katastrofa, odnosno štetu svesti samo na materijalnu vrednost uređaja koji je u katastrofi prestao da funkcioniše. Potrebno je da sistem ima implementirane mehanizme za *backup* i *recovery*. **Backup** sistema se može vršiti na dnevnom, nedeljnem ili mesečnom nivou zavisno od tempa kojim se menjaju digitalni dokumenti i od značaja dokumenata koji se nalaze u sistemu. U sistemima za upravljanje dokumentima često su *backup*-i zahtevni po pitanju procesora i memorije. Ako sistem ima veliki broj korisnika, treba pažljivo birati algoritam i vreme kada će se obaviti *backup* kako u tom periodu korisnici sistema ne bi imali slab odziv sistema. Ako sistem ima veliku količinu dokumenata, mora se pažljivo birati i strategija za *backup* kako bi bilo dovoljno memorijskih resursa za čuvanje *backup*-a. U nekim slučajevima se koristi *incremental backup* strategija koja ne čuva kompletну kopiju podataka

sistema, nego samo razliku podataka u odnosu na neki prethodni *backup*. *Backup* treba da se čuva na fizički udaljenom uređaju. **Recovery** se obavlja samo u slučaju potrebe, ali se funkcionalnost ovog mehanizma mora dobro testirati da se ne bi ispostavilo da nakon pretrpljene katastrofe *backup* ili *recovery* mehanizam ima propusta.

Arhiviranje je funkcionalnost sistema koja omogućuje arhiviranje nekog dokumenta koji više nije u upotrebi, odnosno omogućuje formiranje arhive. Mora se omogućiti pregled arhive kao i pretraga arhiviranih dokumenata. Arhiva je baza znanja. Obično se sprečava izmena dokumenata koji su u arhivi. U nekim sistemima je potrebno neograničeno čuvanje dokumenata koji su u arhivi. Kao što tradicionalni papirni dokument može biti upropasten zubom vremena to se može desiti i digitalnom dokumentu. Sve ima svoj vek trajanja: papir, CD, hard-disk, elektronika, itd. Postoje tehnike koje omogućavaju da se knjige koje su značajne za istoriju čovečanstva čuvaju kroz vekove. Ovo je zaista značajna oblast u bibliotekarstvu i postoje naučni časopisi koji objavljuju samo naučne rezultate koji pripadaju ovoj oblasti kao što je časopis *“Restaurator: International Journal for the Preservation of Library and Archival Material”*. Oblast koja se bavi ovom temom zove se **dugotrajno skladištenje** (eng. *long-term preservation*). Pored činjenice da nakon određenog vremena može doći do oštećenja medija na kojem se nalazi digitalni zapis (CD, DVD, hard-disk) postoji još jedan bitan faktor o kome moramo voditi računa ako želimo da obezbedimo da se kreirani digitalni dokumenti mogu konzumirati nakon dužeg vremenskog perioda. Programi za konzumiranje digitalnih dokumenata se menjaju i menjaju formate koje podržavaju. Nakon određenog broja godina korisnici će se verovatno prilično mučiti da otvorimo neki dokument koji nije oštećen, ali je u *doc* formatu. Pitanje je da li će postojati softver koji otvara ovaj format i da li će aktuelni operativni sistemi u tom momentu podržavati taj softver. Potrebno je obezrediti transformaciju izvornog formata digitalnog dokumenta u novi format koji je aktuelan bez izmene suštinskog sadržaja koji se nalazi u digitalnom dokumentu, odnosno sadržaj i formatiranje sadržaja treba da ostanu isti.

Distribucija je funkcionalnost koja nam obezbeđuje mehanizme za distribuiranje digitalnih dokumenata. Bavi se pitanjima kako obavestiti javnost da je neki dokument nastao i kako obavestiti javnost da je neki dokument promenjen. U digitalnom dobu u kojem živimo okruženi smo velikim brojem digitalnih sadržaja i ako za neki digitalni dokument niko ne zna da postoji, to je gotovo isto kao i da ne postoji. Kao što je već rečeno u prethodnom poglavlju konzumenti digitalnog dokumenta mogu biti milioni ljudi i dokument se može koristiti za regulisanje saobraćajnih propisa ili definisanje određenih zakona. U ovakvim situacijama mehanizmi za distribuiranje digitalnih dokumenata su veoma važni.

Upravljanje poslovnim procesima (eng. *workflow management*) za nastajanja i korišćenje nekog dokumenta je važna funkcionalnost sistema za upravljanje dokumentima. U nekim sistemima ovi procesi mogu biti složeni i može biti više učesnika koji učestvuju u kreiranju digitalnog sadržaja. Neophodno je obezbediti saradnju (kolaboraciju) ovih korisnika u formiraju digitalnog dokumenta. Može postojati i uredba definisana na nivou institucije ili države za formiranja dokumenata određene vrste, tako da sistem treba da spreči kreiranje dokumenta mimo te procedure.

2.2 Primeri sistema

Digitalni dokumenti mogu biti skladišteni u različitim vrstama sistema. Neki su opšte, a neki usko specificirane namene. Shodno različitim namenama, različite su i funkcionalnosti koje ovi sistemi implementiraju. Takođe, i ciljne grupe korisnika mogu biti različite. U ovoj sekciji su ukratko opisane neke vrste sisteme u kojima se mogu skladištiti dokumenti.

2.2.1 Poslovni sistem za upravljanje sadržajem

Sistemi za upravljanje dokumentima su sastavni deo poslovnih informacionih sistema. Pojam poslovni informacioni sistem obuhvata pojam **poslovni sistem za upravljanje sadržajem** (ECMS, eng. *Enterprise Content Management System*). Pod upravljanjem sadržajem obično se podrazumeva upravljanje dokumentima, upravljanje zapisima, upravljanje veb sadržajima, upravljanje poslovnim procesima i saradnjama [5]. Osnovna definicija ECMS sistema formirana je u okviru AIIM asocijacije, međunarodne organizacije koja okuplja proizvođače i korisnike ECMS sistema. Funkcionalnosti ECMS sistema neophodne za upravljanje digitalnim dokumentima su sledeće:

- zahvatanje (eng. *capture*) - generisanje, priprema i procesiranje informacija u elektronskom ili drugom obliku,
- skladištenje (eng. *storing*),
- upravljanje životnim ciklusom dokumenata,
- arhiviranje i
- saradnja (eng. *collaboration*) - različitih učesnika u upravljanju digitalnim dokumentom.

Pored toga, karakteristike koje su prisutne u većem broju ECMS sistema su sledeće: upravljanje poslovnim procesima, digitalni potpisi, upravljanje podacima i upravljanje web sadržajima. Predstavnik ovih sistema je **Alfresco** koji predstavlja veb bazirani ECMS sistem koji može da se primenjuje za upravljanje sadržajem kako u malim i srednjim organizacijama tako i u velikim geografski distribuiranim aplikacijama [6]. Koristi samo tehnologije otvorenog izvornog koda (eng. *open source*), baziran je na otvorenim standardima i publikovan je pod licencom *GNU General Public License*. Lako se prilagodava i proširuje za specifične potrebe jedne organizacije korišćenjem XML dokumenata i lako se integriše sa drugim aplikacijama putem otvorenih standarda. Složeniji zahtevi se mogu implementirati izmenom otvorenog izvornog koda. Alfresco obezbeđuje organizaciji sve potrebne

servise za kreiranje i upravljanje digitalnim dokumentima. Podržava kontrolu verzija, ima mogućnost pretrage implementirane putem *Lucene* biblioteke [7]. Integrисани систем za upravljanje poslovnim procesima omogućava punu kontrolu nad životnim ciklusom digitalnih dokumenata, kao i upravljanje poslovnim procesima u kojima učestvuju digitalni dokumenti.

2.2.2 Digitalne biblioteke

Nastanak digitalnih biblioteka je prirodan put prolaska ljudske civilizacije kroz informaciono doba i uvođenje savremenih tehnologija u razne aspekte ljudskog društva [8]. **Digitalne biblioteke** su logično proširenje fizičkih biblioteka novim resursima i servisima u digitalnom dobu u kojem živimo. U odnosu na klasične/fizičke biblioteke, digitalne biblioteke imaju neke prednosti, a jedna od najznačajnijih prednosti je lakša diseminacija (širenje) sadržaja. Digitalne biblioteke mogu biti opšte ili specifične namene kao što su na primer digitalne biblioteke doktorskih disertacija.

Postoje softverske platforme otvorenog izvornog koda koje se mogu iskoristiti za kreiranje digitalne biblioteke kao što su *DSpace* (www.dspace.org) i *EPrints* (www.eprints.org). Ove softverske platforme se lako instaliraju i prilagođavaju specifičnim potrebama digitalne biblioteke. Mogu se iskoristiti za kreiranje digitalne biblioteke opšte namene kao i za kreiranje digitalne biblioteke specifične namene.

DSpace je softver otvorenog koda koji je razvijen tokom dvo-godišnje saradnje *Hewlett-Packard* kompanije i Univerziteta MIT (eng. *Massachusetts Institute of Technology*). Namenjen je za kreiranje digitalnih biblioteka koje sadrže tekstualne i multimedijalne dokumente. Osnovne osobine ovog proizvoda su:

- skladištenje dokumenata, multimedije i podataka,
- podržana višejezičnost,
- OAI-PMH kompatibilnost u Dublin Core formatu,
- dugotrajno skladištenje.

U prvoj godini od nastajanja DSpace platforme preko 2.500 organizacija su preuzele ovu softversku platformu za izgradnju svoje digitalne biblioteke ili institucionalnog repozitorijuma [9]. Početkom 2013. godine na zvaničnom sajtu je objavljena lista od 1405 DSpace registrovanih instanci, dok je stvaran broj korisnika ove platforme teško proceniti.

EPrints je softver otvorenog koda koji je razvijen na Fakultetu za elektroniku i računarske nauke, Univerziteta u Sautemp-tonu. Predstavlja fleksibilnu platformu za izgradnju digitalnih biblioteka. Upotreboom ove platforme lako i brzo se može kreirati digitalna biblioteka opšte namene ili digitalna biblioteka specifične namene. Osnovne osobine ove platforme su:

- Skladištenje dokumenata, multimedije i podataka,
- Podržana višejezičnost,
- OAI-PMH kompatibilnost u Dublin Core formatu.

Trenutno, postoji oko 270 repozitorijuma sa oko 600.000 zapisu koji su kreirani upotrebom EPrints platforme.

Kao što je već rečeno DSpace i EPrints platforme se mogu iskoristiti za potrebe implementacije digitalne biblioteke specifične namene kao što je na primer digitalna biblioteka doktorskih disertacija [10–14]. Ove platforme ipak nisu planski razvijane za digitalne biblioteke doktorskih disertacija te im zbog toga neke funkcionalnosti za sisteme ove vrste nedostaju, odnosno ove platforme bi trebale biti proširene podrškom za poslovni proces za prijavu e-teze, podrškom za saradnju mentora i kandidata, podrškom za ETD-MS umesto Dublin Core formata, bogatijim servisom za pretraživanje i bogatijom komponentom za izveštavanje [15]. Kako su ove platforme otvorenog izvornog koda, neki univerziteti su proširivali ove platforme za potrebe implementacije svoje digitalne biblioteke doktorskih disertacija [16, 17].

Postoje i softverske platforme koje su implementirane za potrebe kreiranja digitalne biblioteke specifične namene i imaju podršku za sve bitne funkcionalnosti tih specifičnih digitalnih biblioteka. Na primer, za razvoj digitalnih biblioteka doktorskih diser-

tacija mogu se koristiti *ETD-db* ili *OpenDLT* platforma. **OpenDLT** je *platforma koja se razvija na Univerzitetu u Novom Sadu* (<http://opendlt.uns.ac.rs/>). Ova softverska platforma je iskorišćena za kreiranje digitalne biblioteke disertacije odbranjenih na Univerzitetu u Novom Sadu koja je integrisana unutar Informacionog sistema naučne delatnosti Univerziteta u Novom Sadu (CRIS UNS - <http://cris.uns.ac.rs/>). Razvoj informacionog sistema naučne delatnosti je implementiran kroz naučna istraživanja koja su publikovana u radovima [18–28]. Razvoj softverske platforme OpenDLT i primena ove platforme za implementaciju digitalne biblioteke doktorskih disertacija Univerziteta u Novom Sadu su opisani u radovima [4, 29–33]. Pretraga digitalne biblioteke disertacija Univerziteta u Novom Sadu je dostupna na ovom linku: <http://www.cris.uns.ac.rs/searchDissertations.jsf>. Pored pretrage digitalna biblioteke, sistem podržava uvoz metapodataka o doktorskim disertacijama iz raznih izvora kao i izvoz podataka putem OAI-PMH protokola u različitim formatima kao što su MARC 21, Dublin Core, ETD-MS. Ovi protokoli i formati su tema poglavlja 3 ove knjige. Iskustva stečena u razvoju ove digitalne biblioteke su korišćena prilikom pisanja ove knjige.

2.2.3 Institucionalni repozitorijumi

Institucionalni repozitorijum je softverski sistem za prikupljanje, čuvanje i preuzimanje digitalnih sadržaja koji predstavljaju digitalne dokumente relevantne za jednu instituciju. Na primer, može se kreirati institucionalni repozitorijum naučne institucije kao što je univerzitet u kojem se skladište digitalni sadržaji kao što su radovi publikovani u časopisu, radovi prezentovani na naučnoj konferenciji, magistarske teze, doktorske disertacije, itd.

Postojanje određenog naučnog rezultata u digitalnoj formi u nekom institucionalnom repozitorijumu povećava se dostupnost ovog rezultata. Pored mogućnosti preuzimanja digitalnih dokumenata, većina ovih sistema omogućuje i preuzimanje metapodataka o njima po protokolu OAI-PMH u Dublin Core formatu. Po-

stoje mnogi softverski sistemi koji predstavljaju institucionalne repozitorijume, a najviše korišćene platforme za kreiranje institucionalnih repozitorijuma su EPrints i Dspace.

Da bi se još više povećala vidljivost naučnih rezultata pojaviće se težnja da se kreiraju mreže institucionalnih repozitorijuma kako bi se kreirao jedan virtualni repozitorijum. Primer jedne ovakve mreže je **OpenAIRE** (www.openaire.eu).

Rezime

- Standardne poslovne aplikacije rukuju čvrsto strukturiranim podacima koji su najčešće smešteni u relacionim bazama podataka.
- Za digitalne dokumente koji imaju nestrukturirane sadržaje koji nisu pogodni za čuvanje u relacionim bazama podataka potrebna je nova vrsta sistema koji su namenjeni praćenju i skladištenju digitalnih dokumenata koji se često zovu sistemi za upravljanje dokumentima.
- Sistemi za upravljanje dokumentima bi trebali da imaju sledeće funkcionalnosti: skladištenje dokumenata, katalogizacija, pretraživanje, zaštita, oporavak od katastrofe, arhiviranje, distribucija, podrška za poslovni proces.
- Digitalni dokumenti mogu biti skladišteni u različitim vrstama sistema, neki su opšte a neki usko specifične namene, shodno različitim namenama različite su i funkcionalnosti koje ovi sistemi implementiraju.
- Poslovni sistem za upravljanje sadržajem (ECMS) podržava upravljanje dokumentima, upravljanje zapisima, upravljanje veb sadržajima, upravljanje poslovnim procesima i saradnjama. Predstavnik ovih sistema je Alfresco koji predstavlja veb bazirani ECMS sistem koji može da se primenjuje za upravljanje sadržajem kako u malim i srednjim organizacijama tako i u velikim geografski distribuiranim aplikacijama.
- Digitalne biblioteke su logično proširenje fizičkih biblioteka novim resursima i servisima u digitalnom dobu u kojem živimo. U odnosu na klasične, fizičke biblioteke, digitalne biblioteke imaju neke prednosti, a jedna od najznačajnijih je lakša diseminacija (širenje) sadržaja.

- Postoje softverske platforme otvorenog izvornog koda koje se mogu iskoristiti za kreiranje digitalne biblioteke kao što su DSpace i EPrints. Ove softverske platforme se lako instaliraju i prilagođavaju specifičnim potrebama digitalne biblioteke.
- Postoje i softverske platforme koje su implementirane za potrebe kreiranja digitalne biblioteke specifične namene i imaju podršku za sve bitne funkcionalnosti tih specifičnih digitalnih biblioteka. Na primer za razvoj digitalnih biblioteka doktorskih disertacija mogu se koristiti ETD-db ili OpenDLT platforma.
- Institucionalni repozitorijum je softverski sistem za prikupljanje, čuvanje i preuzimanje digitalnih sadržaja koji predstavljaju digitalne dokumente relevantne za jednu instituciju.

Pitanja

1. Navesti osobine standardnih poslovnih aplikacija.
2. Koja je osnovna namena sistema za upravljanje dokumentima?
3. Čime se bavi disciplina upravljanje digitalnim dokumentima?
4. Koje su funkcije sistema za upravljanje dokumentima?
5. U kojim vrstama sistema mogu biti skladišteni digitalni dokumenti?

Glava 3

Standardi u sistemima digitalnih dokumenata

Ovo poglavlje ima za cilj da opiše standarde koji se koriste u oblasti upravljanja digitalnim dokumentima. U ovom poglavlju biće prikazani:

1. *ISO IEC 82045* standard za informatičko uređenje sistema za upravljanje dokumentima,
2. standardizovani formati metapodataka: MARC21, Dublin Core, ETD-MS,
3. standardizovani protokoli za komunikaciju između sistema za upravljanje digitalnim dokumentima: OAI-PMH, Z39.50, SRU.

Prilikom opisa formata metapodataka i protokola za komunikaciju za razmenu podataka polazna osnova bio je pregled ovih formata i protokola prikazan u doktorskoj disertaciji "Modeliranje i implementacija digitalne biblioteke" [4]. Za opis protokola za udaljeno pretraživanje korišćeni su tekstovi iz doktorske disertacije "Model za distribuirano i rangirano pretraživanje bibliotečkih in-

formacionih sistema” [34]. Takođe, deo informacija prezentovanih u ovom poglavlju su preuzeti iz izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Standardi koji se primenjuju u oblasti upravljanja digitalnim dokumentima pripadaju različitim oblastima:

- uređenje sistema za upravljanje dokumentima,
- formati za reprezentaciju metapodataka,
- protokoli
 - za razmenu metapodataka i
 - za udaljeno pretraživanje.

ISO IEC 82045 je standard za informatičko uređenje sistema za upravljanje dokumentima. Sastoji se iz delova, neki od značajnijih delova su navedeni u nastavku. *IEC 82045-1:2001 Document Management: Part 1 - Principles and Methods* definiše principe i metode za definisanje metapodataka namenjenih upravljanju dokumentima u okviru životnog ciklusa [1]. *IEC 82045-2:2004 Document Management: Part 2 - Metadata Elements and Information Reference Model* definiše skup standardizovanih metapodataka za upravljanje dokumentima [35]. Nameđen je za razmenu podataka između sistema i može biti osnova za implementaciju sistema za upravljanje dokumentima. Standard je definisan u formi XML gramatike definisane DTD-om. *ISO 82045-5:2005 Document Management: Part 5 - Application of Metadata for the Construction and Facility Management Sector* definiše skup metapodataka bitnih za upravljački sloj organizacije i metode za razmenu ovih metapodataka [36].

ISO IEC 82045 standard ne mora biti podržan u potpunosti, odnosno sam standard definiše nekoliko nivoa (klasa) podrške u različitim aspektima. Usklađenost sa standardom prema nivou podrške za rad sa verzijama ima dve klase:

- *Conformance class A*: podrška za sekvencijalno važeće verzije,
- *Conformance class B*: podrška za sekvencijalno važeće verzije i podrška za konkurentno važeće verzije.

Usklađenost sa standardom prema stepenu pokrivanja referentnog modela podataka ima tri klase:

- *Conformance class 1*: Podržava samo koncept statičkog dokumenta. Nema životnog ciklusa, veza, verzija. Verzije se mogu čuvati odvojeno, bez podataka o nameni ili roku važenja.
- *Conformance class 2*: Podrška za životni ciklus, verzije, istoriju, veze i referenciranje objekata koji se nalaze izvan sistema.
- *Conformance class 3*: Distribucija i pretplata na dokumente u skladu sa definisanim pravilima. Podrška za arhiviranje. Referenciranje na dokumente u drugim sistemima za upravljanje dokumentima.

3.1 Formati metapodataka

Postoje standardizovani **formati metapodataka** koji se koriste u sistemima za upravljanje dokumentima. Neki od tih standardizovanih formata metapodataka su nastali u drugim oblastima, ali su našli svoju primenu i u upravljanju digitalnim dokumentima. Neki od formata su opšte namene, a neki su specifične namene i mogu se koristiti samo za digitalne dokumente određenog tipa. Na primer, MARC 21 format je opšte namene i koristi se za katalogizaciju svih vrsta bibliotečke građe, dok je ETD-MS je format specifične namene koji je namenjen za opis metapodataka digitalnih dokumenata doktorskih disertacija. U narednim sekcijama opisani su formati metapodataka MARC 21, Dublin Core i ETD-MS.

3.1.1 MARC 21

MARC 21 (eng. *MAchine-Readable Cataloging for 21st century* - <http://www.loc.gov/marc/bibliographic/bdintro.html>) je kreiran u Kongresnoj biblioteci u Sjedinjenim Američkim državama. Ovaj format čine sledećih pet formata:

1. *MARC 21 Format for Bibliographic Data* - za bibliografske podatke (www.loc.gov/marc/bibliographic/ecbdhome.html),
2. *MARC 21 Format for Authority Data* - za normativne podatke (www.loc.gov/marc/authority/ecadhome.html),
3. *MARC 21 Format for Holdings Data* - za podatke o fondovima (www.loc.gov/marc/holdings/echdhome.html),
4. *MARC 21 Format for Classification Data* - za klasifikacione podatke (www.loc.gov/marc/classification/eccdhome.html),
5. *MARC 21 Format for Community Information* - za informacije o zajednicama (www.loc.gov/marc/community/ecihome.html).

Specifikacija MARC zapisa sastoje se od tri elementa: strukture zapisa, oznake sadržaja i sadržaja u podacima zapisa. Struktura zapisa je implementacija standarda *American National Standard for Information Interchange (ANSI/NISO Z39.2)* i njegovog ISO ekvivalenta *ISO 2709*. Oznake sadržaja su kodovi i konvencije eksplicitno uspostavljene da identifikuju i dalje karakterišu elemente podataka unutar zapisa, kao i da podrže rad sa ovim podacima. Definišu se u svakom od pomenutih formata. Sadržaj elemenata podataka koji čine MARC zapis definiše se obično standardima izvan ovih formata kao što su *International Standard Bibliographic Description (ISBD)*, *Anglo-American Cataloguing Rules, 2nd edition 0028AACR 2*), *Library of Congress Subject Headings (LCSH)*, *Holding Statements Summary Level (ISO 10324)*, *American National Standard for Serial Holdings Statements (ANSI/NISO Z39.44)*, *Library of Congress Classification (LCC)* ili drugim konvencijama koje koristi organizacija koja kreira zapis.

Struktura zapisa se sastoji od *zaglavja* koje je prvo polje u svim MARC zapisima, nepromenljive dužine sa 24 karakterske pozicije koje obezbeđuje informacije za obradu zapisa; *direkto-rijuma* koji se sastoji od niza jedinica dužine 12 karakterskih pozicija koje sadrže oznaku polja, dužinu i početnu lokaciju svakog promenljivog polja unutar nekog zapisa; *promenljivih polja* - podaci u MARC 21 se organizuju u polja promenljive dužine od kojih se svako identificuje numeričkom oznakom polja od tri karaktera koja se memorišu u jedinicu direktorijuma za to polje. Postoje dva tipa promenljivih polja: *promenljiva kontrolna polja* koja su sa oznakama polja 00X i *polja za podatke promenljivih dužina* koja su sa oznakama polja 01X-8XX. U okviru polja za podatke promenljivih dužina koriste se dve vrste oznake sadržaja i to indikatorske pozicije koje se nalaze na početku svakog polja i kodovi potpolja tj. dva karaktera koja prethode svakom elementu podataka u polju.

Primer MARC 21 bibliografskog formata u skladu sa ISO 2709 standardom za jednu doktorsku disertaciju dat je na listingu 3.1. Ovaj primer pored jedinstvenog identifikatora disertacije (polja *001* i *003*), sadrži jezik na kome je pisana disertacija (polje *041*), podatke o autoru (*100*), naslov (*245*), datum odbrane (*260*), fizički opis disertacije (*300*), stečenu titulu (*502*), naučnu oblast (*650*), podatke o mentoru (polje *700* koje u potpolju *4* ima vrednost *ths*), podatke o članovima komisije (polja *700* koja u potpolju *4* imaju vrednost *exp*), podatke o predsedniku komisije (prvo polje *700* koje u potpolju *4* ima vrednost *exp*), podatke o instituciji na kojoj je disertacija odbranjena (polje *710*). Pored osnovnih podataka o autoru (*100*), mentoru (*700*), članovima komisije i predsedniku komisije (*700*), kao i o instituciji na kojoj je disertacija odbranjena (*710*), u potpolju *0* ovih polja nalazi se jedinstveni identifikatori normativnih zapisa koji sadrže detaljne podatke o osobi ili instituciji sa kojom je disertacija u vezi. Među osnovnim podacima o mentoru, članovima komisije i predsedniku komisije se pored prezimena i imena koje se skladište u potpolju *a* polja *700*, u potpolju *c* nalaze i podaci o tituli i zvanju, a u potpolju *g* naziv institucije mentora, člana komisije ili predsednika u momentu odbrane teze ili disertacije.

Listing 3.1: Primer bibliografskog MARC 21 zapisa po ISO 2709 standardu

3.1.2 Dublin core

Dublin Core (<http://dublincore.org>) je standard za reprezentaciju metapodataka. Nije jedini standard u ovoj oblasti, ali je jedan od najčešće korišćenih formata za opis i razmenu metapodataka između informacionih sistema koji mogu biti i različitih vrsta. Za razvoj ovog formata je zadužena DCMI (*Dublin Core Metadata Initiative*). Definiše osnovni skup elemenata i pripadajućih atributa za opis resursa, ali se ovaj skup može i proširivati.

Takođe, definiše i jednostavan i standardizovan skup konvencija za opisivanje resursa. Dublin Core ima dva nivoa

- *Simple Dublin Core*: 15 elemenata za opisivanje metapodataka
- *Qualified Dublin Core*: dodatna 3 elementa i mehanizam rafinacije metapodataka

Simple Dublin Core sadrži petnaest elemenata za opis resursa, i to su:

1. *title*,
2. *creator*,
3. *subject*,
4. *description*,
5. *publisher*,
6. *contributor*,
7. *date*,
8. *type*,
9. *format*,
10. *identifier*,
11. *source*,
12. *language*,
13. *relation*,
14. *coverage*,
15. *rights*.

Dublin Core nije vezan za konkretan format podataka, mogu se koristiti različiti formati, a obično se koristi XML gramatika. Dublin Core definiše XML namespace sa elementima koji odgovaraju Dublin Core elementima. Ove elemente možemo koristiti u okviru sopstvene XML gramatike ili nekog opštег standarda.

Svih petnaest elemenata *Simple Dublin Core*-a imaju tekstualnu vrednost, ponovljivi su i opcioni. Ukoliko se elementi ponavljaju definiše im se *atribut lang* koji definiše jezik na kojem je sadržaj elementa. Ovaj atribut je u potpunosti odvojen od elementa *language* koji definiše jezik resursa, odnosno jezik na kojem je napisan dokument koji je opisan Dublin Core formatom. Široka primena petnaest elemenata Dublin Core formata počinje kada je ovaj format postao standardni format podataka unutar OAI-PMH (eng. *Open Archives Initiative Protocol for Metadata Harvesting*). Format podataka je ratifikovan kao IETF RFC 5013, ANSI/NISO Standard Z39.85-2007 i ISO Standard 15836:2009.

Primer Dublin Core zapisa za jednu doktorsku disertaciju koji se može koristiti za razmenu podataka putem OAI-PMH protokola je dat na listingu 3.2.

```
<oai_dc:dc
  xmlns=...>
  <dc:title>
    Kreiranje i korišćenje digitalnih dokumenata pravne
    regulative
  </dc:title>
  <dc:creator>Gostojić Stevan</dc:creator>
  <dc:contributor>Milosavljević Branko</dc:contributor>
  <dc:date>2012-09-21</dc:date>
  <dc:type>Text</dc:type>
  <dc:identifier>
    http://cris.uns.ac.rs/?recordId=77122
  </dc:identifier>
  <dc:language>Serbian</dc:language>
</oai_dc:dc>
```

Listing 3.2: Primer Dublin Core zapisa

Kao što je već rečeno *Qualified Dublin Core* definiše tri nova elementa i mogućnost rafinacije elemenata. Novi elementi su:

- *audience*,
- *provenance*,
- *rightsHolder*.

Mehanizmima rafinacije elemenata mogu se dodatno opisati neki elementi. Primer za rafinaciju je *alternative* element koji

je definisan kao podelement elementa *title* i predstavlja njegov alternativni naslov ili prevod naslova.

3.1.3 ETD-MS

ETD-MS (eng. *Electronic Theses and Dissertations Metadata Standard* - www.ndltd.org/standards/metadata) je *proširenje Dublin Core formata* sa novim elementima. Standard definiše skup metapodataka koji se koristi za opis magistarske teze ili doktorske disertacije. Metapodaci ovog standarda opisuju autora, njegov rad i kontekst u kojem je to delo nastalo na način koji će biti koristan ne samo istraživaču već i bibliotekaru i/ili tehničkom osoblju zaduženom za održavanje rada u elektronskoj formi. Ovaj format se jedino koristi kao format za opis metapodataka teza i disertacija i koristi se unutar svetske mreže digitalnih biblioteka teza i disertacija NDLTD (www.ndltd.org). Novi elementi kojim ovaj format proširuje Dublin Core su:

- *thesis.degree.name* - sadrži naziv stepena koji je vezan sa radom na primer *Diplomirani inženjer - master elektrotehnike i računarstva*.
- *thesis.degree.level* - nivo obrazovanja koji je u vezi sa dokumentom. Moguće su tri vrednosti: 0 (Osnovne), 1 (Master), 2 (Doktorske).
- *thesis.degree.discipline* - oblast kojom se bavi teza ili disertacija.
- *thesis.degree.grantor* - sadrži naziv institucije na kojoj je teza ili disertacija odbranjena i koja se pojavljuje na koricama istih.

Pored standardnih atributa Dublin Core formata, ETD-MS format dodaje i sledeće atrbute:

- *translated* - ovaj atribut ukazuje da je vrednost elementa prevod preveden od strane nekog ko nije autor. Za prevod dobijen od autora se koristi *lang* atribut.

- *scheme* - ovaj atribut omogućava opis rečnika ili šeme korišćene za definisanje ključnih reči unutar elementa *subject*.
- *resource* - ovaj atribut sadrži URI ka podacima o osobi ili instituciji: *creator*, *contributor*, itd.

Kod Dublin Core formata svi elementi su opcioni, dok su kod ETD-MS formata neki elementi obavezni: *title*, *creator*, *subject*, *date*, *type*, *identifier*. Kod Dublin Core formata svi elementi su ponovljivi, dok kod ETD-MS formata samo element *date* nije ponovljiv, a svi ostali jesu. Metapodaci o tezama i disertacijama mogu da se izvoze/uvoze u ETD-MS formatu putem OAI-PMH protokola što se i koristi unutar NDLTD mreže. XML šema ETD-MS formata dostupna je na sajtu: www.ndltd.org/standards/meta-data/etdms/1.0/etdms.xsd. Primer jednog zapisa u ETD-MS formatu je dat na listingu 3.3.

```
<thesis
  xmlns="http://www.ndltd.org/standards/meta-data/etdms/1.0/etdms.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <title>
    Kreiranje i korišćenje digitalnih dokumenata pravne
    regulative
  </title>
  <creator
    resource="http://cris.uns.ac.rs/?recordId=446">
    Gostojić Stevan
  </creator>
  <contributor role="advisor">
    Milosavljević Branko
  </contributor>
  <date>2012-09-21</date>
  <type>Text</type>
  <identifier>
    http://cris.uns.ac.rs/?recordId=77122
  </identifier>
  <language>Serbian</language>
  <degree>
    <level>PhD (dr)</level>
    <grantor
      resource="http://cris.uns.ac.rs/?recordId=5928">
      Fakultet tehničkih nauka u Novom Sadu,
      Univerzitet u Novom Sadu
    </grantor>
  </degree>

```

```
</degree>  
</thesis>
```

Listing 3.3: Primer ETD-MS zapisa

3.2 Protokoli

Pored standardnih funkcija, skupova metapodataka i njihovih formata, u oblasti upravljanja digitalnim dokumentima postoje i standardizovani **protokoli**. Ovi protokoli se mogu podeliti na dve vrste. Prva vrsta protokola se odnosi na protokole za razmenu podataka, a druga vrsta na protokole za pretraživanje udaljene kolekcije, odnosno kolekcije koja se ne nalazi na istom računaru kao i program putem kojeg se zadaje upit. U narednim sekcijama su opisani neki od predstavnika ovih protokola.

3.2.1 Razmena podataka

Razmena podataka između sistema putem određenih protokola omogućava jednu vrstu interoperabilnosti sistema. Postoje standardizovani **protokoli za razmenu podataka**, ali sa druge strane postoje i nestandardizovani protokoli koji su razvijeni za specifične potrebe razmene podataka između dva sistema. U slučaju sistema za upravljanje digitalnim dokumentima mogu se razmenjivati i dokumenti zajedno sa metapodacima koji ih opisuju, ali je mnogo zastupljeniji slučaj da se razmenjuju samo metapodaci i da se među tim metapodacima nalazi i metapodatak koji sadrži URL ka digitalnom dokumentu koji ostaje u svom izvornom repozitorijumu. Na ovaj način se briga o održavanju i pravima pristupa ka digitalnom dokumentu ostavlja izvornom repozitorijumu, a razmenom metapodataka se omogućuje vidljivost ovog digitalnog dokumenta iz drugih sistema i samim tim digitalni dokument postaje lakši za pronalaženje. Protokol koji je dominantan u ovakovom pristupu je OAI-PMH protokol koji će biti opisan u nastavku ove sekcije.

OAI-PMH (eng. *The Open Archives Initiative Protocol for Metadata Harvesting* - www.openarchives.org/pmh) je protokol koji omogućuje razmenu podataka između sistema. Postoje dve klase učesnika u OAI-PMH i to su: *Data Provider* - serverska strana protokola koja daje metapodatke iz svog repozitorijuma i *Service Provider* - klijentska strana protokola koja preuzima podatke. *Service Provider* inicira komunikaciju, tj. izdaje zahteve na koje *Data Provider* odgovara. *Service Provider* preuzete metapodatke koristi za pružanje različitih vrsta servisa: pretraga, pregledanje, izveštavanje, itd. OAI-PMH zahtev se zadaje u formi HTTP GET ili POST zahteva. Vrednost parametra *verb* u HTTP zahtevu sadrži naziv OAI-PMH zahteva, a parametri OAI-PMH zahteva se prosleđuju takođe kao odgovarajući parametri u HTTP zahtevu. OAI-PMH odgovor je u formi XML zapisa koji je upakovan u HTTP odgovor. HTTP odgovor sadrži zaglavlje u kojem je specificiran Content-type: text/xml, nakon kojeg sledi XML odgovor koji je u skladu sa XML šemom dostupnom na adresi www.openarchives.org/OAI/2.0/OAI-PMH.xsd.

OAI-PMH protokol definiše šest vrsta zahteva:

1. *Identify* - se koristi za preuzimanje informacija o repozitorijumu.
2. *GetRecord* - se koristi za preuzimanje pojedinačnih metapodataka zapisa iz repozitorijuma pri čemu se moraju nавести zahtevani argumenti kao što su identifikator objekta traženog zapisa i format metapodataka u kojem se zahteva odgovor.
3. *ListMetadataFormats* - koristi se za preuzimanje formata metapodataka dostupnih u repozitorijumu.
4. *ListRecords* - koristi se za prikupljanje liste zapisa iz repozitorijuma. Zahtevani argument je prefiks metapodataka, čijim unosom će biti vraćeni samo oni zapisi koji postoje u repozitorijumu u zahtevanom formatu.
5. *ListIdentifiers* - ovo je skraćeni oblik ListRecords za preuzimanje zaglavlja zapisa. Zahtevani argument je prefiks metapodataka.

6. *ListSets* - koristi se za preuzimanje liste skupova repozitorijuma što je korisno za selektivno prikupljanje metapodataka.

OAI-PMH protokol podržava nekoliko vidova selekcije zapisa koji se preuzimaju. *Set* je logički skup objekata grupisanih po nekom kriterijumu. Repozitorijumi mogu organizovati objekte u skupove. Organizacija skupova može biti ravna ili hijerarhijska. *ResumptionToken* je argument kod zahteva koji vraćaju listu rezultata sa vrednošću koja ukazuje koji deo liste je preuzet, odnosno u slučaju novog zahteva daje informaciju odakle nastaviti preuzimanje podataka. Dakle, na ovaj način je obezbeđena preuzimanje liste rezulata u segmentima. Takođe, moguće je definisati *vremenski opseg*, početni i krajnji datum, da bi se dobili samo zapisi koji su u ovom periodu dodati ili menjani u repozitorijumu. Naravno, svi navedeni vidovi selekcije se mogu kombinovati, odnosno koristiti zajedno.

OAI-PMH protokol je vrlo popularan i pomoću ovog protokola je kreirano nekoliko popularnih mreža digitalnih biblioteka ili institucionalnih repozitorijuma kao što su: *NDLTD*, *DART-Europe*, *OPENAire*, *OATD*. Cilj formiranja ovih mreža je da se poveća vidljivost digitalnih dokumenata koji se nalaze u repozitorijumima koji su članovi mreža. Sve ove mreže imaju jedan centralni čvor i stotine perifernih čvorova koji sadrže digitalne dokumente. Član mreže se postaje tako što se omogući pristup metapodacima digitalnih dokumenata koji se skladište u repozitorijumu perifernog čvora putem OAI-PMH protokola. OAI kompatibilnost se pre uključenja repozitorijuma u mrežu proverava upotrebom neke aplikacije za ovu namenu kao što je <http://re.cs.uct.ac.za/>. Server centralnog čvora periodično preuzima metapodatke o digitalnim dokumentima koji se nalaze u perifernim čvorovima mreže. Među metapodacima pored naslova, autora i ostalih osnovnih podataka nalazi se i URL ka digitalnom dokumentu u perifernom čvoru. Ovako formirani centralni katalog se obično može pretraživati putem namenski razvijene aplikacije kao što je www.dart-europe.eu/basic-search.php. Ovakav pristup pretrage mreže repozitorijuma ima sledeće nedostatke:

- Ažurnost podataka koji se pretražuju - server *periodično* preuzima podatke.
- Od kvaliteta implementacije namenski razvijene aplikacije za pretragu zavisi rezultat pretraga - tamo se vrši pretprocesiranje teksta, pitanje je da li imaju dobru podršku za srpski jezik i da li im je pretraživanje nezavisno od dva ravnopravna pisma u Srbiji (ćirilica i latinica).
- Ne može se pretraživati mreža po sadržaju digitalnih dokumenata, već samo po metapodacima.

3.2.2 Pretraživanje udaljene kolekcije

Kada se želi napraviti pretraživanje mreže repozitorijuma to se može uraditi na dva načina:

- kreiranjem centralnog repozitorijuma koji periodično preuzima podatke od članova mreže putem nekog protokola za razmenu podataka kao što je prethodno opisani OAI-PMH,
- korišćenjem **protokola za udaljeno pretraživanje** repozitorijuma.

Kao što je već rečeno prvi pristup ima sledeće nedostatke:

- Ažurnost podataka koji se pretražuju - centralni repozitorijum *periodično* preuzima podatke.
- Od kvaliteta implementacije namenski razvijene aplikacije za pretragu centralnog repozitorijuma zavisi rezultat pretraga - tamo se vrši pretprocesiranje teksta, pitanje je da li imaju dobru podršku za srpski jezik i da li im je pretraživanje nezavisno od dva ravnopravna pisma u Srbiji (ćirilica i latinica).
- Ne može se pretraživati mreža po sadržaju digitalnih dokumenata, već samo po metapodacima koji se izvoze u centralni repozitorijum.

Drugi pristup otklanja ove nedostatke, ali sa druge strane ima i on svojih nedostataka:

- *performanse* - sistem zavisi od mrežnog protoka i od brzine kojom udaljeni server daje odgovor,
- *otkaz udaljenog servera*,
- *sortiranje* rezultata dobijenih sa različitih udaljenih kolekcija i *eliminacija duplikata*.

U ovoj sekciji su opisana dva protokola za pretraživanje udaljene kolekcije: *Z39.50 protokol*, i njegov naslednik *SRU protokol*.

Z39.50 protokol je binarni klijent-server protokol koji koristi transportni sloj Interneta TCP/IP za komunikaciju klijenta i servera i ne zavisi od platforme. Osnovna funkcionalnost protokola Z39.50 je da uspostavi komunikaciju između klijentske i serverske aplikacije, zatim da izvrši zahtev za pretraživanje i na kraju da vrati formatiranu listu rezultata pretrage. Ovaj protokol grupiše određene koncepte protokola u mehanizme (eng. *facilities*): mehanizam za inicijalizaciju veze, mehanizam za pretraživanje, mehanizam za prezentaciju rezultata pretrage, ostali mehanizmi. Z39.50 omogućuje bogat upitni jezik koji omogućava korišćenje bulovih izraza, skraćivanje reči, kao i izbor naprednih opcija za pretragu. Jedan od nedostataka ovog protokola je relaksirana specifikacija protokola koja je ostavila mogućnost da se određeni mehanizmi ne moraju implementirati, kao i da se mogu implementirati na različite načine. Detaljna specifikacija protokola se može pronaći na www.loc.gov/z3950/agency.

SRU protokol je XML orijentisani naslednik Z39.50 protokola. Nastao je kao pokušaj da se premosti jaz koji je nastao između klasične klijent-server implementacije koja se koristi u specifikaciji Z39.50 protokola i standarda koji su se razvili i postali široko prihvaćeni ekspanzijom Interneta. Za transport podataka SRU koristi XML dokumente, a za specifikaciju upita koristi se CQL (eng. *Contextual Query Language*). Pošto je SRU nastao sa ciljem da obezbedi istu funkcionalnost kao i Z39.50, većina mehanizama i mogućnosti koje pruža Z39.50 preslikana

je i na SRU protokol. Ključni napredak je što se za komunikaciju u SRU protokolu koriste otvoreni i široko prihvaćeni standardi. Detaljna specifikacija SRU protokola može se pronaći na www.loc.gov/standards/sru

Rezime

- ISO IEC 82045 je standard za uređenje sistema za upravljanje dokumentima, sastoji se iz delova, ne mora biti podržan u potpunosti, odnosno sam standard definiše nekoliko nivoa (klasa) podrške u različitim aspektima.
- Postoje različiti standardizovani formati metapodataka koji se mogu koristiti u sistemima za upravljanje dokumentima.
- Bibliotečki format MARC 21 je kreiran u Kongresnoj biblioteci u Sjedinjenim Američkim državama. MARC 21 zapis se sastoji od tri elementa: strukture zapisa, oznake sadržaja i sadržaja u podacima zapisa.
- Dublin Core je jedan od najčešće korišćenih formata za opis i razmenu metapodataka između istih ili različitih informacionih sistema koji su ga implementirali. Definiše osnovni skup elemenata i pripadajućih atributa za opis resursa, ali se ovaj skup može i proširivati. Takođe, definiše i jednostavan i standardizovan skup konvencija za opisivanje resursa.
- ETD-MS je proširenje Dublin Core formata sa novim elementima. Standard definiše skup metapodataka koji se koristi za opis završnih radova (magistarске/master teze ili doktorske disertacije).
- Standardizovani protokoli u oblasti upravljanja digitalnim dokumentima se mogu podeliti u dve vrste: protokole za razmenu podataka i protokole za pretraživanje udaljene kolekcije. Predstavnik prve vrste je OAI-PMH protokol, a predstavnik druge vrste je Z39.50 i njegov naslednik SRU protokol.

Pitanja

1. Šta je ISO IEC 82045 standard?
2. Navesti neke standardizovane formate metapodataka i opisati ih?
3. Šta su protokoli za razmenu podataka?
4. Koje su osnovne karakteristike OAI-PMH protokola?
5. Šta su protokoli za udaljeno pretraživanje?
6. Koje su osnovne karakteristike Z39.50 i SRU protokola?

Glava 4

Pronalaženje informacija

Ovo poglavlje ima za cilj da uvede čitaoca u osnovne pojmove u oblasti pronalaženja informacija i da odgovori na sledeća pitanja:

- Čime se bavi oblast pronalaženja informacija?
- Kako se pronalaženje informacija nestrukturiranih sadržaja razlikuje od pronalaženja podataka koji imaju jasnu strukturu i koji su skladišteni u bazama podataka?
- Kako se kroz istoriju razvijala oblast pronalaženja informacija i sistemi za pretragu?

Prilikom kreiranja strukture ovog poglavlja i definisanja osnovnih pojmoveva polazna osnova bile su knjige „*Modern Information retrieval*“ [37] i „*Introduction to information retrieval*“ [38]. Deo informacija prezentovanih u ovom poglavlju su preuzeti iz raznih izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavljiju.

Pronalaženje informacija (eng. *information retrieval*) je oblast koja se bavi tehnikama za reprezentaciju, skladištenje, organizaciju, pristup i pronalaženje informacija. Ove tehnike treba da obezbede pronalaženje materijala (najčešće dokumenata) nestrukturirane prirode (najčešće tekstualnih) u okviru velike kolekcije koji zadovoljava korisnikove potrebe za informacijama. Sistemi za pronalaženje informacija najčešće imaju odvojene procese *indeksiranja* i *pretraživanja*. **Indeksiranje** predstavlja pripremu informacija za efikasno pretraživanje i uključuje tehnike za reprezentaciju, skladištenje i organizaciju informacija. **Pretraživanje** predstavlja proces obrade upita i pronalaženje informacija koje korisnik traži. Pretraživanje uključuje tehnike za efikasan pristup i pronalaženje informacija u prethodno kreiranim indeksima u procesu indeksiranja.

Kao što je već rečeno u drugom poglavlju u standardnim poslovnim aplikacijama podaci se čuvaju u relacionim bazama podataka. Sistemi za upravljanje bazama podataka se bave tehnikama za reprezentaciju, skladištenje, organizaciju i pristup podacima koje treba da obezbede skladištenje i pronalaženje podataka u okviru baze podataka.

Koja je razlika između **pronalaženja podataka** u bazama podataka (eng. *data retrieval*) i pronalaženja informacija (eng. *information retrieval*) u velikim kolekcijama sadržaja koji nisu "čvrsto" strukturirani?

Data retrieval se bavi pronalaženjem podataka koji zadovoljavaju precizno definisan kriterijum pri čemu se očekuje od korisnika da poznaje strukturu podataka u bazi koju pretražuje. Podaci u kolekciji koja se pretražuje imaju dobro definisanu strukturu i semantiku. *Data retrieval* samo određuje koji zapisi u kolekciji sadrže ključne reči koje je korisnik naveo u upitu, što često dovodi do toga da korisnik ne može da pronađe ono što želi.

Kod **information retrieval**-a se od korisnika ne očekuje da poznaje kolekciju koju pretražuje, korisnika interesuju informacije o nekoj temi, a ne podaci koji zadovoljavaju upit. Podrazumeva se *nepreciznost* u zadavanju upita, kao i nepreciznost u listi pronađenih rezultata, odnosno u listi rezultata na postavljeni

upit se mogu pojaviti dokumenti koji nisu relevantni za korisnika, odnosno koji ne zadovoljavaju korisnikovu *informacionu potrebu*. Razlog za ovu nepreciznost je činjenica da su informacije u dokumentima i upitima iskazane prirodnim jezicima zbog čega mogu biti semantički neprecizne ili više značne. Pored toga što liste rezultata sadrže nerelevantne dokumente često su ove liste i dugačke pa je *neophodno rangiranje rezultata* koje će obezbediti da su više relevantni dokumenti na početku liste. Dakle, od sistema za pronalaženje informacija se očekuje da nekako “interpretira” sadržaj dokumenta da bi mogao da odredi stepen relevantnosti dokumenta u odnosu na korisnikovu informacionu potrebu i da to iskoriste za sortiranje rezultata pretrage. Ova “interpretacija” sadržaja dokumenta uključuje ekstrahovanje sintaksnih i semantičkih informacija iz dokumenta i iz upita korisnika, kao i upotrebu ovih informacija da se utvrdi da li dokument odgovara informacionoj potrebi korisnika. *Relevantnost dokumenta* je centralni pojam u pronalaženju informacija. Cilj svakog sistema za pronalaženje informacija (eng. *information retrieval system*) je da pronađe sve relevantne dokumente za informacionu potrebu korisnika, a da pri tome u listi rezultata bude što je moguće manje nerelevantnih dokumenata - idealan slučaj je da ih nema, ali je on najčešće teško ostvariv. U nekim sistemima i u nekim modelima pretraživanja informacija stepen relevantnosti je binarna vrednost, dokument je ili relevantan (stepen relevantnosti je 1) ili nije relevantan (stepen relevantnosti je 0). U ovom slučaju nemamo mogućnost sortiranja rezultata pretrage što može da utiče da stepen zadovoljstva korisnika rezultatima pretrage.

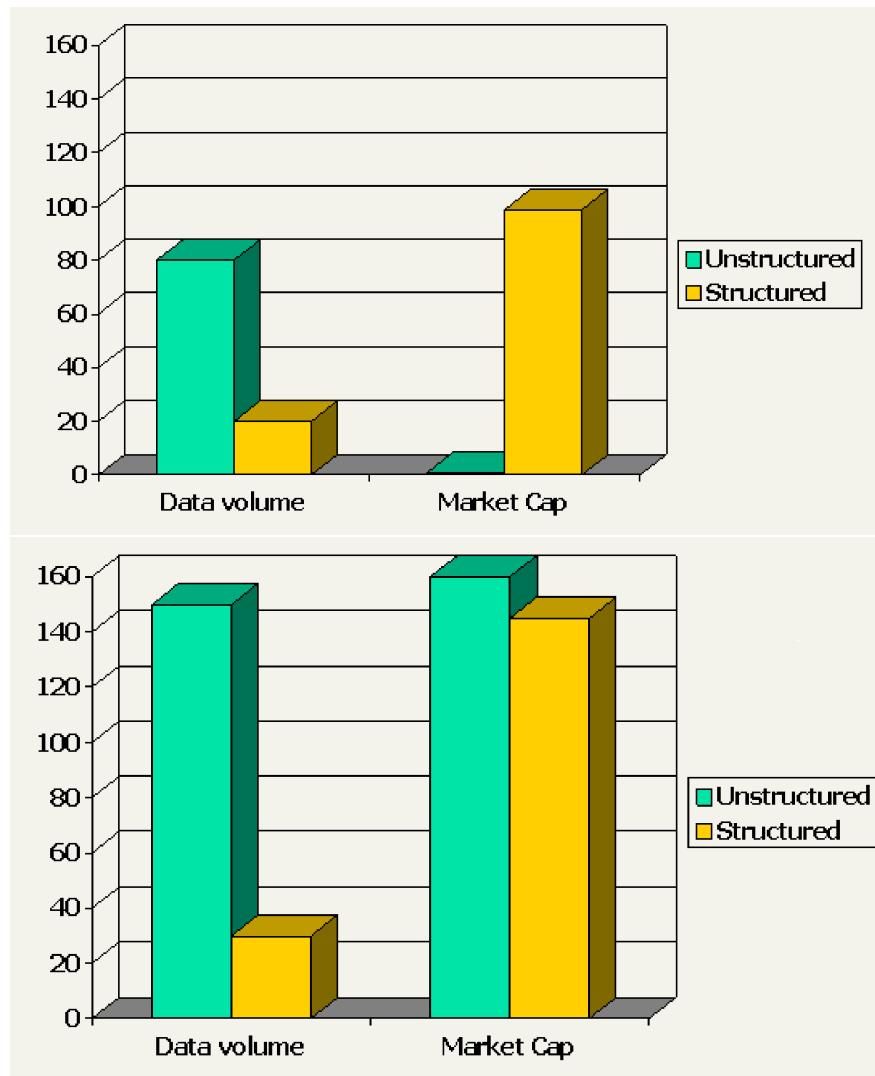
4.1 Istorija razvoja

Organizacionom informacija za kasnije pronalaženje i upotrebu ljudi se bave poslednjih 4.000 godina [37]. Tipičan primer organizacije informacija je pravljenje sadržaja u knjizi (eng. *table of contents*). Kako su veličine knjiga rasle, tako je rasla i količina informacija koje se mogu pronaći u jednoj knjizi, pa je često pravljenje sadržaja bilo nedovoljno dobra organizacija za pretragu

informacija. Zbog toga su uvedeni indeksi pojmove koji se često navode u knjigama. Kako je rastao broj knjiga, tako se javljao i problem da je postajalo nemoguće otvarati svaku knjigu i u njoj čitati sadržaj i indeks pojmove da bi se pronašla određena informacija. Počeli su da se kreiraju indeksi na nivou biblioteke knjiga. Ovi indeksi su dugi niz godina ručno kreirani od strane bibliotekara sve do pojava računara i razvoja IKT kada se deo ovih poslova počeo obavljati upotrebotom računara. Indeksi su bili kreirani u formi hijerarhijski organizovanih klasa. Dakle, oblast pronalaženja informacija, odnosno *pretraga dokumenata je nastala pre pojave računara* i za razvoj ove oblasti najzaslužnije su biblioteke. Naravno, pretraga digitalnih dokumenata je nastala nakon pojave računara.

Nekada je pronalaženje informacija bila oblast koja je bila interesantna samo za oblast bibliotekarstva. Ulaskom u digitalno doba, razvojem IKT i razvojem veba, oblast pronalaženja informacija dobija na popularnosti. Ulaskom u digitalno doba, iz godine u godinu je količina nestrukturiranih sadržaja sve veća (eng. *data volume*), kao i tržišna vrednost ovih sadržaja (eng. *market cap*). Nekada je tržišna vrednost strukturiranih podataka skladištenih u bazama podataka bila dominantna, odnosno podaci koji su bili značajni za poslovanje su gotovo isključivo bili u formi strukturiranih podataka. Na slici 4.1 gornji grafikon prikazuje situaciju iz 1996., a donji iz 2006. godine [39]. Povećanje tržišne vrednosti nestrukturiranih podataka je uticao na značaj pronalaženja informacija u ovim podacima.

Za razvoj oblasti pronalaženja informacija zaslužan je i razvoj veba (eng. *World Wide Web*). Veb je postao univerzalni repozitorijum ljudskog znanja i kulture koji omogućava novi nivo deljenja ideja i informacija u evoluciji čovečanstva. Postoji nekoliko razloga za eksponencijalni razvoj veba koji se desio u poslednje dve decenije. Prvo, korisnici ne moraju poznavati protokol koji se koristi, ne moraju znati na kom se zapravo računaru nalazi sadržaj koji im treba i korisnici mogu koristiti različite operativne sisteme. Drugo, bilo koji korisnik može kreirati veb sadržaj i linkovati (povezati) ga sa bilo kojim drugim veb sadržajem.



Slika 4.1: Strukturirani i nestrukturirani podaci 1996. i 2006. godine

Dakle, veb je danas medijum za publikovanje koji je dostupan svima. Zbog ove dve činjenice veb je privukao na milione korisnika koji ga danas koriste svakodnevno. Kada su korisnici već bili tu, počele su da se pojavljuju veb aplikacije koje omogućuju korisnicima ne samo da pregledaju veb sadržaje, nego i da obave

određene transakcije: kupovina putem veba, e-banking, itd. Povjavljuju se i društvene mreže, kao i veb aplikacije za skladištenje i pregledanje multimedijalnih sadržaja. Nekoliko godina nakon svog nastanka veb je zbog svoje popularnosti toliko narastao u svojim sadržajima da je hiperprostor (eng. *hyperspace*) postalo nemoguće pregledati putem linkova između veb sadržaja. Pojavila se potreba za nastanak veb pretraživača (eng. *Web search engine*).

Zašto je veb popularan izvor informacija?

- pristup vebu je jeftin (gotovo besplatan),
- razvoj IKT je doprineo da se vebu može pristupiti sa različitih mesta upotrebom različitih uređaja,
- sloboda u publikovanju, svako može publikovati šta misli da je značajno na veb - u nekim situacijama ovo je vrlo loša strana veba kao izvora informacija.

Oblast pronalaženja informacija je oblast koja se danas izучava na većini fakulteta čiji se studijski programi bave računarskim naukama. Danas imamo situaciju da je *Google* jedna od najmoćnijih svetskih kompanija, a njena osnovna oblast delovanja je pronalaženje informacija. Oblast pronalaženja informacija je danas multidisciplinarna oblast, više se ne bavi samo indeksiranjem i pretraživanjem. Ova oblast sada uključuje modelovanje, klasifikaciju i klasterovanje dokumenata, korisnički interfejs, vizualizaciju podataka, arhitekturu sistema, prepoznavanja karaktera, semantički veb, itd.

4.2 Razvoj sistema za pretragu

Kako je pronalaženje informacija kao oblast nastala u bibliotekarstvu, logično je da su biblioteke bile prve koje su imale implementirane računarski podržane sisteme za pretragu. Prvi sistemi ove vrste bili su razvijani u naučnim institucijama, na univerzitetima i naučnim institutima, a kasnije su se u razvoj

ovih sistema uključili i komercijalni proizvođači. U prvoj generaciji ovih sistema samo je bila podržana evidencija kataloških kartica i pretraga po imenu autora i naslovu dela. Nakon toga, u narednoj generaciji bibliotečkih informacionih sistema pojavila se i mogućnost pretrage kolekcije po ključnim rečima, oblasti kojoj knjiga pripada i pojavile su se neke naprednije mogućnosti pretrage: kombinovanje upita, unos samo početnih slova reči, itd. Treća, trenutno aktuelna generacija značajno unapređuje korisnički interfejs i mogućnost interakcije korisnika sa sistemom prilikom pretrage kolekcije. Takođe, uključuje rad sa digitalnim dokumentima i tehnike za automatsko ekstrahovanje teksta iz digitalnih dokumenata, kao i tehnike za analizu dobijenih tekstova.

Pretraživači veba su nastali krajem 90-tih godina 20. veka. Ovi pretraživači rešavaju nekoliko novih problema sa kojima se susreću koji se razlikuju od sistema za pretragu biblioteka: velika količina sadržaja, ne postoji kontrola u izmeni sadržaja koji se menja u svakom momentu sa različitim mesta i kroz različite veb aplikacije, kako održati indekse ažurnim, pojavljuje se i potreba da se analiziraju i veze između dokumenata i da se ove informacije takođe koriste prilikom rangiranja rezultata pretrage.

Sa stanovišta arhitekture sistemi za pronalaženje informacija se dele na:

- centralizovane i
- distribuirane.

Pri ovoj podeli ne misli se na način skladištenja digitalnih sadržaja koji se pretražuju, nego na način skladištenja indeksa koji se koriste prilikom pretrage i na način obrade upita. Pretraživači veba imaju složene memorijske zahteve po pitanju indeksa koji se ne mogu čuvati na jednom računaru. Takođe, ovi sistemi imaju i veliki broj korisnika i visoke zahteve po pitanju brzine dobijanja odgovora, tako da im nije dovoljna procesorska moć jednog računara. Pretraživači veba nisu jedini sistemi za pretragu informacija koji koriste distribuiranu arhitekturu za pronalaženje informacija. Za komunikaciju između računara koji formiraju ovakav

distribuirani sistem mogu se koristiti standardizovani protokoli koji su opisani u poslednjoj sekciji prethodnog poglavlja SRU i z39.50, ili neki nestandardizovani protokol razvijen za potrebe jednog distribuiranog sistema.

Oblast pronalaženja informacija je razvojem IKT dobila i novu namenu, ova oblast se više ne bavi samo tekstualnim dokumentima. Prema sadržajima u kolekciji koji se pretražuju imamo sledeću podelu:

- pretraga tekstualnih sadržaja:
 - nestrukturiranih sadržaja,
 - strukturiranih tekstualnih sadržaja koji iako imaju strukturu u nekim poljima svoje strukture imaju velike količine tekstova,
- Pretraga linkovanih tekstualnih sadržaja (pretraga veba),
- Pretraga multimedijalnih sadržaja: uključuje sliku, zvuk, video.
- Pretraga ostalih vrsta sadržaja:
 - kolekcija programskih izvornih kodova,
 - kolekcija 3D objekata,
 - itd.

Sistemi za pretraživanje shodno svojoj nameni i svojim sadržajima mogu biti izgrađeni na različitim modelima pretraživanja:

- **Klasični modeli :**

- Bulov model,
- Vektorski model,
- Probabilistički model,

- **Alternativni modeli :**

- Prošireni Bulov model,
- Fuzzy model,

- Model neuronske mreže,
- Jezički model.

Tri navedena klasična modela se potpuno razlikuju po metodologiji i pristupu u pronalaženju informacija. Nastali su tokom evolucije *information retrieval-a* kao oblasti i sva tri imaju svojih prednosti i mana. Zavisno od sadržaja u kolekciji, od ciljne grupe korisnika i od namene pretraživača, određeni model može biti pogodniji za primenu od drugih. Alternativni modeli su modifikovane verzije klasičnih modela u pokušaju da se prevaziđu neka ograničenja i manjkavosti klasičnih modela. Obično se primenjuju kod kolekcija specifičnih sadržaja.

Bulov model pretraživanja je zasnovan na teoriji skupova i Bulovoj algebri. Traženi pojam se ili nalazi ili ne nalazi u dokumentu. Ovaj model je imao najveću popularnost u 20. veku zbog svoje jednostavnosti. Kako je količina sadržaja u sistemima rasla, tako je jedna velika mana ovog modela sve više dolazila do izražaja. Velika mana je što ovaj model nema mogućnost rangiranja rezultata, odnosno stepen slaganja upita i dokumenta je binarna vrednost. Dakle, nema parcijalnog poklapanja upita i dokumenta, dokument je ili relevantan (ocena relevantnosti je 1) ili nije relevantan (ocena relevantnosti je 0). Na primer, ako imamo konjukciju tri podupita (logičko I) jednako se posmatra i dokument koji nije u skladu ni sa jednim podupitom i dokument koji je u skladu sa dva podupita - oba dokumenta nisu relevantna, ocena relevantnosti je 0. Bulov model je detaljnije opisan u nadrenom poglavlju ove knjige (poglavlje 5.2).

Vektorski model pretraživanja pokušava da prevaziđe problem koji ima Bulov model tako što uvodi da je stepen slaganja upita i dokumenta vrednost koja je veća ili jednak nuli, ali nije celobrojna. Na primer, stepen slaganja može imati vrednosti 0, 12. Na osnovu ovako definisanog stepena slaganja može se vršiti rangiranje rezultata, jer stepen slaganja podržava parcijalno poklapanje upita i dokumenta. Model je zasnovan na vektorima u n-dimenzionalnom prostoru, pri čemu je broj dimenzija prostora obično jako veliki. Ugao koji zaklapaju vektori upita i dokumenta je obrnuto сразмерan relevantnosti dokumenta za po-

stavljeni upit. Vektorski model je detaljnije opisan u narednom poglavlju ove knjige (poglavlje 5.3).

Probabilistički model pretraživanja je zasnovan na teoriji verovatnoće. Polazi se od pretpostavke da za svaki postavljeni upit postoji idealan skup dokumenata. Proces pretrage je specificiranje osobina idealnog skupa dokumenata. U ovom procesu se počinje od nekog inicijalnog idealnog skupa i onda se kroz određeni broj koraka taj skup pomera ka konačnom skupu rezultata pretrage. Centralni problem ovog modela je kako odrediti inicijalni idealni skup od kog se kreće. Model pokušava da proceni verovatnoću da će korisnik smatrati određeni dokument relevantnim. U određenim slučajevima ovaj model je efikasniji od vektorskog modela, ali u velikim kolekcijama slobodnog sadržaja obično je vektorski model bolji. Postoji više vrsta probabilističkih modela koji se razlikuju u načinima računanja verovatnoća i određivanju inicijalnog idealnog skupa. Više detalja o dva probabilistička modela može se pronaći u radovima [40] i [41].

Rezime

- Pronalaženje informacija (eng. *information retrieval*) je oblast koja se bavi tehnikama za reprezentaciju, skladištenje, organizaciju, pristup i pronalaženje informacija koje treba da obezbede pronalaženje materijala (najčešće dokumenata) nestrukturirane prirode (najčešće tekstualnih) u okviru velike kolekcije koji zadovoljava korisnikove potrebe za informacijama.
- Od korisnika se ne očekuje da poznaje kolekciju koju pretražuje, korisnika interesuju informacije o nekoj temi, a ne podaci koji zadovoljavaju upit.
- Podrazumeva se nepreciznost u zadavanju upita, kao i nepreciznost u listi pronađenih rezultata.
- Cilj svakog sistema za pronalaženje informacija je da pronađe sve relevantne dokumente za informacionu potrebu korisnika, a da pri tome u listi rezultata bude što je moguće manje nerelevantnih dokumenata (idealni slučaj je da ih nema, ali je on najčešće teško ostvariv).
- Organizacijom informacija za kasnije pronalaženje i upotrebu ljudi se bave poslednjih 4.000 godina.
- Za inicijalni razvoj ove oblasti najzaslužnija je oblast bibliotekarstva, a oblast je doživela svoju ekspanziju razvojem veba.
- Prvi računarski podržani sistemi za pretragu su se pojavili u bibliotekama, a danas ih imamo svuda uključujući i pretraživače veba.
- Sistemi za pretraživanje mogu imati različitu arhitekturu, mogu se razlikovati po vrsti sadržaja koja se pretražuje, a mogu se razlikovati i po usvojenom modelu pretraživanja.

Pitanja

1. Čime se bavi oblast pronalaženja informacija (eng. *information retrieval*)?
2. Koja je razlika između pronalaženja podataka (eng. *data retrieval*) i pronalaženja informacija (eng. *information retrieval*)?
3. Kako se razvijala oblast pronalaženja informacija?
4. Kako su se razvijali sistemi za pretragu?
5. Kakve arhitekture mogu imati sistemi za pretragu?
6. Koje vrste sadržaja mogu biti pretraživane putem sistema za pretragu?
7. Koji modeli za pretraživanje se koriste u sistemima za pretragu?

Glava 5

Pretraga tekstualnih dokumenata

Ovo poglavlje ima za cilj da definiše osnovne pojmove pretrage tekstualnih digitalnih dokumenta i da odgovori na sledeća pitanja:

- Čemu služi pretprocesiranje teksta?
- Koje su osnovne razlike između dva dominantna modela za pretraživanje tekstualnih digitalnih dokumenata: Bulovog i Vektorskog modela?
- Kako se mogu pretraživati kolekcije strukturiranih i polustrukturiranih tekstualnih digitalnih sadržaja?

Prilikom kreiranja strukture ovog poglavlja i definisanja osnovnih pojnova polazna osnova bila je knjiga “*Introduction to information retrieval*” [38]. Deo informacija prezentovanih u ovom poglavlju su preuzeti iz raznih izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Ovo poglavlje se bavi **pretragom kolekcije tekstualnih digitalnih dokumenata**. Pre nego što pređemo na ostale aspekte pretrage razmotrimo prvo dva osnovna pojma:

- **tekstualni digitalni dokument** i
- **term.**

Oba ova pojma mogu biti *prilično složeni* i mogu uneti određene komplikacije u sistem za pretragu tekstualnih digitalnih dokumenata. Prva komplikacija mogu biti različiti *formati dokumenta*: pdf, doc, xls, odt, itd. Takođe, određenu komplikaciju mogu uvesti i različiti *jezici dokumenata* kao i različiti *kodni rasporedi* koji se koriste u kolekciji digitalnih dokumenata. Jedna kolekcija digitalnih dokumenata može sadržati dokumente pisane na različitim jezicima, odnosno sistem za pretraživanje ove kolekcije u kreiranom indeksu za pretragu sadrži reči iz različitih jezika. **Cross-language information retrieval** je podoblast *information retrieval*-a koja se bavi pretragama kod kojih je upit na drugom jeziku u odnosu na dokumente u kolekciji. **Multilingual information retrieval** je podoblast *information retrieval*-a koja se bavi pretragama kod kojih se u kolekciji nalaze dokumenti koji su na različitim jezicima.

Nekada jedan dokument, odnosno njegovi delovi koriste različite jezike i formate. Na primer, rezime knjige je preveden na engleski jezik, a sve ostalo je na srpskom jeziku. Situacija može biti i složenija, na primer email na francuskom sa PDF prilogom na španskom.

U nekim situacijama nije jasno ni šta bi trebalo da bude dokument u kolekciji, odnosno šta bi trebalo da je dokument koji se može pretraživati i biti rezultat pretrage:

- kompletan dokument,
- deo dokumenta koji predstavlja jedan član zakona,
- email poruka bez priloga,
- email poruka sa prilozima ili

- grupa fajlova, na primer PPT konvertovan u HTML stranice.

Situacija može biti još složenija i možemo razlikovati definiciju dokumenta koji se može pretraživati i dokumenta koji može biti rezultat pretrage. Na primer, može se pretraživati ceo dokument, ali se samo deo dokumenta može videti kao rezultat pretrage zbog autorskih prava.

Tekstualni digitalni dokumenti se sastoje od **reči** koje predstavljaju određeni broj znakova u tekstu. Instanca reči koja se pojavljuje u tekstu zove se **token** u okviru oblasti pronalaženja informacija. Jedna klasa ekvivalencije reči se zove **term**. Ako se reči nakon primene određenih pravila za **preprocesiranje teksta** svode na isti niz znakova, onda pripadaju istoj klasi ekvivalencije. Pravila za preprocesiranje teksta mogu biti različita i umnogome od ovih pravila zavisi kvalitet sistema za pretragu tekstualnih digitalnih dokumenata. Primer pravila je prebacivanje svih slova u mala slova, prebacivanje imenice u jedninu, prebacivanje imenica u isti padež, itd. Preprocesiranje teksta je detaljnije objašnjeno u narednoj sekciji. Na primeru sledeće rečenice možemo diskutovati razliku između tokena i terma:

U julu ћу ићи на одмор у Грчку, на ostrvo Krit.

Prethodna rečenica ima 11 tokena. U srpskom jeziku se reči razdvajaju znacima interpukcije ili belim znacima (eng. *white-space characters*). Broj termova zavisi od preprocesiranja teksta. Ako prebacujemo sva slova u mala onda su prvi token "U" i sedmi token "u" zapravo isti term. U svakom slučaju peti i deveti token "na" su jedan term.

5.1 Preprocesiranje teksta

Preprocesiranje teksta je primena određenih pravila na tekstualne sadržaje koji obezbeđuju fleksibilnost pretrage. Isto preprocesiranje teksta se primenjuje na tekstove u dokumentima

prilikom indeksiranja, kao i na reči u upitu prilikom pretraživanja. Ako je na primer jedno od pravila da se sva slova prebacuju u mala slova na ovaj način se postiže nezavisnost pretrage od velikih i malih slova (eng. *case insensitive search*), odnosno korisnik može kucati sve reči u upitu malim slovima ali će biti pronađeni i dokumenti koji sadrže te reči napisane velikim slovima.

Kao ulaz pretprocesiranje teksta ima određeni tekst (ili upit), a kao izlaz daje listu termova koje je potrebno dodati (ili pronaći) u listi pojava u indeksu za pretraživanje.

Na primer ulaz može biti:

Srbija je zemlja koja je šampion Evrope u vaterpolu, a nekada je bila i šampion sveta u košarci.

A izlaz, odnosno lista pojava, može biti:

srbiја bitи земља која шампион европе у ватерполо
а некада бити и света кошарци

Svaki token jeste kandidat za stavku u listi pojava, ali zavisno od pravila pretprocesiranja teksta neki će se tokeni pojaviti u listi pojava, a neki ne. U svakom slučaju prvi korak je ustanoviti listu tokena, odnosno **tokenizacija teksta**. Tokenizacija može biti vrlo složena, čak i za srpski jezik, a posebno za neke jezike koji nemaju jasno definisana pravila za razdvajanje reči.

Na primer, da li imate nekih dilema ako želite da tokenizujete sledeću rečenicu?

S druge strane, informacioni sistemi naučno-istraživačkog domena i e-obrazovanja često sadrže nestrukturirane ili polustrukturirane digitalne sadržaje.

Naredne fraze takođe nije lako tokenizovati, odnosno tokenizacija se može razlikovati zavisno od namene sistema i implementacije sistema za pretragu digitalnih dokumenata:

- eUprava,
- e-Uprava,
- Novi Sad,

- Banja Luka,
- baza podataka,
- pregled literature.

Datumi i brojevi telefona mogu biti zapisani u različitim oblicima, kako njih tokenizovati:

- 8/3/11
- 3/8/11
- Mart 8, 2011
- (987) 254-2765
- 987 / 254 2765

Stariji sistemi za pretraživanje ne stavljuju brojeve u liste pojava iako brojevi često nose određenu informativnost, odnosno u njima ima informacija čija pretraga značajno može povećati zadovoljstvo korisnika sistema. Noviji sistemi za pretraživanje najčešće ne izostavljaju brojeve u preprocesiranju i omogućuju pretragu po brojevima.

Kinesko pismo nema belih znakova (eng. *whitespace characters*) što unosi dodatni problem u tokenizaciju tekstova napisanih kineskim pismom (preuzeto iz [38]).

莎拉波娃现在居住在美国东南部的佛罗里达。今年4月
9日，莎拉波娃在美国第一大城市纽约度过了18岁生
日。生日派对上，莎拉波娃露出了甜美的微笑。

Segmentacija teksta na kineskom ne mora biti jednoznačna što još više komplikuje tokenizaciju. Naredna dva znaka mogu biti tretirani kao jedna reč "sveštenik" ili sekvenca dve reči "i" i "još" (preuzeto iz [38]).

和尚

U nekim jezima složenice mogu biti sastavljene od reči koje imaju zasebno značenje i nekad bi bilo dobro omogućiti njihovu pretragu po delu složene reči (preuzeto iz [38]):

- Nemački: Lebensversicherungsgesellschaftsangestellter → leben + versicherung + gesellschaft + angestellter (Životno osiguranje zaposlenih),
- Inuitski: tusaatsiarunnannngittualuujunga (Ne čujem dobro),
- Švedski, finski, grčki i mnogi drugi jezici imaju složenice.

Još jedan primer mogućih komplikacija prilikom tokenizacije teksta je japanski jezik koji koristi 4 različita alfabeta: kineski znaci, hiragana za glavne gramatičke varijante, katakana za transkripciju stranih reči i latinica. Nema belih znakova kao i u kineskom. Upit se može izraziti kompletno pomoću hiragane.

Postoje i pisma koja se pišu sa desna u levo kao što je arapsko pismo, a u kojima se brojevi piše sa leva u desno (preuzeto iz [38]).

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.
 ← → ← → ← START
 'Algeria achieved its independence in 1962 after 132 years of French occupation.'

Pored toga što se piše sa desna u levo arapsko pismo ima još jednu komplikaciju, ima dosta ligatura (eng. *ligature*) - spajanje susedna dva karaktera u jedan glif (eng. *glyph*).

كِتَابٌ ← * كِتابُ
 un b ā t i k
 /kitābun/ ‘a book’

Dvosmernost i upotreba ligatura nije problem ako imamo tekstualni digitalni dokument u kome je tekst zapisan u Unicode rasporedu, ali može biti problem za tehnike prepoznavanja teksta (eng. *optical character recognition* - OCR), ako želimo da od papirnog dokumenta napravimo tekstualni digitalni dokument.

Nakon tokenizacije dokumenta u listu tokena, a takođe i upita, najjednostavniji je slučaj ako se tokeni upita potpuno poklapaju sa tokenima dokumenta. Ovo je najjednostavniji slučaj, ali nije najčešći. Vrlo često iako se sekvence znakova u upitu i dokumentu ne poklapaju u potpunosti smatramo da je dokument relevantan za korisnikov upit. Potrebno je izvršiti **normalizaciju**, odnosno svesti termove u indeksiranom tekstu i u upitima u isti oblik. Na primer, želimo da izjednačimo *U.S.A.* i *USA*. Najčešće se definišu *klase ekvivalencije* termova, pri čemu reči pripadaju istoj klasi ako se nakon primene određenog skupa pravila svode na isti niz znakova. Alternativa ovom pristupu je *asimetrično proširivanje* koje svaki token iz indeksiranog teksta proširuje određenim skupom tokena. Primer asimetričnog proširenja je dat u nastavku (preuzeto iz [38]).

- *window* → *window*, *windows*
- *windows* → *Windows*, *windows*, *window*
- *Windows* → *Windows*

Glavna prednost ovog pristupa je asimetričnost i to što se ovo proširenje može primeniti samo nakon tokenizacije dokumenata, odnosno ne mora se primeniti i na upit, pa je samim tim brži odgovor sistema na postavljeni upit. Za navedeno asimetrično proširenje upiti koji sadrže reč *windows* će pronaći i dokumente koji spominju *window* kao prozor i *Windows* kao operativni sistem, ali upiti koji sadrže reč *window* neće pronaći dokumente koji govore o operativnom sistemu. Glavni nedostatak ovog pristupa je zauzeće memorije. Potrebno je više memorije za skladištenje indeksa koji se kasnije pretražuje. Dominantniji je prvi pristup - svođenje reči primenom određenog skupa pravila (transformacija) na jednog predstavnika klase ekvivalencije. U narednom tekstu diskutovane su neke moguće transformacije.

Dijakritik je glif koji je dodat na neko osnovno slovo. U nekim situacijama ovi dijakritici predstavljaju akcente, odnosno način izgovora glasa koji je označen slovom ispod dijakritika, a u nekim slučajevima oni predstavljaju različite glasove. Dijakritici se često uklanjanju nakon tokenizacije. U nekim jezicima dve reči

mogu razlikovati samo po dijakriticima, pa njihovo uklanjanje može dovesti da se dve reči različitog značenja svedu na isti niz znakova, na primer "kuća" i "kuca". I pored ove činjenice, u sistemima za pretragu se najčešće uklanjuju dijakritici, zato što ih korisnici sistema često izostavljaju prilikom pisanja upita zbog: bržeg unosa upita, lenjosti, limitiranog softvera koji ne dozvoljava unos karaktera sa dijakriticima, itd. Čak i u jezicima koji redovno koriste dijakritike, korisnici ih retko pišu. Određeni broj korisnika u Srbiji u email-ovima i nekim neformalnim tekstualnim dokumentima ne koristi slova koja imaju dijakritike. Zbog ovoga će većina sistema za pretraživanje reč "kuća" prebaciti u reč "kuca", odnosno ukloniti dijakritike. Postoje i jezici u kojima se po nepisanom pravilu neki karakteri sa dijakritikom zamenjuju sa dva karaktera: Universität se menja u Universitaet (umlauti - zamena nizom slova "ae"). Najvažniji kriterijum za definisanje svih ovih zamena je kako će korisnici najverovatnije pisati upite za ovakve reči. U Srbiji se poslednjih godina pojavljuje trend da se u tekstu srpska slova sa dijakriticima zamenjuju sa dva slova. Na primer, š se često piše kao sh. Ovaj trend bi mogao da uvede određene komplikacije u implementaciji sistema za pretragu tekstualnih digitalnih dokumenta napisanih na srpskom jeziku.

Velika slova se najčešće svode na mala slova. U nekim situacijama mogu postojati izuzeci - reči u sredini rečenice koje počinju velikim slovom: MIT vs. mit, Fed vs. fed, itd. Često je najbolje sve prebaciti u mala slova jer će i korisnici tako pisati upite, bez obzira na ispravno pisanje.

Stop reči su česte reči koje nisu korisne prilikom pretraživanja, odnosno reči koje imaju malu informativnost. Ovakve reči postoje gotovo u svakom dokumentu. Svaki jezik ima listu svojih stop reči. Na primer, stop reči za engleski su *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*. Na primer, stop reči za srpski su *i, ili, kod, ka, sa, za...*. Eliminacija stop reči je ranije bila uobičajena u klasičnim sistemima za pretraživanje, ali kako su se povećali memorijski kapaciteti i procesorske moći računara danas imamo sisteme za pretraživanje koje ne izbacuju stop reči. Stop reči su

nam potrebne za fraze, na primer "prijava za upis", "*To be or not to be*". Većina veb pretraživača ne izbacuje stop reči. U sistemima koji izbacuju stop reči potrebno je definisati listu stop reči za jezik na kojem su napisani dokumenti koji su u kolekciji. Ako su u kolekciji dokumenti na različitim jezicima, potrebno je definisati listu stop reči za svaki od korišćenih jezika, prepoznati jezik na kome je pisan dokument i primeniti eliminaciju stop reči. Normalizacija teksta i detekcija jezika su međuzavisni i za neke druge transformacije tokena, na primer za prebacivanje velikih slova u mala:

- *PETER WILL NICHT MIT.* → MIT = mit
- *He got his PhD from MIT.* → MIT ≠ mit

U nekim sistemima se u pretpresiranju koriste i tezaurusi koji sadrže semantičku ekvivalenciju reči, odnosno koriste se **reč-nici sinonima**: kola = automobil.

Soundex algoritam se često koriste u jezicima kod kojih ne važi pravilo čitaj kao što je napisano. Ovi algoritmi pokušavaju da utvrde fonetsku ekvivalenciju, odnosno da niz napisanih karaktera prebaci u niz karaktera koji predstavljaju izgovor te reči. Sledeće dve reči su fonetski ekvivalentne: *Tchebyshev* = *Chebys-heff*. Na ovaj način korisnik može da dobije od sistema rezultate pretrage iako je pogrešno napisao određenu reč u upitu, ali bi izgovor tako pogrešno napisane reči bio sličan, odnosno fonetski ekvivalentan izgovoru korektno napisane te reči.

Reči mogu biti u različitim gramatičkim oblicima: promena po licima, rodovima, padežima, vremenima... Redukovanje raznih gramatičkih oblika na baznu formu se zove **lematizacija**. Primeri za lematizaciju su:

- bih, bi, bismo, biste, biše → biti
- automobil, automobili, automobila, automobilu, itd. → automobil
- voleo bih da kupim automobil → voleo biti da kupiti automobil

Lematizacija podrazumeva "pravilnu" redukciju na osnovni rečnički oblik (*lemu*), zbog čega je gotovo nemoguće implementirati algoritam za lematizaciju, odnosno lematizacija je najčešće zasnovana na rečnicima lema i gramatičkih oblika ovih lema. Izgradnja rečnika je vremenski zahtevna, kad je konačno izgrađen već je star nekoliko godina (jezici se menjaju), a treba ga i u budućnosti ažurirati što je konstantan posao. Lematizacija uključuje *inflepcionu morfologiju* (knjige → knjiga) - ista vrsta reči samo u drugom obliku (mnogočina, lice, glagolski oblik, itd.). Takođe, lematizacija uključuje i *derivacionu morfologiju* (brzina → brzo) - nova vrsta reči ali sa istom osnovom.

Zbog problema oko implementacije lematizacije često se u sistemima za pretraživanje koristi **steming** (eng. *stemming*). Steming je grubi heuristički proces koji odseca krajeve reči sa ciljem da postigne rezultat što sličniji onome koji postiže pravilna lematizacija bazirana na lingvističkom znanju. Steming je deo procesa normalizacije teksta, često se očekuje da su pre steminga prebačena velika u mala slova, izbačene stop reči... Takođe, apostrofi i znaci interpunkcije obično ne stižu do stemera. Pre steminga se može vršiti pretprocesiranje teksta upotrebom rečnika sinonima ili nekih drugih rečnika. Dobijeni niz znakova se zove *stem* koji ne mora biti reč koja postoji u jeziku, ne mora biti lema, odnosno koren (eng. *root*) početne reči. Dakle, stem može biti nešto što ne postoji u jeziku, bitno je samo da steming povećava performanse sistema za pretraživanje. O performansama sistema za pretraživanje biće reči u kasnjem poglavlju ove knjige (poglavlje 8). Nekad je steming bio baziran samo na književnom jeziku, poslednjih godina postoji potreba da se steming bavi i slengom, jer razvojem IKT i veba publikuje se dosta digitalnog sadržaja koji sadrži sleng.

Steming može biti zasnovan na algoritmu ili na rečniku. **Steming zasnovan na algoritmu** je definisan skupom pravila, konvencija i definisanjem redosleda njihove primene. Vrlo su značajni za sisteme za pretraživanje, često se koriste jer ih je lakše kreirati nego stemere zasnovane na rečniku. Obično ne skida prefikse,

samo sufikse. Sufiksi su nastavci koji se dodaju baznim rečima. Vrste sufiksa su:

- a-sufiksi (eng. *attached suffixes*) - jedna reč nakačena na kraj druge, ima ih u Italijanskom, Španskom, Portugalskom (**mandargli** - to send + **to him**). Obično se skidaju ovi sufiksi.
- i-sufiksi (eng. *inflectional suffixes*) - infleksiona morfologija, ista vrsta reči u različitom obliku: množina, padeži, glagolska vremena, itd. Uvek se skidaju ovi sufiksi.
- d-sufiksi (eng. *derivational suffixes*) - derivaciona morfologija, različite vrste reči sa istim korenom: radnik, raditi, radan (vredan), itd. U retkim situacijama se skidaju ovi sufiksi.

Za dve reči koje se svode na isti stem kažemo da su spojene (eng. *conflated*). Iako je cilj stemminga da poveća performanse sistema, usled stemminga mogu nastati sledeće greške:

- ***Under-stemming*** - Skidanje previše malog sufiksa. Problem je što u ovom slučaju stemming ne spaja reči koje imaju isto značenje. Dakle, može ostati isti efekat kao da nema stemminga. Ne mora da izazove probleme, ako nema spajanja sa stemovima reči drugačijeg značenja. U najboljem slučaju ne poboljšava povrat, a nadamo se da ne pogoršava preciznost, odnosno ne unapređuje performanse sistema za pretraživanje, ali se nadamo da ga ne unazadjuje (u poglavljiju 8 su definisani pojmovi preciznost i povrat).
- ***Over-stemming*** - Skidanje previše velikog sufiksa tako da se reč spaja sa stemom reči drugačijeg značenja. Ove greške smanjuju preciznost. Uvodi nove homonime u jezik - reči (odnosno stemove) koji se isto pišu, a ne znače isto. Ovo je najopasnija vrsta greške koja se često pokušava popraviti ili izbeći uvodenjem rečnika.
- ***Mis-stemming*** - Skidanje nečega što je deo stema, što se čini kao sufiks. Ne mora da izazove probleme, ako nema

poklapanja sa drugim stemovima. Može da poboljšava povrat, a nadamo se da ne pogoršava preciznost. Nije bitno što je dobijeno nešto što ljudima ne izgleda kao stem te reči.

Nepravilni gramatički oblici su najveći problem stemminga zasnovanog na algoritmu i rešavaju se na dva načina - formiranjem rečnika izuzetaka ili dodavanjem posebnih pravila u algoritmima za ove nepravilne oblike. Ako se ovi nepravilni gramatički oblici javljaju kod retkih reči ili kod reči koji nisu od velikog značaja za sistem za pretraživanje, ne moraju se rešavati ni na jedan od ova dva načina. To ostaje kao greška stemera. Nijedan stemer nije savršen, ali steminzi zasnovani na algoritmu rade brzo i začuđujuće dobro - postoji čuveno pitanje Krovetz-a na 89 strani njegove disertacije [42]: *Why does it do so well?*. Kao što je već rečeno stemming je često samo deo normalizacije teksta, zbog čega su često svesno izostavljena pravila za stemovanje stop reči. Dakle, koji je odgovor na pitanje: *Da li stemming unapređuje performanse sistema za pretraživanje?*. U opštem slučaju stemming popravlja performanse za neke upite, ali za neke druge pogoršava. Pogoršava u slučajevima kada se desi neka od prethodno nabrojanih grešaka.

Steming nije konceptualno primenljiv na sve jezike. Nije primenljiv na Kineski jezik. Primenljiv je na jezike koji imaju uobičajene paterne kreiranja varijanti jedne reči kao što su Indo-Evropski jezici.

Snowball je pogodan za opis stemminga zasnovanog na algoritmu, lako ga je razumeti, odnosno pravilno tumačiti. Stemer opisan *Snowball* jezikom se lako može prevesti u programske jezike Java i ANSI C, a postoje načini i da se koriste u Python-u, Objektnom PASCAL-u i drugim jezicima.

Postoje i **stemerii zasnovani na rečniku**, ne koristi se *Snowball*, teži su za kreiranje, ali mogu imati bolje performanse. Kao i kod lematizacije zasnovane na rečniku i ovde važi da je izgradnja rečnika vremenski zahtevna, kad je konačno izgrađen rečnik već je star nekoliko godina (jezici se menjaju), a treba ga i u budućnosti ažurirati što je konstantan posao. Ove dve vrste stemera nisu striktno razdvojene. Stemerii zasnovani na algoritmu mogu

imati duge liste izuzetaka (praktično rečnika) koji se koriste da bi se smanjile greške. Stemeri zasnovani na rečniku obično skidaju određene nastavke pre nego što izvrše *look-up* u rečniku (uklanjanje množine, . . .) - ovo ukljanjanje je zasnovano na algoritmu.

Porterov algoritam je najpoznatiji stemming algoritam za engleski jezik kreiran od strane Martina Portera [43]. Prvi put je objavljen 1980. godine, a svoju veliku popularnost je stekao krajem 20. veka eksponencijalnim razvojem oblasti pronalaženja informacija. Algoritam se sastoji iz 5 faza redukcije i skupa konvencija koja se primenjuju. Svaka faza predstavlja skup komandi. Faze se primenjuju sekvensijalno. Primer jedne komande: obriši zadnji *ement* ako je ono što ostaje duže od jednog sloga: *replacement* → *replac*, *cement* → *cement*. Još nekoliko komandi koje je definisao Porter (preuzeto iz [43]):

Pravilo	Primer
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

Primer jedne konvencije Porterovog stemming algoritma: od skupa pravila primeni onu koja se odnosi na najduži sufiks.

Iako star preko 30 godina, po rezultatima Porterov stemming algoritam je dobar barem koliko i druge alternative. U narednim paragrafima je upoređen ovaj algoritam sa još dva poznata stemera za engleski jezik: *Lovins stemmer* i *Paice stemmer* (preuzeto iz [38]).

Uzorak originalnog teksta:

Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer:

such an analys can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Lovins stemmer:

such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Paice stemmer:

such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

5.2 Bulov model pretraživanja

Bulov model pretraživanja je klasični model pretraživanja koji je prvi bio široko prihvaćen za upotrebu u sistemima za pretraživanja. Ovaj model je zasnovan na *teoriji skupova* i *Bullovoj algebri*. Tražene informacije se izražavaju upitom. Upiti su logički izrazi, npr. **Caesar AND Brutus**. Pretraživač vraća sve dokumente koji zadovoljavaju logički izraz. U 20. veku ovaj model je bio dominantan u sistemima za pretraživanje zbog svoje jednostavnosti. Bio je osnovni mehanizam za pretraživanje preko 30 godina. Neki korisnici (npr. advokati) i dalje vole da ga koriste zato što se tačno zna šta je rezultat pretrage. Koristi se u mnogo različitim sistema (npr. email klijent). Model može odgovoriti na svaki upit koji je Bulov izraz. Svaki dokument se posmatra kao skup termova. Model je nedvosmislen, koristi osnovni princip traženi pojam se ili nalazi ili ne nalazi u dokumentu. Kako je količina sadržaja u sistemima rasla tako je ovaj osnovni princip postajao sve veća mana ovog modela. Stepen slaganja upita i dokumenta

je binarna vrednost. Dakle, nema parcijalnog poklapanja upita i dokumenta, dokument je ili relevantan (ocena relevantnosti je 1) ili nije relevantan (ocena relevantnosti je 0). Na primer, ako imamo konjukciju tri terma (logičko I) jednako se posmatra i dokument koji nema ni jedan term i dokument koji ima dva terma - oba dokumenta nisu relevantna (ocena relevantnosti je 0). Zbog ovakvog pristupa Bulov model nema mogućnost rangiranja, a sa tim ni mogućnost sortiranja rezultata. Zbog ove činjenice ovakav model nije prihvatljiv za pretraživače velikih kolekcija ili za veb pretraživače kao što je Google.

Primer pretraživača zasnovanog na Bulovom modelu je *Westlaw* (www.westlaw.com). Ovo je komercijalni pretraživač pravnih materijala koji ima najviše pretplatnika od svih ovakvih pretraživača. Preko pola miliona pretplatnika postavlja milione upita dnevno nad desetinama terabajta teksta. Servis je pokrenut 1975. U 2005. Bulovi upiti ("Terms and Connectors") su i dalje podrazumevani tip upita, i koristi ga veliki procenat korisnika, iako rangirana pretraga postoji od 1992. godine.

Westlaw definiše određenu sintaksu za kreiranje upita. Mogu se koristiti fraze upotreboom navodnika, kao i blizinski operatori: */s* - u istoj rečenici, */p* - u istom paragrafu, */4* - u okviru četiri reči... Uzvičnik označava da reč u upitu nije kompletna reč, nego je samo početak reči. Razmak je disjunkcija, a ne konjunkcija, što je praktično bila konvencija u eri pre *Google* pretraživača.

Sledi nekoliko primera zahteva i odgovarajućih upita u *Westlaw* sintaksi.

Zahtev:

Informacije o pravnoj teoriji o sprečavanju odavanja poslovnih tajni od strane zaposlenih koji su prethodno bili zaposleni u konkurentskoj firmi.

Upit:

"poslovna tajna" /s oda! /s spreč! /s zaposle!

Zahetv:

Uslovi za osobe sa posebnim potrebama kod pristupa radnom mestu.

Upit:

posebn! /s potreb! /p pristup /s radnom mestu (zaposlen /4 mesto)

Zahetv:

Slučajevi o odgovornosti domaćina prema pijanom gostu.

Upit:

domaćin! /p odgovor! /p pijan! /p gost!

Profesionalni korisnici vole Bulov model zbog preciznosti, potpune kontrole i transparentnosti. Da li je Bulov model najbolji za pretragu zavisi od potreba korisnika, kolekcije dokumenata, ali i od veštine korisnika.

5.2.1 Invertovani indeks

Razmotrimo sada kako implementirati pretragu zasnovanu na Bulovom modelu. Na primer, imamo kolekciju dokumenata koje predstavljaju doktorske disertacije odbranjene na Fakultetu tehničkih nauka u Novom Sadu i treba da odgovorimo na pitanje: Koje disertacije sadrže reči **Lucene** i (operator **AND**) **pretrage**, ali ne (operator **NE**) **multimedijalnih**? Dakle, potrebne su nam disertacije koje se bave temom pretrage upotrebom Lucene biblioteke, ali se ne bave pretragom multimedijalnih sadržaja. Prvo rešenje je da se uradi **grep** nad svim disertacijama tražeći **Lucene** i **pretrage**, i onda odbace svi dokumenti koje sadrže **multimedijalnih**. Ovo nije dobro rešenje iz više razloga: sporo za velike kolekcije,

“NOT multimedijalnih” nije trivijalno za implementaciju, druge operacije (npr. nađi reč Lucene blizu reči pretrage, složeno pretprocesiranje upita i teksta nije izvodljivo, rangiranje pogodaka ne postoji).

Bolje rešenje je kreiranje **invertovanog indeksa** koji predstavlja strukturu podataka koja mapira reči i brojeve iz sadržaja dokumenata na dokumente u kojima se javljaju, a u boljoj varijanti i na lokaciju na kojoj se te reči i brojevi javljaju u dokumentima.

Pre nego što dođemo do pravog invertovanog indeksa pogledajmo prvo narednu **matricu incidencije term/dokument**.

	Ivanović D. [19]	Milosavljević B. [44]	Gostojić S. [45]	Zarić M. [34]	...
digitalan	1	1	1	1	
lucene	1	1	1	1	
dokument	1	1	1	1	
obrazovanje	0	0	1	0	
pretraga	1	1	1	1	
multimedijalan	0	1	1	1	
evaluacija	0	0	0	0	
...					

U zaglavljima kolona su prezimena i prvo slovo imena autora doktorskih disertacija koje se nalaze u kolekciji, a u zaglavljima vrsta su termovi koji su nakon pretprocesiranja tekstova ovih dokumenata izdvojeni. Vrednost u matrici incidencije je 1 ako se term pojavljuje u dokumentu. Na primer: **obrazovanje** se javlja u disertaciji čiji je autor *Gostojić S.* Vrednost u matrici incidencije je 0 ako se term ne pojavljuje u dokumentu. Na primer: **obrazovanje** se ne javlja u disertaciji čiji je autor *Milosavljević B.*

Vektor incidencije je zapravo jedna vrsta u ovoj matrici, odnosno vektor sa elementima 0/1 za svaki term. Algoritam za odgovor na upit **Lucene AND pretraga AND NOT multimedijalnih** upotreboom matrice incidencije je sledeći:

1. pretprocesiranje upita da bi se tokeni iz upita prebacili u termove: **lucene AND pretraga AND NOT multimedijalan**,

2. uzimanje vektora incidencije za **lucene** (*1111*), **pretraga** (*1111*) i **multimedijalan** (*0111*),
3. izračunavanje komplementa za vektor incidencije za **multimedijalan** (*1000*),
4. izračunavanje logičkog I (AND) po bitovima (eng. *bitwise and*) za tri vektora *1111 AND 1111 AND 1000 = 1000*.

Odgovor na upit je:

Ivanović D., Informacioni sistem naučno-istraživačke delatnosti

...IndexSearcher klasa pripada biblioteci Apache **Lucene**. Ova klasa je namenjena **pretrazi** dokumenata koji su prethodno indeksirani upotreboom biblioteke Apache **Lucene**...

Prepostavimo da imamo mnogo veću kolekciju koja ima $N = 10^6$ dokumenata, svaki sa 1.000 tokena, i da nam je u proseku potrebno 6 bajtova po tokenu, uključujući razmake i interpunkciju \Rightarrow veličina kolekcije je oko 6 GB. Naravno, nisu svi tokeni različiti, prepostavimo da ima $M = 500.000$ različitih termova u kolekciji. U ovom slučaju imamo ogromnu matricu incidencije: $M = 500.000 \times 10^6 =$ pola biliona nula i jedinica. Matrica je vrlo retka i sadrži manje od milion jedinica. *Ima li bolje reprezentacije?* Ima, invertovani indeks - za svaki term t , čuvamo listu dokumenata koji sadrže term t .

lucene	→	2	31	54	101
--------	---	---	----	----	-----

pretraga	→	1	2	4	5	6	16	57	132	...
----------	---	---	---	---	---	---	----	----	-----	-----

multimedijalan	→	1	2	4	11	31	45	173	174
----------------	---	---	---	---	----	----	----	-----	-----

⋮

 rečnik

 pojave

Algoritam za konstrukciju invertovanog indeksa je sledeći:

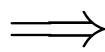
1. prikupljanje dokumenata koje treba indeksirati:
Lucene biblioteka je otvorenog koda.
Koristi se za pretragu tekstualnih sadržaja. . . ,
2. tokenizacija teksta - pretvaranje svakog dokumenta u listu tokena:
Lucene biblioteka je otvorenog . . . ,
3. preprocesiranje teksta - formiranje liste normalizovanih tokena, tj. termova koji će biti u rečniku:
lucene biblioteka biti otvoren . . . ,
4. indeksiranje dokumenata - formiranje invertovanog indeksa koji ima rečnik i pojave.

Prvi korak se može obaviti na razne načine, neki od tih načina su potpuno automatizovani, a neki zahtevaju rad ljudi.

Drugi i treći korak predstavljaju tokenizaciju i preprocesiranje. Na sledećim primerima možemo videti koji rezultat ovi koraci mogu dati:

Dokument 1. Apache
Lucene je javno dostupna
biblioteka pisana u Javi
namenjena pretraživanju
teksta

Dokument 2. Program-
ska biblioteka Apache Lu-
cene omogućava full-text
pretraživanje sa rangira-
njem rezultata



Dokument 1. apache lu-
cene biti javno dostu-
pan biblioteka pisan u
java namenjen pretrazi-
vanje tekst

Dokument 2. program-
ski biblioteka apache lu-
cene omoguciti full text
pretrazivanje sa rangira-
njem rezultat

Implementacija četvrtog koraka zahteva da se za svaki term dođen na kraju trećeg koraka kreira lista pojava ovog terma u dokumentima koji su u kolekciji. Kreiranje pojave termova u dokumentima možemo videti na sledećem primeru:

term	docID
apache	1
lucene	1
biti	1
javan	1
dostupan	1
biblioteka	1
pisan	1
u	1
java	1
namenjen	1
pretrazivanje	1
tekst	1
programske	2
biblioteka	2
apache	2
lucene	2
omoguciti	2
full	2
text	2
pretrazivanje	2
sa	2
rangiranje	2
rezultat	2

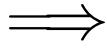
Dokument 1. apache lucene
biti javan dostupan biblioteka pi-
san u java namenjen pretraziva-
nje tekst

Dokument 2. programske bibli-
oteka apache lucene omoguciti
full text pretrazivanje sa rangi-
ranje rezultat

====>

Da bismo formirali listu pojava za svaki term u rečniku, potrebno je sortirati prethodno prikazane pojave u dokumentima:

term	docID	term	docID
apache	1	apache	1
lucene	1	apache	2
biti	1	biblioteka	1
javan	1	biblioteka	2
dostupan	1	biti	1
biblioteka	1	dostupan	1
pisan	1	full	2
u	1	java	1
java	1	javan	1
namenjen	1	lucene	1
pretrazivanje	1	lucene	2
tekst	1	namenjen	1
programske	2	omoguciti	2
biblioteka	2	pisan	1
apache	2	pretrazivanje	1
lucene	2	pretrazivanje	2
omoguciti	2	programske	2
full	2	rangiranje	2
text	2	rezultat	2
pretrazivanje	2	sa	2
sa	2	tekst	1
rangiranje	2	text	2
rezultat	2	u	1



Nakon sortiranja moguće je kreirati liste pojava za svaki term i izračunati frekvencije pojavljivanja:

term	docID	term	frekv.	→	liste pojava
apache	1	apache	2	→	1 → 2
apache	2	biblioteka	2	→	1 → 2
biblioteka	1	biti	1	→	1
biblioteka	2	dostupan	1	→	1
biti	1	full	1	→	2
dostupan	1	java	1	→	1
full	2	javan	1	→	1
java	1	lucene	2	→	1 → 2
javan	1	namenjen	1	→	1
lucene	1	omoguciti	1	→	2
lucene	2	pisan	1	→	1
namenjen	1	pretrazivanje	2	→	1 → 2
omoguciti	2	programske	1	→	2
pisan	1	rangiranje	1	→	2
pretrazivanje	1	rezultat	1	→	2
pretrazivanje	2	sa	1	→	2
programske	2	tekst	1	→	1
rangiranje	2	text	1	→	2
rezultat	2	u	1	→	1
sa	2				
tekst	1				
text	2				
u	1				

rečnik

pojave

Na kraju je potrebno podeliti rezultat u fajl za rečnik i fajl za pojave i na taj način je kreiran invertovani indeks.

Za kreiranje prethodno opisanog invertovanog indeksa postoji nekoliko bitnih pitanja koji se moraju rešavati prilikom implementacije sistema za pretraživanje. Kako kreirati indeks za ogromne kolekcije? Kako proceniti koliko memorije je potrebno za rečnik i indeks? Da li postoje neki mehanizmi da se kompresuju indeksi, odnosno kako efikasno skladištiti i obrađivati indeks za velike kolekcije? Kako kreirati invertovani indeks kada je potreban “najbolji” odgovor?

5.2.2 Procesiranje upita

Do sada smo razmatrali na koji način se kreira invertovani indeks, a sada se postavlja pitanje koji je algoritam za računanje odgovora na upit upotrebom invertovanog indeksa. Razmotrimo jednostavan konjunktivni upit dva tokena **Lucene AND multimedijalnih**. Algoritam za pronalaženje svih relevantnih dokumenata za ovaj upit pomoću invertovanog indeksa kreiranog na prethodno opisani način je sledeći:

1. preprocesiranje upita nakon čega se dobija konjuktivni upit dva terma (ne dva tokena, nego dva terma): **lucene AND multimedijalan**,
2. pronalaženje terma **lucene** u rečniku termova,
3. učitavanje liste pojava ovog terma iz fajla sa pojavama,
4. pronalaženje terma **multimedijalan** u rečniku termova,
5. učitavanje liste pojava ovog terma iz fajla sa pojavama,
6. izračunavanje **preseka** ove dve liste pojava,
7. vraćanje rezultata korisniku - vraćanje dokumenata koji se nalazu u prethodno izračunatom preseku.

Prvi korak ovog algoritma može biti prilično komplikovan i kao što je ranije rečeno od kvaliteta implementacije preprocesi-

ranja teksta i upita umnogome zavisi kvalitet dobijene liste rezultata pretrage. Naredna četiri koraka ovog algoritma su trivialna za implementaciju, a izračunavanje preseka dve liste pojava umnogome određuje efikasnost odnosno brzinu procesiranja upita. Jedan jednostavan (ali ne mnogo efikasan) algoritam za računanje ovog preseka je dat u nastavku (preuzeto iz [38]). Iako algoritamska složenost nije predmet ovog udžbenika, u nastavku će biti razmatrani algoritmi koji unapređuju performanse pretrage.

```

INTERSECT( $p_1, p_2$ )
1   answer  $\leftarrow \langle \rangle$ 
2   while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3     do if  $docID(p_1) = docID(p_2)$ 
4       then ADD(answer,  $docID(p_1)$ )
5          $p_1 \leftarrow next(p_1)$ 
6          $p_2 \leftarrow next(p_2)$ 
7       else if  $docID(p_1) < docID(p_2)$ 
8         then  $p_1 \leftarrow next(p_1)$ 
9       else  $p_2 \leftarrow next(p_2)$ 
10  return answer
```

Na primer, prethodni algoritam za sledeće liste pojava termova **lucene** i **multimedijalan** daje prikazani rezultat (presek):

lucene	\longrightarrow	$[2] \rightarrow [31] \rightarrow [54] \rightarrow [101]$
multimedijalan	\longrightarrow	$[1] \rightarrow [2] \rightarrow [4] \rightarrow [11] \rightarrow [31] \rightarrow [45] \rightarrow [173] \rightarrow [174]$
presek	\Longrightarrow	$[2] \rightarrow [31]$

Ovaj algoritam je linearno zavisao od dužine liste i radi samo ako su liste sortirane. Koja je minimalna složenost za procesiranje velikih Bulovih upita i kako se to može optimizovati? Posmatramo upit koji je konjunkcija (**AND**) od n termova, $n > 2$. Dakle, potrebno je za svaki term pribaviti njegovu listu pojava, onda izračunati **AND** za sve njih. Operator **AND** je asocijativan. Kojim redom ga računati da bi se optimizovalo procesiranje upita? Jed-

nostavna i efikasna optimizacija ovog procesiranja je da se upit obradi u rastućem redosledu frekvencije termova, odnosno da se liste pojava sortiraju od najmanje do najveće i da se počne obrada upita od najkraće liste pojava. Optimizovani algoritam za izračunavanje preseka lista pojava je dat u nastavku (preuzeto iz [38]).

```

INTERSECT( $\langle t_1, \dots, t_n \rangle$ )
1   $terms \leftarrow \text{SORTBYINCREASINGFREQUENCY}(\langle t_1, \dots, t_n \rangle)$ 
2   $result \leftarrow \text{postings}(first(terms))$ 
3   $terms \leftarrow rest(terms)$ 
4  while  $terms \neq \text{NIL}$  and  $result \neq \text{NIL}$ 
5    do  $result \leftarrow \text{INTERSECT}(result, \text{postings}(first(terms)))$ 
6     $terms \leftarrow rest(terms)$ 
7  return  $result$ 

```

Na primer, imamo upit:

lucene AND multimedijalan AND tekstura

Liste pojava odgovarajućih termova su:

lucene	\longrightarrow	$[2] \rightarrow [31] \rightarrow [54] \rightarrow [101]$
multimedijalan	\longrightarrow	$[1] \rightarrow [2] \rightarrow [4] \rightarrow [11] \rightarrow [31] \rightarrow [45] \rightarrow [173] \rightarrow [174]$
tekstura	\longrightarrow	$[5] \rightarrow [31]$

Primenom prethodnog optimizovanog algoritma prvo će se sortirati liste pojava i u tom redosledu uraditi Bulova operacija **AND** između njih. U ovom primeru: prvo **tekstura AND lucene**, potom dobijeni rezultat **AND multimedijalan**. Kako izvršiti optimizaciju za nešto složeniji upit koji kombinuje operatore **AND** i **OR**? Primer takvog upita je:

(lucene OR sphinx) AND (multimedijalan OR slika) AND (tekstura OR šablon)

U ovom slučaju možemo primeniti sledeći mehanizam:

- proceniti veličinu rezultata svakog OR-a kao zbir frekvencija operanada (konzervativni pristup) i
- obraditi upit u rastućem redosledu veličine rezultata OR-ova.

5.2.3 Pointeri za preskakanje

Izvršili smo neka unapređenja algoritma za računanje preseka lista pojava, ali je i dalje taj algoritam linearno zavisao od dužine lista. Neke liste pojava sadrže više miliona elemenata - brzina može biti problem ako je algoritam linearan. Može li se računanje preseka još ubrzati? Mogu se koristiti **pointeri za preskakanje** (eng. *skip pointers*). Pointeri za preskakanje omogućavaju *preskakanje* pojava koje svakako neće biti u rezultatu. Algoritam za računanje preseka pojava upotrebom pointera za preskakanje je dat u nastavku (preuzeto iz [38]).

```

INTERSECTWITHSKIPS( $p_1, p_2$ )
1    $answer \leftarrow \langle \rangle$ 
2   while  $p_1 \neq NIL$  and  $p_2 \neq NIL$ 
3     do if  $docID(p_1) = docID(p_2)$ 
4       then ADD( $answer, docID(p_1)$ )
5          $p_1 \leftarrow next(p_1)$ 
6          $p_2 \leftarrow next(p_2)$ 
7     else if  $docID(p_1) < docID(p_2)$ 
8       then if  $hasSkip(p_1)$ 
9         and( $docID(skip(p_1)) \leq docID(p_2)$ )
10        then while  $hasSkip(p_1)$ 
11          and( $docID(skip(p_1)) \leq docID(p_2)$ )
12          do  $p_1 \leftarrow skip(p_1)$ 
13        else  $p_1 \leftarrow next(p_1)$ 
14      else if  $hasSkip(p_2)$ 
15        and( $docID(skip(p_2)) \leq docID(p_1)$ )
16        then while  $hasSkip(p_2)$ 
17          and( $docID(skip(p_2)) \leq docID(p_1)$ )
18

```

```

19           do  $p_2 \leftarrow \text{skip}(p_2)$ 
20           else  $p_2 \leftarrow \text{next}(p_2)$ 
21   return answer

```

Gde da stavimo pointere za preskakanje tako da računanje bude što je moguće efikasnije (naravno mora biti i tačno u prvom redu)? Treba napraviti kompromis između broja preskočenih elemenata i učestalosti skakanja upotrebom pointera za preskakanje:

- više skokova: svaki pointer preskače malo elemenata, ali ga možemo češće koristiti,
- manje skokova: svaki pointer preskače puno elemenata, ali ga možemo retko koristiti.

Primer jedne jednostavne heuristike za postavljanje pointera za preskakanje je: za liste pojava dužine P , napraviti \sqrt{P} skip pointer-a jednake dužine. Ova heuristika ignoriše distribuciju termova, odnosno podrazumeva uniformnu raspodelu. Primena ove heuristike je jednostavna ako je indeks sporo promenljiv, ali ako je indeks često promenljiv zahteva česte izmene pointera za preskakanje. Dok su procesorske moći računara bile slabije, pointeri za preskakanje su se intenzivno koristili u sistemima za pretraživanje i bili su jako korisni. Sa današnjim procesorima nisu više presudni za utisak i zadovoljstvo korisnika sistema za pretraživanje.

5.2.4 Upiti fraze

Korisnici sistema za pretraživanje često imaju potrebu da pronađu sadržaje koji sadrže određenu frazu. Zbog toga imaju potrebu da u upitu mogu da koriste i odgovarajuću frazu - tj. da kreiraju **upite fraze** kao što je na primer “**Fakultet tehničkih nauka**”. Kada se koncept fraze pojavio u sistemima za pretraživanje korisnici su ga jako brzo usvojili i danas oko 10% upita na vebu su upiti fraze. Ako je postavljen upit “**sive pantalone**” - kao fraza (niz reči), onda *Voli da nosi sive košulje i plave pantalone*

nije pogodak. Invertovani indeks koji smo do sada razmatrali ne može da odgovori na ovakve upite. Dakle, nije dovoljno u listi pojava imati samo termove za koje su vezane liste identifikatora dokumenata u kojima se ovi termovi pojavljuju.

Prvi pristup koji može da obezbedi odgovor na upite fraze su **dvorečni indeksi**. Dvorečni indeksi pored termova indeksiraju i svaki susedni par reči u tekstu kao frazu. Na primer, *sive markirane pantalone* će dati dva para reči: “*sive markirane*” i “*markirane pantalone*”. Pored standardnih termova (sive, markirane, pantalone), svaki od parova reči se tretira kao term u rečniku. Upotreboom ovako kreiranog indeksa lako je odgovoriti na dvorečne upite fraze “*sive markirane*” i “*markirane pantalone*”, ali nije lako odgovoriti na duže upite fraze: “*sive markirane pantalone*”. Ova fraza može da se prikaže kao:

“*sive markirane*” AND “*markirane pantalone*”

Da li su svi dokumenti koji su odgovor na ovaj upit ujedno i odgovori na upit “*sive markirane pantalone*”? Nisu! *Voli da nosi sive markirane pantalone* jeste odgovor na trorečnu upit fazu, ali *Voli da nosi sive markirane košulje i plave markirane pantalone* nije odgovor na ovaj upit, ali jeste odgovor na upit koji je izražen pomoću **AND** operatora. Dakle, dvorečni indeks može da odgovori na duže upite fraze, ali nakon procesiranja upita koji je izražen pomoću **AND** operatora mora se uraditi filtriranje podataka da izdvojimo samo one dokumente koji stvarno sadrže celu fazu. Ovo filtriranje može značajno uticati na brzinu procesiranja upita. Ako želimo da zbog brzine procesiranja upita preskočimo filtriranje rezultata, onda nam rezultati sistema za pretraživanje mogu biti *false positive* - pronađeni dokument koji zapravo ne sadrži celu dugačku fazu. Ovo je prvi veliki problem dvorečnog indeksa, a drugi veliki problem je što može dovesti do eksplozije indeksa zbog velikog broja termova u rečniku.

Problem sa eksplozijom indeksa je utoliko veći jer se često u rečnik indeksa stavljaju **proširene dvoreči**. Rečnik sa proširenim dvorečima se kreira na sledeći način:

- parsiranje dokumenta i označavanje vrste reči,
- vrste reči podeliti na imenice (N) i predloge/priloge (X),
- dodavanje svakog niza reči oblika NX*N kao *proširenu dvoreč* u rečnik.

Primeri proširenih dvoreči:

Univerzitet	u	Beogradu
N	X	N
protokol za udaljenu pretragu		
N	X	X N

Pozicioni indeksi su dobra alternativa za dvorečne indekse.

Omogućuju odgovore na upite fraze proizvoljne dužine. Za jedan term u *nepozicionom* indeksu je vezana lista pojava ovog terma u dokumentima, pri čemu je svaka pojava *docID* - identifikator dokumenta u kolekciji koji sadrži term. Kod *pozicionog indeksa* svaka pojava je docID i *lista pozicija*. Na primer, imamo upit *to₁ be₂ or₃ not₄ to₅ be₆* i pozicioni indeks (preuzeto iz [38]):

to, 993427:

```
⟨ 1, 6: ⟨7, 18, 33, 72, 86, 231⟩;
  2, 5: ⟨1, 17, 74, 222, 255⟩;
  4, 5: ⟨8, 16, 190, 429, 433⟩;
  5, 2: ⟨363, 367⟩;
  7, 3: ⟨13, 23, 191⟩; ... ⟩
```

be, 178239:

```
⟨ 1, 2: ⟨17, 25⟩;
  4, 5: ⟨17, 191, 291, 430, 434⟩;
  5, 3: ⟨14, 19, 101⟩; ... ⟩
```

Document sa identifikatorom (*docID*) 4 je pogodak!

Deo pozicionog indeksa u formatu term: doc1: ⟨position1, position2, ...⟩; doc2: ⟨position1, position2, ...⟩; je prikazan u nastavku (preuzeto iz [38]).

angels: 2: ⟨36,174,252,651⟩; 4: ⟨12,22,102,432⟩; 7: ⟨17⟩;
fools: 2: ⟨1,17,74,222⟩; 4: ⟨8,78,108,458⟩; 7: ⟨3,13,23,193⟩;
fear: 2: ⟨87,704,722,901⟩; 4: ⟨13,43,113,433⟩; 7: ⟨18,328,528⟩;

in: 2: ⟨3,37,76,444,851⟩; 4: ⟨10,20,110,470,500⟩; 7: ⟨5,15,25,195⟩;
rush: 2: ⟨2,66,194,321,702⟩; 4: ⟨9,69,149,429,569⟩; 7: ⟨4,14,404⟩;
to: 2: ⟨47,86,234,999⟩; 4: ⟨14,24,774,944⟩; 7: ⟨199,319,599,709⟩;
tread: 2: ⟨57,94,333⟩; 4: ⟨15,35,155⟩; 7: ⟨20,320⟩;
where: 2: ⟨67,124,393,1001⟩; 4: ⟨11,41,101,421,431⟩; 7: ⟨16,36,736⟩;

Koji dokument(i) zadovoljava(ju) sledeće upite-fraze:

“fools rush in”
“fools rush in” AND “angels fear to tread”

Pozicioni indeks se može koristiti i za **blizinsku pretragu**. Na primer, imamo upit **pretraga /3 metapodatak** koji ima sledeće značenje: pronađi sve dokumente koji sadrže **pretraga** i **metapodatak** na rastojanju od najviše tri reči. *Pretraga po metapodatacima može biti implementirana upotrebom Lucene biblioteke* je pogodak, a *Rezultati pretraga digitalne biblioteke su prikazani u MARC 21 formatu metapodataka* nije pogodak.

Najjednostavniji algoritam za implementaciju blizinske pretrage je implementacija Dekartovog proizvoda pozicija za term **employment** i term **place** (preuzeto iz [38]). Ovakav algoritam je neefikasan za česte reči koje imaju dugačke liste pojave kao što su stop reči. Algoritam za blizinsku pretragu bi trebalo da vraća i pozicije, a ne samo listu dokumenata. Te vraćene pozicije su važne za dinamičke sažetke koji se koriste u veb pretraživačima i o dinamičkim sažecima će biti reči u poglavlju 8.

```

POSITIONALINTERSECT( $p_1, p_2, k$ )
1   answer  $\leftarrow \langle \rangle$ 
2   while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3     do if  $docID(p_1) = docID(p_2)$ 
4       then l  $\leftarrow \langle \rangle$ 
5          $pp_1 \leftarrow positions(p_1)$ 
6          $pp_2 \leftarrow positions(p_2)$ 
7         while  $pp_1 \neq \text{NIL}$ 

```

```

8           do while  $pp_2 \neq \text{NIL}$ 
9             do if  $|pos(pp_1) - pos(pp_2)| \leq k$ 
10            then ADD( $l, pos(pp_2)$ )
11            else if  $pos(pp_2) > pos(pp_1)$ 
12              then break
13               $pp_2 \leftarrow next(pp_2)$ 
14            while  $l \neq \langle \rangle$  and  $|l[0] - pos(pp_1)| > k$ 
15              do DELETE( $l[0]$ )
16            for each  $ps \in l$ 
17              do ADD( $answer, \langle docID(p_1),$ 
18                   $pos(pp_1), ps \rangle$ )
19                   $pp_1 \leftarrow next(pp_1)$ 
20                   $p_1 \leftarrow next(p_1)$ 
21                   $p_2 \leftarrow next(p_2)$ 
22                else if  $docID(p_1) < docID(p_2)$ 
23                  then  $p_1 \leftarrow next(p_1)$ 
24                  else  $p_2 \leftarrow next(p_2)$ 
25 return  $answer$ 

```

Dvorečni i pozicioni indeksi se mogu uspešno kombinovati. Za pretraživače pozicioni upiti su mnogo skupljii (sporiji) od običnih Bulovih upita zato što se ne prolazi samo kroz listu dokumenata u kojima se javlja neki term, nego se u određenim slučajevima prolazi i kroz listu pojava tog terma u dokumentu. Dakle, sam algoritam je složeniji. Mnoge dvoreči su jako česte u kolekciji dokumenata i u upitima korisnika: Michael Jackson, Britney Spears... Za ovako česte dvoreči ubrzanje koje se dobija stavljanjem ovih dvoreči u rečnik u odnosu na pozicioni indeks je značajno. Način za kombinovanje je da se uključe česte dvoreči kao termini u rečnik, a ostale fraze da se računaju pomoću pozicionog indeksa. Primer jedne druge kombinovane pretragu koja je brža od pozicionog indeksa uz 26% više prostora za indeks je opisana u radu Williams-a i njegovih kolega iz 2004 godine [46]. Ova pretraga je implementirana tako što se pored identifikatora dokumenta u kojem se nalazi term i liste pozicija skladišti i naredni term u dokumentu.

5.3 Vektorski model pretraživanja

Vektorski model uvodi suštinski drugačiji pristup koji ispravlja neke nedostatke koje je imao Bulov model. Težinski faktori vezani za pojedine termove u odnosu na dokumente i upite su pozitivne vrednosti koje nisu celobrojne. I termovi iz upita imaju težinske faktore. Ovaj model omogućuje parcijalno poklapanje upita i dokumenta i samim tim omogućuje rangiranje i sortiranje rezultata. I upit i dokument se predstavljaju kao n-dimenzionalni vektor gde je n broj termova u rečniku. U ovom vektorskem prostoru ugao koji zaklapaju vektori upita i dokumenta je obrnuto srazmeran relevantnosti dokumenta za postavljeni upit.

U prethodnoj sekciji smo razmatrali Bulov model kod koga su svi upiti Bulovi i dokumenti ili odgovaraju upitu, ili ne odgovaraju - nema parcijalnog poklapanja. Ovo je dobar model za korisnike-eksperte sa preciznim razumevanjem svojih potreba i sadržaja kolekcije. Dobar je i za aplikacije jer one lako mogu da obrade hiljade rezultata. Ali ovakav model nije dobar za većinu korisnika koji nisu u stanju da pišu Bulove upite (ili jesu, ali ih mrzi). Većina korisnika ne želi da pregleda hiljade pogodaka koji mogu biti rezultat upita, posebno ako se radi o pretrazi veba gde možemo da imamo veliku količinu pogodaka, a da pri tome nemamo garanciju o kvalitetu tih pogodaka. Centralni problem Bulovih upita je što kao rezultat obično daju previše ili premalo. Bulovi upiti često rezultuju sa malo (0) ili previše (1.000 i više) pogodaka:

- upit 1: "standard user dlink 650" → 200.000 pogodaka,
- upit 2: "standard user dlink 650 *no card found*": 0 pogodaka.

Potrebna je veština da se napiše upit koji će vratiti razuman broj pogodaka. Ako imamo rangirani skup pogodaka, broj pogodaka postaje manje bitan. Da bismo imali rangiranje potrebno nam je ocenjivanje pogodaka. Želimo da nađemo dokumente koji su najkorisniji za korisnika - želimo sortiran rezultat. Kako možemo da rangiramo dokumente u odnosu na upit? Potrebno je da

dodelimo ocenu (eng. *score*) svakom dokumentu. Ta ocena, na primer, može biti iz intervala $[0, 1]$.

5.3.1 Ocena relevantnosti

Ocena je mera koliko se dokument i upit “poklapaju” (eng. *match*). Treba nam način za dodelu ocene svakom paru upit/dokument. Razmotrimo prvo upit sa jednim termom. Ako se term ne pojavljuje u dokumentu, ocena bi trebalo da bude 0. Što češće se term pojavljuje u dokumentu, ocena bi trebalo da bude veća. Razmotrimo da ovu ocenu definишemo pomoću *Jaccard*-ovog koeficijenta. *Jaccard*-ov koeficijent je uobičajena mera preklapanja dva skupa definisana na sledeći način:

1. Neka su A i B skupovi (bar jedan je neprazan),
2. *Jaccard*-ov koeficijent:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

,

3. $\text{JACCARD}(A, A) = 1$,
4. $\text{JACCARD}(A, B) = 0$ ako je $A \cap B = 0$.

Skupovi koji se posmatraju ne moraju imati isti broj elemenata. *Jaccard*-ov koeficijent je uvek broj između 0 i 1. Primer za vežbu - koja je ocena upit/dokument dobijena pomoću *Jaccard*-ovog koeficijenta za:

- upit: “ides of March”,
- dokument: “Caesar died in March”.

Odgovor je 0,1667, odnosno $1/6$.

Šta nije dobro kod korišćenja *Jaccard*-ovog koeficijenta za ocenu para upit/dokument:

- ne uzima u obzir frekvenciju terma (koliko puta se term pojavljuje),
- retki termovi su informativniji od čestih - *Jaccard* ovo ne uzima u obzir.

Treba nam bolji način za normalizaciju dužine, kasnije ćemo koristiti $|A \cap B| / \sqrt{|A \cup B|}$ (cosine) umesto $|A \cap B| / |A \cup B|$ (Jaccard) za normalizaciju dužine.

5.3.2 Frekvencija terma

Podsetimo se binarne matrice incidencije.

	Ivanović D. [19]	Milosavljević B. [44]	Gostojić S. [45]	Zarić M. [34]	...
digitalan	1	1	1	1	
ucene	1	1	1	1	
dokument	1	1	1	1	
obrazovanje	0	0	1	0	
pretraga	1	1	1	1	
multimedijalan	0	1	1	1	
evaluacija	0	0	0	0	
...					

Svaki dokument je prikazan pomoću binarnog vektora.

Od sada ćemo koristiti matricu koja sadrži informaciju o broju ponavljanja terma u dokumentu - **brojačku matricu**.

	Ivanović D. [19]	Milosavljević B. [44]	Gostojić S. [45]	Zarić M. [34]	...
digitalan	8	46	7	5	
ucene	11	68	3	2	
dokument	41	953	105	204	
obrazovanje	0	0	1	0	
pretraga	11	56	10	30	
multimedijalan	0	96	1	7	
evaluacija	0	0	0	0	
...					

Svaki dokument je prikazan pomoću vektora broja pojavljivanja.

Za razmatranje vektorskog modela koristićemo *model "vreće sa rečima"* (eng. *bag of words*) koji ne uzimamo u obzir *redosled* reči u dokumentu: *John is quicker than Mary* i *Mary is quicker than John* su prikazani na isti način. Ovo je korak unazad u odnosu na pozicioni indeks koji razlikuje ova dva dokumenta i koji se može primeniti i na vektorski model, ali ovo nije od bilo kakve važnosti za razumevanje suštine vektorskog modela, pa ćemo stoga pojednostaviti priču upotrebom modela "vreće sa rečima".

Frekvencija terma $\text{tf}_{t,d}$ terma t u dokumentu d definiše se kao *broj pojavljivanja t u d*. Želimo da koristimo tf kada računamo upit/dokument ocene. Postavlja se pitanje kako da koristimo tf. Sirova frekvencija terma nije ono što nam treba. Dokument sa 10 pojava jednog terma je relevantniji od dokumenta sa jednom pojavom istog terma, ali nije 10 puta relevantniji, jer relevantnost ne raste proporcionalno sa frekvencijom terma. Koristićemo logaritmsku težinu frekvencije terma t u d koja se definiše kao:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \text{tf}_{t,d} & \text{ako je } \text{tf}_{t,d} > 0 \\ 0 & \text{inače} \end{cases}$$

Za određene vrednosti frekvencije terma logaritamska težina frekvencije terma je:

$$0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4\dots$$

Ocena za par upit/dokument ćemo definisati kao sumu po termovima t za q i d :

$$\text{ocena} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$$

Ocena je 0 ako i samo ako nijedan term iz upita nije prisutan u dokumentu.

5.3.3 Frekvencija dokumenta

Retki termovi su infomativniji od čestih. Razmotrimo term koji je *redak* u kolekciji (npr. *digitalizacija*). Dokument koji sadrži

ovaj term je verovatno relevantan → želimo *veliku težinu za retke termove* kao što je *digitalizacija*. Razmotrimo term koji je *čest* u kolekciji (npr. *visoko*, *teško*, *analiza*). Dokument koji sadrži ovaj term je verovatno relevantniji od onog koji ga ne sadrži, ali to nije siguran indikator relevantnosti → želimo pozitivne težine za *za česte termove* kao što su *visoko*, *teško* i *analiza*, ali *manje težine* nego za retke termove. Koristićemo **frekvenciju dokumenta** da uzmemo informativnost terma u obzir prilikom računanja ocene.

df_t je oznaka za frekvenciju dokumenta i predstavlja broj dokumenata u kojima se pojavljuje term t . df je inverzna mera *informativnosti* terma, zbog čega definišemo **idf težinu** terma t kao:

$$idf_t = \log_{10} \frac{N}{df_t}$$

idf je mera *srazmerna informativnosti* terma - nije inverzna mera informativnosti. N je ukupan broj dokumenata u kolekciji. Koristićemo $\log N/df_t$ umesto N/df_t da “ublažimo efekat” idf-a. Dakle, koristimo logaritamsku funkciju i za frekvenciju terma i za frekvenciju dokumenta.

Pod prepostavkom da je u kolekciji 1.000.000 dokumenata u narednoj tabeli su prikazane idf_t za određene vrednosti df_t .

term	df_t	idf_t
XMIRS	1	6
digitalizacija	100	4
nedelja	1000	3
analiza	10.000	2
ispod	100.000	1
i	1.000.000	0

idf utiče na rangiranje samo ako upit ima bar dva terma. Na primer, u upitu “digitalizacija dokumenata”, idf težina povećava relativnu težinu za *digitalizacija* i smanjuje relativnu težinu za *dokumenata*. idf nema uticaja na rangiranje rezultata upita sa jednim termom.

5.3.4 Frekvencija kolekcije

Frekvencija kolekcije terma t je broj pojavljivanja t u kolekciji i označava se sa cf_t . Ova mera se retko koristi za utvrđivanje informativnosti terma, mnogo se češće koristi frekvencija dokumenta. Prepostavimo da imamo date sledeće termove i njihove frekvencije kolekcije i frekvencije dokumenta.

Reč	cf	df
osiguranje	10440	3997
pokušati	10422	8760

Koji term je informativniji i trebalo bi da dobije veću težinu? Zdrava logika nam kaže da je term **osiguranje** informativniji od reči **pokušati**, ali su frekvencije kolekcije ova dva terma gotovo identične. Želimo da manji broj dokumenata koji sadrži **osiguranje** dobije veći značaj u odnosu na veliki broj dokumenata koji sadrže **pokušati**. Ovaj primer sugerise da je df bolji za težine nego cf , i ovo je često problem sa kolekcijom frekvencije zbog čega se ona retko koristi.

Podsećanja radi, u nastavku su prikazane definicije tri prethodno opisane frekvencije na jednom mestu.

Veličina	Simbol	Definicija
frekv. terma	$tf_{t,d}$	broj pojavljivanja t u d
frekv. dokumenta	df_t	broj dokumenata u kolekciji koji sadrže t
frekv. kolekcije	cf_t	ukupan broj pojavljivanja t u kolekciji

Koja je veza između df i cf ? Koja je veza između tf i cf ?

5.3.5 tf-idf

Jedna od najpoznatijih težina u oblasti pronalaženja informacija je **tf-idf težina** koja se često označava i kao tf.idf ili tf x idf. tf-idf težina terma je *proizvod njegove tf težine i njegove idf težine*:

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$

tf-idf težina je zavisna od terma i od dokumenta, odnosno tf-idf težinu možemo dodeliti svakom termu t za svaki dokument d . Ova težina raste sa brojem pojavljivanja terma u dokumentu i sa retkošću terma u kolekciji. To je upravo ono što smo želeli da postignemo.

5.3.6 Težinska matrica

Podsećanja radi, prvo smo imali binarna matricu koja je sadržala samo informaciju da li se određeni term pojavljuje u dokumentu. Nakon toga smo imali brojačku matricu koja je sadržala informacije koliko puta se određeni term pojavljuje u dokumentu. Sada je vreme da definišemo **težinsku matricu** koja u svojim celijama sadrži tf-idf težine.

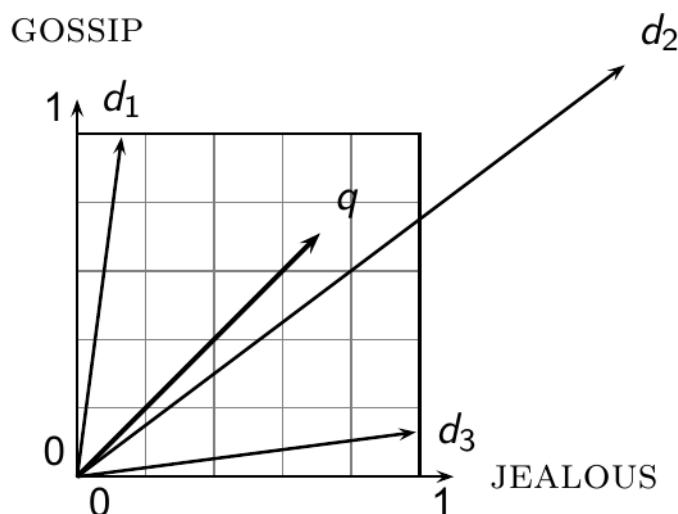
	Ivanović D. [19]	Milosavljević B. [44]	Gostojić S. [45]	Zarić M. [34]	...
digitalan	0.25	1.51	0.12	0.05	
ucene	0.75	3.2	0.28	0.18	
dokument	0.12	2.3	0.51	0.83	
obrazovanje	0	0	0.5	0	
pretraga	0.25	1.8	0.24	0.26	
multimedijalan	0	8.25	0.12	0.23	
evaluacija	0	0	0	0	
...					

Svaki dokument je predstavljen vektorom realnih vrednosti tf-idf težina $\in \mathbb{R}^{|V|}$. Tako imamo $|V|$ -dimenzionalni vektorski prostor. Termovi su *ose* prostora. Dokumenti su *tačke* ili *vektori* u ovom prostoru. Prostor ima visoku dimenzionalnost: desetak miliona dimenzija kada se primeni na veb pretraživač. Dokumenti su vrlo retki vektori - vrednosti na većini osa su 0.

Upite, kao i dokumente, možemo predstaviti kao vektore u vektorskem prostoru. Kada to uradimo onda možemo rangirati

dokumente prema njihovoj *blizini* u vektorskem prostoru sa upitom. Dakle, blizina nam zapravo predstavlja sličnost dokumenta i upita. Blizinu kao mjeru treba definisati tako da je obrnuto сразмерna rastojanju vektora - što je rastojanje manje, mera za blzinu treba da bude veća i obrnuto. Podsećanja radi, ovo radimo da prevaziđemo problem "ili jesi ili nisi" Bulovog modela.

Kako formalno opisati sličnost u vektorskem prostoru, odnosno blizinu vektora? Pokušajmo prvo da to definišemo kao rastojanje između dve tačke, odnosno rastojanje između krajnjih tačaka dvaju vektora. Rastojanje između dve tačke možemo definisati kao Euklidsko rastojanje, ali to je loša ideja jer je ovako definisano rastojanje veliko za vektore različitih dužina čak i ako su oni slični (slika 5.1).



Slika 5.1: Euklidsko rastojanje (preuzeto iz [38])

Euklidsko \vec{q} i \vec{d}_2 je veliko iako je distribucija termova u upitu q i dokumentu d_2 vrlo slična.

Pokušajmo da rangiramo dokumente prema uglu koji zaklapaju sa upitom. Uzmimo dokument d i dodajmo ga još jednom na njegov kraj - nazovimo to d' . "Semantički" d i d' imaju isti sa-

držaj. Ugao između dokumenata je 0, što odgovara maksimalnoj sličnosti. Euklidsko rastojanje između d i d' je veliko - u svakom slučaju > 0 . Dakle, rangiranje prema uglu je bolje nego rangiranje po Euklidskom rastojanju vrhova vektora.

Rekli smo da želimo da blizinu kao meru definišemo tako da je obrnuto srazmerna rastojanju vektora, odnosno da je srazmerna sličnosti dokumenta i upita. Ako koristimo ugao između vektora to nije ispunjeno. Što je ugao manji, to je sličnost veća i obrnuto. Zbog toga ćemo koristiti kosinus ugla koji je monotono opadajuća funkcija ugla u intervalu $[0^\circ, 180^\circ]$. Sledeće dve stvari su ekvivalentne i sortiraju dokumente od najsličnijeg do najmanje sličnog sa nekim upitom:

- rangiraj dokumente prema uglu između upita i dokumenta u rastućem redosledu,
- rangiraj dokumente prema $\cos(\text{query}, \text{document})$ u opadajućem redosledu.

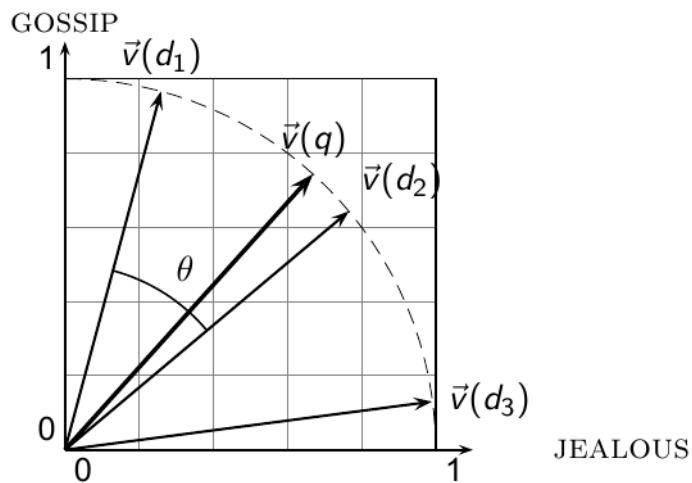
Dakle, pronašli smo meru za sličnost upita i dokumenta: $\text{kosinus ugla između upita i dokumenta}$. Kako da izračunamo ovu meru? Kako da izračunamo kosinus? Kosinus ugla između \vec{q} i \vec{d} se računa na sledeći način:

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| |\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i je tf-idf težina terma i u upitu,
- d_i je tf-idf težina terma i u dokumentu,
- $|\vec{q}|$ i $|\vec{d}|$ su dužine \vec{q} i \vec{d} .

Vektor se može normalizovati deljenjem svake komponente njegovom dužinom - ovde koristimo L_2 normu: $\|x\|_2 = \sqrt{\sum_i x_i^2}$. Ovo premešta sve vektore u jediničnu sferu jer je nakon normalizacije: $\|x\|_2 = \sqrt{\sum_i x_i^2} = 1$. Kao rezultat, i kratki i dugački

dokumenti imaju težine istog reda veličine. Efekat primene normalizacije na dva dokumenta d i d' (d dodat na samog sebe) iz prethodno opisanog primera je da imaju *identične vektore* nakon normalizacije. Zašto je bitna normalizacija? Pojednostavljuje se računanje kosinusa. Ako su \vec{q} i \vec{d} normalizovani onda je $\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2} = 1$, pa je kosinus jednak skalarnom proizvodu $\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$. Na ovaj način indeksiranje, odnosno kreiranje težinske matrice nam traje duže, jer se zahteva i normalizacija, odnosno deljenje svake komponente dužinom vektora. Sa druge strane, pretraživanje nam traje kraće, tu više nemamo deljenja. Na slici 5.2 je demonstracija kako se na osnovu normalizovanih vektora dokumenata i upita pomoću kosinusa ugla može utvrditi sličnost dokumenata i upita. Zbog preglednosti primera, vektorski prostor ima samo dve dimenzije, odnosno imamo samo dva terma: GOSSIP i JEALOUS.



Slika 5.2: Kosinus ugla kao sličnost upita i dokumenta (preuzeto iz [38])

Na sledećim primerima je pokazan postupak kreiranja težinske matrice i procesiranja upita u Vektorskem modelu. Demonstracije radi, pretpostavimo da imamo tri disertacije u kolekciji doktorskih disertacija:

- ID: Ivanović D. [19],
- MB: Milosavljević B. [44],
- GS: Gostojić S. [45].

Pretpostavimo da imamo takvo preprocesiranje teksta da su nam ostali samo sledeći termovi (ostali su izbačeni):

- indeksiranje,
- dokument,
- obrazovanje,
- multimedijalan.

Brojačka matrica je data u nastavku.

frekv. terma (broj)

term	ID	MB	GS
indeksiranje	5	58	0
dokument	41	953	105
obrazovanje	0	0	1
multimedijalan	0	96	1

Kako ćemo kreirati težinsku matricu?

Prvo ćemo na osnovu datih $tf_{t,d}$ izračunati $w_{t,d}$ koristeći formula:

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{inače} \end{cases}$$

Nakon ovog računa dobija se sledeća matrica:

log frekv.

term	ID	MB	GS
indeksiranje	1,7	2,76	0
dokument	2,61	3,98	3,02
obrazovanje	0	0	1
multimedijalan	0	2,98	1

Izračunaćemo idf za svaki term koristeći formulu:

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

N je 3 (imamo tri dokumenta u kolekciji), df_t za term **dokument** je 3, za termove **indeksiranje** i **multimedijalan** je 2, a za term **obrazovanje** je 1.

Dobijene vrednosti su sledeće:

frekv. dok

term	idf
indeksiranje	0,176
dokument	0
obrazovanje	0,477
multimedijalan	0,176

Sada ćemo izračunati tf_idf za svaki term i dokument tako što ćemo pomnožiti prethodno izračunate tf i idf vrednosti prikazane u prethodne dve matrice. Dobijamo sledeću matricu:

tf_idf

term	ID	MB	GS
indeksiranje	0,29	0,49	0
dokument	0	0	0
obrazovanje	0	0	0,48
multimedijalan	0	0,52	0,18

Sada je potrebno uraditi normalizaciju, odnosno podeliti svaki element matrice sa kvadratnim korenom zbiru kvadrata svih elemenata u toj koloni. Elemente u prvoj koloni ćemo deliti sa $\sqrt{0,29^2 + 0^2 + 0^2 + 0^2} = 0,29$, elemente u drugoj ćemo deliti sa $\sqrt{0,49^2 + 0^2 + 0^2 + 0,52^2} = 0,71$, a elemente u trećoj koloni sa $\sqrt{0^2 + 0^2 + 0,48^2 + 0,18^2} = 0,51$.

tf_idf
& normalizacija

term	ID	MB	GS
indeksiranje	1	0,69	0
dokument	0	0	0
obrazovanje	0	0	0,94
multimedijalan	0	0,73	0,35

Na ovaj način smo dobili težinsku matricu. Razmotrimo kako bismo odgovorili na upit:

...multimedijalnih ...indeksiranje ...multimedijalnog

pri čemu ... predstavljaju delove upita koji će nakon pretprocesiranja upita biti izbačeni. Potreban nam je i vektor za upit.

Kolona iz brojačke matrice za ovaj upit je:

frekv. terma (broj)

term	upit
indeksiranje	1
dokument	0
obrazovanje	0
multimedijalan	2

Na osnovu datih $tf_{t,d}$ izračunaćemo $w_{t,d}$:

log frekv.

term	upit
indeksiranje	1
dokument	0
obrazovanje	0
multimedijalan	1,3

Za idf koristimo prethodno izračunate vrednosti. idf se ne računa ponovo zato što imamo i upit - upit se ne računa da je dokument u kolekciji iako računamo vektor za njega.

frekv. dok

term	idf
indeksiranje	0,176
dokument	0
obrazovanje	0,477
multimedijalan	0,176

Sada ćemo izračunati tf_idf za svaki term i upit tako što ćemo pomnožiti prethodno izračunate tf i idf vrednosti prikazane u prethodne dve matrice. Dobijamo sledeću matricu:

$$tf_idf$$

term	upit
indeksiranje	0,18
dokument	0
obrazovanje	0
multimedijalan	0,23

Sada je potrebno uraditi normalizaciju vektora upita, odnosno podeliti svaki element sa $\sqrt{0,18^2 + 0^2 + 0^2 + 0,23^2} = 0,5323$. Ukupna težinska matrica (i dokumenata i upita) je:

$$tf_idf \\ \& \text{normalizacija}$$

term	ID	MB	GS	upit
indeksiranje	1	0,69	0	0,62
dokument	0	0	0	0
obrazovanje	0	0	0,94	0
multimedijalan	0	0,73	0,35	0,79

Izračunajmo sada kosinuse uglova između ovih normalizovanih vektorâ:

- $\cos(\text{ID}, \text{upit}) = 1 * 0,62 + 0 * 0 + 0 * 0 + 0 * 0,79 \approx 0,62$
- $\cos(\text{MB}, \text{upit}) = 0,69 * 0,62 + 0 * 0 + 0 * 0 + 0,73 * 0,79 \approx 1$
- $\cos(\text{GS}, \text{upit}) = 0 * 0,62 + 0 * 0 + 0,94 * 0 + 0,35 * 0,79 \approx 0,28$

Odgovor koji najbolje odgovara upitu je disertacija MB koji ima oba terma iz upita, nakon njega su disertacije DI i GS koje

nemaju oba terma iz upita. Dakle, više nam model nije "ili jesи ili nisi" i rezultate možemo da rangiramo. Možemo da vratimo i samo K najboljih odgovora, a možemo i da odredimo donju liniju kosinusne sličnosti ispod koje smatramo da rezultat nije relevantan - ne mora ta donja linija biti vrednost 0.

Prethodno opisani algoritam za pretragu putem vektorskog modela možemo opisati na sledeći način:

1. predstavljanje svakog dokumenta u formi normalizovanog tf-idf vektora,
2. predstavljanje upita u formi normalizovanog tf-idf vektora,
3. računanje kosinusnih sličnosti između upita i svakog dokumenta,
4. rangiranje dokumenata prema sličnosti,
5. prikazivanje najboljih K (npr. $K = 10$) dokumenata korisniku.

5.4 Pretraga strukturiranih tekstualnih dokumenata

Kao što je već ranije rečeno prema sadržajima u kolekciji sistemi za pretragu tekstualnih sadržaja se dele na dve vrste - sisteme za pretragu nestrukturiranih tekstualnih sadržaja i sisteme za pretragu strukturiranih tekstualnih sadržaja. **Pretraga strukturiranih tekstualnih sadržaja** je nešto između pretrage baze podataka i pretrage nestrukturiranih tekstualnih sadržaja. U pitanju je pretraga sadržaja koji imaju strukturu, ali su elementi u strukturi bogati tekstualnim sadržajima (dugački tekstuálni sadržaji), pa je zgodno imati neke osobine karakteristične za sisteme za pretraživanje, kao što su normalizacija upita i tekstova, relevantnost dokumenta za određeni upit, sortiranje odgovora po relevantnosti... U literaturi se ponekad koristi i termin *semistructured retrieval* da bi se razlikovalo od pretrage baze podataka. U pretrazi ovakvih sadržaja se kombinuju tekstualni kriterijumi i

5.4 Pretraga strukturiranih tekstualnih dokumenata 107

strukturalni kriterijumi. Pretraga strukturiranih tekstualnih sadržaja se deli na dve vrste:

- pretraga po parametrima i zonama (eng. *Parametric and zone search*),
- pretraga složenijih struktura - to je najčešće XML, iako može i neka druga struktura.

5.4.1 Pretraga po parametrima i zonama

Pretraga po parametrima i zonama kao što joj sam naziv kaže omogućuje pretragu po

- parametrima: datum izmene, pripadnost nekoj grupi, geografska pripadnost, redni broj... ,
- zonama: naslov, apstrakt, uvod, ključne reči... .

Sadržaji koji se nalaze u parametrima se obično ne pretrprocesiraju, ali se može pretraživati po njima. Dakle, pretraga po parametrima je zapravo neki vid filtriranja rezultata. Sadržaji koji se nalaze u zonama se pretrpocesiraju što omogućuje fleksibilnu pretragu po zonama nezavisnu od velikih i malih slova, oblika reči, korišćenog sinonima... Zone mogu biti svi metapodaci jednog digitalnog dokumenta. Takođe, jedna zona može biti i potpuni tekst digitalnog dokumenta. Kod sistema za pretragu po parametrima i zonama obično se korisniku dozvoljava unos slobodnog teksta i odabir zone po kojoj se pretražuje (slika 5.3), a pretraga po parametrima nema unos slobodnog teksta, nego se biraju neke vrednosti iz šifarnika.

Da bismo omogućili pretragu po parametrima i zonama neophodno je da drugačije organizujemo indekse, odnosno potrebna nam je drugačija struktura podataka koja omogućuje ovakvu pretragu. U nastavku je prikazana jedna struktura za invertovani indeks za Bulov model koja omogućuje pretragu po parametrima i zonama. Često je neophodno omogućiti i pretragu po svim zonama, kao na primer: pronađi sve dokumente koji imaju term

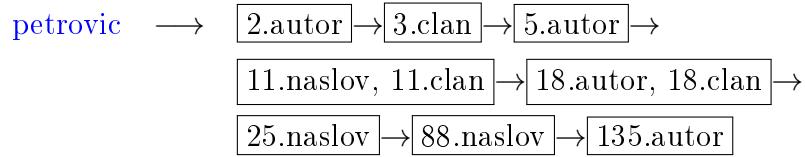
Slika 5.3: Pretraga po zonama

[petrovic](#) u bilo kojoj zoni. U ovakvoj strukturi ovo se može postići pisanjem odgovarajućeg Bulovog OR upita sa svim zonama. Drugi način da se ovo postigne je da se prilikom indeksiranja dodaje zona [sve](#) koja je kreirana upravo za ovu namenu.

petrovic.naslov	→	[11] → [25] → [88]
petrovic.autor	→	[2] → [5] → [18] → [135]
petrovic.clan	→	[3] → [11] → [18]
petrovic.sve	→	[2] → [3] → [5] → [11] → [18] → [25] → [88] → [135]

U nastavku je prikazana druga struktura za invertovani indeks za Bulov model koji omogućuje pretragu po parametrima i zonama. Ovakva struktura može lako i potpuno prirodno da odgovori na prethodno diskutovani upit koji zahteva da se pronađe određeni term u bilo kojoj zoni, ali je procesiranje upita koji zahtevaju da se određeni term pronađe u tačno određenoj zoni nešto sporije, jer je lista pojava duža.

5.4 Pretraga strukturiranih tekstualnih dokumenata 109



Ako želimo da imamo rangiranu pretragu, neophodno je da koristimo model koji to podržava kao što je vektorski model za pretraživanje. U tom slučaju nekoj zoni možemo dati veću težinu, na primer: naslov ima veću težinu nego autor koji ima veću težinu nego član komisije. Ako je upit **Petrović**, relevantnija je disertacija koja u naslovu ima ovu reč od disertacije čije je jedan od članova komisije sa ovim prezimenom. U obzir treba uzeti i tf-idf - ako jedan dokument ima **Petrović** u naslovu jedanput, i nema je nigde više, a drugi dokument nema u naslovu ovu reč, ali je autor **Petrović** i među članovima komisije je **Petrović** i spominje 1.000 puta u tekstu ovo prezime, postavlja se pitanje šta je relevantnije? Težine su samo koeficijenti sa kojima se množe mere koje označavaju relevantnost (npr. tf-idf). Težine koje se dodeljuju zonama određuju ili projektanti sistema za pretragu, ili korisnici putem interfejsa ili se koriste *machine learning* tehnike za utvrđivanje težina zonama na osnovu prethodno postavljenih pitanja i selektovanih odgovora.

5.4.2 Pretraga tekstualnih sadržaja složenih struktura

Iako **pretraga strukturiranih ili “polustrukturiranih” tekstualnih sadržaja** složenijih struktura ne specificira striktno format strukture, mi ćemo se u nastavku ograničiti na **pretragu XML sadržaja**. XML je pogodna struktura za opis strukturiranog ili “polustrukturiranog” sadržaja. Sastoji se iz elemenata (eng. *node*) koji imaju otvarajuće i zatvarajuće tagove. Ovi elementi se mogu ugnježdavati i formirati proizvoljnu hijerarhiju. Ne smeju se ukrštati, odnos između tagova je takav da se uvek zna ko kome pripada. Elementi mogu (a ne moraju) imati jedan ili više atributa. Atributi imaju ime i vrednost, i navode se u otvarajućem tagu. Krajnji elementi se zovu list elementi (eng.

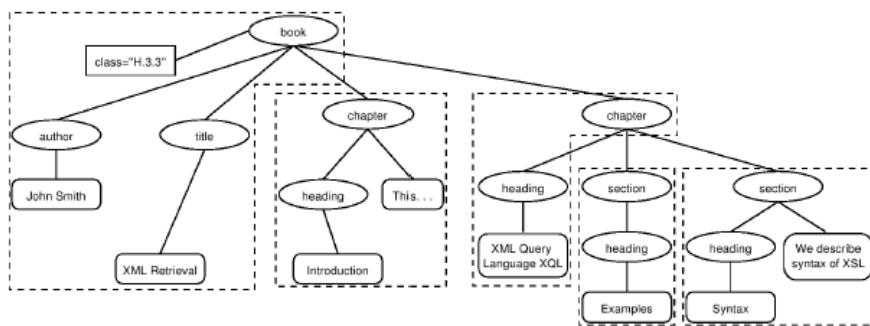
leaf nodes). Tekstualni elementi su list elementi koji imaju samo tekst, nemaju tagove. *DTD* i *XML Scheme* su dva standardizovana načina za opis šeme XML dokumenta. *XPath* je standard za izraze kojima se vrši selekcija elemenata. *Document object model (DOM)* predstavlja objektnu reprezentaciju XML-a koja se često koristi u programskoj obradi XML-a.

XML se može koristiti kao sredstvo za komunikaciju između aplikacija, pri čemu su u XML zapisu najčešće strukturirani podaci iz relacione baze. Ovakav XML se zove **data-centric XML**. Sa druge strane, XML može sadržati “polustrukturirani” sadržaj, odnosno čvorovi XML-a mogu biti bogati tekstrom. Ovakav XML se zove **document-centric XML**. Zapravo, u pitanju su duži tekstualni dokumenti tagovani XML-om: tehnička uputstva, časopisi... Ako imamo kolekciju ovakvih XML-ova, onda imamo potrebu za pretraživanjem ovih sadržaja. Na primer: “daj mi `<section>` u kome se objašnjava kako se menja sijalica”, “nadi ISBN brojeve knjiga u kojima se bar tri poglavlja bave proizvodnjom kafe, rangirane po ceni knjige”... Osnovni princip za pretragu složenih strukturiranih tekstualnih sadržaja je da sistem uvek treba da vrati kao rezultat najspecifičniji (najmanji) deo dokumenta koji odgovara na informacionu potrebu korisnika. Ovo nije uvek tako jednostavno. Na primer, imamo upit `title:Macbeth`, naslov cele tragedije je *Macbeth* i naslov jedne scene *Act I, Scene vii, Macbeth's castle* su oboje relevantni, jer sadrže term `Macbeth`. Šta treba vratiti kao odgovor, XML element koji predstavlja celu tragediju ili njegov podelement koji predstavlja *Scenu vii*? U nekim situacijama ne treba vratiti manji nego veći deo XML-a. Za razliku od pretrage XML strukturiranih sadržaja, kod prethodno opisane pretrage po parametrima i zonama se zna šta je dokument koji se indeksira i koji je rezultat pretrage, nema hijerarhije u strukturi teksta, ima manje atributa i čvorova (zona) nego kod XML strukturiranih sadržaja.

Dakle, jedan od osnovnih problema kod pretrage sadržaja iz XML dokumenata je ustanoviti *šta je dokument koji se indeksira* (eng. *indexing unit*) i *šta je dokument koji je odgovor na upit* (eng. *result unit*). Ovde imamo nekoliko poznatih pristupa. Od

5.4 Pretraga strukturiranih tekstualnih dokumenata 111

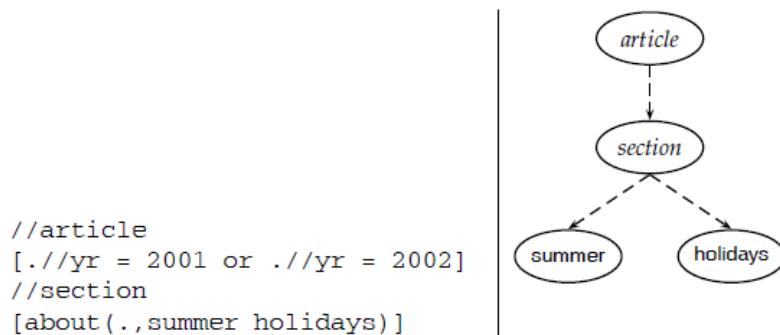
pristupa za odabir *indexing unit*-a nam zavisi kako računamo tf i idf ako se odlučimo za vektorski model pretraživanja. Prvi pristup je da je *indexing unit* najveći *node*. U ovom pristupu imamo potrebu za post-procesiranjem rezultata, ako je jedan veliki *node* odgovor, onda ulazimo u njega i gledamo koji to njegov deo treba prikazati kao odgovor. Problem kod ovog pristupa je što najrelevantniji veliki *node* često ne sadrži najrelevantniji podelement - on može biti u nekom elementu koji u celini nije ocenjen kao mnogo relevantan. Drugi pristup je suprotan - *indexing unit* je najmanji *node*, odnosno *leaf*. Ako taj pronađeni *node* proglašimo odgovorom često je nedovoljno informativan - trebalo bi ga proširiti. Takođe, često imamo i multiplikiranje odgovora, pronađeni su identični mali *node*-ovi više puta u dokumentu. Treći pristup je da se svi XML *node*-ovi uzimaju za *indexing unit*. I kod ovog pristupa je neophodno post-procesiranje rezultata, ako je jedan mali *node* odgovor, onda ga proširijumo da bi dobili nešto što treba prikazati kao odgovor. Ako je odgovor preveliki, onda tražimo relevantne podelemente, pa opet dolazimo do problema da najrelevantniji mali podelement često ne pripada najrelevantnjem inicijalnom odgovoru. Četvrti pristup je deljenje *node*-ova na nepreklapajuće *indexing unit*-e. U ovom slučaju odgovori nisu koherentni i često zbnjuju korisnike - jedan odgovor je samo apstrakt, drugi je sekcija u knjizi... Primer ove podele je prikazan na slici 5.4.



Slika 5.4: Deljenje elemenata na nepreklapajuće *indexing unit*-e (preuzeto iz [38])

Još jedan problem kod pretrage sadržaja u XML dokumentima je *heterogenost šema*. Idealno bi bilo da postoji samo jedna šema koju korisnici razumeju. U praksi je to jako retko. Obično postoji više šema koje nisu poznate unapred. Šeme se menjaju i korisnici ih ne razumeju. Potrebno je identifikovati slične elemente u različitim šemama i proširiti upit. Na primer, podaci o zaposlenima se čuvaju u različitim elementima u različitim šemama.

Prilikom razvoja sistema za pretragu neophodno je voditi računa o dobrom *korisničkom interfejsu* koji će omogućiti korisniku da pronađe relevantne čvorove u različitim šemama: *author*, *editor*, *contributor*, *sender*. Korisnički interfejse treba da predstavlja poseban sloj između korisnika i XML-a. Postoje specijalizovani XML upitni jezici kao što je *Narrowed Extended XPath I (NEXI)*. NEXI je standardizovani format za XML upite koji podržava definisanje *relational attribute constraints* ($./\text{yr} = 2001$ or $./\text{yr} = 2002$) i *ranking constraints* (*summer holidays*) kao što je prikazano na slici 5.5).



Slika 5.5: NEXI upitni jezik (preuzeto iz [38])

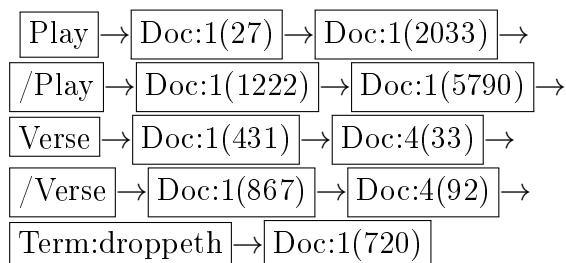
U većini slučajeva korisnik nije dovoljno vešt da se snalazi sa ovakvim upitnim jezicima, pa je potrebno implementirati forme putem kojih korisnik izražava svoju informacionu potrebu. Poslednjih godina je aktuelna i ideja koja je centralna tema jedne COST akcije finansirane od strane Evropske Unije (www.cost.eu). Naziv akcije je *Semantic keyword-based search on structured data*

5.4 Pretraga strukturiranih tekstualnih dokumenata 113

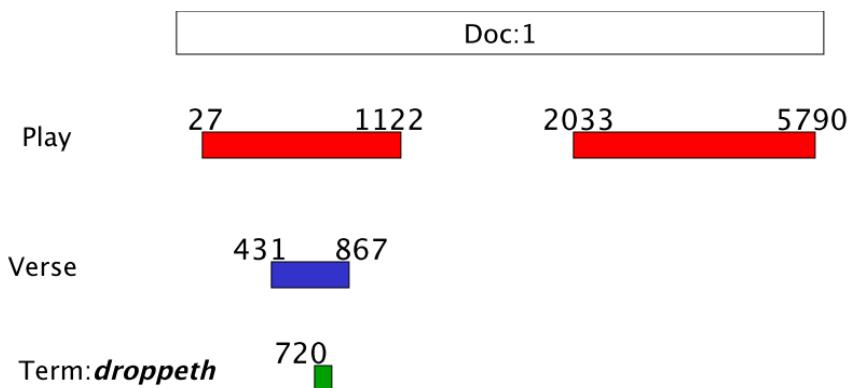
sources (www.keystone-cost.eu). Osnovna ideja je da korisnik unosi upit u formi ključnih reči, a da sistem prepozna korisnikovu potrebu i kreira strukturirani upit prilagođen strukturiranim podacima koji se nalaze u kolekciji.

Kako organizovati indekse koji će omogućiti pretragu XML dokumenata. Prethodno je već diskutovano da prvo treba odlučiti šta je *indexing unit* i šta je *result unit*. Odgovor na tekstualne pretrage koje zahtevaju sve elemente koji zadovoljavaju tekstualni upit q se može postići tako što se tretira svaki *indexing unit* kao poseban dokument u invertovanom indeksu. U pretrazi XML dokumenata česti su i strukturni zahtevi. Primer stukturnog zahteva je: potrebni su mi svi elementi koji su deca *book* elementa. Такође, neretki su upiti koji kombinuju tekstualnu pretragu i pretragu po strukturi. Kako opisati veze između roditelja i deteta u XML strukturi elemenata tako da je moguće efikasno odgovoriti na strukturne zahteve? Prvi pristup je dodeliti broj svakom elementu i održavati listu veza roditelj/dete: na primer, Chapter:21 ← Book:8. Ovaj pristup je jednostavan za održavanje veze sa neposrednim pretkom. Šta se dešava ako imamo upit “reč Hamlet ispod Scene elementa ispod Play elementa”? Ovo “ispod” ne znači da je neposredni potomak. Ne zna se u kom elementu je reč Hamlet.

Posmatrajmo sad stvari malo drugačije. Posmatrajmo i dalje XML dokument kao tekstualni dokument, ali napravimo pozicioni indeks za svaki element, odnosno označimo početak i kraj svakog elementa. Na primer:



U ovakovom modelu sadržavanje se može posmatrati kao spašanje (eng. *merge*) liste pojava (slika 5.6).

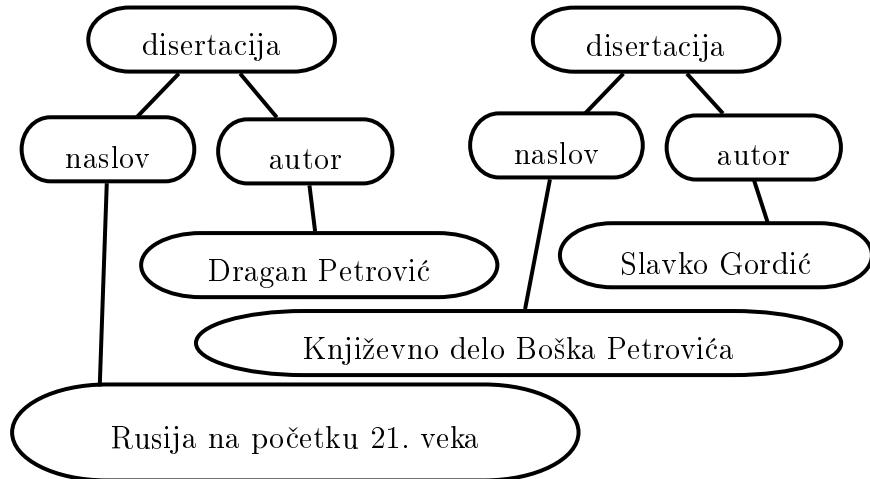


Slika 5.6: Sadržavanje elemenata

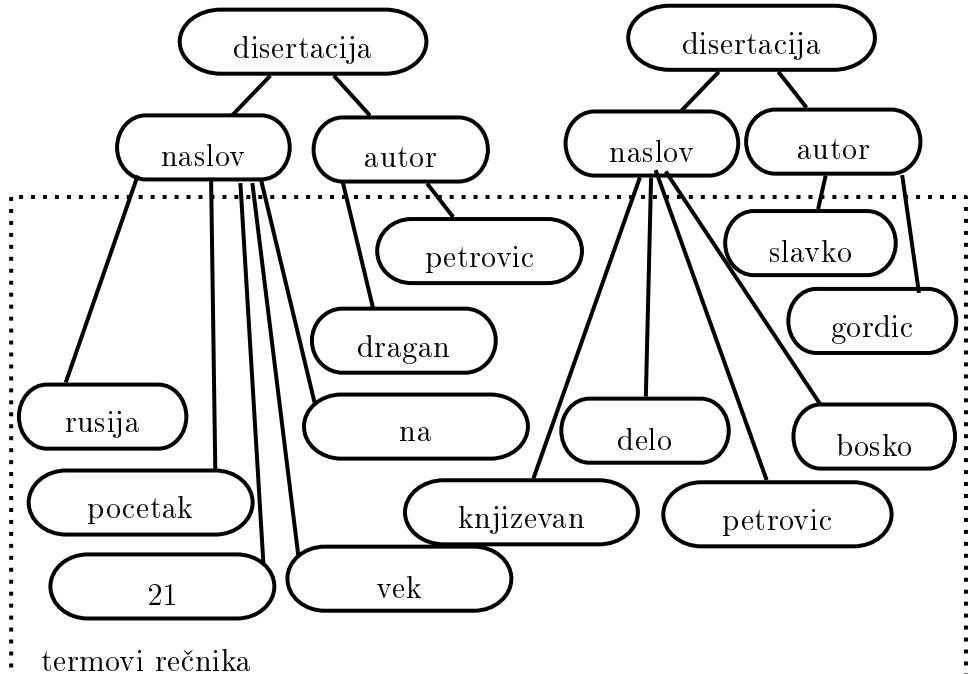
Dakle, sadržavanje podelemenata se može rešiti pozicionim invertovanim indeksom. Pretraživanje podrazumeva "spajanje" liste pojava. Međutim, komplikacije nastaju prilikom dodavanja/uklanjanja elemenata. Ako umetnemo jedan podelement u XML, potrebno je ažurirati listu pojava u invertovanom indeksu za sve sadržaje XML-a koji su iza umetnutog podelementa. Dakle, ako umetnemo element negde blizu početka XML dokumenta, potrebno je ažurirati dobar deo invertovanog indeksa.

Možemo li upotrebiti vektorski model koji je potvrđen u praksi za *document-centric* pretraživanje XML-a? Kako predstaviti tekstualni dokument u vektorskem prostoru je već ranije diskutovano, ali novi problem koji imamo kod pretraživanja XML-a je kako prikazati strukturu XML dokumenta u vektorskem prostoru. Trebalo bi napraviti razliku između sledeća dva slučaja pojavljivanja reči [Petrović](#).

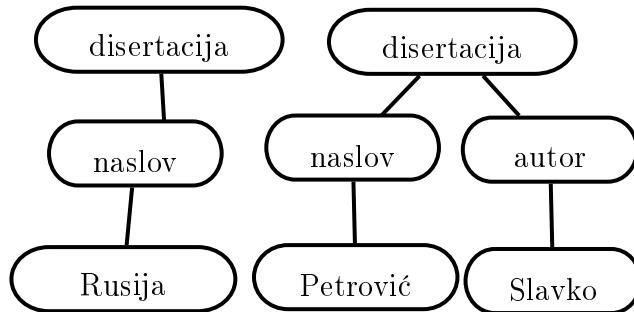
5.4 Pretraga strukturiranih tekstualnih dokumenata 115



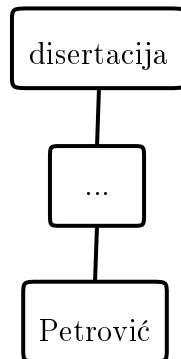
Ose klasičnog vektorskog prostora su termovi. Nakon procesiranja teksta, postojala bi jedna osa za term *petrovic*. Sada treba da razdvojimo njeno pojavljivanje u različitim elementima, **autor** i **naslov**. Ose moraju da opišu ne samo term nego i njegov položaj u stablu dokumenta.



Da bismo ustanovili kako da prilagodimo vektorski model potrebama pretraživanja XML dokumenata, razmotrimo prvo na koje vrste upita je potrebno odgovoriti. Prva vrsta upita je upit kao podstablo dokumenta: tražimo disertacije koje u naslovu imaju reč Rusija; ili tražimo disertacije koje u naslovu imaju reč Petrović, a čiji se autor zove Boško, itd. Ovi upiti predstavljaju traženje podstabla.

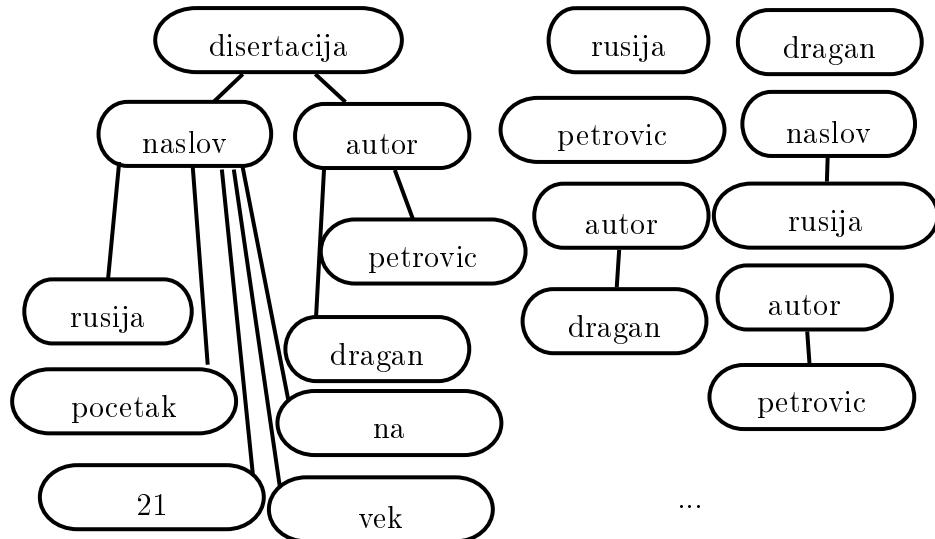


Postoje situacije kada ne želimo da specificiramo tačno podstablo u kojem očekujemo da pronađemo određeni term, samo želimo da pronađemo disertacije u kojima se spominje Petrović, nije bitno u kom podelementu: *Petrović* negde ispod *disertacija*.



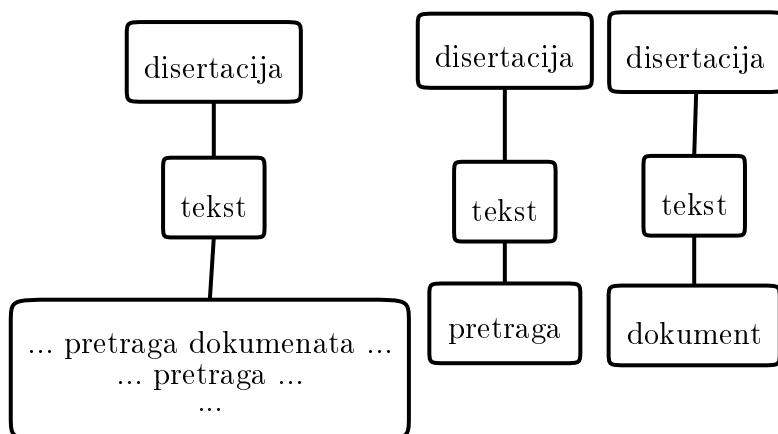
Posmatramo sva podstabla koja sadrže bar jedan term rečnika. Nazovimo sva podstabla **strukturnim termovima**.

5.4 Pretraga strukturiranih tekstualnih dokumenata 117



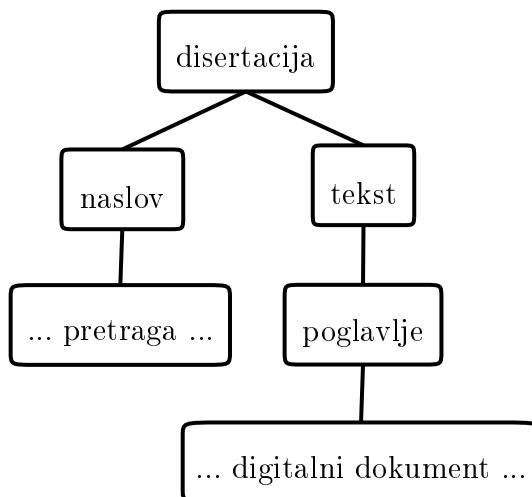
Isti strukturni termovi se mogu pojaviti više puta u dokumentu. Definišimo po jednu osu u vektorskom prostoru za svaki različiti strukturni term. Sve uobičajjene operacije nad tokenima (*lowercase*, *lemmatization*...) se obavljaju pre indeksiranja, odnosno preprocesiranje teksta se obavlja na način kako ga definiše inženjeri sistema za pretraživanje.

Težine ćemo definisati prema broju pojavljivanja strukturnih termova (slično kao tf).



Strukturni term koji sadrži *pretraga* ima veću težinu nego strukturni term koji sadrži *dokument*. I ovde se može logaritmom ublažiti razlika u broju pojavljivanja.

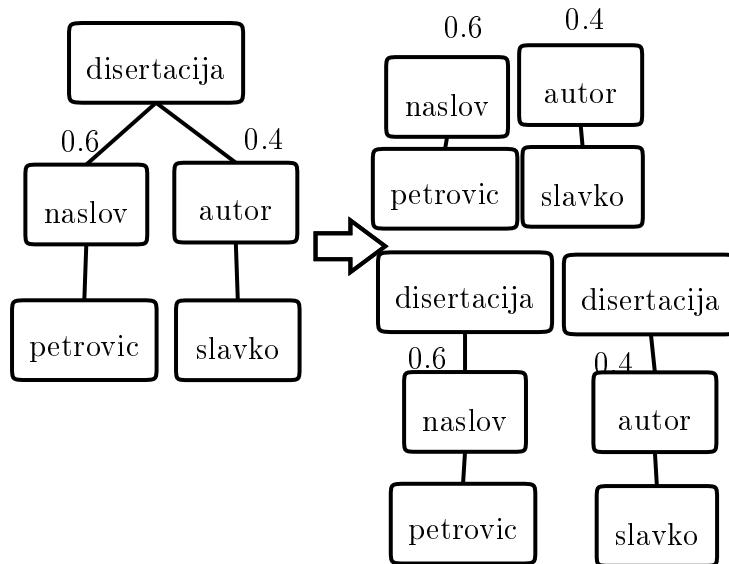
Težina strukturnog terma ne zavisi samo od broja pojavljivanja tog strukturnog terma u XML dokumentu, nego i od dubine na kojoj se pojavljuje taj strukturni term.



Term *pretraga* bi trebalo da ima veću težinu nego *dokument*. To se postiže tako što pomnožimo tf doprinos terma t čvoru koji se nalazi k nivoa iznad sa γ^k za neko $\gamma < 1$. Za prethodni dokument, tf težina terma *pretraga* se množi sa 0,8, a terma *dokument* se množi sa $0,8^2 = 0,64$ za svaki strukturni term sa korenom *disertacija*.

Pojam strukturnog terma ne zavisi od šeme dokumenta, što je zgodno za heterogene kolekcije XML dokumenata. Dokumente predstavljamo kao vektore u prostoru strukturalnih termova. Upit se takođe može rastaviti na strukturne termove i prikazati kao vektor. Mogu se koristiti i različite težine za delove upita.

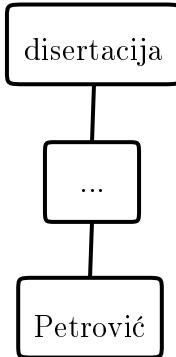
5.4 Pretraga strukturiranih tekstualnih dokumenata 119



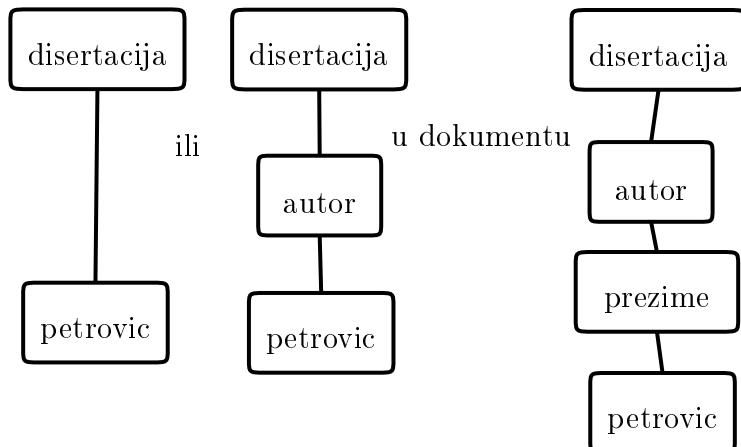
Dodela težina 0,6 i 0,4 u prethodnom primeru je bila suviše pojednostavljena, odnosno dodata težina može biti finija. U nekim situacijama težine generiše aplikacija, ne definiše ih korisnik. Vektori upita i dokumenata se normalizuju, odnosno kreiraju se normalizovani vektori jedinične dužine. Mera sličnosti je kosinussna mera, kao i u klasičnom slučaju.

Koliko dimenzija ima ovako kreirani vektorski prostor? Broj dimenzija može da raste eksponencijalno sa veličinom dokumenta. Indeksirati sva podstabla verovatno nije isplativo. U nekom momentu može postati beznadežno praviti indeks. Većina podstabala se nikad neće koristiti u upitima. Bilo bi idealno znati koja podstabla će se javljati u upitima, onda bismo mogli ograničiti broj strukturnih termova. Na primer, ako nikad nećemo tražiti **autor** čvor, nego samo **naslov** i **disertacija** čvorove, tada u indeks neće ući strukturni termovi čiji koren je **autor**.

Korišćenjem prethodno opisanog modela i dalje se ne može odgovoriti na upit kao što je:



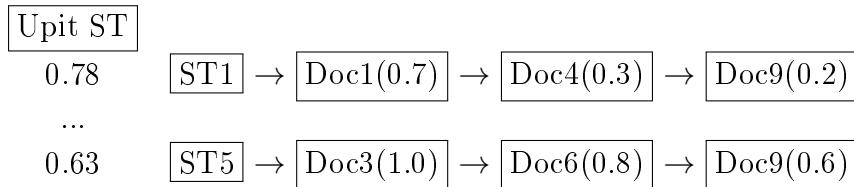
Da bismo omogućili pretragu po potomcima moramo definisati funkciju za poređenje podstabala koja vraća rezultat iz $[0, 1]$, odnosno ako su strukturni termovi putanje, meri poklapanje ovih putanja. Funkcija treba da bude definisana tako da važi: što veće poklapanje, veća ocena, i obrnuto.



Kako da koristimo ovako definisanu funkciju u pretraživanju? Izdvojimo sve strukturne termove u upitu. Pretražimo rečnik strukturalnih termova pri čemu rezultat nije binaran (term postoji/ne postoji u rečniku), nego se stepen poklapanja sa termom izražava brojem iz $[0, 1]$. Dobavimo dokumente iz liste koja je vezana za te strukturne termove i težine koje pronađemo množimo sa prethodno dobijenim stepenom poklapanja, računamo

5.4 Pretraga strukturiranih tekstualnih dokumenata 121

kosinusnu meru sličnosti, elemente rangiramo prema kosinusnoj meri.



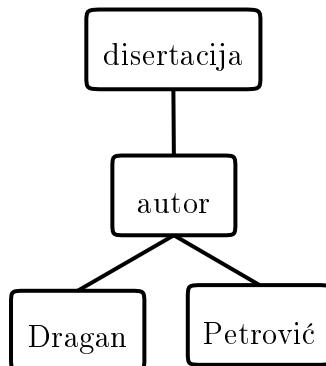
- ST = strukturni upit

Na kakve upite ne možemo odgovoriti u ovakovom vektorskom prostoru?

- “Nađi slike koji opisuju strukturu EJB komponente i pasuse *koji se referenciraju na te slike*” - treba nam u upitnom jeziku nešto slično kao *join* dve tabele u SQL sintaksi.
- “Nađi naslove članaka iz trećeg odeljka u avgustovskom broju časopisa *IEEE Trans on Software Engineering*” - zavisi od redosleda čvorova-braće.

Kod klasičnog vektorskog modela računa se i idf da bi se bolje odredili vektori dokumenata i upita i njihova sličnost. Može li se računati idf u slučaju prethodno opisanog vektorskog modela za pretraživanje XML dokumenata? Da, ali nema smisla računati idf na nivou cele kolekcije, može imati smisla računati za sav tekst u okviru nekog elementa. Na taj način bi dobili tf-idf težinu svakog terma u okviru datog elementa. Komplikovano pitanje koje se ovde javlja je kako propagirati težine u roditeljske čvorove.

Recimo da term **petrovic** ima visok idf u okviru elementa **autor**. Kako da izračunamo tf-idf za **disertaciju**? Da koristimo idf za **petrovic** u elementu **autor** ili elementu **disertacija**?



Rezime

- Jedna klasa ekvivalencije reči se zove term. Ako se reči nakon primene određenih pravila za normalizaciju svode na isti niz znakova onda pripadaju istoj klasi ekvivalencije.
- Tokenizacija teksta je izdvajanje tokena (reči) iz tekstualnog digitalnog dokumenta. Može biti vrlo složena, posebno za jezike koji nemaju jasno definisano razdvajanje reči. Svaki token je kandidat za term.
- Nakon dobijene liste tokena vrši se normalizacija termova koja može uključiti sledeće operacije: prebacivanje velikih u mala slova, izbacivanje dijakritika, izbacivanje stop reči, lematizaciju, stemming, itd.
- Lematizacija podrazumeva pravilnu redukciju na osnovni rečnički oblik (eng. *lemu*), a stemming je grubi heuristički proces koji odseca krajeve reči sa ciljem da postigne rezultat što sličniji onome koji postiže prava lematizacija.
- Bulov model pretraživanja je klasični model pretraživanja koji je prvi bio široko prihvачen u sistemima za pretraživanje. Zasnovan je na teoriji skupova i Bulovoj algebri. Mana mu je što nema mogućnost rangiranja

5.4 Pretraga strukturiranih tekstualnih dokumenata 123

rezultata. Dokument je ili relevantan ili nije, ne postoji parcijalno poklapanje upita i dokumenta.

- Pointeri za preskakanje se koriste kako bi se ubrzao algoritam za odgovor na korisnikov upit kod Bulovog modela pretraživanja. Za odgovor na upite fraze koriste se ili dvorečni indeksi ili pozicioni indeksi.
- Vektorski model omogućuje parcijalno poklapanje upita i odgovora, a samim tim omogućuje i rangiranje rezultata. Dokumente, kao i upit, predstavlja u vektorskom prostoru velike dimenzionalnosti. Koliko ima termova u rečniku toliko ima dimenzija u vektorskom prostoru. Za određivanje ovih težina najčešće se koristi tf-idf mera koja uzima u obzir broj pojavljivanja terma u određenom dokumentu i broj različitih dokumenata u kojima se pojavljuje određeni term.
- Pretraga strukturiranih tekstualnih sadržaja je nešto između pretraga baza podataka i pretrage nestrukturiranih tekstualnih sadržaja. Deli se na pretragu po parametrima i zonama i na pretragu složenijih struktura koje su najčešće izražene upotrebom XML-a.
- Kod sistema za pretragu po parametrima i zonama obično se korisniku dozvoljava unos slobodnog teksta i odabir zone po kojoj se pretražuje, a pretraga po parametrima nema mogućnost unosa slobodnog teksta, nego se biraju neke vrednosti iz šifarnika. Zone i parametri mogu biti metapodaci o digitalnim dokumentima, kao i ekstrahovani tekst iz digitalnog dokumenta. Potrebno je drugačije organizovati indekse nego kod klasične pretrage nestrukturiranih tekstualnih sadržaja.
- Za razliku od pretrage XML strukturiranih sadržaja složenije strukture, kod pretrage po parametrima i zonama se zna šta je dokument koji se indeksira i koji je rezultat pretrage, nema hijerarhije u strukturi teksta,

ima manje atributa i čvorova (zona) nego kod XML strukturiranih sadržaja.

- Za pretragu kolekcije XML dokumenata može se koristiti modifikovani vektorski model koji u rečniku sadrži strukturne termove. U dodeli težina strukturnim termova obično se koristi tf - koliko se puta taj strukturni term pojavljuje u XML dokumentu. Takođe, koristi se i informacija o dubini na kojoj se pojavljuje taj strukturni dokument u hijerarhiji XML dokumenta.
- Da bismo omogućili pretragu po potomcima moramo definisati i funkciju za poređenje podstabla koja vraća rezultat iz [0, 1], odnosno ako su strukturni termovi putanje, meri poklapanje ovih putanja.

Pitanja

1. Koja je razlika između terma i tokena?
2. Šta je tokenizacija i koji problemi postoje u ovoj fazi preprocesiranja?
3. Zašto se vrši „normalizacija“ reči?
4. Šta je to stemming?
5. Šta je to lematizacija?
6. Objasniti Bulov model pretraživanja.
7. Šta je to invertovani indeks i kako se kreira?
8. Objasniti procesiranje upita kod Bulovog modela.
9. Šta su pointeri za preskakanje?
10. Šta se može koristiti ako je potrebno podržati upite fraze?
11. Šta je to dvorečni indeks?
12. Šta je to pozicioni indeks?

5.4 Pretraga strukturiranih tekstualnih dokumenata 125

13. Objasniti Vektorski model pretraživanja.
14. Šta je ocena relevantnosti?
15. Šta je frekvencija terma?
16. Šta je frekvencija dokumenta?
17. Šta je tf-idf?
18. Objasniti kreiranje težinske matrice.
19. Koje su razlike između Bulovog i Vektorskog modela pretraživanja?
20. Navesti osnovne karakteristike pretrage po zonama i parametrima.
21. Navesti osnovne karakteristike pretrage tekstualnih sadržaja složenih struktura.
22. Koje su izmene u klasičnom Vektorskom modelu potrebne za pretragu kolekcije XML dokumenata?

Glava 6

Pretraga veba

Ovo poglavlje ima za cilj da definiše osnovne pojmove pretrage veba i da odgovori na sledeća pitanja:

- Koje su osnovne razlike pretrage kolekcije tekstuálnih digitalnih dokumenta i pretrage veba?
- Kako se vrši prikupljanje veb sadržaja putem veb *crawler-a*?
- Da li se linkovi između veb sadržaja mogu iskoristiti za utvrđivanje kvaliteta veb sadržaja?

Prilikom kreiranja strukture ovog poglavlja i definisanja osnovnih pojnova polazna osnova bila je knjiga “*Introduction to information retrieval*” [38]. Deo informacija prezentovanih u ovom poglavlju su preuzeti iz raznih izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Veb je danas ogromna, globalna baza znanja čija je pretraga neophodna za razvoj društva zasnovanog na znanju. Veb je danas nešto što se smatra normalnim načinom života.

The Web has become the “new normal” in the American way of life; those who don’t go online constitute an ever-shrinking minority. - [Pew Foundation report, January 2005]

Zašto kreirati veb sadržaj ako ga neće moći pronaći korisnici u moru veb sadržaja? Drugi vidovi pronalaženja informacija upotrebom taksonomije ili *bookmark*-a nisu dobri zbog veličine veba. **Pretraga veba** nam omogućava gotovo neograničenu selekciju onoga što želimo da saznamo ili kupimo. **Veb pretraživač** je softverski sistem koji omogućava pretragu informacija na vebu. U srpskom jeziku se često termin veb pretraživač koristi kao termin koji označava *web browser*, ali to nije slučaj u ovoj knjizi. Na dalje u ovom poglavlu će biti korišćen termin veb pretraživač koji označava *web search engine* kao što su Google, Yahoo!, Bing... Pomoću veb pretraživača moguća je agregacija interesa:

- kreiranje zajednica ljudi koji imaju isto interesovanje,
- *online* prodavnice koje prodaju usku paletu prodavnica

Veb je danas ogromno, globalno tržište. Zarada od reklama na veb pretraživačima je ogromno, jer na veb pretraživače dolazi jako puno ljudi.

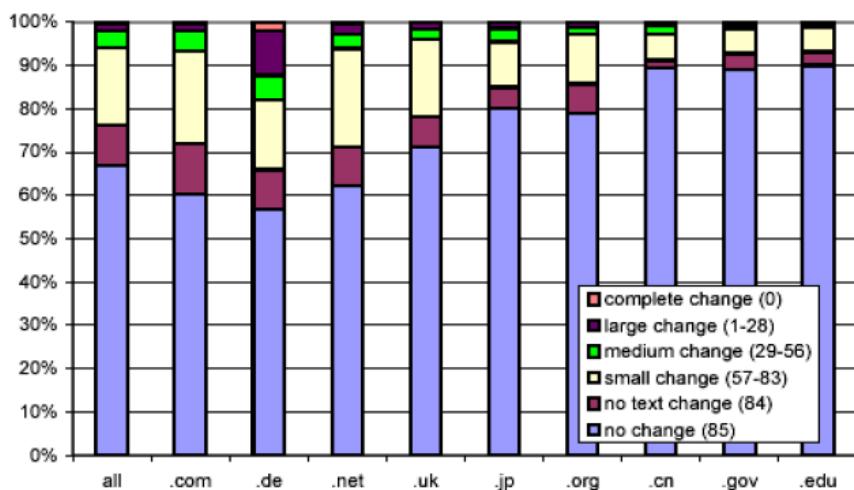
Koja je razlika između pretraživača klasične kolekcije digitalnih dokumenta i pretraživača veba?

Prva razlika je u veličini kolekcije. *Veličina veb sadržaja* prevazilazi sve poznate kolekcije digitalnih dokumenata. Veb je mnogo veći i raste mnogo brže od kolekcija u ostalim IR sistemima. Koliko je *host*-ova, koliko je stranica (statičkih), koja je količina podataka na vebu su pitanja kojima se bavi *NETCRAFT* - <http://news.netcraft.com/>. Tačan broj stranica je praktično nemoguće utvrditi, ali postoje neke numeričke procene o ovom broju. Broj novih veb sadržaja je veći iz dana u dan, iako više ne važi ono što je bilo u jednom momentu razvoja veba “*volume doubling every few months*”, pre svega jer je veb postao veći.

Takođe, razlika je u *načinu kreiranja sadržaja*. Ono što je karakteristično za veb je da nema koordinacije u kreiranju sa-

držaja, odnosno da vlada potpuna demokratija u kreiranju sadržaja koji se može kreirati i distribuirano. *Wikipeadia* se proslavila na distribuiranom kreiranju sadržaja. Na vebu sadržaje kreiraju ljudi različitog znanja i interesa, zbog čega na vebu imamo i dobrih i loših sadržaja. Sadržaji mogu biti nestrukturirani (text, html, ...), polustrukturirani (XML, anotirane fotografije), strukturirani (baze podataka, veb servisi, *RESTfull* servisi). Među sadržajima postoje i dinamički generisani sadržaji - dinamički generisane HTML strane u momentu prijema zahteva. HTTP zahtev za dinamički generisane HTML strane često ima karakter "?". Dinamički generisane HTML strane se generišu u momentu prijema HTTP zahteva. Prilikom generisanja HTML strane uzimaju se trenutni podaci iz baze podataka, na primer: trenutno stanje leta AA129, raspoloživost soba u hotelu... *Spider-i*, odnosno *Crawler-i*, o kojima će biti reči kasnije u ovoj knjizi često ignorisu dinamičke sadržaje da ne bi upali u maliciozne zamke, ali u nekim aplikacijama su vesti dinamički sadržaji, pa se koristi *application-specific spidering*. Na taj način, *statically indexable web* koji predstavlja sve što veb pretraživači indeksiraju, uključuje i neke dinamički generisane HTML strane.

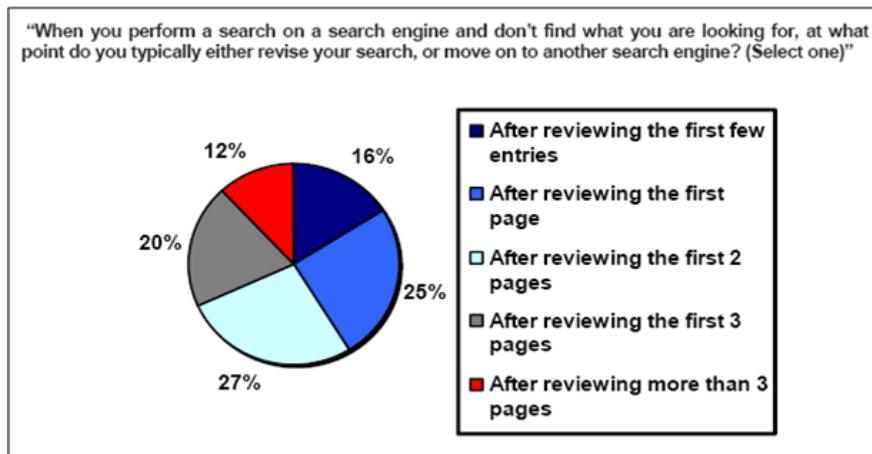
Veb sadržaji se *često menjaju*. Postoji više istraživanja o promeni veb sadržaja, ali najveće je opisano u radu iz 2003. godine [47] - nekoliko puta su pregledani podaci o 150 miliona stranica tokom 11 nedelja istraživanja, postoji 85 različitih nivoa izmena sadržaja (slika 6.1).



Slika 6.1: Promena veb sadržaja (preuzeto iz [48])

Korisnici veb pretraživača pripadaju najširoj mogućoj lepezi korisnika, veb pretraživače koriste ljudi od 5 do 105 godina iz različitih delova sveta i različitih kultura koji mogu biti vrlo pismeni ili nasuprot njih gotovo nepismeni. Tipični korisnici veb pretraživača ne vole da se opterećuju operatorima i složenim upitnim konstrukcijama. Uglavnom formiraju kratke upite - 80% upita na veb pretraživačima ima jednu ili dve reči. Upiti su često nепrecizni, jer tipični korisnici ne ulažu puno intelektualnih napora prilikom kreiranja upita. Iako su upiti nепrecizni vrlo retko se dodatno reformulišu. Na slici 6.2 je prikazana statistika ponašanja korisnika u slučaju da ne mogu da pronađu među odgovorima ono što su tražili.

Postoje velike razlike među korisnicima u potrebama, očekivanjima, znanju, tehničkoj i komunikacionoj opremi, godinama. Cilj pretrage upotrebom veb pretraživača ne mora biti samo ispunjenje *informacione potrebe*, može biti i ispunjenje *navigacione* i *transakcione potrebe*. Ako korisnik želi da nauči nešto o *information retrieval-u* on formira upit koji treba da ispuni njegovu **informacionu potrebu**. Sa druge strane, ako korisnik želi da pronađe veb sajt *Wizzair* kompanije ili veb sajt teniskog turnira



Slika 6.2: Ponašanje korisnika (preuzeto iz [48])

US Open, korisnik postavlja odgovarajući upit koji treba da ispunji njegovu **navigacionu potrebu**. Dakle, korisnik ne želi da sazna šta je to *Wizzair* ili *US Open*, on već zna nešto o tome, ali ne zna njihov veb sajt i koristi veb pretraživač samo kao sponu da bi pronašao veb sajt. Ako korisnik želi da preuzme odgovarajuću datoteku, pristupi odgovaraajućem servisu ili obavi određenu transakciju ili kupovinu, korisnik izražava upitom svoju **transakcionu potrebu**. Na primer, korisniku je potrebno da preuzme *Lucene* biblioteku za implementaciju sistema za pretraživanje, korisnik postavlja upit *download lucene* na nekom veb pretraživaču. Dakle, korisnik ne želi da sazna više o *Lucene* biblioteci, on je već slušao o toj biblioteci na predavanjima na svojim studijama, a sada mu je potrebno preuzimanje te datoteke. Sledеći primer za transakcionu potrebu je da korisnik želi da konvertuje 723 dinara u evre. U tom slučaju zadaje upit *convert 723 RSD to EUR* i veb pretraživač treba da mu pronađe sajt koji može da obavi ovu konverziju. Primer transakcione potrebe za kupovinu određenog upita može biti upit *buy Canon S410*. Većina kompanija koje razvijaju veb pretraživače ulažu velike napore da što bolje prepoznaju koja je korisnikova potreba. U svakom slučaju korisnici bi trebali da se trude da u svom upitu zaista postoji opisana njihova potreba. Ako je u pitanju transakciona potreba

preporučljivo je koristiti glagol koji opisuje transakciju: *download, convert, buy...* Nije uvek tako jednostavno svrstati korisnikovu potrebu u jednu od prethodno opisane tri potrebe: informacionu, navigacionu, transakcionu. Recimo, nekad je korisniku potrebna veb stranica koja daje opšte informacije o nekoj temi sa linkovima na veb sajtove na kojima su detaljno opisane određene informacije ili mogućnost da obavi određenu transakciju. Na primer, korisnikov upit *car rental brasil* može značiti da bi korisnik da dobije veb sajt koji sadrži opšte informacije o iznajmljivanju automobila u Brazilu, kao i mogućnost da pronađe automobil koji želi da iznajmi i obavi rezervaciju istog. Ako želite da putujete na Krit, onda biste mogli postaviti sledeći upit veb pretraživaču: *crete attractions*. Dakle, potreban Vam je veb sajt na kojem su opisane atrakcije koje ima da ponudi Krit, po mogućству nekako klasifikovane i rangirane, i linkovi koji Vas vode na detaljnije informacije o određenoj atrakciji, mogućnost rezervacije ili kupovine karte...

Od veb pretraživača se očekuje da se rezultati prikažu korakno i brzo. Takođe, očekuje se jednostavan interfejs, tolerantan na greške korisnika: *spell checking, did you mean*. Vrlo je korisna i mogućnost da se vide *similar pages* za određenu stranicu. Korisnici ne vole da ih veb pretraživač davi reklamama, *pop up*ima... U prilog ovoj činjenici ide i uspeh *Google* veb pretraživača koji je od starta najmanje opterećivao korisnike reklamama. Korisnici očekuju da dobiju odgovore iz pouzanih izvora, da odgovori nisu duplirani, odnosno da su u listi odgovora samo različiti odgovori koji su lepo organizovani. Korisnicima ne smeta puno što među odgovorima ima i onog što im ne treba, pod uslovom da ima onog što im treba. Bitno je da to što im treba bude izlistano među prvim rezultatima, bitna je preciznost prvih k odgovora - 85% korisnika gleda samo prvu stranu sa rezultatima. Mišljenje jednog korisnika može biti beznačajno, ali mišljenje velike količine korisnika se mora analizirati i uzeti u obzir. Ozbiljni veb pretraživači se mišljenjem korisnika ozbiljno bave i imaju vrlo ozbiljna istraživanja i analize logova (eng. *log mining*). Pošto su upiti korisnika često neprecizni od veb pretraživača se očekuje da čitaju između redova. Ako upit ima samo ime grada najpre će biti

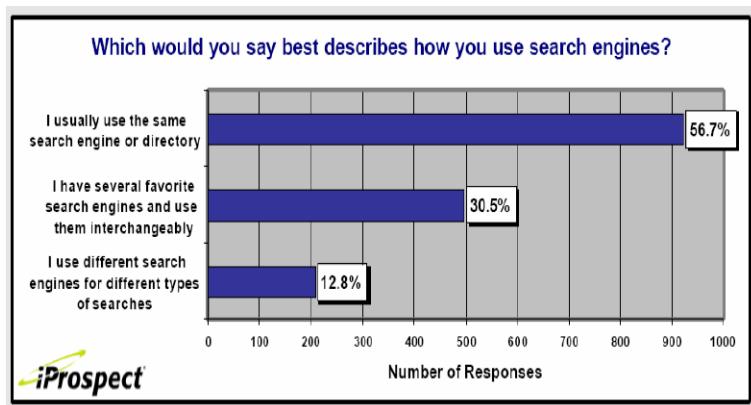
rezultati sa mapom i turističkim vodičima za taj grad. Korisnici, barem oni realni, su zadovoljni ako neprecizno postave upit, jer ni sami nisu sigurni šta traže, a veb pretraživač ih uputi na ono što im treba. Čitanje između redova se ne odnosi samo na *did you mean*, koriste se i tehnike *data mining*-a. Često se vrši i reformulacija upita izraženog prirodnim jezikom tako da se dobije bolja lista odgovora - ne zaboravite i da slabo pismeni koriste veb pretraživače.

Personalizacija pretrage je vrlo poželjna kod veb pretraživača i predstavlja prilagođavanje veb pretraživačima navikama i potrebama jednog korisnika. Ako ste ulogovani na *Google* nalog i koristite veb pretraživač možete videti svoju statistiku pretraga, a onda *Google* to koristi za Vaše buduće pretrage, slično je i kod ostalih veb pretraživača. Čak i da nemate nalog veb pretraživača ili da niste ulogovani, pomoću korisničke sesije se mogu utvrditi Vaši prethodni upiti i na osnovu njih izvesti određeni zaključci. Ti zaključci mogu dovesti do restrikcije rezultata - uklanjanje nedovarajućih, kao i do izmena rangiranja - prvo se radi generičko rangiranje, a onda se personalizuje rangiranje prvih n rezultata. Odgovori na upit mogu zavisiti i od lokacije korisnika. *Google.rs* i *Google.fr* mogu dati drugačije rezultate za isti upit. Nekad se i na osnovu IP adrese utvrđuje odakle ste.

Statistika koja pokazuje lojalnost korisnika je prikazana na slici 6.3.

Za *relevantnost dokumenta*, odnosno za rangiranje rezultata kod veb pretraživača se koristi mnogo više parametara, ne samo *tf-idf*, kako bi se maksimalno razdvojili kvalitetni i nekvalitetni veb sadržaji. Veb sadržaji uključuju istine, laži, zastarele informacije, kontradiktornosti. Veb sadržaji su povezani velikom količinom linkova, u proseku postoji preko 8 linkova sa jedne strane. Velika količina veb sadržaja i linkova između njih dovodi do kompleksnog grafa veb strana. Iz ovih grafova se mogu crpeti informacije o kvalitetu veb sadržaja - *analiza linkova* je veoma značajna za veb pretraživače.

Veb sadrži veliku količinu *dupliciranog sadržaja*. Prema istraživanju sprovedenom još 1997. godine od strane Brodera i nje-



Slika 6.3: Lojalnost korisnika (preuzeto iz [48])

govih kolega [49], 35-40% veb sadržaja predstavljaju (približne) sintaksne duplike. Semantičkih duplikata se ne zna koliko ima, ali je sigurno značajan procenat. Na sve ovo treba dodati da postoje milijarde stranica koje predstavljaju spam. Potpuni duplikati se mogu lako detektovati ali njih nema toliko puno. Ali postoji mnogo, mnogo slučajeva približnih duplikata (eng. *near duplicates*) - 35-40% u kojima je na primer različit samo datum poslednje izmene, ili je različito zaglavje, na različitim forumima isto pitanje i isti odgovori...

Potpuni duplikati se mogu detektovati *fingerprinting* algoritma [50] koji veliki niz podataka pretvaraju u jedinstveni niz bitova koji ga identificuje, na primer hash funkcija. Može se povući analogija sa identifikacijom čoveka pomoću otiska prstiju. Da bismo detektovali približne duplike potrebna nam je mera sličnosti (eng. *edit distance*). Mora se odrediti i krajnja granica iznad koje se smatra da su dokumenti slični. Na primer, sličnost $>80\%$ se uzima za približni duplikat. Iako mera sličnosti nije tranzitivna u nekim situacijama je veb pretraživač koriste tranzitivno. Da bi se utvrdilo da li su dokumenti približni duplikati prvo se dokumenti izdele na delove ili na *shingles* - N-grame reči. Formiramo skup istih delova ili istih *shingles* dva dokumenta koji se porede - odnosno formiramo presek dva skupa. Formiramo i uniju delova ili *shingles* dva dokumenta koji se porede. Količnik

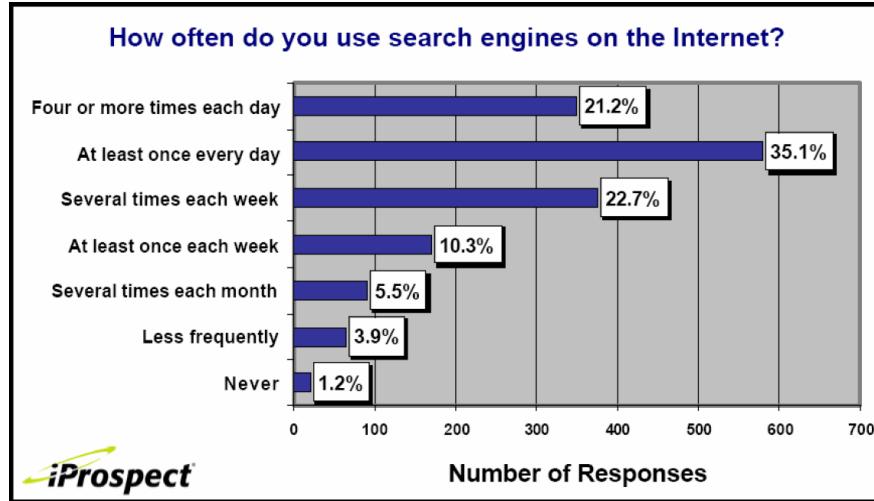
veličine preseka i unije je mera sličnosti. Ovo računanje je nekad skupo, pa se uzima samo uzorak delova ili *shingles* iz dokumenata - *sketch*. Postoje tehnike za odabir uzorka - uvek pitanje da li je uzorak dobar visi u vazduhu. Matrice i *Jaccard*-ov proizvod se takođe koriste za brže računanje mere sličnosti.

6.1 Veb pretraživači

U srpskom jeziku se često termin veb pretraživač koristi kao termin koji označava *web browser*, ali to nije slučaj u ovoj knjizi. Na dalje u ovom poglavlju će biti korišćen termin **veb pretraživač** koji označava *web search engine* - softverski sistem koji omogućava pretragu informacija na vebu kao što su Google, Yahoo!, Bing... Prvi veb pretraživači bazirani na ključnim rečima su se pojavili u periodu 1995-1997: *Altavista*, *Excite*, *Infoseek*, *Inktomi*, *Lycos*. U ovim prvim veb pretraživačima rangiranje određenog sajta na upit je zavisilo od toga koliko su ti sajtovi platili (eng. *paid search ranking*). Postojale su aukcije za ključne reči - *casino* je bila jako skupa ključna reč. 1998. godine pojavilo se link bazirano rangiranje uvedeno od strane *Google*-a. *Google* kompaniji je bilo najvažnije da su korisnici zadovoljni, da ih privuku na svoj pretraživač, a kasnije su naravno zaradili od njih i izgradili čitavu imperiju. Pobedili su konkurenčiju jer su korisnici bili zadovoljni. U prvi mah su se odrekli zarade koji su drugi pretraživači zarađivali jer su bili zasnovani na *paid search ranking*-u. Na primer *Goto* veb pretraživač je do 1998. godini imao godišnji prihod od preko milijardu dolara. Kasnije je ovaj pretraživač promenio naziv u *Overture.com* i 2003. godine *Yahoo!* ga je kupio za 1.63 milijardi dolara.

Naravno, *Google* se odrekao zarade samo u prvi mah, kasnije je dodao sekciju *Ads* koja je nezavisna od osnovnih rezultata pretrage (desna polovina ekrana) i da bi se neki veb sajt reklamirao tamo potrebno je platiti - slično je usvojio i *Yahoo!* i ostali veb pretraživači. *Google* danas ima najviše korisnika, samim tim verovatno i najviše zarade.

Koliko često korisnici koriste veb pretraživače se može videti na slici 6.4 koja je rezultat analize napravljene od strane *iProspect* organizacije (www.iprospect.com).



Slika 6.4: Korišćenje veb pretraživača (preuzeto iz [48])

Sredinom 2014. godine postoji više dostupnih veb pretraživača od kojih je najpopularniji *Google* putem kojeg se postavi blizu 70% svih upita za pretragu veba (slika 6.5).

comScore Explicit Core Search Share Report*			
February 2014 vs. January 2014			
Total U.S. – Home & Work Locations			
Source: comScore qSearch			
Core Search Entity	Explicit Core Search Share (%)		
	Jan-14	Feb-14	Point Change
Total Explicit Core Search	100.0%	100.0%	N/A
Google Sites	67.6%	67.5%	-0.1
Microsoft Sites	18.3%	18.4%	0.1
Yahoo Sites	10.4%	10.3%	-0.1
Ask Network	2.4%	2.4%	0.0
AOL, Inc.	1.3%	1.3%	0.0

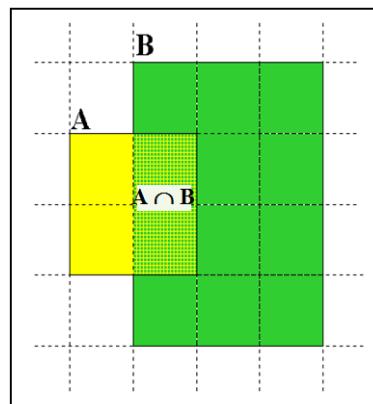
Slika 6.5: Popularnost veb pretraživača (preuzeto iz [48])

Jedan od parametara kvaliteta veb pretraživača je *veličina kolekcije* koja se indeksira i pretražuje. Pojam broja strana koje

indeksira neki veb pretraživač je relativno dobro definisan, ali postoje i neki problemi:

- pretraživači indeksiraju i stranice koje još nisu preuzeli putem *crawler-a - anchor text* (poglavlje 6.3),
- pretraživači uvode određene restrikcije - na primer, indeksirano je samo prvih n reči u stranici.

Različiti parametri mogu biti definisani za veb pretraživače - *max url depth, max count/host, anti-spam rules, priority rules...* Za isti URL se mogu indeksirati različiti sadržaji - *frames, meta-keywords, document restrictions, document extensions...* Postoje mehanizmi za relativno poređenje pokrivanja dva različita veb pretraživača (slika 6.6). Jedan takav mehanizam se



Slika 6.6: Poređenje pokrivenosti veb pretraživača
sastoji iz sledećih koraka:

1. odabratи uzorke - random *URL-ovi* iz pretraživača A,
2. proveriti da li tako odabrani *URL* postoji i u B,
3. uraditi i obrnuto,
4. odnos veličine kolekcije koju pokriva pretraživač A i veličine kolekcije koju pokriva pretraživač B je srazmeran odnosu broja slučajnih uzoraka iz B pronadjenih u A i broja slučajnih uzoraka iz A pronadjenih u B.

U ovom algoritmu ostaju otvorena dva pitanja - kako birati uzorke i kako vršiti provere. Odabir uzorka se može vršiti na sledeći način:

1. generisanje random upita - kreira se *Lexicon* koji je sačinjen od par stotina hiljada reči prikupljenih putem *crawler-a*, odabere se nekoliko termina iz leksikona i izvrši konjukcija [w1 AND w2](#),
2. odaberu se *URL-ovi* prvih 100 rezultata kada se ovaj upit izvrši na pretraživaču A,
3. odabere se slučajan predstavnik od ovih 100 koji će služiti kao uzorak da se proveri njegova prisutnost u pretraživaču B.

Postoje i drugi mehanizmi za odabir uzorka: *random search*, *random IP addresses*, *random walks*...

Drugo otvoreno pitanje je kako vršiti provere da li neki veb pretraživač indeksira određeni veb sadržaj? Provera da li pretraživač B indeksira dokument D se može uraditi na sledeći način:

1. preuzme se dokument D i uzme se njegova lista reči,
2. napravi se konjukcija između 8 reči male frekvencije i postavi taj upit pretraživaču B,
3. proveri se da li je među odgovorima i dokument D.

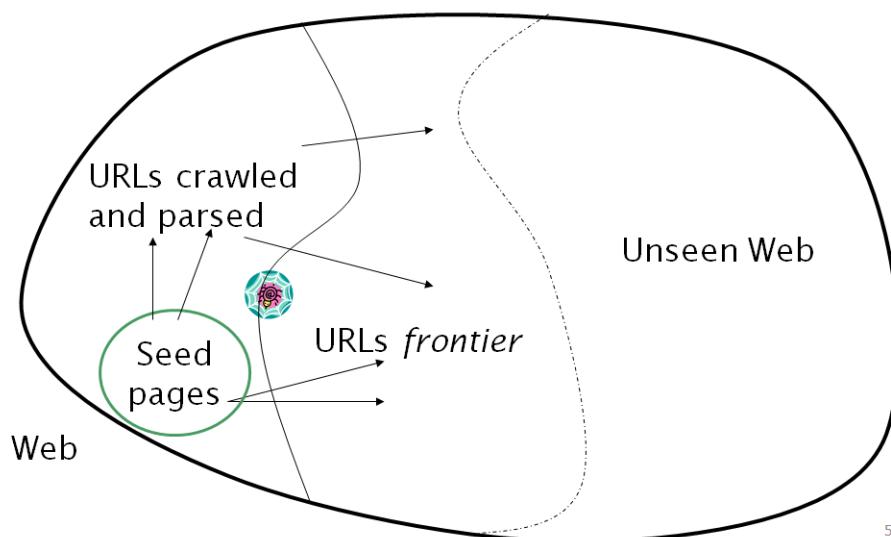
Problemi u prethodno opisanom algoritmu mogu biti približni duplikati (nešto što liči na dokument D) i redirekcije (nešto što redirektuje na dokument D). Takođe, ostaje i pitanje da li je 8 reči dovoljno, odnosno da li je lista odgovora prevelika da bi se proverilo da li je u toj listi i dokument D.

6.2 Veb *crawling*

Crawler (poznat i kao *Spider*, odnosno *Web robot*) je program za automatsko kretanje kroz graf veba i preuzimanje veb

stranica radi neke dalje obrade. Obično se ta dalja obrada odnosi na indeksiranje zarad pretraživanja. Kao što je već rečeno graf veba je ogroman što ovaj zadatak prilično komplikuje. Stranice na vebu se svakodnevno menjaju, što još više komplikuje zadatak. Takođe, zadatak komplikuje i postojanje dinamički generisanih stranica. *Crawler-i* često ignorisu dinamičke sadržaje, da ne bi upali u maliciozne zamke, ali u nekim aplikacijama su vesti dinamički sadržaji, pa se koristi *application-specific spidering*. Na taj način *statically indexable web* koji predstavlja sve što veb pretraživači indeksiraju uključuje i neke dinamički generisane HTML strane.

Crawler počinje sa poznatim *seed URL-ovima* (slika 6.7). Preuzme ih i parsira. Nakon toga ekstrahuje URL-ove iz ovih



Slika 6.7: *Crawling*

fajlova koje postavlja u listu. Za svaki URL iz liste ponavlja prethodno nabrojane zadatke.

Proces izgleda prilično jednostavno, ali zapravo postoje mnogobrojne komplikacije u ovom procesu. Gotovo je nemoguće raditi *crawling* veba sa jednim računarcem - sve operacije su distribuirane na više računara. Postoje maliciozne veb strane - spam

strane, *spider traps*, itd. I sa veb stranama koje nisu maliciozne postoje problemi - protok ka i kašnjenje odgovora sa udaljenih servera varira, koliko duboko u hijerarhiju sajta treba ići, *site mirrors* i približni duplikati, itd.

Postoje neka pravila u kreiranju *crawler-a* pre svega iz ugla šta jedan *crawler* mora da ispunи i šta bi bilo poželjno da ispunи. *Crawler* mora biti *učitiv*, poštovati implicitne i eksplisitne zahteve veb sajtova. Eksplisitni zahtevi veb sajtova su izraženi upotrebom *robots.txt* fajla. U ovom fajlu je specificirano u propisanoj formi šta može biti preuzeto *crawler-om*. Ovaj protokol za ograničenja pristupa *crawler-a* (ili robota) veb sajtu je nastao 1994. godine (www.robotstxt.org). Fajl *robots.txt* se stavlja u koren (eng. *root*) veb sajta i u njemu veb sajt navodi šta *crawler* sme, a šta ne sme da preuzme. Na primer, sledeći *robots.txt* zabranjuje bilo kom *crawler-u* da preuzme sadržaje čiji URL počinje sa */yoursite/temp/*.

```
User-agent: *
Disallow: /yoursite/temp/
```

Implicitno učitivost podrazumeva čak iako nema specifikacije, da *crawler* mora izbegavati preuzimanje puno strana sa jednog veb sajta u kratkom vremenskom roku, odnosno mora voditi računa da ne uputi previše HTTP zahteva jednom veb sajtu koji bi mogli da ugrose odziv tog sajta njegovim redovnim korisnicima.

Pored poštovanja učitivosti, *crawler* mora biti *robustan* - mora na odgovarajući način da reaguje na nenormalne situacije, maliciozne strane, klopke za *crawler-e*.

O čemu bi još bilo poželjno voditi računa prilikom kreiranja *crawler-a*?

Distribuiranost - gotovo je nemoguće raditi *crawling* veba sa jednim računarom, vrlo je poželjno da postoji mogućnost distribuiranog izvršavanja operacija, odnosno *crawler* treba da bude dizajniran da može da se izvršava na više računara.

Skalabilnost - poželjno je da se može dodavati još računara u sistem ako za to bude potrebe.

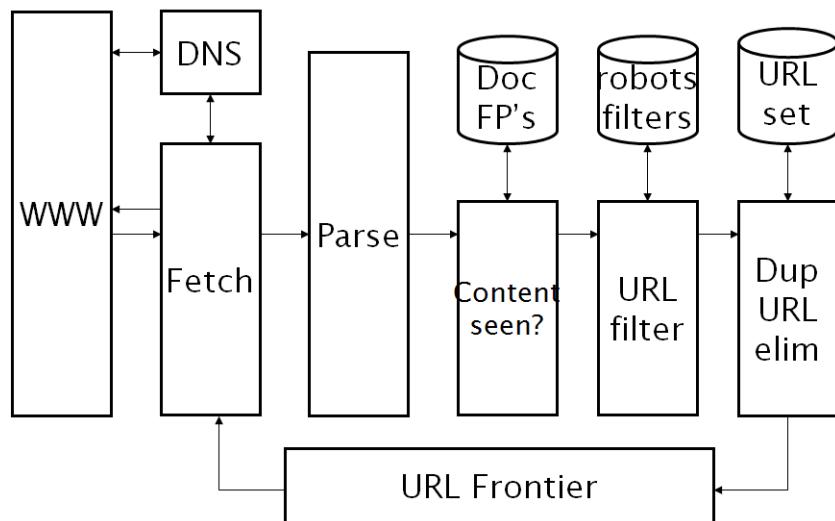
Performanse (efikasnost) maksimalne - poželjna je najbolja moguća iskorišćenost procesnih i mrežnih resursa.

Prioritetnije preuzimanje strana dobrog kvaliteta je još jedna poželjna karakteristika dobro *crawler-a*.

Iako postoje slučajevi kada nam je potrebno jednokratno preuzimanja kompletног veba ili jednog njegovog dela, *ponovno preuzimanje* novih sadržaja sa veb strana koje su u međuvremenu izmenjene je poželjna (i gotovo neophodna) osobina većine *crawler-a*. Ovo se posebno odnosi ako koristimo *crawler* za potrebe indeksiranja i pretraživanja veba.

Proširivost novim formatima i protokolima je takođe vrlo poželjna osobina *crawler-a*, jer se pojavljuju novi formati na vebu, a iako su u ovom momentu dominantni neki protokoli za preuzimanje sadržaja, nema garancija da će tako biti i u budućnosti.

Osnovna arhitektura jednog *crawler-a* je prikazana na slici 6.8.



Slika 6.8: Osnovna arhitektura *crawler-a* (preuzeto iz [38])

URL frontier sadrži URL-ove koje je potrebno preuzimati. Može uključiti više strana sa istog *host-a*, ali zbog učitivosti mora voditi računa da se izbegava njihovo preuzimanje u istom ili pri-

bližnom trenutku - definiše se minimalni *time gap* između dva zahteva istom *host*-u. Na neki način on diriguje proces i mora pokušati da u svakom momentu koristi sve resurse *crawler*-a.

Fetch komponenta preuzima naredni URL iz *URL frontier*-a. Koji je naredni URL je vrlo komplikovano pitanje i postoji nekoliko pristupa kako se to određuje. Nakon preuzimanja narednog URL-a, preuzima se veb sadržaj sa tog URL-a upotrebom *DNS* komponente. *Domain name server* za simboličko ime vraća IP adresu. Prilikom linkovanja strana najčešće se koriste simbolička imena. *Domain name server* je distribuiran sistem koji može imati spor odziv (nekoliko sekundi). DNS *lookup*-i su blokirajuće operacije. DNS komponenta implementira DNS keširanje - ako se jedanput traži IP adresa za neko simboličko ime, rezultat se kešira. Na taj način ako jedan veb sajt ima na stotine veb strana samo se jednom obraća *Domain name server*-u koji može imati spor odziv. Takođe, DNS komponenta obično implementira *batch DNS resolver* - prikupljanje zahteva i njihovo grupno slanje.

Nakon dobavljanja veb sadržaja ako je sadržaj tekstualni dokument vrši se parsiranje tog sadržaja upotrebom komponente *Parse*. Parsiranje se vrši da bi se utvrdilo da li je sadržaj duplikat i da bi se ekstrahovali URL-ovi ka drugim veb sadržajima. Kada se preuzeti dokument parsira neki od linkova su relativni i ovi linkovi se prebacuju u absolutne linkove.

Utvrdjivanje da li je sadržaj duplikat ili približni duplikat obavlja *Content seen?* komponenta. Duplikata i približnih duplikata ima puno na vebu. Ako je preuzeti veb sadržaj već indeksiran, dalje se ne procesira, a ako se ispostavi da nije duplikat, vrši se indeksiranje. Provera se vrši poznatim tehnikama za utvrđivanje duplikata: *fingerprinting algorithms* ili *shingles* za približne duplike. Koriste se podaci iz *Doc FP's* baze sa *fingerprint*-ovima već indeksiranih dokumenata.

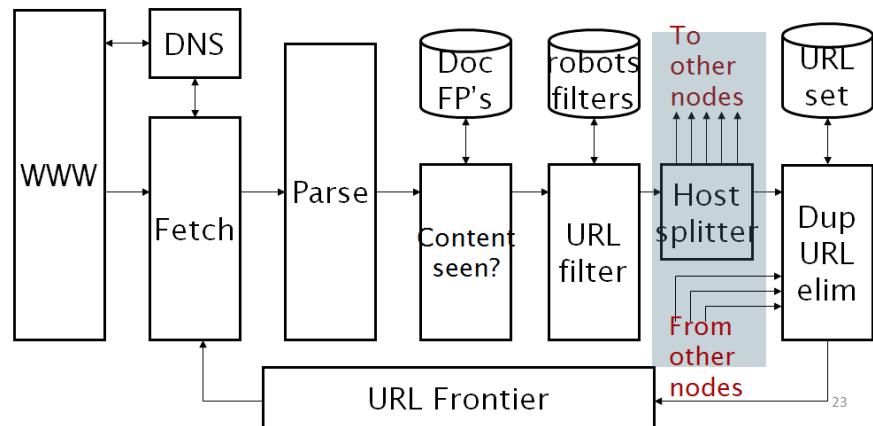
Komponenta *URL filter* proverava za svaki ekstrahovani URL da li prolazi sve filtere - filteri *crawler*-a i veb sajta definisani pomoću *robots.txt*. Filteri *crawler*-a su regularni izrazi kojima se definiše koje URL-ove *crawler* treba da preuzima radi indeksiranja - na primer ne pravimo pretraživač celog veba, želimo samo

domen *edu*. Pored filtera definisanih u samom *crawler*-u, mora se poštovati i *robots.txt* preuzet za određeni veb sajt. Tekstualni dokument *robots.txt* se preuzima samo jednom i kešira se u *robots filters* bazi. Postupak je ovakav, jer je ponovno preuzimanje *robots.txt* još jedan zahtev za veb server, a zbog implicitne učitivosti i zbog performansi *crawler* to se izbegava.

Dupl URL elim komponenta proverava za sve URL-ove koji su prošli filtere da li oni već postoje u *URL frontier*-u, odnosno da li ih treba staviti u listu u *URL frontier*. Za ovu namenu koristi *URL set* bazu. Kada *crawler* treba da samo jedanput preuzme sadržaje, onda je ova komponenta vrlo prosta. Ako je potrebno da periodično preuzima sadržaje kako bi imao njihove ažurne verzije, što je slučaj kod *crawler*-a za veb pretraživače, onda je to nešto složenije. U ovom slučaju, to što je URL već bio u *URL frontier*-u, ne znači da tamo više ne treba da se pojavi. Takođe, treba voditi računa o prioritetima (*freshness*) - potrebno je preuzimanje nekih strana češće od drugih (na primer sajtove sa dnevnim vestima koje se često menjaju). U ovom slučaju, *URL frontier* ima složeniju arhitekturu, jer je potrebno voditi računa i o prioriterima (*freshness*) i o učitivosti. Poznata je *Mercator URL frontier scheme* [51] koja se sastoji iz *front queues* koji vode računa o prioritetima (*freshness*) i *back queues* koji vode računa o učitivosti.

Kao što je već rečeno gotovo je nemoguće raditi *crawling* veba sa jednim računarom. Potrebno je distribuirati ovaj zadatak da bi se *crawler* što uspešnije izborio sa ogromnim grafom veba. Potrebno je pokrenuti više niti na različitim računarima koji mogu biti i fizički udaljeni. Kako komuniciraju ovi računari i ko je zadužen za koji URL? Jedna arhitektura koja ovo omogućuje je prikazana na slici 6.9.

U odnosu na prethodnu arhitekturu dodata je komponenta *Host splitter* čiji je zadatak da za URL koji je prošao sve filtre utvrdi koji je računar u distribuiranom sistemu zadužen za taj URL i da pozove njegovu *Dupl URL elim* komponentu. Baze podataka se ne dupliraju na svakom računaru koji se nalazi u distribuiranoj sistemu - baze podataka su centralizovane. Sve ostale



Slika 6.9: Osnovna arhitektura distribuiranog *crawler*-a (preuzeto iz [38])

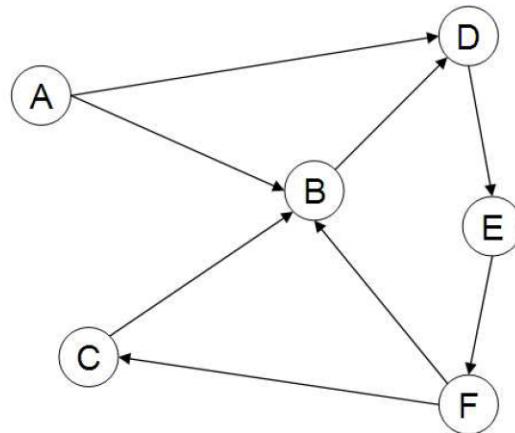
komponente su prisutne na svim računarima u distribuiranom sistemu.

6.3 Analiza linkova

Na vebu pored sadržaja dokumenata imamo i linkove kojima su dokumenti povezani. Moguće je formirati graf čiji su čvorovi dokumenti na vebu, a relacije su linkovi između njih (slika 6.10). Na ovaj način kreiramo **graf veba** i onda se može koristiti teorija grafova za neke zaključke.

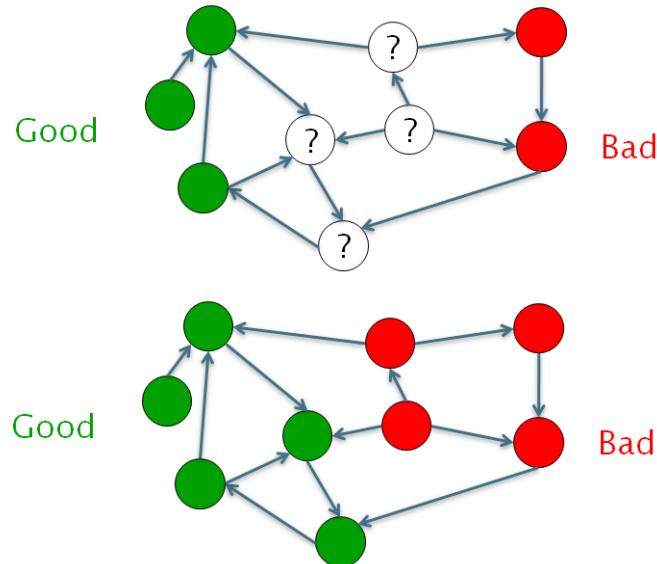
Podsećanje: korisnici veb pretraživača žele pouzdane odgovore. Da li količina linkova ka nekom dokumentu ukazuje na značaj i kvalitet tog dokumenta ili te veb strane? Da li se linkovi mogu uključiti u rangiranje rezultata? Koliko je verovatno da stranica na koju linkuje CERN veb stranica govori o fizici visoke energije?

U veb grafu postoje *dobri i loši čvorovi*, ali i čvorovi za koje se ne zna da li su dobri ili loši. Dobri čvorovi nemaju linkove ka lošim čvorovima, a sve ostale kombinacije su moguće. Ako čvor ima linkove ka lošim čvorovima i on je loš. Ako dobri čvorovi



Slika 6.10: Građevinski graf (veba) (preuzeto iz [38])

imaju link ka nekom čvoru i taj čvor je dobar (slika 6.11). Ovo



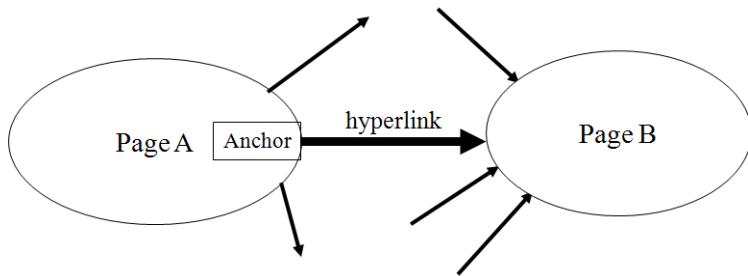
Slika 6.11: Dobri i loši čvorovi

je jedna ideja kako analizirati građevinski graf (vebu) i utvrditi da li je neki čvor dobar ili nije, ali u praksi je to mnogo složenije, i ima dosta čvorova za koje je teško utvrditi da li su dobri ili loši.

Link analize se primenjuje u raznim oblastima [52] sa različitim ciljevima:

- utvrđivanje kvaliteta veb strane, odnosno primena u rangiranju rezultata kod pretrage veba, klasterovanje veb strana, klasifikacija veb strana, *crawling*,
- društvene mreže - grupe ljudi sličnih interesovanja, pronađenje potencijalnih kupaca (onaj ko ima puno prijatelja koji troše mnogo, i sam troši mnogo),
- nauka - veze između radova.

Tekst linka(eng. *anchor text*) se može iskoristiti za kvalitetniji opis stranice na koju link upućuje (slika 6.12). Da bismo ovo

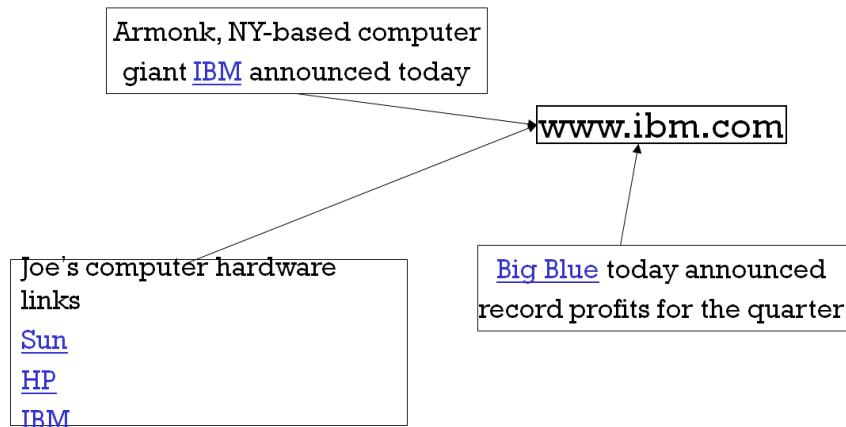


Slika 6.12: Tekst linka

zaključili polazimo od sledeće dve hipoteze.

- *Hipoteza 1.* Link ka nekoj strani doprinosi meri kvaliteta strane.
- *Hipoteza 2.* Tekst na stranici A nad kojim je definisan link (tekst linka) opisuje stranicu B.

Kada se indeksira veb sadržaj B, možemo uključiti sa određenom težinom tekstove linkova na stranicama koje upućuju na B. Na ovaj način, dokument B može biti indeksiran, može se i naći među odgovorima, čak i ako dokument B još uvek nije preuzet *crawler-om* (slika 6.13). Nekad je sam link na drugi dokument jasna fraza, a nekad je potrebno uzeti rečenicu ili deo rečenice kojoj



Slika 6.13: Indeksiranje pomoću teksta linka

link pripada i tako formirati tekst linka. Potrebno je dati težinu tom tekstu shodno kvalitetu veb stranice A sa koje link upućuje na stranicu B. Na primer, tekst linka sa www.cnn.com je dobar za indeksiranje stranice na koju se upućuje, jer je *www.cnn.com* dobar i pouzdan sadržaj. Nezavisno od kvaliteta veb sadržaja, tekstovi relativnih linkova imaju manju težinu nego tekstovi apsolutnih linkova. Upotreba tekstova linkova u indeksiranju i pretraživanju veb sadržaja može nekad imati neočekivane bočne efekte, jer nije ispunjena druga hipoteza - *evil empire* je tekst linka koji često upućuje na Microsoft sajt.

Tekstovi linkova mogu imati i druge primene. Na osnovu teksta linka mogu se davati težine veza u veb grafu. Takođe, opis strane se može kreirati na osnovu teksta linkova koji upućuju na tu stranu. Ako znamo klasifikaciju strane A koja upućuje na neku stranu B, i znamo tekst linka, nekad je to dovoljno za klasifikaciju strane B.

Konekcioni serveri skladište i omogućuju pretragu *inlinks* i *outlinks* URL-ova i mogu da odgovore na dve vrste upita vezanih za graf veba:

- Ko upućuje na dokument čije je URL u upitu?

- Na koga upućuje dokument čije je URL u upitu?

Konekcioni serveri su korisni za kontrolu i optimizaciju *crawling-a*, za analizu veb grafova i za link analizu koja se koristi za rangiranje rezultata. Gotovo svi današnji veb pretraživači imaju konekcione servere i dosta pažnje im posvećuju. Bitno je da rade brzo, daju informacije koje im trebaju u *crawling-u* i indeksiranju i ne zauzimaju previše memorije. Jedna kompresiona tehnika za graf veba koja omogućuje da konekcioni serveri ne troše previše memorije je opisana u radu [53].

Pre nego što razmotrimo kako da koristimo graf veba prilikom indeksiranja da bismo omogućili bolje rangiranje rezultata, razmotrimo prvo poreklo ove ideje. Naučni radovi koji citiraju iste radove su slične tematike. Analiza citata, odnosno graf naučnih radova, je iskorišćena da se utvrdi rejting rada ili časopisa. Da li sledeće tri vrste citata imaju istu težinu:

- jedan rad koji je objavljen u dobrom časopisu i ima dosta citata citira drugi rad,
- Jedan rad koji je objavljen u časopisu slabog rejtinga i nema citata citira drugi rad,
- Jedan rad citira rad koji ima iste autore - autocitat.

Veb i skup naučnih radova, odnosno graf veba i graf naučnih radova imaju značajnih razlika. Na vebu ima mnogo više učenika, različiti su interesi i ciljevi. Spam je daleko prisutniji na vebu - postoje i u nauci radovi koji predstavljaju spam i koji su nastali kao šala, ali ih je beznačajno malo [54]. Od kada su se počeli linkovi uzimati u obzir prilikom rangiranja rezultata u veb pretraživačima (1998), link spamovi rastu. Postoje i link farme - grupe veb sajtova koje se međusobno citiraju da bi dobili bolji rejting.

Zamislimo da se *web browser* šeta slučajnom putanjom kroz veb sajtove, odnosno kroz graf veba. Krene sa slučajno izabrane veb strane i dalje ide nekim od linkova koji se nalaze na toj strani, pri čemu je izbor linka slučajan i jednakoverojan - ako ima tri

linka onda je težina za odlazak na svaku stranu $1/3$. Želimo da u dugoj šetnji svaka veb strana ima težinu za posetu - *page score*. Problem je što veb graf nije jako povezan graf, nego je sačinjen od mnogo podgrafova koji su slabo povezani. Rešenje za ovaj problem je uvođenje pojma *teleportovanja*. Ako smo došli do mrtvog čvora, teleportujemo se u proizvoljni čvor pri čemu je težina za prelaz jednaka $\frac{1}{\text{ukupan broj čvorova}}$. Sada definišemo prelazak na novu veb stranicu na sledeći način: u bilo kom čvoru koji nije mrtav se može teleportovati u proizvoljni čvor pri čemu je težina za prelaz jednaka $\frac{\alpha}{\text{ukupan broj čvorova}}$, ili u čvor prema kojem postoji link pri čemu je težina za prelaz jednaka $\frac{1-\alpha}{\text{broj izlaznih linkova}}$. α je parametar u intervalu $[0,1]$ koji predstavlja verovatnoću da korisnik unese veb adresu direktno u polje za adresu. Ovaj parametar obično ima vrednost $0,1$. Na ovaj način više se kretanje ne može zaglaviti u podgrafu veba, odnosno zaista svaka veb strana ima težinu za posetu - **page score**.

Markovljev lanac se sastoji iz n stanja i matrice prelaza $n \times n$ koja sadrži verovatnoće prelaza u drugo stanje. Za svako $1 \leq i \leq n$ i $1 \leq j \leq n$, element matrice P_{ij} je verovatnoća da je j sledeće stanje ako se nalazimo u stanju i , pri čemu važi $\forall i, \sum_{j=1}^n P_{ij} = 1$. *Ergodic Markov chain* je Markovljev lanac takav da iz proizvoljnog stanja *možemo* doći u bilo koje stanje, tako da u dugotrajnoj slobodnoj šetnji možemo obići sva stanja. Probabilistički vektor $x = (x_1, \dots, x_n)$ sadrži verovatnoće da je šetnja u stanju i sa verovatnoćom x_i , pri čemu važi $\forall i, \sum_{i=1}^n x_i = 1$. Verovatnoće za sledeće stanje su $x \times P$, $x \times P^2 \dots$. Želimo da opišemo verovatnoću da smo u slobodnoj šetnji na nekoj veb strani nezavisno kada i odakle smo krenuli u slobodnu šetnju po grafu. Dakle, potreban nam je vektor nepromenjivih verovatnoća $a = (a_1, \dots, a_n)$, takvih da važi $a = a \times P$, $a = a \times P^2 \dots$. Računanje vektora a se svodi na računanje (levog) *eigenvector-a* matrice P . a_i je **pagerank** stranice i , odnosno njen **page score**.

Prethodno definisani *pagerank* stranice se može iskoristiti prilikom rangiranja rezultata. Na neki način ovo je mera kvaliteta stranice. Procedura za računanje ovog parametra je sledeća:

1. kreiramo matricu P upotrebom izlaznih linkova sa veb strana i unapred definisanog parametra α ,
2. izračunamo vektor a koji predstavlja (levi) *eigenvector* matrice P ,
3. a_i je *pagerank* stranice i .

Ovu procedura je nezavisna od upita i ne radimo je pre svakog upita. Procedura se izvršava periodično, jer se sadržaj veba menja, a samim tim se menja i graf veba i *pagerank* vrednosti stranica.

Nakon pronađenih veb sadržaja koji odgovaraju upitu možemo ih rangirati upotrebom *pagerank*-a. Problem je što *pagerank* ne zavisi od upita, a samim tim i rangiranje koje je zasnovano samo na *pagerank*-u ne zavisi od upita. Rešenje za ovaj problem je da se *pagerank* koristi samo kao jedan parametar u rangiranju. Dakle, ovaj parametar pokušava da razgraniči kvalitetne i nekvalitetne sadržaje, a ostali parametri kao što je *tf-idf* bi trebali da rangiraju rezultate shodno relevantnosti u odnosu na korisnikovu potrebu predstavljenju postavljenim upitom.

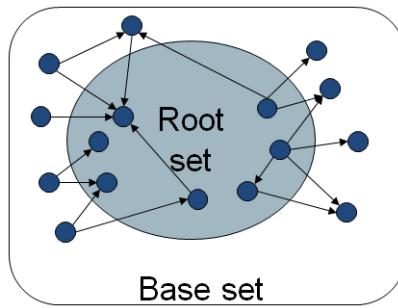
Pored *pagerank*-a za rangiranje rezultata pomoću informacija koje se crpe iz grafa veba koristi se i **HITS** (eng. *Hyperlink-Induced Topic Search*). Ideja je da u grafu veba uočimo dva skupa čvorova:

- **Hub** - čvorne tačke koje imaju puno linkova na veb strane iz odgovarajuće oblasti, na primer - “Bob’s list of cancer-related links”,
- **Authority** - značajan broj *Hub*-oba upućuje na ove strane koje se smatraju autoritetima u odgovarajućoj oblasti.

Na veb pretraživačima često imamo informacione potrebe koje nisu usko specificirane, nego se želi neko opšte mišljenje i informacije o nekoj oblasti. Dobri čvorovi određene teme imaju puno linkova ka autorativnim stranama. Dobre autorativne strane se spominju na puno čvorova. Ovo je cirkularna definicija autorativnih strana i čvorova. Zbog toga je neophodno iterativno

utvrđivanje autoratitivnih strana i čvorova. Kao prvi korak se odabere početni skup čvorova i autoriteta. Nakon toga se ovaj skup iterativno menja (obično se smanjuje).

Kako odabrati početni skup i kako vršiti iteracije? *Korenski skup* (eng. *Root set*) je sve što je odgovor na upit, na primer upit može biti *browser*. *Početni skup* (eng. *Base set*) je *Root set* + *in-links* + *out-links* (slika 6.14).



Slika 6.14: Početni skup

Inicijalna vrednost *hub score* pre prve iteracije za svako x je $h(x) = 1$, a *authority score* je $a(x) = 1$. Nakon jedne iteracije *hub score* je

$$h(x) = \sum_{x \rightarrow y} a(y) \quad (6.1)$$

Nakon jedne iteracije *authority score* je

$$a(x) = \sum_{y \rightarrow x} h(y) \quad (6.2)$$

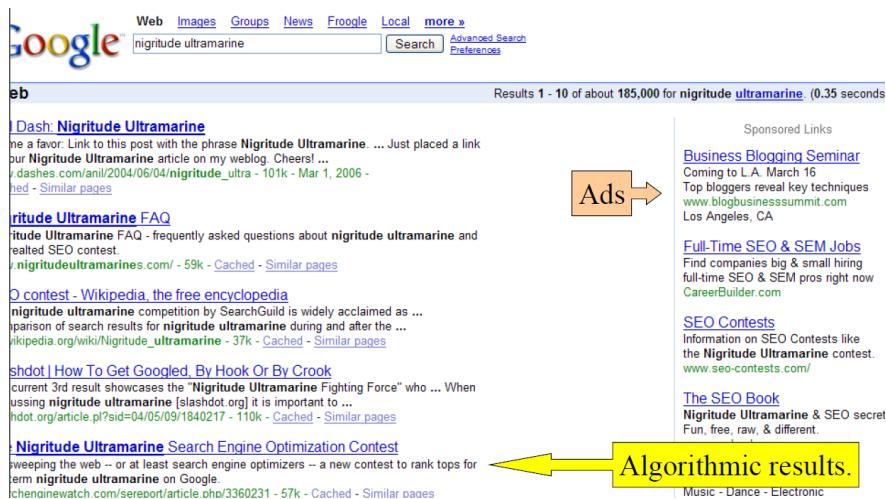
Nakon jedne iteracije uzmemosamo najboljih n čvorova i autoriteta na osnovu prethodno izračunatih vrednosti i počnemosmo narednu iteraciju, ali ne vraćamo vrednosti na 1. Eventualno vrednosti umanjimo k puta da ne bismo imali problem sa eksponencijalnim rastom vrednosti. U praksi 5 iteracija je dovoljno da čvorovi i autoriteti konvergiraju.

HITS za razliku od *pagerank-a* nije nezavisan od upita. Iteracije ne zavise od upita, samo osnovni skup. Ako se u osnovnom

skupu nađe stranica koja je imala malu relevantnost može uticati na autoritete i čvorove. Povezani sajтови sebi povećavaju *hub score* i *authority score*. Veb pretraživači pokušavaju da prepoznaju informacione potrebe koje nisu usko specificirane, nego se želi neko opšte mišljenje i informacije o nekoj oblasti, kako bi prilikom kreiranja odgovora na ove upite za rangiranje koristili *HITS*.

6.4 Search engine optimization

Među veb pretraživačima danas je dominantan pristup da je osnovna pretraga nezavisna od plaćanja, ali da postoji i prostor na ekranu za reklame gde je aukcija za ključne reči (slika 6.15). Ko da više novca povećava svoje šanse da bude dobro rangiran u



Slika 6.15: Razdvojenost rezultata i reklama (preuzeto iz [48])

ovom delu ekrana, iako količina datog novca nije jedini parametar za rangiranje u ovom delu ekrana. Ovo razdvajanje rezultata na rezultate koji su nastali nakon primene određenog algoritma i na rezultate koji predstavljaju plaćene reklame je prvi uvela *Google* kompanija na svom pretraživaču veba, a kasnije su ovaj koncept usvojili ostali pretraživači (*Yahoo!*, *Bing*). Ove dve liste

rezulata su međusobno nezavisne (slika 6.16). Prostor za rezul-



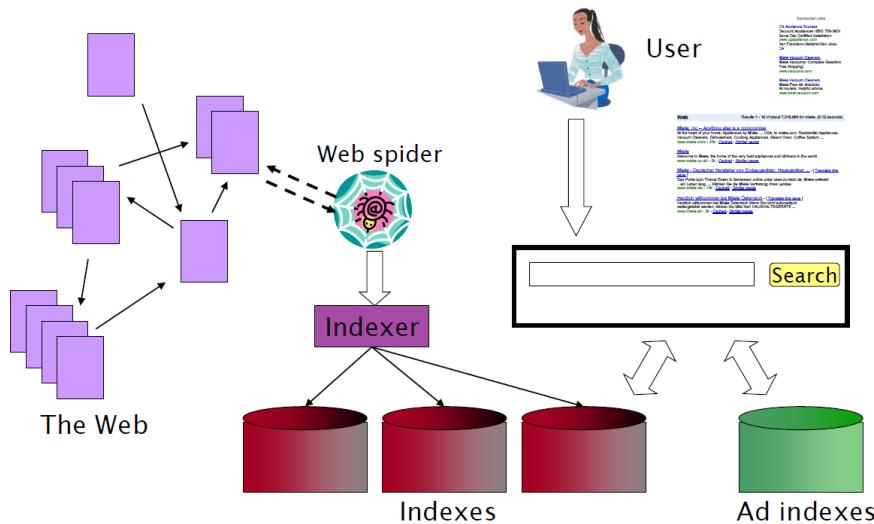
Slika 6.16: Nezavisnost rezultata pretrage i reklama (preuzeto iz [48])

tate pretrage koji predstavljaju reklame je glavni izvor prihoda veb pretraživača. Što više imaju korisnika, to veb pretraživači mogu bolje da naplate ovaj prostor. Indeksi koji sadrže osnovne veb sadržaje i indeksi koji sadrže reklame su razdvojeni kao što je to prikazano na slici 6.17).

Pošto je veb danas ogromno, globalno tržište, kompanijama i raznim organizacijama je jako stalo da su dobro rangirani na pretraživačima i to ne samo u prostoru na ekranu za reklame, nego i u osnovnim rezultatima pretrage. Zbog toga se neretko kompanije služe i nelegalnim sredstvima da bi došli do dobrog rejtinga. Na ovaj način pojavljuje se **spam** - veb stranice su dobro rangirani odgovor na upit korisnika, iako to po svom sadržaju ne bi trebale da budu. Spam stranice mogu biti čak širenje nekih verskih ili političkih ideja koje nemaju vezu sa korisnikovim upitom.

Koji je problem sa plaćenim mestima za *Ads*? - [cena](#).

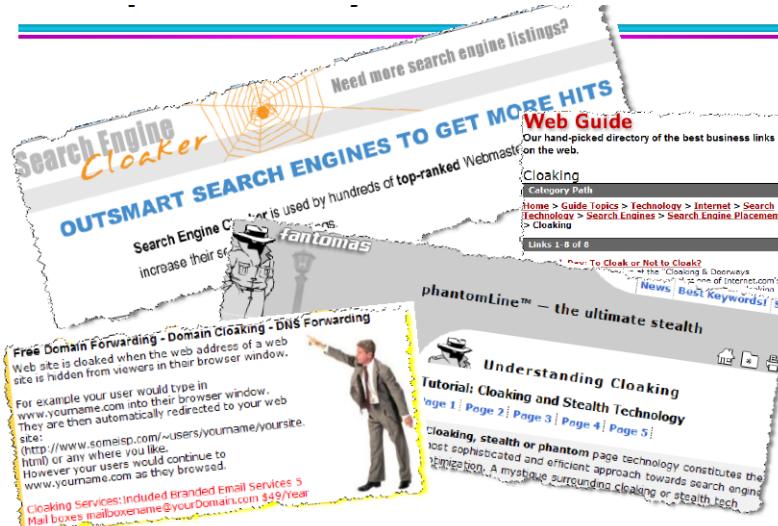
Koja je alternativa? - **Search Engine Optimization (SEO)**



Slika 6.17: Organizacija indeksa za reklame (preuzeto iz [48])

Osnovna ideja SEO tehnika je da se izmeni veb strana tako da ona bude dobro rangirana u algoritamskoj pretrazi veb pretraživača (levi deo ekrana) za odgovarajuće ključne reči u upitu. Na taj način se ne mora platiti veb pretraživačima ništa, a i korisnici više cene rezultate koji nisu reklama. Ali to ne znači da se ne mora ništa plati - mora se platiti SEO stručnjacima da urade ovu optimizaciju. SEO stručnjaci su ili zaposleni u marketing odeljenju jedne kompanije ili su iznajmljeni za potrebe optimizacije veb sajta kompanije. SEO-om se bave kompanije, webmasters, konsultanti (slika 6.18).

Postoje takmičenja iz oblasti tehnika korišćenih u SEO [55], a postoje i naučni skupovi na kojima se ove tehnike razmatraju [56]. Dakle, oblast SEO ne mora nužno uključivati korišćenje nelegalnih tehnika. Veb pretraživači obično imaju dokument u kojem navode šta je dozvoljeno [57]. Za dobromamerne to je polazna tačka šta da urade da povećaju vidljivost svoje stranice. Za *spammers* ovo je polazna tačka da pronađu "rupe u zakonu", opet sa istim ciljem, da povećaju vidljivost svoje stranice, a da ne moraju da plate, ili da šire svoje političke, religiozne ili druge ideje. Ako

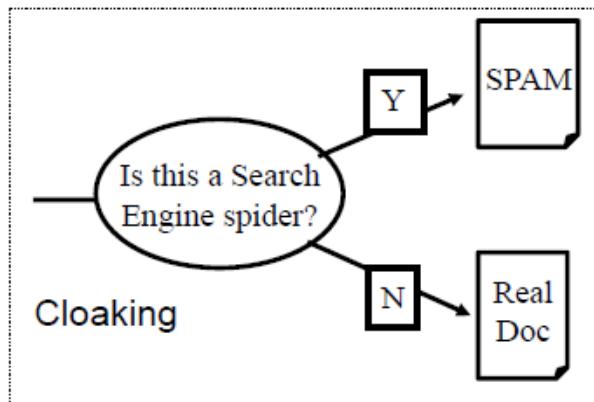


Slika 6.18: SEO industrija (preuzeto iz [48])

dobijete veb stranicu kao najrelevantniji odgovor na Vaš upit, a ona zapravo nije najbolji odgovor smatra se da je to spam.

Prve generacije veb pretraživača su se oslanjale na tf/idf u rangiranju rezultata. Top rangirane strane za upit "maui resort" su bile one koje su sadržale najviše reči "maui" i "resort". Nelegalne SEO tehnike su u html strane ponavljale kompletne sadržaje (ili bar bitne ključne reči) u boji pozadine. Tako da su ovi ponavljajući termovi bili preuzeti i indeksirani od strane veb pretraživača, samim tim i uticali na tf/idf. Sa druge strane ovi ponavljajući termovi nisu bili vidljivi za ljude koji otvaraju stranice i nisu smetali čitaocima. Pored stavljanja ponavljajućih termova u boji pozadine koristili su se i trikovi sa *cascade style sheet* (stavljanje u div tag koji nije prikazan). Namerno ponavljanje ključnih reči u veb stranici (tako da se one vide) je na primer legitimna SEO tehnika. Još jedna nelegalna tehnika je bila da se popularne reči koje su netačne i nevezane za sadržaj veb stranice stavljuju u meta-tags: *London hotels, hotel, holiday inn, hilton, discount, booking, reservation, sex, mp3, britney spears, viagra*. *Cloaking* je takođe jedna popularna nelegitimna SEO radnja koja pred-

stavlja poturanje specijalno pripremljenih sadržaja *crawler-ima* (slika 6.19).



Slika 6.19: *Cloaking* (preuzeto iz [38])

Takođe, *doorway pages* je još jedna nelegitimna tehnika koja predstavlja kreiranje stranica optimizovanih za jednu kљučnu reč od interesa i te stranice samo redirektuju do prave stranice.

Kada su veb pretraživači počeli da koriste i link analizu za rangiranje rezultata pretrage, tada su nastale i nove nelegitimne SEO tehnike koje uključuju: skrivene linkove, međusobno pomanjanje - link farme su grupe veb sajtova koje se međusobno citiraju, *Domain flooding* - gomila domena koji imaju link ili čak redirektuju na ciljanu stranicu. Takođe, veb pretraživači se upotreboom programskih robova napadaju kreiranjem miliona prijava putem *Add-Url*.

Jasno je da su veb pretraživači u ratu sa sajtvima koji koriste nelegitimne SEO tehnike. Veb pretraživači žele da imaju najbolje odgovore. Pokušavaju da kreiraju algoritme koji više gledaju linkove sa autorativnih strana, kao i da koriste glasove od autora i korisnika. Kreiraju se algoritmi koji sprovode link analizu da se proba detektovati spam. Mašinsko učenje se koristi u detekciji spama - obučavajući skup su poznate spam stranice. Pokušavaju se detektovati veb strane koje se međusobno podržavaju (*family friendly filters*). Formiraju se *blacklist-e*, kao i mehanizmi da ko-

risnici izraze žalbe na određene veb sadržaje. *Add-Url* obično ima implementirane anti robot testove.

Rezime

- Veb je danas ogromna, globalna baza znanja čija je pretraga neophodna za razvoj društva zasnovanog na znanju.
- Postoji nekoliko značajnih razlika u pretrazi veba i pretrazi ostalih kolekcija digitalnih dokumenata. Veličina veb sadržaja prevazilazi sve poznate kolekcije digitalnih dokumenata. Nema koordinacije u kreiranju sadržaja zbog čega imamo i dobrih i loših sadržaja. Veb sadržaji se često menjaju. Korisnici veb pretraživača pripadaju najširoj mogućoj lepezi korisnika. Sadržaji su povezani linkovima što se može koristiti prilikom rangiranja rezultata. Veb sadrži puno dupliranog sadržaja.
- *Google* je danas najčešće korišćeni pretraživač.
- *Crawler* je program za automatsko kretanje kroz graf veba i preuzimanje veb sadržaja radi neke dalje obrade. Mora biti učiv i robustan. Najčešće je distribuirane arhitekture.
- Količina linkova ka nekom veb sadržaju se može iskoristiti da bi se odredio kvalitet i pouzdanost nekog veb sadržaja. Tekstovi linkova se mogu koristiti prilikom indeksiranja. *pagerank* i *HITS* su dva pristupa u korišćenju linkova prilikom rangiranja rezultata.
- Osnovana ideja SEO tehnika je da se izmeni veb sadržaj tako da bude dobro rangiran u algoritamskoj pretrazi za odgovarajuće ključne reči u upitu. Oblast SEO ne uključuje nužno korišćenje nelegalnih tehnika.

Pitanja

1. Koja je razlika između pretraživača klasične kolekcije digitalnih dokumenata i pretraživača veba?
2. Koje su tri vrste potreba korisnika na koje veb pretraživači treba da odgovore?
3. Kako se može porebiti pokrivanje veba od strane dva različita veb pretraživača?
4. Šta je *crawler*, šta on mora, a šta bi bilo poželjno da ispunii?
5. Opisati arhitekturu *crawler-a*. Diskutovati i proširenja *crawler-a* distribuirane arhitekture.
6. Šta je graf veba?
7. Šta je tekst linka i za šta se on može iskoristiti?
8. Objasniti kako se *pagerank* može koristiti za rangiranje rezultata.
9. Objasniti kako se *HITS* može koristiti za rangiranje rezultata.
10. Objasniti sta su tehnike *Search Engine Optimization-a* i navesti nekoliko primera ovih tehnika.

Glava 7

Pretraga multimedijalnih dokumenata

Ovo poglavlje ima za cilj da definiše osnovne pojmove pretrage multimedijalnih dokumenata i da odgovori na sledeća pitanja:

- Koji sve mediji postoje za prenos informacija?
- Kako se mogu pretraživati kolekcije digitalnih dokumenata koji nisu tekstualni?
- Koje su karakteristike i problemi pretrage kolekcije slika, zvučnih i video zapisa?

Prilikom kreiranja strukture ovog poglavlja i definisanja osnovnih pojmoveva polazna osnova bila je disertacija "Proširivi sistem za pronalaženje multimedijalnih dokumenata" [44]. Takođe, prilikom kreiranja sekcije u kojoj se obrađuje pretraga zvučnih zapisa polazna osnova bila je magistarska teza "Adaptivni sistem za pretraživanje zvučnih zapisa" [58]. Deo informacija prezentovanih u ovom poglavlju su preuzeti iz raznih izvora otvorenog pristupa dostupnih putem Interneta, kao i iz

literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Informacije mogu biti prenete u obliku teksta, slike, veb stranice, videa ili zvuka. Svaki medij za prenos informacija ima svoje karakteristike, prednosti i slabosti. Kada kažemo medij za prenos informacija ne mislimo na medij za fizički prenos (CD, DVD, papir, stena sa uklesanim tekstom), nego mislimo na oblik u kome se informacije prenose - tekst, slika, video, animacija, zvuk. Odabir najprikladnijeg medija za prenos informacija za neku svrhu je težak zadatak. Mediji se mogu kombinovati u **multimediju** kako bi se što kvalitetnije prenele informacije konzumentima. Linkovani multimedijalni sadržaji dostupni na vebu se zovu **hipermedija**. Prema zavisnosti od vremena, mediji se dele na vremenski zavisne medije i vremenski nezavisne (statične) medije. *Vremenski zavisni mediji* su oni koji se menjaju kako vreme teče od početka konzumiranja medija: video, animacija i zvuk. *Statični mediji* se ne menjaju kroz vreme: tekst i slika. Prema načinu konzumiranja mediji se dele na linearne i nelinearne. Kod *linearnih medija* korisnik koji konzumira medij to uvek radi ustaljenim redosledom, dok kod *nelinearnih medija* korisnik bira put kojim konzumira medij - hipertekst sa linkovima je primer ovakvog medija.

Digitalni mediji i multimediji se mogu obrađivati i konzumirati upotrebom odgovarajućeg hardvera i softvera. Sve je više hardvera koji može da kreira multimedijalni sadržaj. Takođe, sve je više softvera koji može da kreira multimedijalni sadržaj. Zbog ovoga je i sve više multimedijalnih sadržaja. Takođe, sve je više kolekcija multimedijalnih sadržaja i na vebu i van njega. Zbog postojanja velike količine ovih sadržaja, teško je pronaći, odabrati, filterisati multimedijalne sadržaje. Potrebne su nam informacije o tome šta je u multimedijalnom sadržaju i potrebna nam je **pretraga kolekcije multimedijalnih sadržaja**.

Dakle, cilj je da napravimo multimedijalne sadržaje pretraživim kao što su tekstualni sadržaji. U današnjem svetu prepunom digitalnih podataka vrednost sadržaja ne zavisi samo od kvaliteta sadržaja nego i od toga koliko ga je lako pronaći. Potreban

nam je efikasan način opisa multimedijalnih sadržaja, kao i način kreiranja upita i pretraživanja ovih sadržaja.

Za implementaciju pretraga ovih vrsta kolekcija postoji nekoliko problema. Najčešće imamo samo sirovu reprezentaciju podataka bez metapodataka. Problem je i kako ustanoviti način kreiranja upita - kako opisati šta nam treba. Takođe, postoji dosta redudanse u ovim vrstama medija - na primer, iste slike u različitoj rezoluciji.

Kako izraziti upit nad kolekcijom multimedijalnih sadržaja? Prvi način je da se upit izrazi rečima za šta nam je potreban specifičan upitni jezik i oslanjamо se da mediji imaju kvalitetne metapodatke. Drugi način je da upit izrazimo uzorkom, odnosno da tražimo medije slične datom uzorku. U ovom pristupu se postavlja pitanje šta je kriterijum sličnosti uzorka i medija u kolekciji? Često je korisno da se mogu definisati neki dodatni uslovi: minimalna rezolucija tražene slike, godina objavlјivanja, potrebni su samo multimediji koji imaju i video i zvuk... Nekad se javljaju i složeniji uslovi - na primer, potrebne su slike koje imaju crvenu kuću. Naravno, potrebna je i mogućnost pregledanja kolekcije i mogućnost navigacije.

Kakav indeks koristiti za pretraživanje medija? Ako koristimo vektorski model, broj dimenzija kojima se reprezentuje jedan multimedijalni dokument zavisi od broja osobina kojima se želi predstaviti multimedijalni dokument. Osobine medija prema nivoima se mogu podeliti na:

1. osobine visokog nivoa (eng. *high-level features*) - sadržaj na slici, videu, prepoznavanje govora... ,
2. osobine srednjeg nivoa (eng. *medium-level features*) - detektor lica, klasifikacija regiona, žanr muzike... ,
3. osobine niskog nivoa (eng. *low-level features*) - Furijeova transformacija, *texture histograms*, *colour histograms*, *shape primitives*... .

Izračunavanje sličnosti vektora u visoko-dimenzionalnom prostoru može biti previše vremenski zahtevno ako se koristi sekvencijalna pretraga. Jedan način da se ovaj problem prevaziđe je da

se koristi neka od struktura za višedimenzionalne vektorske prostore - grid file, k-d tree, quad-tree, K-D-B tree, hB-tree, R-tree. Drugi način je da se problem pretrage multimedijalnih sadržaja svede na pretragu tekstualnih sadržaja - ovo nije uvek moguće. Treći način je da se koriste specijalizovani indeksi za pojedine osobine medija.

Postoje dva osnovna pristupa u pretrazi multimedijalnih sadržaja:

1. *text based* - pretraga multimedijalnih sadržaja bazirana na tekstu kojim je opisana,
2. *content based* - pretraga multimedijalnih sadržaja bazirana na sadržaju.

Za implementaciju **text based pretrage multimedijalnih sadržaja** neophodno je angažovati čoveka koji je zadužen za opisivanje sadržaja, odnosno za formiranje metapodataka. Opis predstavlja karakteristike visokog nivoa (eng. *high-level features*) i koriste se alfanumerički tipovi podataka: ključne reči, datumi, imena, koordinate. Za pretraživanje se koristi klasična baza podataka ili tehnike za pretraživanje teksta. Formiranje metapodataka je skupo i dugotrajno. Takođe, opis sadržaja zavisi od konteksta onoga ko opisuje što može da bude različito od konteksta onoga ko pretražuje. U ovom pristupu problem se svodi na prethodni - pretraživanje teksta.

Za implementaciju **content based pretrage multimedijalnih sadržaja** nije potrebno angažovati čoveka - koristi se računar. Opisivanje sadržaja se realizuje ekstrahovanjem osobina medija (slike, zvuka, videa). Opis predstavlja osobine niskog nivoa (eng. *low-level features*). Ove osobine se *izračunavaju* na osnovu prostog sadržaja medija - na primer na osnovu piksela slike. Za pretraživanje u ovom pristupu se koristi specijalizovani indeks. Ovaj sistem karakteriše brzo i jeftino kreiranje sistema indeksa, ali je manji kvalitet pretrage.

7.1 Slika

Slike se prikazuju kao nizovi piksela, a reprezentovani su korišćenjem internog modela: rasterskom grafikom (eng. *bitmaps*) ili vektorskom grafikom. Generisanje piksela od modela zove se renderovanje. **Rasterska grafika** reprezentuje sliku kao niz logičkih piksela (odnosno kao niz skladištenih vrednosti boja) koji mogu biti direktno mapirani na fizičke piksele na ekranu. Rasterska grafika pruža velike mogućnosti, zauzima dosta prostora, pati od problema u prepoznavanju oblika i kompleksnosti prilikom transformacije slike. U **vektorskoj grafici** slike su skladištene kao matematički opis kolekcije linija, krivi i oblika (često u formi nekog XML-a). Kod vektorske grafike renderovanje slike je složenije, ali su prepoznavanje oblika i transformacija slike mnogo jednostavniji.

Pretraživanje kolekcije slika je *vrlo aktuelna tema*, jer imamo puno uređaja i programa za kreiranje slika, a samim tim imamo i velike kolekcije slika (najčešće rasterskih slika). *Google* ima posebnu veb stranicu za pretragu slika koja omogućuje i *text based* i *content based* pretragu slika. U prilog aktuelnosti pretrage kolekcija slika govori i postojanje android aplikacije *Novi Sad Priča* (eng. *Novi Sad Talking*). Ova aplikacija omogućuje da se na osnovu slike koja se vidi kamerom na mobilnom telefonu i na osnovu lokacije mobilnog telefona pretraži kolekcija slika o Novom Sadu i pronađu dodatni podaci o zgradama ili spomeniku koji se vidi pomoću kamere mobilnog telefona [59].

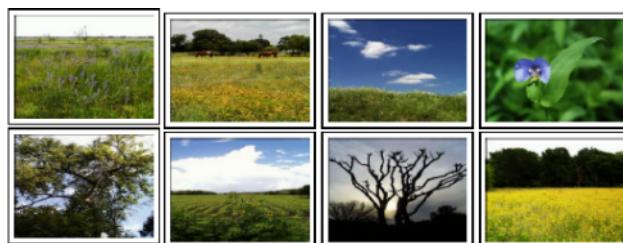
Pretraživanje kolekcija slika je *komplikовано*. Šta je tema slike 7.1)? Kojim ključnim rečima bismo je mogli opisati? Koje reči bismo koristili u pretrazi?

Kod pretrage kolekcije slika *nije lako rangirati rezultate pretrage*: "Potrebna je slika na kojoj je letnji pejzaž" (slika 7.2).

Problem kod pretrage slika je *semantički jaz* između različitih tumačenja slika. Jedna slika vredi 1000 reči i može se interpretirati na različite načine. Značenje (interpretacija sadržaja) slike je vrlo individualno i subjektivno. Po čemu su slične, a po čemu različite slike prikazane na slici 7.3?



Slika 7.1: Brodovi



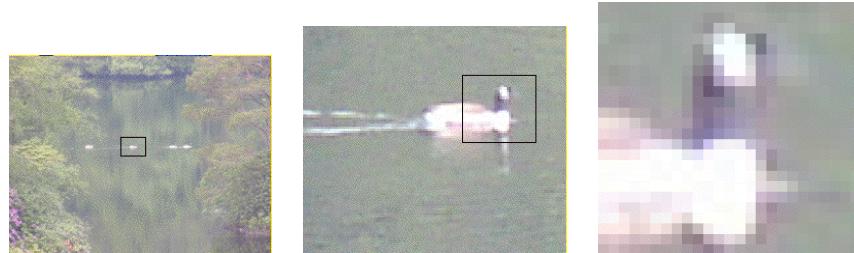
Slika 7.2: Pejzaži



Slika 7.3: Jabuke

Koliko god da su napredovale IKT, *rezolucija slike je konačna* i to može prestavljati problem za tumačenje slike od strane računara. Na slici 7.4 mi vidimo patke, a šta vidi računar?

Postoji dva načina da izrazimo upit nad kolekcijom slika. Prvi način je da *upit izrazimo rečima* pri čemu se oslanjamamo na metapodatke o slikama. Problem je kako formirati kolekciju slika



Slika 7.4: Patke

koja ima kvalitetne metapodatke. Drugi način je da *upit izrazimo uzorkom*, odnosno da tražimo slike slične datom uzorku. Ovde je problem kako definisati kriterijum sličnosti uzorka i slike u kolekciji. Na primer, upit je dat uzorkom prikazanom na slici 7.5. Ako u kriterijumu sličnosti figurira boja, lista odgovora na ovaj



Slika 7.5: Upit

upit bi mogla biti prikazan na slici 7.6.

Postoji dva načina da se izvuku **informacije o sadržaju slike**.

Prvi način uključuje angažovanje *čoveka* koji treba da opiše sadržaj formiranjem metapodataka. U ovom slučaju opis sadržaja predstavlja karakteristike visokog nivoa (eng. *high-level features*) i koristi alfanumeričke tipove podataka: ključne reči, datume, imena, koordinate. Ako na ovaj način opišemo sadržaje slika za pretraživanje se može koristi klasična baza podataka ili klasične tehnike za pretraživanje teksta. Problem je što u ovom pristupu opisivanje sadržaja slika skupo i dugotrajno. Opis sadržaja zavisi od konteksta onoga ko opisuje, što se može značajno razlikovati od konteksta onoga ko pretražuje.

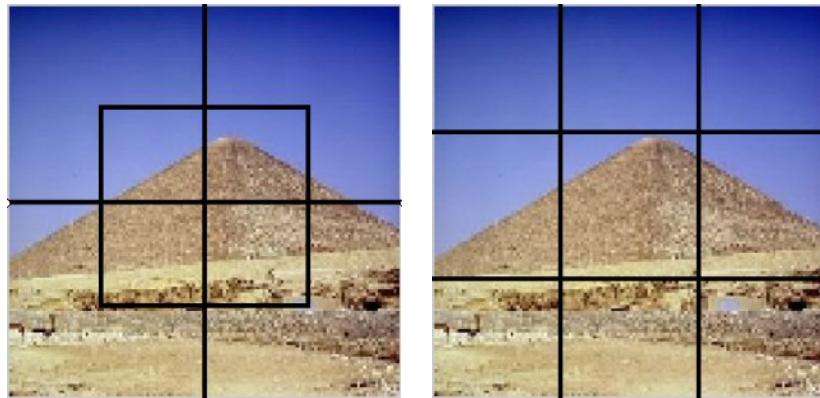
Drugi način isključuje angažovanje čoveka i potrebno je samo angažovati *računar*. Opisivanje sadržaja slike se svodi na ekstra-



Slika 7.6: Lista odgovora

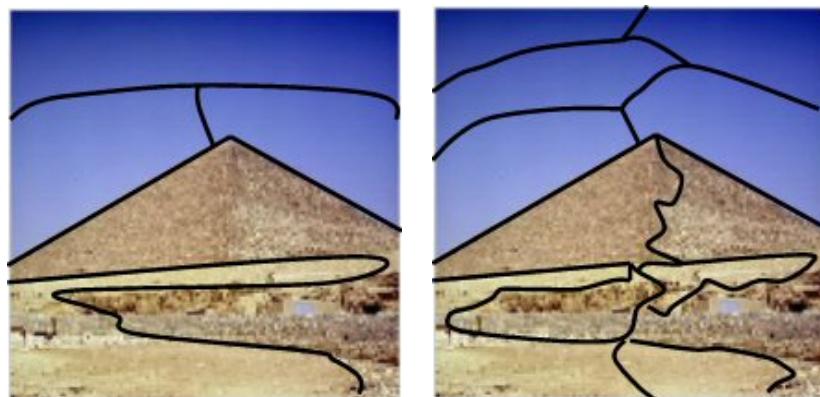
hovanje osobina slika niskog nivoa (eng. *low-level features*). Najčešće osobine niskog nivoa koji se ekstrahuju za potrebe pretraživanja su boja, oblik, tekstura i međusobni položaj objekata. Ove osobine se *izračunavaju* na osnovu prostog sadržaja slike - na osnovu piksela. Ovaj pristup omogućuje brže i jeftinije opisivanje sadržaja slika u kolekcijama, ali je kvalitet pretrage manji nego u prvom pristupu. Osobine niskog nivoa se mogu računati za celu sliku - **globalne osobine**, ili po delovima slike - **lokalne osobine**. Ekstrahovanje globalnih osobina je računski jednostavnije, ali ne odslikava čovekovo poimanje slike. Postoje uspešne implementacije ekstrahovanja globalnih osobina. Vrednosti globalnih osobina predstavljaju prosek za celu sliku. Gubi se razlikovanje prednjeg plana i pozadine, sve se isto posmatra. Ako želimo da ekstrahujemo lokalne osobine potrebno je da prvo uradimo **segmentaciju slike** i za tako dobijene delove izvršimo ekstrahovanje osobina. Postoji dva načina da se uradi segmentacija slike. *Tile-based* segmentacija je podela slike na pravilan raspored delova. Implementacija ove segmentacije je računski jednostavna, ali nakon ovakve segmentacije ekstrahovanje lokalnih osobina pati od sličnih problema kao ekstrahovanje globalnih osobina: ne odslikava čovekovo poimanje slike, gubi se razlikovanje prednjeg plana

i pozadine. Ovakva vrsta segmentacije se može raditi po različitim šemama (slika 7.7).



Slika 7.7: *Tile-based* segmentacija

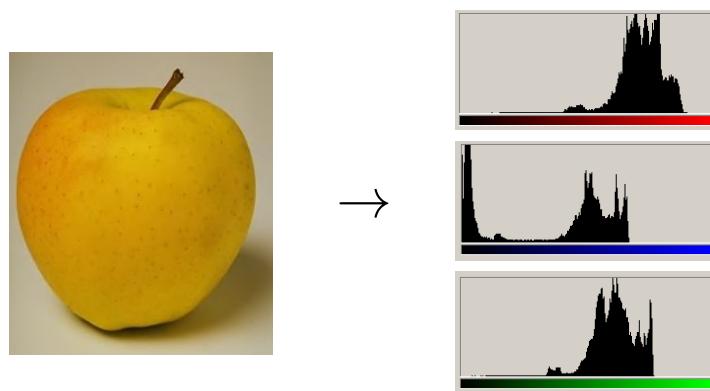
Region-based segmentacija je podela slike po regionima na osnovu sadržaja, odnosno podela slike na vizuelno koherentne zone. Može da identificuje značajne objekte, ali je ovakva segmentacija računski složena i nepouzdana (slika 7.8).



Slika 7.8: *Region-based* segmentacija

Bilo da vršimo segmentaciju ili ne, potrebno nam je ekstrahovanje osobina niskog nivoa. Jedna od osobina slike je **boja**. Može

se izračunati *signatura* regiona/slike na osnovu boje piksela. Jedan od načina da koristimo boju kao osobinu je da kreiramo *histogram boja*. Prvo je potrebno prebrojati koliko ima piksela za svaku boju u regionu/slici, nakon čega se od dobijenih vrednosti sastavlja histogram. Za slike u boji to se ponavlja za svaki kanal zasebno, na primer R, G i B ako se koristi RGB model boja (slika 7.9).



Slika 7.9: Histogram boja

Ovakav histogram boja je proizveo 3 niza celobrojnih vrednosti. Obično se radi normalizacija ovih brojeva u nizovima da histogram ne bi zavisio od veličine slike. Nakon normalizacije se ova tri niza spoje u jedan niz koji predstavlja signaturu regiona/slike. Histogram boja je prilično neosetljiv na male promene. Da bi se koristio histogram boja u pretrazi kolekcije slike, sve slike moraju imati isti model boja. Pretraživanje se svodi na poređenje signatura u vektorskom prostoru.

Drugi način da koristimo boju kao osobinu prilikom pretrage kolekcije slika je upotreba *vektora koherencije boja*. Da bi se kreirali ovi vektori prvo je neophodno piksele klasifikovati u grupe po boji: pikseli sa vrednošću u opsegu $[0, 10] \rightarrow$ klasa 1, $[11, 20] \rightarrow$ klasa 2... Nakon ovoga moguće je uočiti regione obojene istom klasom - pikseli povezani 8-susedstvom. Pikseli se označavaju kao koheretni ili nekoherentni, pri čemu je koherentan piksel deo regije piksela iste klase, a nekoherentan onaj koji to nije. Na kraju

se formira binarni vektor po horizontali ili vertikali koji predstavlja vektor koherencije boja, odnosno signaturu regiona/slike. Primer određivanja vektora koherencije jedne slike je dat u nastavku.

$$\begin{bmatrix} 22 & 10 & 21 & 22 & 15 & 16 \\ 24 & 21 & 13 & 20 & 14 & 17 \\ 23 & 17 & 38 & 23 & 17 & 16 \\ 25 & 25 & 22 & 14 & 15 & 21 \\ 27 & 22 & 12 & 11 & 21 & 20 \\ 24 & 21 & 10 & 12 & 22 & 23 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 & 1 & 1 \\ 2 & 1 & 3 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 & 1 & 1 \\ 2 & 1 & 3 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 2 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} BCBBAA \\ BBCBAA \\ BCDBAA \\ BBBAAE \\ BBAEAE \\ BBAEAE \end{bmatrix}$$

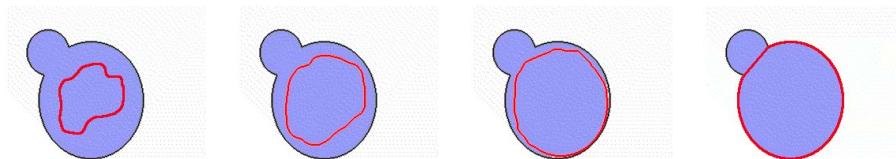
$$\begin{bmatrix} BCBBAA \\ BBCBAA \\ BCDBAA \\ BBBAAE \\ BBAEAE \\ BBAEAE \end{bmatrix} \rightarrow \begin{bmatrix} BCBBAAABCBAAABCDBAA \\ BBAAEBBAAEEBBAAEE \end{bmatrix}$$

Boja nije jedina osobina slike i nije uvek dovoljna za pretraživanje kolekcije slika. Na primer, ako bismo koristili samo boju kao osobinu slike onda bi slike žuto-zelane jabuke i žuto-zelene kruške bile slične (slika 7.10).

Druga bitna osobina je **oblik**, odnosno prepoznavanje oblika objekata na slici. Već je ranije konstatovano da je rasterska grafika danas mnogo više prisutna od vektorske grafike, a i da ova grafika za razliku od vektorske grafike pati od mogućnosti prepoznavanja oblika. Ipak, postoje neke tehnike za prepoznavanje oblika na rasterskoj grafici. *Aktivne konture* se koriste da se pronađu oblici objekata na slici [60]. Aktivne konture generišu krive koje se kreću unutar slike i traže granicu objekta (slika 7.11). Zavise od početne pozicije i mogu da završe u lokalnom minimumu.



Slika 7.10: Boja nije dovoljna



Slika 7.11: Aktivne konture

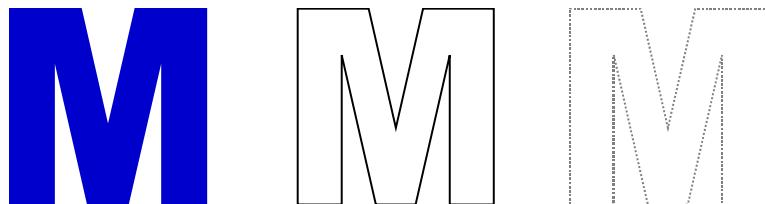
Gradient Vector Flow su usavršene aktivne konture (slika 7.12) koje u nekim situacijama rade bolje od aktivnih kontura, odnosno teže završavaju u lokalnom minimumu [61].



Slika 7.12: *Gradient Vector Flow*

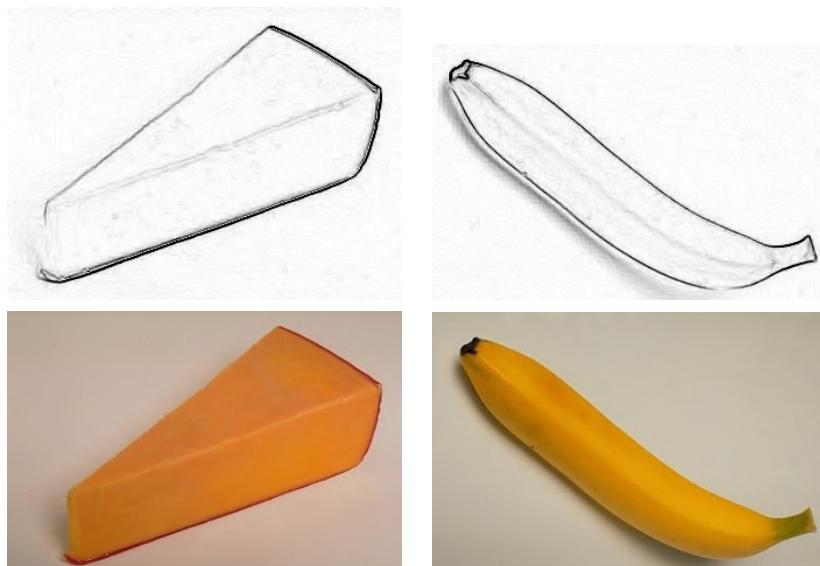
Furijeovi deskriptori se takođe koriste za prepoznavanje oblika na slici [62]. Ova tehnika radi tako što se izračunavaju varijante

Furijeove transformacije za ivice objekta. Ova tehnika za prepoznavanje oblika je otporna na geometrijske transformacije i šum (slika 7.13).



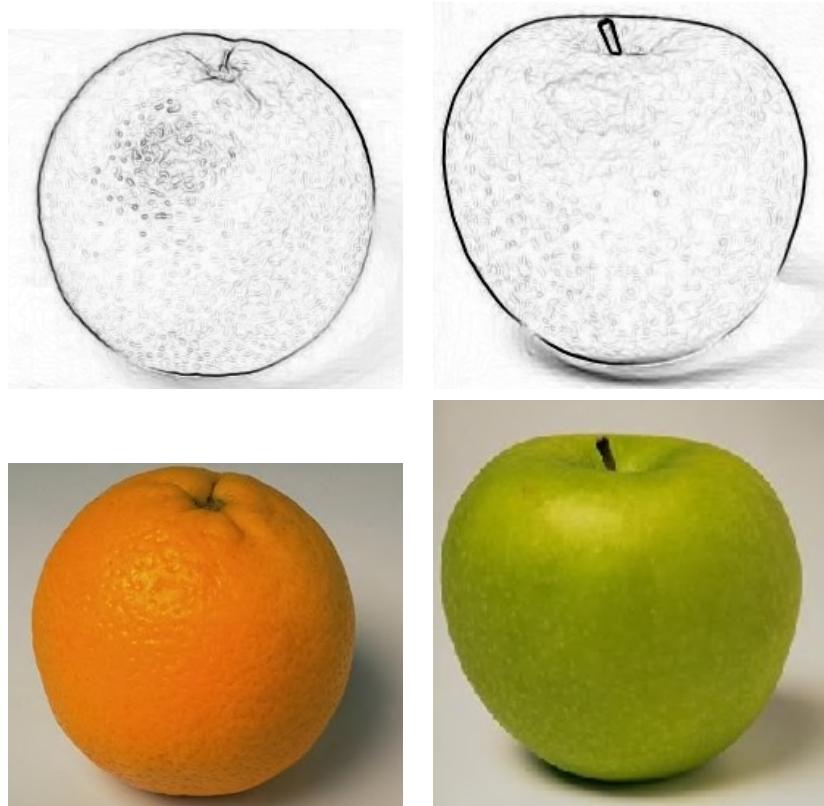
Slika 7.13: Furijeovi deskriptori

Neke slike se razlikuju po obliku objekata na slici, a ne razlikuju se mnogo po boji (slika 7.14).



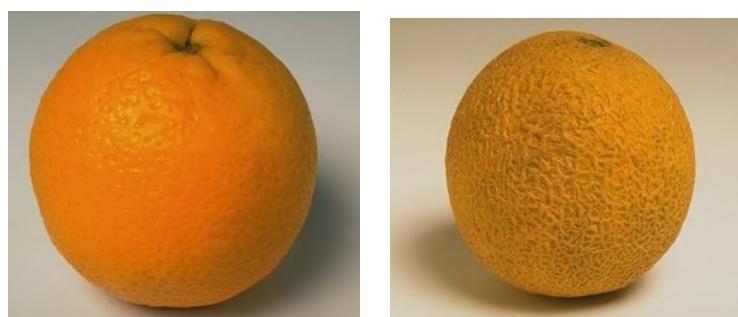
Slika 7.14: Različiti oblici, slične boje

Sa druge strane neke slike je teško razlikovati po obliku, a lako ih je razlikovati po boji (slika 7.15).



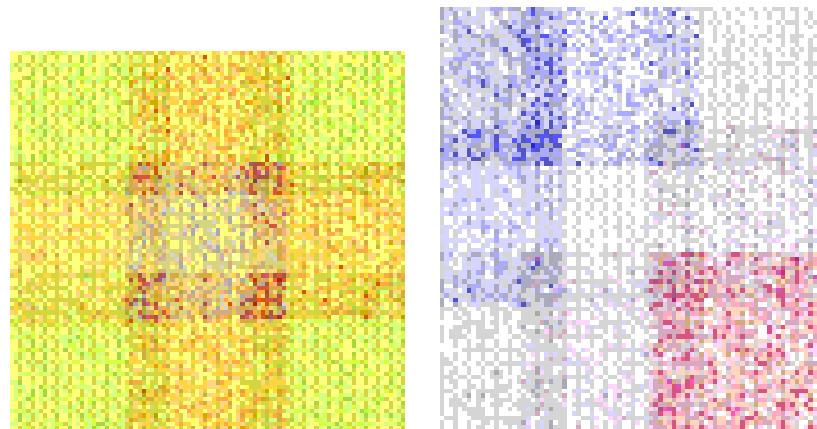
Slika 7.15: Slični oblici, različite boje

Zato je najbolje koristiti obe ove osobine - boju i oblik. U određenim situacijama ni ovo nije dovoljno (slika 7.16).



Slika 7.16: Slični oblici, slične boje, različite teksture

Nekad je neophodno uključiti i prepoznavanje **teksture**. Ovo je još jedna osobina koja se često koristi u pretrazi kolekcije slika. Kao što slike mogu biti slične po oblicima i bojama, a različite po teksturi objekata, isto tako slike mogu biti slične po teksturi, iako su različite po nekim drugim osobinama kao što je boja (slika 7.17).



Slika 7.17: Slične teksture, različite boje

Tekstura je matematički opis ponavljajućeg šablonu u slici: glatko, peskovito, zrnasto, trakasto... *Matrica pojavljivanja* može služiti za matematički opis tekture. Ova matrica pojavljivanja P se kreira na sledeći način: P_{ij} je broj pojavljivanja piksela boje i u smeru p u odnosu na piksel boje j . Na primer, ako slika ima tri boje (0, 1 i 2) i smer je “dole desno”, matrica pojavljivanja se kreira na sledeći način:

$$slika = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \Rightarrow P_{ij} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

Pored implementacije zasebnih sistema za pretragu kolekcija slika, postoje i gotove baze podataka koje podržavaju skladištenje i pretraživanje rasterskih slika. ISO/IEC 13249-5:2001 -

“SQL/MM Still Image standard” definiše standardne objektno-relacione tipove podataka za pretraživanje slika po sadržaju slike: *SI_S stillImage*, *SI_AverageColor*, *SI_Color*, *SI_ColorHistogram*, *SI_FeatureList*, *SI_PositionalColor*, *SI_Texture*. *Oracle Multimedia* je proširenje Oracle relacione baze podataka za rad sa slikama, zvukom i videom. U ranijim verzijama Oracle relacione baze ovo proširanje se zvalo *Oracle interMedia*. Slike se smeštaju u BLOB kolone baze podataka i podržano je više formata. Ekstrakcija opisa sadržaja slike je zavisna od formata. Ovo proširenje nudi i konverziju formata, a ono što je najbitnije nudi mogućnost pretraživanja slika po sadržaju (ne samo po metapodacima). Indeksiranje slike obuhvata: segmentaciju slike, analizu boje, analizu oblika, analizu teksture. Rezultat ovih koraka je vektor sa 3.000-4.000 elemenata koji predstavlja signaturu slike. Upit se izražava slikom-uzorkom, a lista odgovora se dobija poređenjem signatura slika i uzorka u vektorskom prostoru.

LIRE (eng. *Lucene Image REtrieval*) je biblioteka otvorenog koda koja je nastala kao proširenje Lucene biblioteke za potrebe pretraživanja slika na osnovu metapodataka i na osnovu sadržaja [63]. API biblioteke je krajnje jednostavan. Koristi mehanizme za pretraživanje teksta: invertovani indeks i tf-idf težine. Prilagođava reprezentaciju osobina slike tako da se mogu smestiti u rečnik Lucene indeksa. Esktrahuje nekoliko osobina niskog nivoa koji se odnose na boju: *ScalableColor*, *ColorLayout* i *EdgeHistogram*, *Auto Color Correlogram*...

7.2 Zvuk

Zvuk se proizvodi konverzijom energije u talase koji se prostiru kroz neki etar. Talasi dolaze do uha gde se konvertuju u nervni impuls što mozak detektuje kao zvuk. Ljudsko uho može detektovati zvuk frekvencije između 20Hz i 20kHz što zavisi i od godina starosti. Zvuk je složen i subjektivan fenomen. Percepcija zvuka ima i psihološku dimenziju - isti zvuk prvo tiše pa glasnije se čini mnogo glasniji nego da je odmah pušten tim intezitetom, Vaše ime na kraju glasne sale ćete lakše prepoznati nego bilo

koju drugu reč, Stereo zvuk proizvodi utisak da je nešto desno, odnosno levo, itd. Teško je napraviti precizan model zvuka. Postoji više različitih reprezentacija zvuka. *Waveform* (predstava talasima) zvuka prikazuje promenu amplitude zvuka kroz vreme. *MIDI* je format za skladištenje instrukcija kako će se zvuk proizvesti - nešto kao vektorska grafika kod slike.

Postoje različite vrste zvuka:

- baziran na govoru - radio program, telefonska komunikacija, snimljeni razgovori,
- baziran na muzici - instrumentalna, vokalna (pevanje),
- ostalo - alarm, zvuci iz prirode.

Pretraga kolekcije zvučnih zapisa je vrlo korisna *na kolekcijama zvučnih zapisa baziranih na govoru* i u ovakvim kolekcijama se najčešće i implementira - pretraga radio vesti, snimljenih razgovora, predavanja, itd. Prilikom implementacije ove pretrage postoji više poteškoća. Iste reči izgovorene od iste osobe pod istim uslovima mogu proizvesti prilično različite talase. Kolike su tek onda razlike kada su u pitanju različiti ljudi? Razlika u tome koliko je glasno nešto izgovoren, različiti akcenti i harmonici takođe predstavljaju ozbiljne izazove u implementaciji pretrage. Već smo ranije rekli da je zvuk vrlo složen fenomen, da ne čujemo svi isto i da postoji subjektivnost u slušanju - poznate termine bolje čujemo od drugih, na primer naše ime na kraju bučne sale.

Kao i kod pretrage kolekcije rasterskih slika, tako je i kod pretrage kolekcije zvučnih zapisa potrebno detektovati neke osobine u ovim zapisima. Naravno, te osobine su drugačije nego kod slike. Kod zvukova baziranih na glasovima osobine visokog nivoa se mogu podeliti u sledeće grupe:

- sadržaj - izgovoreni fonemi, *one-best word recognition, n-best*,
- identitet - identifikacija govornika, podela zvuka po različitim govornicima,
- jezičke osobine - jezik, dijalekt, akcenat,

- ostale osobine - okruženje, kanal (*stereo, surround*), itd.

Naravno, kod pretrage kolekcije zvučnih zapisa baziranih na govoru, najvažnije je prepoznati šta je rečeno. Ovom tematikom se bavi oblast prepoznavanja govora (eng. *speech recognition*). Prvi korak u prepoznavanju govora je prebacivanje *waveform-ova* u foneme (delove reči, slogove). Naredni korak je segmentacija reči i ovaj korak ima za cilj da grupiše foneme u reči, odnosno da ustanovi gde je kraj jedne reči i početak druge. Poslednji korak ima za cilj da prepozna koja je reč izgovorena, odnosno koju reč predstavlja grupa fonema. Sva tri koraka se obično obučavaju nadgledanim mašinskim učenjem. Implementacija prepoznavanja govora je vrlo komplikovana, ali postoje određeni rezultati u ovoj oblasti. Da li Vam se dogodila da nekoga slušate i pogrešno ga razumete? Zamislite koliko je onda teško implementirati da se računaru ovo nikada ne desi u prepoznavanju govora. Nakon prepoznavanja govora u zvučnom zapisu mogu se koristiti klasični modeli za indeksiranje i pretraživanje tekstualnih digitalnih dokumenata.

U određenim slučajevima postoji i potreba za *pretraživanjem kolekcije zvučnih zapisa baziranih na muzici*. Ove kolekcije se obično pretražuju bazirano na uzorku. Na primer, želimo da pronađemo muziku sličnu pesmi koju volimo. Da bi se ovo postiglo prvi korak je da se iz zvučnih zapisa u kolekciji izvuku osobine i onda je potrebno vektore osobina indeksirati nekom strukturu (na primer R-stablon). Potrebno je takođe definisati meru sličnosti dva zapisa. Nakon toga se zapisi klasteruju u MBR (eng. *minimal bounding rectangle*). Nakon svakog postavljenog upita nekim uzorkom, što je u našem slučaju pesma koju volimo, potrebno je iz uzorka izvući osobine i formirati vektor osobina. Za vektor osobina upita utvrđuje se kom klasteru pripada, odnosno utvrđuje se MBR koji se vraća.

Osobine u zvučnim zapisima koje je potrebno ekstrahovati mogu biti razne [58], a neke od njih su:

- *Mean square* - procena applitude zvučnog signala,
- *ZeroCrossings* - broj promena znaka zvučnog signala,

- *Spectrum*[32] - prvih 32 koeficijenta dobijena Diskretnom Furijeovom Transformacijom signala,
- *SpectralSum* - suma razlika susednih članova niza *Spectrum*,
- *Beats per minute* - procena ritma zvučnog zapisa,
- *Avg FFT Delta*[32] - članovi niza *Avg FFT Delta* su razlike susednih elemenata niza *Spectrum*,
- *Haar*[64] - niz 64 koeficijenta dobijena *Harovom wavelet transformacijom*,
- *Song seconds* - vreme trajanja zvučnog zapisa u sekundama,
- *Energy difference* - signal se deli na delove/frejmove, za svaki deo računa se energija, *Energy difference* je suma razlika energija za sve frejmove zvučnog zapisa,
- *Energy Zero Crossing* - događa se kad vrednosti energije dva uzastopna frejma (*Energy difference*) imaju različit znak, *Energy Zero Crossings* je broj ovih događaja za zvučni zapis.

Kao što je prethodno rečeno pored ekstrahovanja osobina zvučnih zapisa, potrebno je definisati meru sličnosti dva zvučna zapisa. To se može uraditi na sledeći način [58]. Formira se obučavajući skup, odnosno mali deo kolekcije zvučnih zapisa se podeli na klase. Iskoristi se ovaj obučavajući skup za algoritam (npr. genetski algoritam) za tjuniranje prostora osobina. Rezultat ovog tjuniranja su koeficijenti koji se stavljuju ispred osobina kako bi se optimizovala metrika sličnosti - ne utiče svaka osobina jednako na sličnost dva zvučna zapisa.

7.3 Video

Video je niz frejmova, odnosno rasterskih slika. Ovo je medij koji je doživeo revoluciju razvojem IKT. Digitalni video je danas normalna pojava, digitalne kamere su u telefonima, laptop računarima, itd. Video sa zvukom i tekstrom je multimedija, ali

se pod pojmom pretraga video zapisa najčešće podrazumeva pretraga kolekcije videa koji sadrži i zvučne i tekstualne komponente (u formi *subtitle-a*).

Postoje velike kolekcije video zapisa i potreba za pretragom ovih kolekcija je sve veća razvojem IKT. Na primer, *BBC* arhiva u toku jedne godine primi više od 500.000 upita čiji bi odgovor mogao da bude video ili deo videa, a *YouTube* primi još više upita na godišnjem nivou.

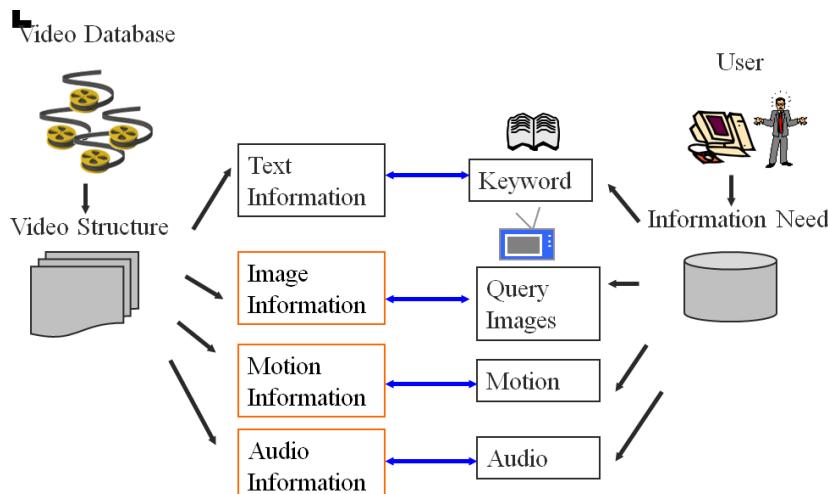
Pretraga video zapisa je vrlo kompleksna. Postoji veliki broj osobina koji se mogu ekstrahovati iz videa. Implementaciju pretrage video zapisa otežavaju kretanje objekata u kadrovima, kretanje kamere, promena *zoom-a*, itd. Koriste se razne tehnike (slika 7.18) kako bi se omogućila kvalitetna pretraga kolekcije video zapisa: *OCR*, *speech recognition*, *face recognition*, *scene recognition*, itd. Odgovor na upit ne treba da bude ceo video, nego njegov deo koji odgovara informacionoj potrebi korisnika, obično 20-40 sekundi - zbog ovoga je segmentacija scena jako bitno.

The screenshot shows a search interface. On the left, there is a thumbnail of a man in a white shirt and tie sitting at a desk. To the right of the thumbnail, the word "Query:" is followed by the text: "Find pictures of Harry Hertz, Director of the National Quality Program, NIST". Below this, under the heading "Speech:", is the transcription of a speech: "We're looking for people that have a broad range of expertise that have business knowledge that have knowledge on quality management on quality improvement and in particular ...". Under the heading "OCR:", is the text: "Harry Hertz a Director aro 7 wa- ,i,,ty Program .Harry Hertz. a. Director ..".

Slika 7.18: Upotreba *OCR* i *speech recognition* tehnike

Informaciona potreba korisnika može biti izražena tekstualnim upitom, ili uzorkom koji može biti slika, video ili zvuk (slika 7.19).

Iako postoje određeni rezultati u ovoj oblasti i dalje postoji ogroman prostor za napredak i u *text-based* pretrazi kolekcije video zapisa, a posebno u *content-based* pretrazi kolekcije video zapisa.



Slika 7.19: Pretraga kolekcije video zapisa

Rezime

- Informacije mogu biti prenete u obliku teksta, slike, veb stranice, videa ili zvuka. Svaki medij ima svoje karakteristike, prednosti i slabosti.
- Za implementaciju pretrage multimedijalnih sadržaja baziranih na tekstu (eng. *text based*) neophodno je angažovanje čoveka koji je zadužen za opisivanje sadržaja, odnosno za formiranje metapodataka.
- Za implementaciju pretrage multimedijalnih sadržaja baziranih na sadržaju (eng. *content based*) nije potrebno angažovati čoveka, koristi se računar. Opis sadržaja se realizuje ekstrahovanjem osobina medija.
- Pretraživanje slika je komplikovano iz više razloga: teško je rangirati rezultate pretrage, postoji semantički

jaz između različitih tumačenja slike, rezolucija slike je konačna.

- Osobine slike se mogu računati za celu sliku ili po delovima slike za šta je neophodna segmentacija slike. Najčešće se ekstrahuju osobine slike koje se odnose na boju, oblike objekata na slici i teksturu.
- Postoje različite vrste zvučnih zapisa: bazirani na govoru, bazirani na muzici, ostalo (alarm, zvuci iz prirode).
- Kod pretrage kolekcije zvučnih zapisa baziranih na govoru potrebno je prepoznati govore u zvučnim zapisima i prebaciti ih u tekstualni oblik.
- Kod pretrage zvučnih zapisa baziranih na muzici najčešće je izražavanje informacione potrebe uzorkom. Potrebno je ekstrahovati određene osobine zvučnih zapisa koji se nalaze u kolekciji i uzorka. Nakon toga potrebno je odrediti meru sličnosti uzorka i zvučnih zapisa u kolekciji.
- Iako je video niz frejmova, odnosno rasterskih slika, pod pojmom pretraga video zapisa najčešće se podrazumeva pretraga kolekcije videa koji sadrži i zvučne i tekstualne komponente. Pretraga video zapisa je vrlo kompleksna. Kretanje objekata u kadrovima, kretanje kamere, promena *zoom-a* otežavaju implementaciju pretrage kolekcije video zapisa.

Pitanja

1. Šta je multimedija?
2. Koja je razlika između *text based* i *content based* pretrage multimedijalnih kolekcija?
3. Šta su to lokalne, a šta globalne osobine slike i kako se računaju?
4. Kako se koristi boja kao osobina za pretragu kolekcije slika?
5. Kako se koristi oblik kao osobina za pretragu kolekcije slika?
6. Kako se koristi tekstura kao osobina za pretragu kolekcije slika?
7. Koje su razlike u implementacijama pretraga kolekcija zvučnih zapisa zasnovanih na govoru i na muzici?
8. Zašto je pretraga kolekcije video zapisa komplikovana?

Glava 8

Performanse pretraživanja

Ovo poglavlje ima za cilj da definiše osnovne pojmove evaluacije i unapređenja performansi sistema za pretraživanje i da odgovori na sledeća pitanja:

- Koje su osnovne mere kvaliteta sistema za pretraživanje?
- Da li se ocenjuje relevantnost dokumenta i upita ili relevantnost dokumenta i informacione potrebe?
- Kako se evaluirala sistem za pretraživanje?
- Kako se unapređuje sistem za pretraživanje?

Prilikom kreiranja strukture ovog poglavlja i definisanja osnovnih pojmoveva polazna osnova bila je knjiga „*Introduction to information retrieval*“ [38]. Deo informacija prezentovanih u ovom poglavlju su preuzeti iz raznih izvora otvorenog pristupa dostupnih putem Interneta, kao i iz literature navedene na kraju knjige koja je citirana u ovom poglavlju.

Kako utvrditi performanse pretraživača kolekcije digitalnih dokumenata? Mera za kvalitet pretraživača može biti koliko brzo indeksira - broj dokumenata/megabajta na sat. Takođe, mera može biti koliko brzo pretražuje - kašnjenje kao funkcija veličine indeksa i broja upita u sekundi. Takođe, mera se može odnositi i na prostor koji indeksi zauzimaju, kao i na to šta se sve može pronaći u kolekciji. Svi prethodno nabrojani kriterijumi su mrljivi - možemo kvantifikovati i brzinu i prostor. Međutim, ključna mera za pretraživač je **zadovoljstvo korisnika**.

Pre nego što pređemo na pitanje šta je zadovoljstvo korisnika, razmotrimo ko je korisnik koga želimo da zadovoljimo. Na veb pretraživačima korisnik je *tragač*. Tragač želi da pronađe ono što traži. Ako se tragači vraćaju na neki veb pretraživač to znači da su zadovoljni tim veb pretraživačom. Takođe, na pretraživačima korisnik je i *zakupac reklama*. On želi da tragači kliknu na njegovu reklamu. *Clickthrough rate* se definiše kao odnos broja kliktanja korisnika na neku reklamu i broja pojavljivanja te reklame. Zakupac reklama je zadovoljan ako njegova reklama ima dobar *clickthrough rate*. Ako imamo veb aplikaciju za prodaju određenih dobara koja ima mogućnost pretrage kolekcije dobara, možemo analizirati dve vrste korisnika. Prva vrsta korisnika je *kupac* koji želi da pronađe i kupi ono zbog čega je došao na sajt. Mera njegovog zadovoljstva je srazmerna vremenu potrošenom do kupovine, a takođe možemo analizirati i procenat konvertovanih tragača u kupce da bismo ustavili koliko su kupci zadovoljni sistemom (ovo naravno zavisi i od cene i kvaliteta dobara, ne samo od sistema). Druga vrsta korisnika je *prodavac* koji želi da prodaje svoju robu putem sistema. Prodavac će biti zadovoljan ako ima profita. Ako imamo aplikaciju u prodavnici koja proverava čega ima na lageru u magacinu i gde se određeni artikal nalazi u magacinu, *direktor* će biti zadovoljan ako su zaposleni produktivniji, jer brzo pronalaze ono što im treba. To može da utiče na potreban broj zaposlenih za obavljanje iste količine posla, a samim tim i na profit firme.

U ovom poglavlju mićemo pre svega analizirati performanse pretrage tekstualnih digitalnih dokumenata iz ugla korisnika koji

koristi sistem da bi pronašao ono što mu treba. Dakle, najviše ćemo pažnje posvetiti korisniku kog smo u prošlom paragrafu nazvali tragačem. Šta je zadovoljstvo korisnika? Faktori *zadovoljstva korisnika* uključuju sledeće kriterijume:

- brzinu dobijanja odgovora,
- veličinu indeksa - šta sve mogu pronaći u kolekciji,
- nezatrpan korisnički interfejs,
- *relevantnost* dobijenih odgovora - najvažniji kriterijum je da li je korisnik dobio ono što je tražio,
- besplatan pristup.

8.1 Relevantnost

Nijedan faktor pojedinačno nije dovoljan: fantastično brzi ali beskorisni odgovori neće korisnika učiniti zadovoljnim. Kako da kvantifikujemo zadovoljstvo korisnika? Iako **relevantnost rezultata pretrage** nije jedini kriterijum zadovoljstva korisnika, ovo je najvažniji kriterijum zadovoljstva i često se zadovoljstvo korisnika izjednačava sa relevantnošću rezultata pretrage. Kako kvantifikovati relevantnost rezultata pretrage, kako meriti relevantnost? Standardna metodologija za merenje relevantnosti rezultata pretrage, odnosno kvaliteta sistema za pretraživanje ima sledeća tri elementa:

- test-kolekciju dokumenata,
- skup test-upita,
- binarnu (ili ne-binarnu) ocenu relevantnosti svakog para upit-dokument.

Ovakvo vrednovanje se često kritikuje, jer predstavlja veštački scenario. Odnosno, koristi se test-kolekcija dokumenata i upita koji možda nisu dobri reprezentanti stvarne kolekcije dokumenata sistema i stvarnih upita koje postavljaju korisnici sistema. Međutim, pored svih ovih kritika, ova metodologija je vrlo uspešno

i ima široku primenu u oblasti utvrđivanja performansi sistema za pretraživanje.

Da li nam treba relevantnost rezultata pretraga u odnosu na upit? Relevantnost u odnosu na upit je vrlo problematična. Na primer, tražimo informacije o tome da li je crno vino bolje za smanjenje rizika od infarkta nego belo vino. Ovo je informaciona potreba (i), a ne upit. Odgovarajući upit za ovu informacionu potrebu (q) je: **wine AND red AND white AND heart AND attack** (preuzeto iz [38]). Razmotrimo sada dokument d :

He then launched into the heart of his speech and attacked the wine industry lobby for downplaying the role of red and white wine in drunk driving.

d je relevantan za upit q , ali **nije** relevantan za informacionu potrebu i . Kao što je već rečeno, korisnik je zadovoljan ako je pronašao ono što je tražio. Dakle, **zadovoljstvo korisnika se može meriti samo prema relevantnosti u odnosu na informacione potrebe, a ne upite**. Dakle, u oblasti pronalaženja informacija kada se govori o relevantnosti upit-dokument misli se na relevantnost informaciona potreba-dokument. Ono što je jedan od težih problema sa kojim se sistemi za pretraživanje susreću je činjenica da se može desiti da korisnik ne reprezentuje svoju informacionu potrebu odgovarajućim upitom.

Relevantnost dokumenta za određenu informacionu potrebu se posmatra izolovano od ostalih rezultata pretrage. Zbog ovoga je često ova mera kritikovana. **Marginalna relevantnost** dokumenta u rezultatu je dodatna informacija koju sadržaj dokumenta donosi. Na primer, duplikat može biti vrlo relevantan, ali ima marginalnu relevantnost 0, jer se u listi odgovara već nalazi isti sadržaj. Marginalna relevantnost je bolja mera zadovoljstva korisnika, ali je gotovo nemoguće kreirati test skupove koji koriste marginalnu relevantnost i pomoći ovakvih skupova testirati performanse sistema.

Ako imamo ocene relevantnosti za parove informaciona potreba-dokument možemo računati mere performansi sistema za pretraživanje definisane na sledeći način.

Preciznost P (eng. *precision*) je udeo pronađenih relevantnih dokumenata u svim pronađenim dokumentima, odnosno u listi rezultata pretrage.

$$\text{Preciznost} = \frac{\#\text{(pronađeni relevantni)}}{\#\text{(svi pronađeni)}} = P(\text{relevantan}|\text{pronađen})$$

Povrat R (eng. *recall*) je udeo pronađenih relevantnih dokumenata u svim relevantnim dokumentima koji postoje u kolekciji. Povrat se u literaturi često naziva *odziv*.

$$\text{Povrat} = \frac{\#\text{(pronađeni relevantni)}}{\#\text{(svi relevantni)}} = P(\text{pronađen}|\text{relevantan})$$

	Relevantan	Nerelevantan
Pronađen	true positives (TP)	false positives (FP)
Nije pronađen	false negatives (FN)	true negatives (TN)

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

Zašto bi koristili složene mere kao preciznost i povrat za merenje kvaliteta pretraživača? Zašto ne nešto jednostavnije - npr. tačnost?

Tačnost je deo odluka (relevantan/nerelevantan) koje su ispravne, u smislu prethodne tabele tačnost = $(TP + TN)/(TP + FP + FN + TN)$. Zašto tačnost nije korisna mera za veb pretraživače? Zato što se može primeniti jednostavan trik da se dobije dobra tačnost u veb pretraživačima - uvek kaži ne i vrati prazan skup. Za svaki upit u velikim kolekcijama sa kojima rade veb pretraživači mnogo je više nerelevantnih nego relevantnih dokumenata. Vraćanjem praznog skupa se postiže 99.99% tačnost za većinu upita. Tragači na veb pretraživačima, i u drugim sistemima za pretragu kolekcija dokumenata, žele da pronađu nešto i imaju

određeni stepen tolerancije na dobijanje određenog broja nerelevantnih odgovora u listi rezultata. Dakle, tačnost nije dobra mera zadovoljstva korisnika, dok su preciznost i povrat, kao i mere izvedene iz ovih mera često koriste za ocene performansi sistema za pretraživanje.

Sistemi za pretraživanje teže da imaju dobre vrednosti i preciznosti i povrata, a ne samo odličnu vrednost jedne od ovih mera. Povrat se može povećati vraćanjem više dokumenata kao rezultat na određeni upit, jer je povrat neopadajuća funkcija broja pronađenih dokumenata. Prema definiciji povrata, sistem koji vraća sve dokumente ima 100% povrat! Suprotno je takođe (često) tačno - lako je imati veliku preciznost ako vraćamo mali broj dokumenta kao rezultat pretrage. Neka je najbolje rangirani dokument uvek relevantan. Kako se može maksimizovati preciznost? Tako što će se uvek vraćati samo jedan dokument kao odgovor na upit. Preciznost je maksimalna, ali povrat nije dobar. Možda u kolekciji ima još na desetine relevantnih dokumenata koje bi korisnik voleo da dobije kao odgovor.

F mera omogućava da se meri kompromis između preciznosti i povrata:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad \text{gde} \quad \beta^2 = \frac{1 - \alpha}{\alpha}$$

$$\alpha \in [0, 1] \text{ pa prema tome } \beta^2 \in [0, \infty]$$

Najčešće korišćen za merenje performansi sistema za pretraživanje je *balansirani F₁* sa $\beta = 1$ ili $\alpha = 0,5$, čime se dobija *harmonijska sredina* P i R : $\frac{1}{F_1} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$, odnosno $F_1 = \frac{2PR}{P+R}$.

Kada je $\beta > 1$ povrat vrednujemo više nego preciznost, a kada je $\beta < 1$ onda preciznost vrednujemo više nego povrat.

Pretpostavimo da je neki sistem za pretraživanje nakon postavljenog upita dao listu rezultata i da važi:

	relevantni	nerelevantni
pronađeni	18	2
nepronađeni	82	1.000.000.000

Kolika je preciznost ovog sistema?

$$\frac{18}{18+2} = \frac{18}{20}$$

Ovo je odlična preciznost. Koliki je povrat?

$$\frac{18}{18+82} = \frac{18}{100}$$

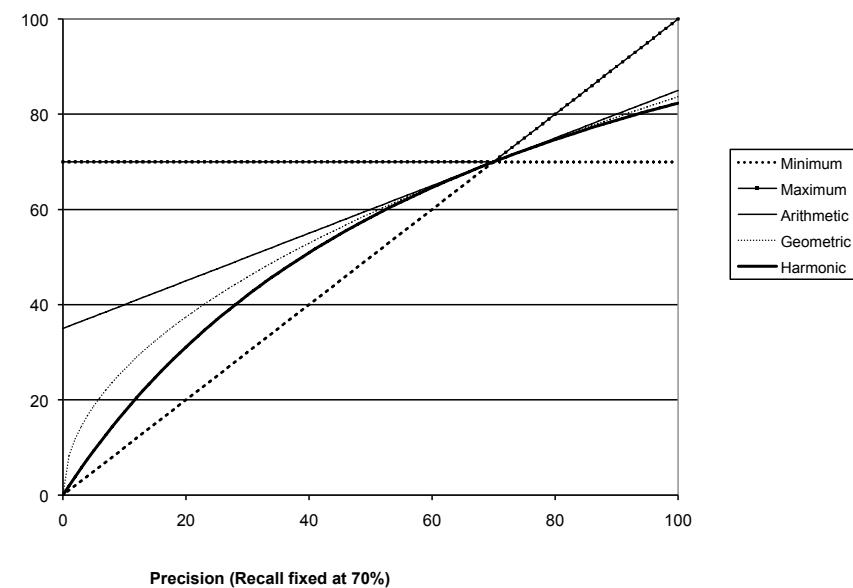
Ovo je loš povrat. Kolika je F_1 mera?

$$\frac{2PR}{P+R} = \frac{2 \frac{18}{20} \frac{18}{100}}{\frac{18}{20} + \frac{18}{100}} = \frac{3}{10}$$

Ovo je prilično loša F_1 mera zato što je povrat loš. Harmonijska sredina se može posmatrati kao meki minimum preciznosti i povrata (slika 8.1). Ako bi se koristila mera koja je aritmetička sredina preciznosti i povrata, pretraživač koji "vraća sve" bi imao vrednost ove mere 50% - što je previše. Cilj je kreirati meru koja kažnjava loše performanse na račun bilo preciznosti ili povrata. Ovo se postiže uzimanjem minimuma - F_1 (harmonijska sredina) je kao meki minimum.

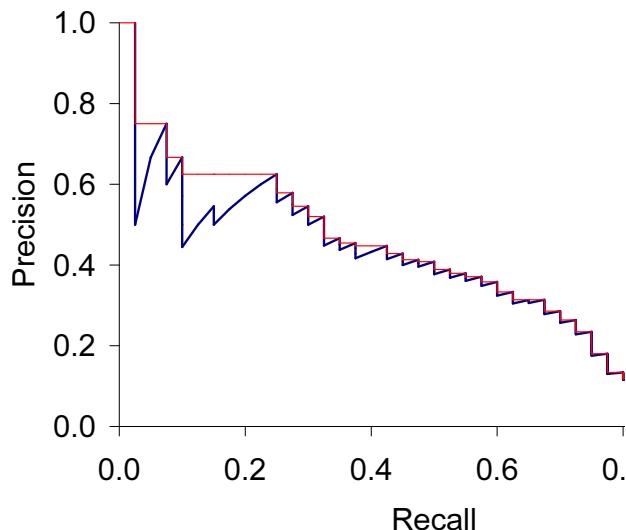
Kao što je već rečeno standardna metodologija za merenje relevantnosti rezultata pretrage, odnosno kvaliteta sistema za pretraživanje ima tri elementa: test-kolekciju dokumenata, skup test-upita, binarnu (ili, ređe, ne-binarnu) ocenu relevantnosti svakog para upit-dokument. Preciznost, povrat i F_1 mera se mogu računati za svaki test-upit na osnovu binarnih ocena relevantnosti para upit-dokument i onda izračunati prosečne vrednosti ovih mera za jedan sistem za pretraživanje.

Preciznost, povrat i F mera su *mere nerangiranih skupova*. Dakle, potrebna je samo informacija da li je dokument relevantan ili nije i da li se nalazi u listi rezultata. Nije važno koliko je dokument relevantan i na kojoj je poziciji u listi rezultata. Ipak, ove mere se mogu koristiti za merenje performansi sistema za pretraživanja koji imaju rangirane liste rezultata. Mogu se izračunati ove mere za svaki "prefiks": najbolji 1 pogodak, najboljih 2 pogotka, najbolja 3 pogotka, najbolja 4 pogotka, itd. Dakle,



Slika 8.1: F_1 mera i druge mere (preuzeto iz [38])

iako je sistem za neki upit kao rezultat vratio listu od, na primer, 223 dokumenata, može se izračunati preciznost i povrat kao da je sistem vratio samo prvih k dokumenata kao odgovor. Izračunavanje preciznosti i povrata na ovaj način daje *preciznost/povrat krivu* (slika 8.2). Svaka tačka odgovara rezultatu za najboljih k rangiranih pogodaka ($k = 1, 2, 3, 4 \dots$). Ova kriva se može interpolirati tako što se uzima maksimum svih budućih tačaka. Kako je povrat neopadajuća funkcija, maksimum budućih tačaka je zapravo maksimalna preciznost budućih tačaka. Razlog za ovu interpolaciju je činjenica da će korisnik hteti da pregleda još pogodaka ako se preciznost i povrat popravljaju. Za prikazanu sliku vrednosti povrata i preciznosti u 11 tačaka su:



Slika 8.2: Preciznost/povrat kriva (preuzeto iz [38])

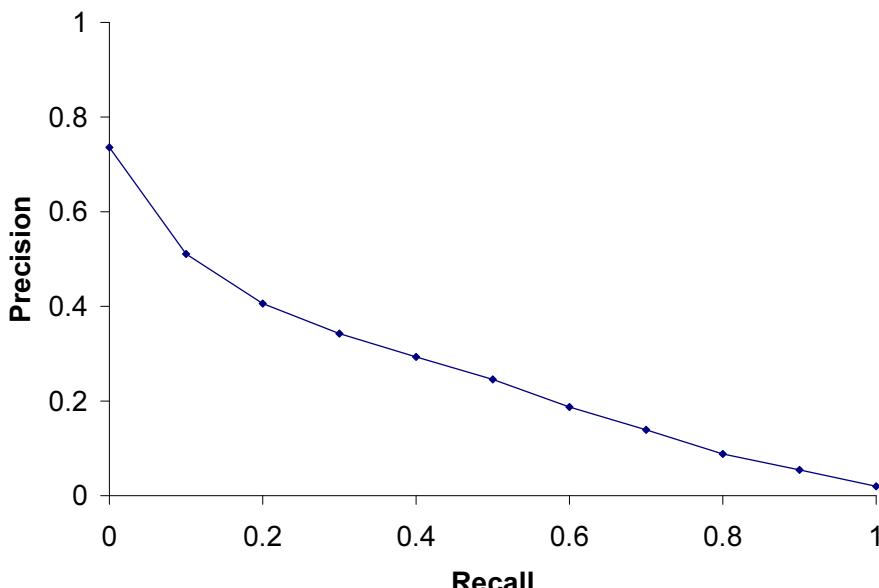
Povrat	Interpolirana Preciznost
0,0	1,00
0,1	0,67
0,2	0,63
0,3	0,55
0,4	0,45
0,5	0,41
0,6	0,36
0,7	0,29
0,8	0,13
0,9	0,10
1,0	0,08

Prosek interpolirane preciznosti za 11 tačaka je $\approx 0,425$.

Ako je povrat 0, to znači da od svih relevantnih dokumenata u kolekciji u listi odgovora se nije našao nijedan relevantan. Ako u listi odgovora nema relevantnih dokumenata, onda je i preciznost 0. Ovo je u većini sistema situacija jedino ako je lista odgovora prazna, odnosno ako posmatramo najboljih 0 odgovora. Međutim, interpolirana preciznost je 1. Ako posmatramo najboljih 1

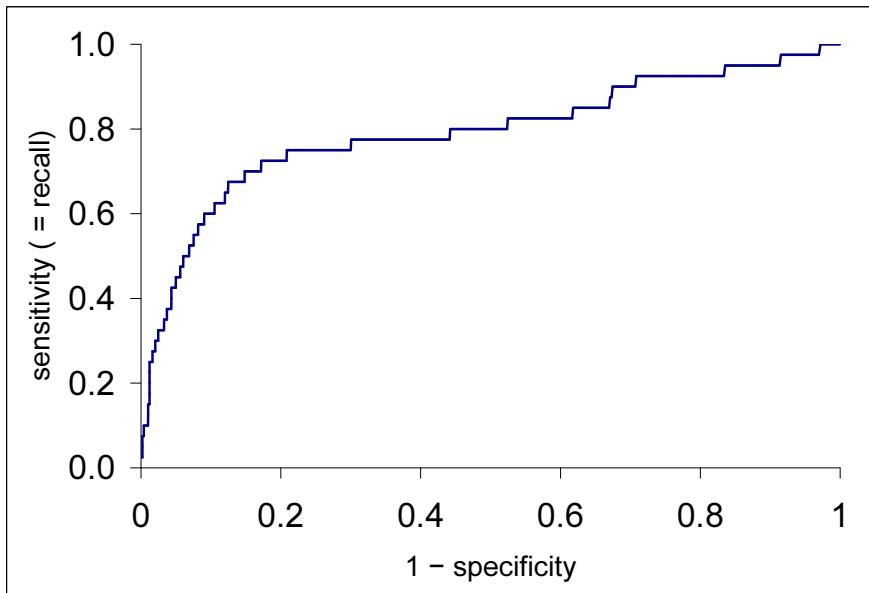
odgovora i ako je taj odgovor relevantan onda je preciznost 1, a povrat je vrlo bliska nuli pod pretpostavkom da u kolekciji ima na stotine ili hiljade relevantnih odgovora na određeni upit.

Preciznost/povrat krivu je potrebno kreirati za svaki test-upit i onda izračunati interpoliranu preciznost za nivoje povrata $0, 0; 0, 1; 0, 2 \dots$. Onda se mogu izračunati prosečne vrednosti interpoliranih preciznosti za sve nivoje povrata (slika 8.3). Ovo je mera performansi za sve nivoje povrata.



Slika 8.3: Uprosečeni precision/recall grafikon za 11 nivoa povrata (preuzeto iz [38])

Za merenje performansi sistema za pretraživanje sa rangiranim rezultatima može se koristiti i ROC kriva (slika 8.4). Specifičnost (eng. *specificity*) se koristi kao x-osa na slici i računa se kao specifičnost = $TN/(TN + FN)$, odnosno to je udeo nepronađenih nerelevantnih dokumenata u svim nepronađenim dokumentima. Ova kriva je slična kao i preciznost/povrat kriva, ali nas zanima samo mala zona u donjem levom uglu. Upravo ova zona je zapravo uvećana na preciznost/povrat krivi.



Slika 8.4: ROC kriva (preuzeto iz [38])

Za test kolekciju uobičajeno je da sistem radi loše za neke informacione potrebe (na primer, $P = 0,2$ za $R = 0,1$), a odlično za neke druge (na primer, $P = 0,95$ za $R = 0,1$). Često je varijansa jednog sistema za više upita mnogo veća nego varijansa različitih sistema za isti upit. Ovo je posledica činjenice da postoje jednostavne i složene informacione potrebe.

8.2 Evaluacija performansi

Povrat se teško meri kod sistema za pretragu velikih kolekcija kao što su veb pretraživači. Veb pretraživači obično koriste preciznost za najboljih k kao meru svojih performansi, na primer $k = 10$, ili koriste mere koje više vrednuju da je prvi podogak bolji nego deseti. Takođe se koriste merae koje nisu zasnovane na relevantnosti:

- *clickthrough* za prvi pogodak,
- Laboratorijske studije ponašanja korisnika,

- A/B testiranje.

Clickthrough za prvi pogodak nije pouzdana mera ako se posmatra jedan *clickthrough*, jer korisnik može da odluči da je dokument nerelevantan nakon kratkog pregledanja stranice, ali je ova mera prilično pouzdana u proseku za veliki broj korisnika i upita.

Laboratorijske studije ponašanja korisnika uključuju posmatranje korisnika i njihovog ponašanja prilikom pretrage. Vrlo je skupo sprovoditi ove studije i često je problem stvoriti uslove za ove studije.

A/B testiranje ima za cilj testiranje jednog unapređenja sistema za pretraživanje. Uslov koji mora biti ispunjen da bi se moglo sprovoditi ovo testiranje je da postoji veliki pretraživač u pogonu sa velikim brojem korisnika i svakodnevnih upita. Većina korisnika pristupa staroj verziji sistema, a mali deo korisnika (recimo 1%) se preusmerava na novu verziju sistema koja ima unapređenje. Vrši se vrednovanje i stare i nove verzije sistema pomoću neke "automatske" mere, npr. *clickthrough* za prvi pogodak. Poređenjem ovih mera može se utvrditi da li unapređenje povećava zadovoljstvo korisnika. Verovatno je ovo metodologija kojoj veliki pretraživači najviše veruju. Jedna varijanta ove metodologije koja se ređe koristi je da se korisnicima ostavi mogućnost da sami izaberu staru ili novu verziju sistema. U ovom slučaju jako mali broj korisnika se odlučuje za novu verziju sistema.

Kao što je već rečeno standardna metodologija za merenje performansi sistema za pretraživanje zahteva *test skup* (eng. *benchmark*) sa tri elementa: test-kolekciju dokumenata, kolekciju test-upita, binarnu (ili, ređe, ne-binarnu) ocenu relevantnosti svakog para upit-dokument. Kolekcija dokumenata treba da bude takva da reprezentuje dokumente koje očekujemo da imamo i u stvarnom slučaju. Kolekcija informacionih potreba, koje ćemo često neispravno nazivati upitim, treba da bude takva da reprezentuje informacione potrebe koje očekujemo i u stvarnom korišćenju sistema. Da bismo dobili ocene relevantnosti svakog para upit-dokument moramo angažovati ocenjivače za ovaj posao. Angažovanje ljudi za ove ocene je skupo i vremenski za-

htevno. Ocenjivači treba da reprezentuju one koje očekujemo i u stvarnom slučaju, odnosno treba da imaju slične karakteristike, interesovanja i znanja kao i očekivani korisnici sistema.

Ocene relevantnosti su korisne samo ako su *konzistentne* među ocenjivačima. Konzistentnost među ocenjivačima možemo meriti **kapa merom** (κ). Dakle, kapa je mera koliko se međusobno ocenjivači slažu i ova mera je dizajnirana za kategorične ocene.

$P(A) =$ koji deo od ukupnog broja slučajeva se ocenjivači slažu, $P(E) =$ koji deo slaganja bismo dobili slučajno,

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

Za slučajno slaganje vrednost kapa mere je 0, a za totalno slaganje vrednost je 1. Vrednosti $\kappa \in [2/3, 1.0]$ se smatraju prihvativim, a sa manjim vrednostima se smatra da je potreban redizajn metodologije ocenjivanja. Razmotrimo računanje kapa mere za slaganja ocenjivača koja su prikazana u narednoj tabeli (primer preuzet iz [38]).

		Ocenj. 2 relevantnost		
		Da	Ne	Total
Ocenj. 1 relevantnost	Da	300	20	320
	Ne	10	70	80
	Total	310	90	400

Odnos puta kada su se ocenjivači složili $P(A) = (300+70)/400 = 370/400 = 0,925$.

$$P(nerelevantan) = (80 + 90)/(400 + 400) = 170/800 = 0,2125$$

$$P(relevantan) = (320 + 310)/(400 + 400) = 630/800 = 0,7878$$

$$\text{Verovatnoća da su se slučajno složili } P(E) = P(nerelevantan)^2 + P(relevantan)^2 = 0,2125^2 + 0,7878^2 = 0,665.$$

$$\text{Kapa mera } \kappa = (P(A) - P(E)) / (1 - P(E)) = (0,925 - 0,665) / (1 - 0,665) = 0,776.$$

Ovo je prihvatljiva kapa mera.

Pored mogućnosti za razvijanje sopstvenog test skupa koje je skupo i vremenski zavisno, postoji nekoliko standardizovanih test skupova za relevantnost. Prvi skup testova za precizno

merenje preformansi sistema za pretraživanje je *Cranfield benchmark*. Nastao je kasnih 1950-tih u Velikoj Britaniji. Sadrži kolekciju dokumenta od 1398 apstrakata iz članaka o aerodinamici, skup od 225 upita, iscrpne ocene relevantnosti za sve parove upit-dokument. Za današnje uslove suviše mali i atipičan uzorak. *Text Retrieval Conference* (TREC) je međunarodno konferencija koju svake godine organizuje *U.S. National Institute of Standards and Technology* (NIST). Za potrebe merenja performansi sistema za pretraživanje koji su se prezentovali na prvih osam konferencija između 1992. i 1999. godine formiran je *TREC Ad Hoc* test skup. Ovaj test skup ima 1,89 milion dokumenata, uglavnom novinskih članaka i 450 informacionih potreba prezentovanih upitima. Nema iscrpnih ocena relevantnosti, jer je to previše skupo za ovu količinu dokumenata i upita. Umesto toga, ocene NIST-ovih ocenjivača postoje samo za dokumente koji su bili među prvih k koje je vratio jedan od sistema u TREC testu. *GOV2* je još jedan TREC/NIST test skup. Sadrži čak 25 miliona veb strana što ovu kolekciju čini najvećom test kolekcijom koja je lako dostupna, ali i dalje je ova količina sadržaja 3 reda veličine manja od *Google* i *Yahoo!* indeksa. Prilikom testiranja sistema za pretraživanje treba odabratи test skup koji ima kolekciju dokumenata takvu da reprezentuje dokumente koje očekujemo da imamo i u stvarnom slučaju. Analogno ovome važi i za kolekciju test-upita. Jedna od bitnih karakteristika i dokumenta i upita u kolekciji je jezik na kome su napisani. Prethodno nabrojani test skupovi su pre svega za pretragu na engleskom jeziku. Postoje i test skupovi koji se razvijaju za druge jezike. NTCIR je test skup za merenje performansi sistema za pretraživanje sadržaja napisanih upotrebom dalekoistočnih jezika. *Cross Language Evaluation Forum* (CLEF) je fokusiran na evropske jezike i na *cross-language* pretraživanje.

Za merenje performansi sistema za pretragu kolekcije strukturiranih tekstualnih digitalnih dokumenata, na primer kolekcije XML dokumenata, potreban je drugačiji test skup od onih koji se koriste za kolekcije tekstualnih digitalnih dokumenata. INEX (eng. *INitiative for the Evaluation of XML retrieval*) je kreirao jedan ovakav test skup za ocenu performansi pretraživanja XML-a [64]. Sastoji se iz skupa XML dokumenata, kolekcije pre-

traživačkih tema, odnosno informacionih potreba i podataka o oceni relevantnosti delova dokumenata za određene informacione potrebe. Ako želimo da ovim test-skupom testiramo neki sistem za pretragu XML dokumenata, potrebno je da taj sistem prvo indeksira XML dokumente iz test skupa, a zatim da neko zapiše pretraživačke teme kao upite koristeći upitni jezik koji sistem (koji je potrebno testirati) razume. Nakon izvršenja ovih upita, sistem vraća elemente unutar dokumenata, ne cele dokumente, i rangira pronađene elemente. Da bise izvršila evaluaciju performansi sistema porede se elementi koje je sistem vratio sa ocenama u INEX test skupu.

INEX test skup sadrži 12.107 publikacija iz *IEEE Computer Society* ukupne veličine 494 megabajta. Prosečna publikacija ima 1.532 XML čvora i prosečna dubina hijerarhije ovih XML-ova je 6,9.

Među pretraživačkim temama, odnosno informacionim potreбama imamo dve vrste tema:

- *content only*: upiti u formi slobodnog teksta,
- *content and structure*: pored teksta imamo i eksplicitna ograničenja na strukturu dokumenta, tj. sadržavanje elemenata.

U INEX test skupu za svaki upit, elementi (delovi dokumenta) su ocenjeni po dva kriterijuma:

- *relevantnost*: koliko je relevantan element (deo dokumenta),
- *pokrivanje*: da li je element suviše uzak ili suviše širok.

Relevantnost je ocenjena na skali od 0 (nerelevantno) do 3 (vrlo relevantno). Ocjenjuje se i pokrivanje, jer za upit koji traži definiciju Furijeove transformacije nije isto da li se dobija kao odgovor samo jednačina (suviše uzak odgovor), celo poglavje (suviše širok), ili tekst definicije. Pokrivanje je ocenjeno na skali od četiri ocene:

- *no coverage*: tema ne odgovara ničemu u elementu,

- *too large*: tema je manji deo pronađenog elementa,
- *too small*: element je premali da pokrije temu,
- *exact*.

Svaki pronađeni element ima ocenu iz skupa $\{0, 1, 2, 3\} \times \{N, S, L, E\}$. Ove ocene se koriste za izračunavanje f -vrednosti pronađenog elementa pomoću jedne od sledeće dve formule (prva je striktna, pa je zato druga mnogo češće u upotrebi):

$$f_{strict}(rel, cov) = \begin{cases} 1 & (rel, cov) = 3E \\ 0 & \text{inače} \end{cases}$$

$$f_{generalized}(rel, cov) = \begin{cases} 1,00 & (rel, cov) = 3E \\ 0,75 & (rel, cov) \in \{2E, 3L, 3S\} \\ 0,50 & (rel, cov) \in \{1E, 2L, 2S\} \\ 0,25 & (rel, cov) \in \{1S, 1L\} \\ 0,00 & (rel, cov) = 0N \end{cases}$$

Dakle, korišćenjem jedne od ove dve formule dobijaju se f -vrednosti koje predstavljaju skalarne mera kvaliteta pronađenog elementa. Mogu se izračunati f -vrednosti za različite brojeve pronađenih elemenata (10, 20, 30) i to koristiti kao sredstvo za poređenje pretraživača XML kolekcija.

8.3 Unapređenje sistema

Prehodno opisana evaluacija performansi sistema za pretraživanje se može koristiti za poređenje dva sistema, ali se često koriste i za unapređenje jednog sistema za pretraživanje. Na primer, promenili smo pretprocesiranje teksta prilikom indeksiranja i obrade upita i želimo da proverimo da li smo tom promenom unapredili sistem. Možemo izmeriti preciznost/povrat mere koristeći neki test skup pre i posle promene pretprocesiranja teksta i uporediti dobijene mere. Na osnovu ovih mera možemo ustaviti da li je potrebno zadržati novo pretprocesiranje teksta ili se

vratiti na staro. U nekim situacijama možemo koristiti i druge pristupe da ustanovimo da li smo unapredili sistem kao što su: *clickthrough* za prvi pogodak, laboratorijske studije ponašanja korisnika, A/B testiranje. Već je rečeno da je najbitnije zadovoljstvo korisnika i da ga često merimo zapravo kroz merenje relevantnosti dobijenih rezultata pretrage. Međutim, sistem za pretraživanje se može unaprediti i u drugim aspektima, a ne samo u preprocesiraju teksta koje menja relevantnost dobijenih rezultata pretrage. Laboratorijskim studijama ponašanja korisnika možemo evaluirati sva unapređenja sistema.

8.3.1 Rezultati pretrage

Vrlo je važno na koji se način korisniku predstavljaju i organizuju **rezultati pretrage**. Najčešće su rezultati predstavljeni kao lista koja pored opisa dokumenta sadrži i linkove na dokumente koji su rezultat pretrage - “10 plavih linkova”. Način na koji su dokumenti opisani u listi može značajno unaprediti zadovoljstvo korisnika. U nekim situacijama korisnik može da odredi relevantne pogotke na osnovu opisa, ne mora da klikne na sve dokumente sekvensialno. Najčešće opis dokumenta u rezultatu uključuje naslov i neke metapodatke. Vrlo je poželjno da opis sadrži i **sažetak**. Postavlja se pitanje kako “izračunati” ovaj sažetak. Postoje dve vrste sažetka:

- statički,
- dinamički.

Statički sažetak dokumenta je uvek isti bez obzira na upit kojim je dokument pronađen. Obično je statički sažetak podskup dokumenta. Može to biti jednostavna heuristika, na primer, prvih 50-tak reči u dokumentu. Može biti i nešto složenija, na primer izvuče se iz dokumenta skup “ključnih” rečenica. Za ovaku heuristiku koriste se jednostavne *natural language processing* (NLP) heuristike za ocenjivanje svake rečenice. Sažetak je sastavljen od najbolje rangiranih rečenica. Ovaj pristup je zasnovan na mašinskom učenju. Najsloženije računanje statičkog sažetka je ono

u kome statički sažetak nije podskup dokumenta nego se koristi složeni NLP sa sinteza/generisanje sažetka. Ovo je najsloženiji pristup, ali pristup koji ako bi radio kako treba doneo bi najveće zadovoljstvo korisnika. Zbog toga se teži razvoju tehnika za kvalitetnu implementaciju ovog pristupa, ali u ovom momentu kvalitet ovako generisanih sažetaka nije na zadovoljavajućem nivou za većinu aplikacija, zbog čega se ovaj pristup retko koristi.

Dinamički sažeci su *zavisi od upita*. Pokušavaju da objasne zašto je dokument pronađen za baš taj upit. Cilj je prikazati jedan ili više fragmenata iz dokumenta koji sadrže termove iz upita. Ovi dinamički sažeci se generišu u skladu sa rangiranjem. Posebno dobri fragmenti su fragmenti gde se traženi termovi pojavljuju kao fraza, ali su dobri i fragmenti gde se traženi termovi pojavljuju zajedno na malom prostoru. Prikazani sažetak sadrži ceo sadržaj fragmenata, a ne samo termove iz upita. Na primer, za upit *pretraga indeksiranje lucene* izdvojeni fragmenti koji će činiti dinamički sažetak su prikazani zadebljanim tekstrom u nadrenom tekstu (preuzeto iz teksta disertacije [19]).

*... Za indeksiranje i pretraživanje tekstualnih sadržaja korišćena je Apache Lucene [57] biblioteka. Apache Lucene je javno dostupna biblioteka pisana u Javi namenjena pretraživanju teksta. Pošto je kriterijum sličnosti definisan (u opisu slučaja korišćenja <Pick journal>) tako da su cirilično i latinično pismo ravnopravni *svi cirilični sadržaji se pre indeksiranja prevode na latinično pismo, a prilikom pretrage podataka svi cirilični upiti se prevode na latinično pismo. To znači da Apache Lucene radi samo sa sadržajima zapisanim latiničnim pismom*, ali se u bazi podataka sadržaji čuvaju onako kako ih je korisnik uneo. Prevođenje ciriličnih sadržaja na latinično pismo je jednoznačno...*

Kada se kreira dinamički sažetak obično se izdvojeni fragmenti teksta razdvaje sa ... i obično se zadebljanim tekstrom označe samo termovi iz upita koji se pojavljuju u ovim fragmentima:

... Za indeksiranje i pretraživanje tekstualnih sadržaja korишćena je Apache **Lucene** [57] biblioteka. Apache **Lucene** je javno dostupna biblioteka ... svi cirilični sadržaji se pre indeksiranja prevode na latinično pismo, a prilikom pretrage podataka svi cirilični upiti se prevode na latinično pismo. To znači da Apache **Lucene** radi samo sa sadržajima zapisanim latiničnim pismom ...

Veb pretraživači gotovo po pravilu imaju dinamički generisane sažetke, a ovo je poželjna osobina i za druge sisteme za pretraživanje. Prostor na stranici sa rezultatima je ograničen zbog čega sažeci moraju biti kratki, ali moraju biti dovoljno dugački da nose neku informaciju. Sažeci bi trebalo da opišu da li i kako dokument odgovara upitu. Idealno bi bilo da je sažetak lingvistički ispravan i da sažetak zadovolji korisnikovu informacionu potrebu - da korisnik ne mora da otvara i pregleda ceo dokument. Dinamički sažeci su važan deo zadovoljstva korisnika, jer mogu brzo da se pregledaju da bi se pronašao relevantan dokument koji se želi otvoriti i pregledati, a u nekim slučajevima, uopšte se ne mora otvarati dokument. Na ovaj način se štedi korisnikovo vreme što značajno utiče na zadovoljstvo korisnika.

Postavalja se pitanje kako generisati dinamičke sažetke? Odašte da dobavimo druge termove (osim onih iz upita) za sažetke? Ne može se konstruisati dinamički sažetak iz invertovanog indeksa - bar ne može efikasno. Potrebno je keširati dokumente. Pozicioni invertovani indeks može odgovoriti na kojim pozicijama u dokumentu se nalazi term iz upita. Ove pozicije su obično *word offset* (redni broj reči od početka dokumenta), jer se tako lakše odgovara na upite fraze, ali bi za generisanje dinamičkih sažetaka bilo mnogo zgodnije da su pozicije izražene kao *byte offset* (redni broj bajta od početka dokumenta). Ove pozicije se koriste da se iz keširanog dokumenta izvuku fragmenti. Indeksi i keširana kopija mogu biti neažurni, samim tim se može desiti da je dinamički generisan sažetak netačan, odnosno fragmenti koji su u njemu ne moraju postojati u dokumentu. U praksi vrlo dugački

dokumenti se obično ne keširaju kompletni, nego samo njihov određeni početni deo.

8.3.2 Klasifikacija

U cilju unapređenja sistema za pretraživanje koriste se i tehnike **klasifikacije**. Polazeći od zadatog skupa, klasifikacija pokušava da utvrdi kojoj klasi ili klasama posmatrani objekat (dокумент) pripada. Pojam klasifikacije je usko vezan za pretraživanje podataka, iako se klasifikacija dokumenata koristi u više domena: istraživanje i analiza teksta i podataka, procesiranje slike (utvrđivanje da li je landscape ili portrait), pretraživanje teksta (kao i drugih vrsta sadržaja), itd.

Klasifikacija se može koristiti kao *korak u preprocesiranju teksta*: utvrđivanje enkodinga, segmentacija reči, utvrđivanje jezika, *truecasing* - da li reč treba da ostane napisana velikim slovima (Fed - fed, CAT - cat). Takođe, može se koristiti za *automatsku detekciju spam strana* ili *automatska detekcija drugih vrsta strana* koja ne treba da se nađu u rezultatima pretrage (na primer, *sexually explicit content*). Klasifikacija se može koristiti za *utvrđivanje sentimenta* (eng. *sentiment detection*). Na primer, klasifikacija komentara na pozitivne i negativne nam daje mogućnost da možemo da pronađemo sve negativne komentare za neki proizvod - kada pročitamo negativne komentare odlučićemo da li da kupimo proizvod. Klasifikacijom se mogu *razdvojiti rezultati po unapred definisanim grupama* ili e-mail-ovi po folderima (spam folder). Takođe, klasifikacija nam omogućava implementaciju *topic-specific ili vertikalnog pretraživanja* - mogu se pronaći *computer science* strane na univerzitetima u Kini pri čemu se ne mora spominjati reč Kina na tim stranicama. *Rangiranje rezultata* se može izvršiti na osnovu klasifikacije - veoma relevantan, prilično relevantan, itd.

Postoje tri pristupa u implementaciji klasifikacije: manuelna - ručna klasifikacija, klasifikacija bazirana na pravilima, klasifikacija bazirana na tehnikama mašinskog učenja. Ručna klasifikacija je dugotrajna i gotovo nemoguće je primeniti ovaj pristup na

klasifikacijama velikih kolekcija. Sa druge strane ova klasifikacija je tačna. Klasifikacija bazirana na pravilima je implementacija ručno zapisanih pravila koja u kontekstu klasifikacije tekstualnih sadržaja opisuju značenje pojedinih reči za smeštanje datog teksta u određenu klasu dokumenata. Klasifikacija bazirana na tehnikama mašinskog učenja je najčešće korišćen pristup za implementaciju klasifikacije. Kriterijumi za odlučivanje se utvrđuju automatski. Sistem se obučava putem skupa obučavajućih podataka. Sami objekti se reprezentuju skupom atributa relevantnih za postupak klasifikacije među kojima je i jedan atribut koji označava oznaku klase kojoj objekat pripada (klasni atribut). Skup obučavajućih podataka se obično ručno klasificuje od strane eksperata - labeling (označavanje, anotacija). Dakle, ovo je nadgledani metod obučavanja i tokom obučavanja cilj je kreirati model (matematički) kojim se klasni atribut izražava kao funkcija vrednosti ostalih atributa. Krajnji cilj je da formirani model omogućuje da se novi objekti koji nisu bili deo obučavajućeg skupa što je moguće tačnije klasifikuju na osnovu vrednosti svojih atributa. Iako najčešće korišćeni pristup, klasifikacija bazirana na tehnikama mašinskog učenja gotovo nikada ne daje savršenu tačnost u klasifikaciji podataka. Kvalitet klasifikatora meri se na test skupu koji predstavlja skup podataka za koji su klase poznate, ali koji nije korišćen za obuku modela. Ako nemamo takav testni skup koristimo obučavajući skup i *cross-validation* tehniku. Osnovna ideja je deljenje svih dostupnih obeleženih podataka na dva skupa: obučavajući skup - na osnovu koga se formira klasifikator; test skup - na kome se klasifikator evaluira. Koristi se k-tostruka unakrsna validacija na sledeći način:

1. podeli se skup u k jednakih delova
2. formiraju se sve moguće kombinacije delova na sledeći način
način:
 - obučiti na $(k_1 + \dots + k_{n-1})$, testirati na k_n
 - obučiti na $(k_1 + \dots + k_{n-2} + k_n)$, testirati na k_{n-1}
 -

3. izračuna se prosek performansi od svih kombinacija - koriste se tačnost, preciznost, povrat, F_1 mera, matrica troškova koja uzima u obzir koliko je "skupa" netačna klasifikacija.

8.3.3 Klasterovanje

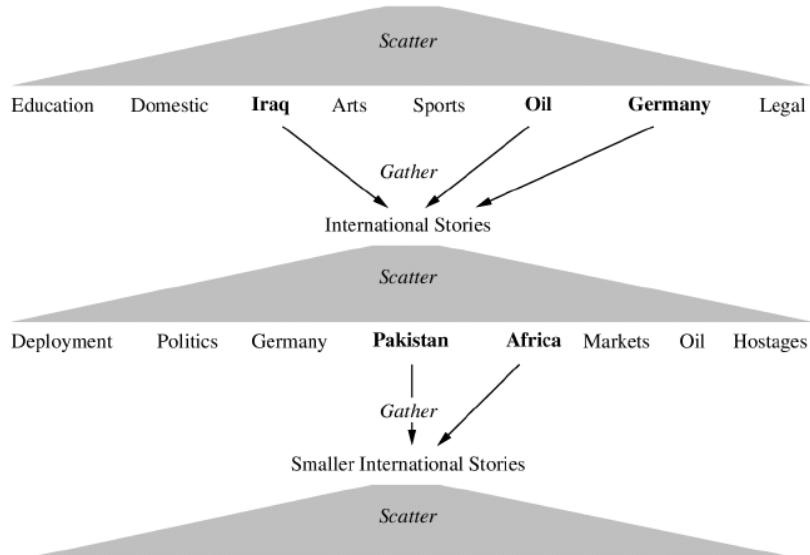
U cilju unapređenja sistema za pretraživanje koriste se i tehnike **klasterovanja**. Klaster analiza je podjela skupa objekata na podskupove i cilj klaster analize je nalaženje grupe objekata takvih da su objekti iz grupe međusobno slični (eng. *as similar as possible*) i da su različiti od objekata iz drugih grupa (eng. *as dissimilar as possible*). I klaster analiza i klasifikacija bazirana na tehnikama mašinskog učenja dele skup na podskupove. Za razliku od klasifikacije koja je bazirana na tehnikama mašinskog učenja, u kreiranju klaster analize koristi se nenadgledani metod obučavanja, nema definisanih klasa i nema obučavajućeg skupa kreiranog od strane eksperta (čoveka). Klasterovanje se koristi za razumevanje grupe povezanih dokumenata za pretraživanje, grupe gena i proteina sa sličnom strukturu, grupe akcija sa sličnom fluktacijom cena, itd. Takođe, klasterovanje se koristi i za sažimanje kolekcije objekata - smanjenje velike količine objekata uzimanjem jednog predstavnika iz klastera.

Kada se klasterovanje koristi u sistemima za pretraživanje polazi se od sledeće pretpostavke: dokumenti iz istog klastera se ponašaju *slično* iz ugla relevantnosti u odnosu na informacionu potrebu, odnosno ako je jedan dokument iz klastera relevantan, *verovatno* su i ostali dokumenti iz klastera relevantni.

Klasterovanje rezultata pretrage (eng. *search result clustering*) grupiše rezultate po klasterima, odnosno rezultati nisu prosta lista relevantnih odgovora. Na primer, za upit *jaguar* sistem vraća rezultate grupisane u nekoliko klastera - kola, životinja, operativni sistem. Ovo olakšava korisniku da se snade među odgovorima na upit. Značajno je kada imamo upit koji nije jednosmislen.

Scatter-Gather je tehnika razbijanja na klastera i spajanje klastera koja se koristi u sistemima za pretragu (slika 8.5). Cilj je

bolji korisnički interfejs. Kolekcija dokumenata je klasterovana, korisnik selektuje klastere koji su mu od interesa. Odgovori (dokumenti) koji pripadaju selektovanim klasterima su ponovo klasterovani, pa korisnik ponovo selektuje klastere koji su mu od interesa. I sve tako dok ne dođe do malog klastera koji mu je od interesa.



Slika 8.5: *Scatter-Gather* (preuzeto iz [38])

Collection clustering je tehnika za klasterovanje kolekcije bez interakcije sa korisnikom i bez postavljenog upita. Ako korisnik želi da čita vesti, on ih uglavnom ne pretražuje, nego želi da čita nove vesti iz neke oblasti. Imamo veliku količinu novih vesti, korisnik želi da ispunи svoju informacionu potrebu koju retko izražava upitom. Google News (<https://news.google.com/>) koristi ovu tehniku.

Language modeling je korišćenje klasterovanja da bi se rešio problem sinonima. Na primer, korisnik je u upitu koristio reč "car", sistem je pronašao nekoliko dokumenata koji imaju ovu reč, ali koji imaju i reči "automobile", "vehicle". Vraćamo i druge rezultate iz istog klastera iako nemaju reč "car", ali imaju neku od

reči ‐automobile‐ ili ‐vehicle‐ zbog čega pripadaju istom klasteru. Ovo može povećati povrat ako se uzme u obzir prepostavka od koje smo pošli, ali može i smanjiti preciznost.

Cluster-based retrieval se koristi da bi se ubrzala pretraga i na taj način povećalo zadovoljstvo korisnika. Računanje sličnosti vektora koji predstavlja upit i dokumenata u kolekciji može biti sporo. Alternativa je poređiti upit sa klasterima (kojih je znatno manje nego dokumenata) i vrati sve dokumente iz klastera. Iz svakog klastera se odabere par dokumenata predstavnika sa kojima se porede upiti (*cluster pruning*). Ovo je manje precizno od klasičnog pristupa u vektorskom modelu, ali je dosta brže, i u praksi se u nekim situacijama pokazalo da su rezultati zadovoljavajući.

U implementacijama klaster analiza postoji razlika u performansama. Mnogo je lakše validirati nadgledanu klasifikaciju nego klasterovanje. Ključni je problem sa čime da se poređimo? Kako da uporedimo dva algoritma za klasterovanje, koji je bolji? Može se koristiti eksterni indeks za merenje stepena slaganja dobijenih oznaka klasa sa oznakama klasa koje su eksterno date. Takođe, može se koristiti i interni indeks za merenje kvaliteta strukture klasteringa bez korišćenja eksternih informacija. Na kraju, može se koristiti i relativni indeks za poređenje različitih klaster analiza ili dobijenih klastera. Parametar u ovim indeksima mogu biti i eksterni i interni indeksi.

8.3.4 *Relevance feedback*

Kao što je već rečeno *Scatter-Gather* tehnika nakon procesiranja korisnikovog inicijalnog upita definiše interakciju između korisnika i sistema sa ciljem da korisnik dođe do informacija koje mu trebaju. Ova tehnika je bazirana na klasterovanju. Postoji još jedna tehnika koja nakon procesiranja inicijalnog korisnikovog upita definiše interakciju korisnika i sistema, opet sa ciljem da korisnik dođe do informacija koje mu trebaju. Ova tehnika nije bazirana na klasterovanju i zove se **relevance feedback**. Re-

levance feedback (RF) takođe značajno može unaprediti sistem, odnosno zadovoljstvo korisnika i radi na sledeći način:

1. korisnik postavlja (kratak, jednostavan) upit,
2. pretraživač vraća skup dokumenata,
3. korisnik označava neke dokumente kao relevantne, a neke kao nerelevantne,
4. pretraživač izračunava novu reprezentaciju informacione potrebe, bolju od inicijalnog upita,
5. pretraživač izvršava novi upit i vraća rezultate korisniku.

Novi rezultati imaju (najčešće) bolji povrat. Ovo može da se radi iterativno, ali se obično završava nakon jednog ciklusa. Razmotrimo sledeći primer koji je preuzet iz [38]. Početni upit je “New space satellite applications” (korak 1), a lista odgovora na ovaj upit koje je sistem vratio (korak 2):

1. 0,539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
2. 0,533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
3. 0,528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
4. 0,526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
5. 0,525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
6. 0,524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
7. 0,516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
8. 0,509, 12/02/87, Telecommunications Tale of Two Companies

Pretpostavimo da je korisnik označio 1., 2. i 8. odgovor kao relevantan, a da ostale smatra nerelevantnim (korak 3). Nakon toga pretraživač izračunava novu reprezentaciju informacione potrebe

na osnovu korisnikovog inicijalnog upita i označenih dokumenta kao relevantnih i nerelevantnih (korak 4):

2,074	new	15,106	space
30,816	satellite	5,660	application
5,991	nasa	5,196	eos
4,196	launch	3,972	aster
3,516	instrument	3,446	arianespace
3,004	bundespost	2,806	ss
2,790	rocket	2,053	scientist
2,003	broadcast	1,172	earth
0,836	oil	0,646	measure

Na kraju pretraživač izvršava novi upit i prikazuje listu odgovora (korak 5):

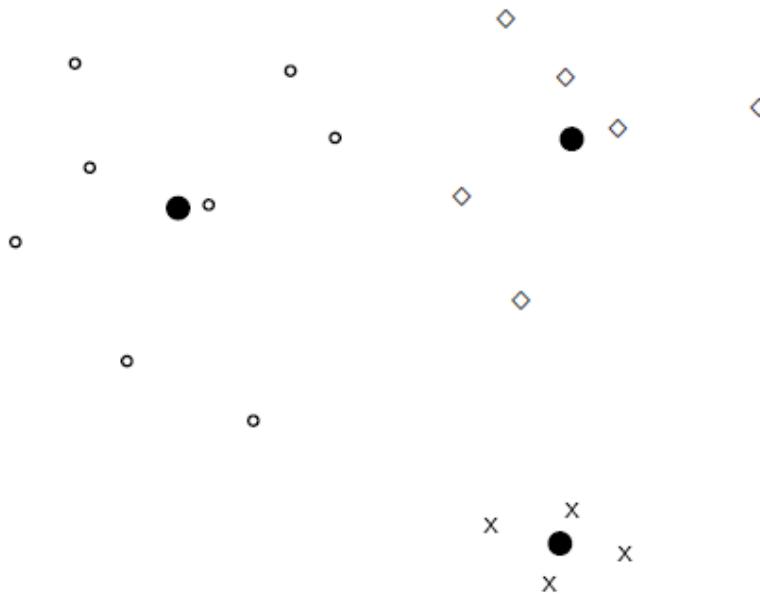
- * 1. 0,513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
- * 2. 0,500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
- 3. 0,493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
- 4. 0,493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit
- * 5. 0,492, 12/02/87, Telecommunications Tale of Two Companies
- 6. 0,491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use
- 7. 0,490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
- 8. 0,490, 06/14/90, Rescue of Satellite By Space Agency To Cost \$90 Million

Odgovori označeni * su bili odgovori i nakon izvršenja inicijalnog upita, a ostali odgovori nisu bili tada u listi odgovora.

Osnovni koncept za RF je centroid. *Centroid* je centar mase za skup tačaka (slika 8.6). Kako su dokumenti u vektorskem modelu predstavljeni kao tačke u visoko-dimenzionalnom prostoru, možemo izračunati centroid za skup dokumenata upotrebom sledeće formule:

$$\vec{\mu}(D) = \frac{1}{|D|} \sum_{d \in D} \vec{v}(d)$$

gde je D skup dokumenata i $\vec{v}(d) = \vec{d}$ je vektor kojim reprezentujemo dokument d .



Slika 8.6: Centroid

Rocchio-ov algoritam implementira RF u vektorskem modelu. Ovaj algoritam bira upit \vec{q}_{opt} koji maksimizuje

$$\vec{q}_{opt} = \max_{\vec{q}} [\text{sim}(\vec{q}, D_r) - \text{sim}(\vec{q}, D_{nr})]$$

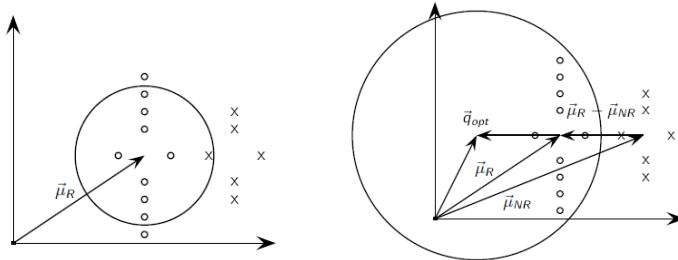
Ideja je da se napravi maksimalno razdvajanje relevantnih i nerelevantnih dokumenata. Optimalni vektor upita je:

$$\vec{q}_{opt} = \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j + \left[\frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \right] \quad (8.1)$$

D_r - skup relevantnih dokumenata; D_{nr} - skup nerelevantnih dokumenata

Na ovaj način pomeramo centroid relevantnih dokumenata za razliku dva centroida (slika 8.7) i to smatramo optimalnim upitom:

$$\vec{q}_{opt} = \text{centroid}_{rel} + (\text{centroid}_{rel} - \text{centroid}_{rel})$$



Slika 8.7: *Rocchio* algoritam

$\vec{\mu}_R$ ne razdvaja dobro relevantne i nerelevantne, dok \vec{q}_{opt} odlično razdvaja relevantne i nerelevantne.

Problem je što mi obično ne znamo sve relevantne i nerelevantne dokumente, i ne znamo da nađemo njihov centroid. Zbog toga se u praksi koristi modifikovan Rocchio algoritam iz 1971. godine koji je popularizovan od strane SMART sistema:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \quad (8.2)$$

q_m : modifikovani vektor upita

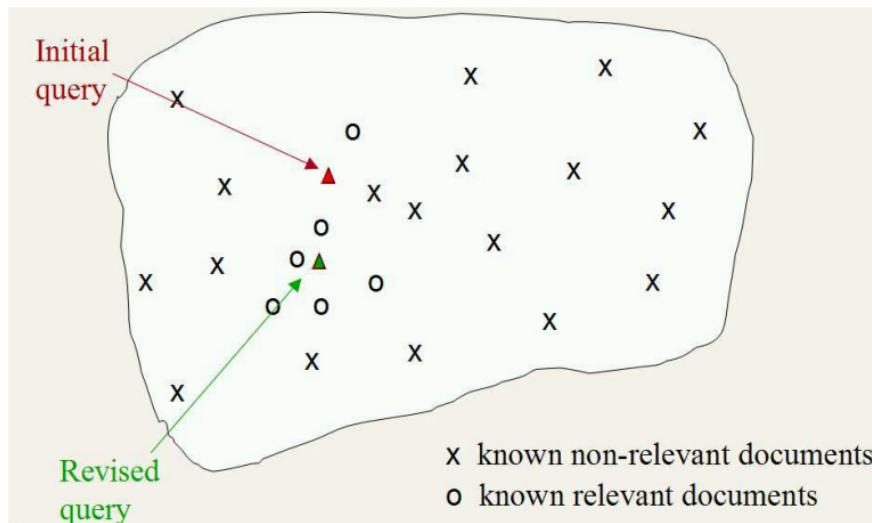
q_0 : originalni vektor upita;

D_r i D_{nr} : skup poznatih relevantnih odnosno nerelevantnih

α , β i γ : težine

Opet je ideja da se novi upit pomera prema (poznatim) relevantnim dokumentima i udaljava od (poznatih) nerelevantnih dokumenata (slika 8.8). Možemo praviti kompromise između α i β/γ u različitim situacijama. Na primer, ako imamo mnogo ocenjenih dokumenata, obično se koristi veći β/γ . Podsećanja radi korisnici sistema za pretraživanje žele da pronađu nešto i imaju

određeni stepen tolerancije na dobijanje određenog broja nerelevantnih odgovora u listi rezultata. Dakle, pozitivni odgovori su vredniji od negativnih. Zbog toga je često $\beta > \gamma$. Na primer, neka je $\beta = 0,75$, onda je $\gamma = 0,25$ da bi se dobile veće težine za pozitivne odgovore. Postoje i sistemi koji uzimaju u obzir samo pozitivne odgovore.



Slika 8.8: Modifikovani *Rocchio* algoritam iz 1971. godine (preuzeto iz [38])

RF pre svega može da unapredi povrat ako su ispunjene sledeće dve pretpostavke:

1. korisnik zna termove u kolekciji dovoljno dobro za početni upit,
2. relevantni dokumenti sadrže slične termove, odnosno nalaze se blizu jedan drugog u vektorskom prostoru.

Na žalost ove pretpostavke nisu uvek ispunjene. Često korisnik ne zna termove u kolekciji dovoljno dobro za početni upit, odnosno postoji neslaganje korisnikovog rečnika i rečnika kolekcije. Na primer, korisnik koristi reč "kosmonaut", a u kolekciji se

mnogo češće koristi reč "astronaut". U nekim situacijama pretprocesiranje teksta koje koristi rečnik sinonima može da pomogne u ovom neslaganju korisnikovog rečnika i rečnika kolekcije. Takođe, relevantni dokumenti nisu uvek slični, odnosno mogu biti značajno udaljeni u vektorskem prostoru. Na primer, ako imamo upit "contradictory government policies", RF verovatno neće značajno povećati povrat za ovaj upit. Razlog za ovo leži u činjenici da je upit takav da imamo nekoliko nevezanih "prototipova" u relevantnim odgovorima, na primer:

- *Subsidies for tobacco farmers vs. anti-smoking campaigns,*
- *Aid for developing countries vs. high tariffs on imports from developing countries.*

RF primenjen na "tobacco" dokumente neće pomoći u pronaalaženju dokumenata o "developing countries". Pored mogućih kršenja polaznih prepostavki, RF ima još nekoliko problema. RF je vremenski skup, jer kreira dugačke modifikovane upite koji su skupi za obradu. Korisnici često oklevaju da daju eksplisitne odgovore koji su dokumenti relevantni, a koji nisu. Često je teško razumeti zašto je neki dokument pronađen nakon što je primenjen jedan ili više ciklusa RF.

Pored svih ovih problema, u određenim situacijama RF unapređuje sistem za pretragu. Kako vrednovati da li je i koliko RF unapredio naš sistem? Uzmimo jednu od mera za performanse IR sistema - recimo preciznost za najboljih 10 dokumenata $P@10$. Možemo izračunati $P@10$ za originalni upit q_0 , a nakon primene RF algoritma možemo izračunati $P@10$ za RF-modifikovani upit q_1 . U većini slučajeva: mera za q_1 je drastično bolja od mere za q_0 . Da li je ovo dobro poređenje? Nije, zato što je korisnik nakon izvršenog upita označava koji su relevantni dokumenti od prvih 10 rezultata. Poređenje mora biti na ostaku kolekcije - dokumentima koje korisnik još nije ocenio. Studije pokazuju da je RF uspešan ako se vrednuje i na ostaku kolekcije. U praksi, jedan RF ciklus je često vrlo uspešan, dok je drugi ciklus marginalno koristan. Pravo vrednovanje koristi RF mora uključiti druge metode koje troše istu količinu korisnikovog vremena. Alternativa

za RF je da korisnik revidira i ponovo pošalje upit. Da li je to bolje korišćenje korisnikov vremena? Korisnicima može da se više sviđa revizija upita nego ocenjivanje dokumenata. RF unapređuje sistem, ali nema jasnog dokaza da je RF “najbolje korišćenje” korisnikovog vremena.

Google koristi RF tehniku za “Similar” opciju. Kada unesemo neki upit na *Google* veb pretraživaču za svaki od dobijenih odgovora imamo opciju “Similar”. Kada odaberemo ovu opciju za neki rezultat pretrage, inicijalni upit će biti proširen termovima iz odabranog dokumenta i tako kreirani upit će biti izvršen.

Pored upotrebe RF za potrebe unapređenja performansi sistema za pretragu, RF se koristi i za održavanje *trajnih upita*. Na primer, želimo da primimo svakog dana listu novinskih članaka objavljenih u prethodnih 24 sata na temu “multicore computer chips”. RF se može koristiti za rafinaciju ovog trajnog upita tokom vremena. Spam filteri rade sličnu stvar, tokom vremena vrši se rafinacija upita za utvrđivanje spama na osnovu korisnikovog *feedback-a*. RF je mnogo praktičniji za trajne upite nego za rafinaciju jednog upita za pretragu.

Pseudo-RF, koji je poznat i pod nazivom *blind relevance feedback*, automatizuje “ručni” deo pravog RF ciklusa i radi na sledeći način:

1. izračuna se rangirana lista pogodaka za korisnikov upit,
2. *prepostavi se da je najboljih k dokumenata relevantno*,
3. uradi se RF ciklus (na primer, *Rocchio* algoritmom).

Pseudo-RF radi dobro u proseku, ali može da vrati katastrofle rezultate za neke upite. Više iteracija može da izazove *klizanje upita* u neželjenom smeru, jer prepostavka da je najboljih *k* dokumenata relevantno može biti pogrešna.

Postoji i *indirektni ili implicitni RF* gde se takođe od korisnika ne očekuje eksplicitno izjašnjavanje da li je nešto relevantno ili nije, ali se koristi implicitno izjašnjavanje korisnika. Na primer, ako se za jedan upit na veb pretraživaču na osnovu dinamičkog sažetka korisnici često odlučuju da otvore određeni

rezultat, veb pretraživači mogu koristiti reči iz tog dinamičkog sažetka da njime prošire inicijalni upit i vrate listu odgovora za proširenji upit.

8.3.5 Globalno proširenje upita

Pored RF postoje i druge tehnike za povećanje povrata kao što je **globalno proširenje upita** (GPU). Termin globalno proširenje upita se koristi kada mislimo na globalne metode za reformulaciju upita. U GPU, upit se menja zavisno od nekog globalnog resursa koji ne zavisi od upita (slika 8.9). Kod RF korisnik daje odgovor (informacije) o dokumentima, dok kod GPU korisnik daje odgovor o rečima ili frazama. RF jeste jedna varijanta proširenja upita, ali su termovi dodati u RF ciklusu zasnovani na lokalnim informacijama u tekućem rezultatu. Sa druge strane termovi dodati u GPU se zasnivaju na globalnim informacijama koje ne zavise od upita. Za dodavanje ovih termova koriste se baze podataka semantički povezanih reči - **tezaurus**. Za svaki term t u upitu, proširuje se upit rečima koje tezaurus navodi kao semantički povezane sa t . Na primer, **hospital** se proširuje sa **medical**. U proseku GPU povećava povrat, ali može značajno da smanji preciznost, naročito sa dvosmislenim termovima. Na primer, proširenje upita **interest rate** sa **interest rate fascinate evaluate** će značajno pogoršati preciznost. GPU je često korišćen metod u specijalizovanim pretraživačima za naučne i inženjerske primene gde postoji dosta termina koji često idu zajedno.



Slika 8.9: Proširenje upita

Postoje dve vrste tezaurusa iz aspekta načina formiranja: ručno i automatski formirani. **Ručni tezaurus** neko održava. Primer ručnog tezaurusa je *PubMed* (slika 8.10). Vrlo je skupo ručno održavanje tezaurusa. Ručni tezaurus je skoro ekvivalentan anotiranju sa *kontrolisanim rečnikom*.

Automatski generisan tezaurus je zasnovan na pokušaju da se generiše tezaurus na osnovu analize distribucije reči u dokumentu. Na primer, može biti zasnovan na statistici zajedničkog pojavljivanja. Osnovni pojam kod automatskog generisanja tezaurusa je sličnost dve reči. Razlikuju se definicije sličnosti dve reči.



Slika 8.10: Ručno formirani tezaurus - PubMed

Jedna definicija može biti da su dve reči slične ako se “pojavljuju zajedno”. Druga definicija može biti da su dve reči slične ako se pojavljaju u istom gramatičkom odnosu sa sličnim rečima. Jabuke i kruške se beru, ljušte, jedu, pripremaju - dakle jabuke i kruške su slične. Primena prve definicije je robusnija, odnosno otpornija na greške u parsiranju, ali primena druge definicije daje tačnije rezultate. Primer tezaurusa baziranog na prvoj definiciji sličnosti dat je u nastavku (preuzeto iz [38]).

reč	najbliži susedi
absolutely	absurd, whatsoever, totally, exactly, nothing
bottomed	dip, copper, drops, topped, slide, trimmed
captivating	shimmer, stunningly, superbly, plucky, witty
doghouse	dog, porch, crawling, beside, downstairs
makeup	repellent, lotion, glossy, sunscreen, skin, gel
mediating	reconciliation, negotiate, case, conciliation
keeping	hoping, bring, wiping, could, some, would
lithographs	drawings, Picasso, Dali, sculptures, Gauguin
pathogens	toxins, bacteria, organisms, bacterial, parasite
senses	grasp, psyche, truly, clumsy, naive, innate

Veb pretraživači često proširuju upit na osnovu ranijih upita (eng. *query log*). Ovakva izmena upita je složenija od jednostavnog proširenja upita i zahteva korišćenje tehnika *log mining-a*.

Dakle, RF i globalno proširenje upita su tehnike koje se koriste za unapređenje sistema za pretraživanje. Ove tehnike povećavaju povrat, ali u određenim slučajevima značajno unazađuju preciznost. Takođe, ove tehnike troše vreme - korisnikovo i sistema za pretraživanje, zbog čega se postavlja pitanje da li postoje tehnike koje bolje koriste korisnikovo vreme.

Rezime

- Ključna mera sistema za pretraživanje je zadovoljstvo korisnika.
- Najvažniji kriterijum zadovoljstva korisnika je relevantnost rezultata pretrage. Relevantnost rezultata pretrage u odnosu na informacionu potrebu, a ne u odnosu na upit, je ono što je bitno.
- Preciznost, povrat i F_1 mera su mere zasnovane na relevantnosti rezultata pretrage koje se koriste za ocenu kvaliteta sistema za pretraživanje.
- Za evaluaciju performansi sistema koriste se standar-dizovani test skupovi ili test skupovi specijalno kreirani za određeni sistem i za ove test skupove se računaju mere nabrojane u prethodnoj stavci.
- Test skup za evaluaciju sistema za pretraživanje strukturiranih tekstualnih digitalnih dokumenata je drugačiji od test skupa za evaluaciju sistema za pretraživanje nestrukturiranih tekstualnih digitalnih dokumenata.
- Veb pretraživači koriste i druge tehnike za evaluaciju performansi: *clickthrough* za prvi pogodak, laboratorijske studije ponašanja korisnika, A/B testiranje.
- Vrlo je važno na koji se način korisniku predstavljaju i organizuju rezultati pretrage. Vrlo je poželjno da opis jednog rezultata sadrži sažetak koji može biti statički (nezavisan od upita) ili dinamički (zavisan od upita).
- U cilju unapredjenja sistema za pretraživanje koriste se i tehnike klasifikacije i tehnike klasterovanja.
- *Relevance feedback* i globalno proširenje upita unapređuju povrat sistema, ali za neke upite mogu značajno unazaditi preciznost.

Pitanja

1. Da li se relevantnost odgovara meri u odnosu na informacionu potrebu ili upit?
2. Šta je preciznost (eng. *precision*)?
3. Šta je povrat (eng. *recall*)?
4. Šta je F mera i zašto je ona relevantnija od korišćenja preciznosti i povrata?
5. Kako se može vršiti evaluacija performansi sistema za pretraživanje?
6. Šta je kapa mera?
7. Zašto je važan način prezentacije rezultata pretrage i koje su dve osnovne vrste sažetka?
8. Kako se mogu koristiti tehnike klasifikacije u sistemima za pretraživanje?
9. Kako se mogu koristiti tehnike klasterovanja u sistemima za pretraživanje?
10. Koji su koraci u obradi upita upotrebom *relevance feedback*-a?
11. Koja je razlika između pseudo *relevance feedback*-a i klasičnog *relevance feedback*-a?
12. Objasniti razliku između globalnog proširenja upita i *relevance feedback*-a?
13. Šta je tezaurus i koje su dve vrste tezaurusa?

Bibliografija

- [1] IEC 82045-1:2001, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=34159
- [2] The man with the perfect memory - just don't ask him to remember what's in it,
<http://www.theguardian.com/science/2005/dec/28/research.highereducation>
- [3] Barker, P. (2005). What is IEEE Learning Object Metadata / IMS Learning Resource Metadata?
<http://zope.cetis.ac.uk/lib/media/WhatIsLOMscreen.pdf>
- [4] Ivanović, L. (2014). Modelovanje i implementacija digitalne biblioteke teza i disertacija, doktorska disertacija, Fakultet tehničkih nauka u Novom Sadu
- [5] Shariff, M. (2006). Alfresco Enterprise Content Management Implementation. Packt Publishing, ISBN 1-904811-11-6
- [6] Sladić, G., Milosavljević, B., & Gostojić, S. (2009). Digitalno potpisivanje dokumenata u Alfresco sistemu.
- [7] McCandless, M., Hatcher, E., & Gospodnetic, O. (2010). Lucene in Action: Covers Apache Lucene 3.0. Manning Publications Co.
- [8] Chowdhury, G., & Chowdhury, S. (2002). Introduction to digital libraries. Facet publishing.

- [9] Shearer, K. (2003). Institutional repositories: towards the identification of critical success factors. Canadian journal of information and library science, Vol. 27, No. 3, pp. 89-108.
- [10] Andrew, T. (2004). Theses Alive!: an E-theses management system for the UK. Edinburgh Research Archive, <http://www.era.lib.ed.ac.uk/handle/1842/423>
- [11] Arabito, S., & Asnicar, F. (2006). OpenstarTs: a “lean” approach to ETD publishing. E-LIS. <http://eprints.rclis.org/10324/>
- [12] Bevan, S. J. (2005). Electronic thesis development at Cranfield University. Program: electronic library and information systems, Vol. 39, No. 2, pp. 100-111.
- [13] Johnson, I. M., & Copeland, S. M. (2008). OpenAIR: The Development of the Institutional Repository at the Robert Gordon University. Library Hi Tech News, Vol. 25, No. 4, pp. 1-4.
- [14] Coles, B., & Johnson, K. (2010). Moving Electronic Theses from ETD-db to EPrints: The Best of Both Worlds: A Project Briefing Presented at the CNI Spring 2010 Membership Meeting, Baltimore, MD, April 12, 2010.
- [15] Vijayakumar, J. K., Murthy, T. A. V., & Khan, M. T. M. (2006). Experimenting with a model digital library of ETDs for Indian universities using D-space. <http://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1105&context=libphilprac>
- [16] Jones, R. D. (2004). The Tapir: Adding e-theses functionality to DSpace. Ariadne, no. 41. <http://www.ariadne.ac.uk/issue41/jones/>
- [17] Koulouris, A., & Anagnostopoulos, A. (2010). Theses e-submission tool at the National Technical University of Athens. OCLC Systems & Services, Vol. 26, No. 2, pp. 123-132.

-
- [18] Ivanović, D., Milosavljević, B., Konjović, Z. & Surla, D. (2009). Funkcionalnost korisničkog interfejsa za bibliotečku obradu radova sa konferencija, SNTPi, Zbornik radova, Beograd, pp. 61-64
 - [19] Ivanović, D. (2010). Informacioni sistem naučno-istraživačke delatnosti, doktorska disertacija, Fakultet tehničkih nauka u Novom Sadu
 - [20] Ivanović, D., Milosavljević, G., Milosavljević, B. and Surla, D. (2010). A CERIF-compatible research management system based on the MARC 21 format. Program: Electronic library and information systems, vol. 44, no. 1, pp. 229-251, DOI: 10.1108/00330331011064249
 - [21] Ivanović, D. & Milosavljević, B. (2010). Software architecture of system of bibliographic data. Proceedings of the XXI Conference on Applied Mathematics PRIM 2009, pp. 85-94.
 - [22] Ivanović, D. (2011). Sistemi za skladištenje naučnih sadržaja. Zadužbina Andrejević
 - [23] Ivanović, D. (2011). Data exchange between CRIS UNS, institutional repositories and library information systems. Proceedings of the 5th International Quality Conference, Kragujevac, May 19-21, pp. 371-378.
 - [24] Ivanović, D., Surla, D. and Konjović, Z. (2011). CERIF compatible data model based on MARC 21 format. The Electronic Library, Vol. 29, No. 1, pp. 52-70, DOI: 10.1108/0264047111111433.
 - [25] Ivanović, D., Surla, D. and Racković, M. (2011). A CERIF data model extension for evaluation and quantitative expression of scientific research results. Scientometrics, Vol. 86, No. 1, pp. 155-172, DOI: 10.1007/s11192-010-0228-2.
 - [26] Ivanović, D. (2012). Software systems for increasing availability of scientific-research outputs. Novi Sad Journal of Mathematics - NS JOM, Vol. 42, No. 1, pp. 37-48.

- [27] Ivanović, D., Surla, D. & Racković, M. (2012). Journal evaluation based on bibliometric indicators and the CERIF data model. *Computer Science and Information Systems*, Vol. 9, No. 2, pp. 791-811, DOI: 10.2298/CSIS110801009I
- [28] Ivanović, L., Dimić-Surla, B., Segedinac, M. & Ivanović, D. (2012). CRISUNS ontology for theses and dissertations. *Proceedings of the ICIST 2012*, Kopaonik, February 29 - March 3, pp. 164-169.
- [29] Ivanović, L., Ivanović, D. and Surla, D. (2012). A data model of theses and dissertations compatible with CERIF, Dublin Core and EDT-MS. *Online Information Review*, Vol. 36, No. 4.
- [30] Ivanovic, L., Ivanovic, D. and Surla, D. (2012). Integration of a Research Management System and an OAI-PMH Compatible ETDs Repository at the University of Novi Sad, Republic of Serbia. *Library Resources & Technical Services*, Vol. 56, No. 2, pp. 104-112.
- [31] Ivanović, L., & Surla, D. (2012). A software module for import of theses and dissertations to CRISSs. *Proceedings of the CRIS 2012 Conference*, Prague, June 6-9, pp. 313-322.
- [32] Ivanovic, L., Ivanovic, D., Surla, D., & Konjovic, Z. (2013). User interface of web application for searching PhD dissertations of the University of Novi Sad. In *Intelligent Systems and Informatics (SISY)*, 2013 IEEE 11th International Symposium on (pp. 117-122). IEEE.
- [33] Ivanović, L. (2013). Search of catalogues of theses and dissertations. *Novi Sad Journal of Mathematics - NS JOM*, Vol. 43, No. 1, pp. 155-165.
- [34] Zarić, M. (2013). Model za distribuirano i rangirano pretraživanje u bibliotečkim informacionim sistemima, doktorska disertacija, Fakultet tehničkih nauka u Novom Sadu.

-
- [35] IEC 82045-2:2004, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=34513
 - [36] ISO 82045-5:2005, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=34952
 - [37] Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval. New York: ACM press, Vol. 463.
 - [38] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge: Cambridge university press.
 - [39] Information Retrieval, Department of Computer Science & Engineering at the University of Ioannina, Greece, <http://www.cs.uoi.gr/pitoura/courses/ir/ir-intro09.pdf>
 - [40] Sparck Jones, K., Walker, S., & Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management*, Vol. 36, No. 6, pp. 779-808.
 - [41] Croft, W. B., & Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance information. *Journal of documentation*, Vol. 35, No. 4, pp. 285-295.
 - [42] Krovetz, B. (1995). Word sense disambiguation for large text databases. PhD Thesis. Department of Computer Science, University of Massachusetts Amherst.
 - [43] Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, Vol. 14, No. 3, pp. 130-137.
 - [44] Milosavljević, B. (2003). Proširivi sistem za pronalaženje multimedijalnih dokumenata. Fakultet tehničkih nauka u Novom Sadu
 - [45] Gostojić, S. (2012). Kreiranje i korišćenje digitalnih dokumenata pravne regulative. (Doktorska disertacija), Fakultet tehničkih nauka u Novom Sadu

- [46] Williams, H. E., Zobel, J., & Bahle, D. (2004). Fast phrase querying with combined indexes. *ACM Transactions on Information Systems (TOIS)*, Vol. 22, No. 4, pp. 573-594.
- [47] Fetterly, D., Manasse, M., Najork, M., & Wiener, J. (2003). A large-scale study of the evolution of web pages. In *Proceedings of the 12th international conference on World Wide Web*, pp. 669-678.
- [48] Luo, S. (2012), Web Search (I). Web Information Retrieval and Management (CS490WIR), Department of Computer Science, Purdue University. https://www.cs.purdue.edu/homes/lsi/CS490W_Fall_2012/slides/WIS_Web_Search_I.pdf
- [49] Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems*, Vol. 29, No. 8, pp. 1157-1166.
- [50] Fingerprint (computing), [http://en.wikipedia.org/wiki/Fingerprint_\(computing\)](http://en.wikipedia.org/wiki/Fingerprint_(computing))
- [51] Mercator URL frontier scheme - <http://nlp.stanford.edu/IR-book/html/htmledition/the-url-frontier-1.html>
- [52] Easley, D., Kleinberg, J. (2010). *Networks, Crowds, and Markets*. Cambridge University, www.cs.cornell.edu/home/kleinber/networks-book/
- [53] Boldi, P., & Vigna, S. (2004). The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pp. 595-602.
- [54] Dyrud, M. A. (2014). Predatory Online Technical Journals: A Question of Ethics. www.asee.org/file_server/papers/attachment/file/0004/4924/ASEE_2014_predatory.pdf
- [55] Nigritude ultramarine SEO competition - <http://www.sim64.co.uk/uk/nigritude-ultramarine/>

-
- [56] Adversarial IR: the unending (technical) battle between SEO's and web search engines research - <http://airweb.cse.lehigh.edu/>
 - [57] Google Search Engine Optimization, https://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en//webmasters/docs/search-engine-optimization-starter-guide.pdf
 - [58] Kovačević, A. (2006). Adaptivni sistem za pretraživanje zvučnih zapisa, Fakultet tehničkih nauka u Novom Sadu
 - [59] Novi Sad Talking android aplikacija - <https://play.google.com/store/apps/details?id=hub.nstalking.com&hl=en>
 - [60] Kass, M., Witkin, A. & Terzopoulos, D. (1988). Snakes: Active contour Models. International Journal of Computer Vision, Vol. 1, pp. 321 - 331.
 - [61] Xu, C. & Prince, J.L. (1998). Snakes, Shapes, and Gradient Vector Flow. IEEE Transactions on Image Processing, pp. 359-369.
 - [62] Kauppinen, H. et.al. (1995). An Experimental Comparison of Autoregressive and Fourier Based Descriptors in 2D Shape Classification. IEEE Transactions on Pattern Recognition and Machine Intelligence, Vol. 17 No. 2.
 - [63] LIRE - Lucene Image REtrieval - www.semanticmetadata.net/lire/
 - [64] INEX - <https://inex.mmc.uni-saarland.de/>

Indeks korišćenih pojmoveva

- A/B testiranje, 194
alternativni modeli pretraživanja, 56
blizinska pretraga, 90
brojačka matrica, 94
Bulov model, 57, 74
centralizovan model kolekcije, 19
clickthrough, 194
crawler, 138
cross-language information retrieval, 62
data retrieval, 50
digitalna biblioteka, 25
 - DSpace, 25
 - EPrints, 26
 - OpenDLT, 27distribuirani model kolekcije, 19
dokument, 2
 - digitalni dokument, 2
 - papirni dokument, 2dugotrajno skladištenje, 22
dvorečni indeks, 88
 - proširene dvoreči, 88F mera, 188
frekvencija dokumenta, 96
frekvencija kolekcije, 97
frekvencija terma, 95
globalno proširanje upita, 214
graf veba, 144
hipermedija, 160
HITS, 150
 - authority, 150
 - hub, 150idf, 96
indeksiranje, 50
informaciona potreba, 130
information retrieval, 20, 50
institucionalni repozitorijum, 27
OpenAIRE, 28
invertovani indeks, 77
ISO IEC 82045 standard, 32
kapa mera, 195
klasični modeli pretraživanja, 56
klasifikacija, 202
klasterovanje, 204
konekcioni server, 147
laboratorijske studije ponašanja korisnika, 194
long-term preservation, 22

- matrica incidencije, 77
metapodaci, 4
 formati metapodataka, 33
 Dublin Core, 36, 39
 MARC 21, 34
multilingual information retrieval, 62
multimedija, 160
navigaciona potreba, 131
normalizacija, 67
 dijakritici, 67
 lematizacija, 69
 Snowball, 72
 soundex algoritam, 69
 steming, 70
 Porterov steming algoritam, 73
 steming zasnovan na algoritmu, 70
 steming zasnovan na rečniku, 72
 stop reči, 68
 velika slova, 68
ocena relevantnosti, 93
page score, 149
pagerank, 149
pointeri za preskakanje, 86
poslovni sistem za upravljanje sadržajem, 24
 Alfresco, 24
povrat, 187
pozicioni indeks, 89
preciznost, 187
pretprocesiranje teksta, 63
pretraživanje, 50
pretraživanje kolekcije slika, 163
 boja, 167
 globalne osobine, 166
 informacije o sadržaju slike, 165
 lokalne osobine, 166
 oblik, 169
 segmentacija slike, 166
 tekstura, 173
pretraga kolekcije multimedijalnih sadržaja, 160
 content based, 162
 text based, 162
pretraga kolekcije tekstualnih digitalnih dokumenata, 62
 normalizacija
 rečnici sinonima, 69
pretraga kolekcije video zapisa, 178
pretraga kolekcije zvučnih zapisu, 175
pretraga po parametrima i zonama, 107
pretraga strukturiranih tekstuálnih sadržaja, 109
 data-centric XML, 110
 document-centric XML, 110
pretraga strukturiranih tekstuálnih sadržaja, 106
pretraga veba, 128
 spam, 153
Probabilistički model, 58
pronalaženje informacija, 20, 50
pronalaženje podataka, 50
protokol, 41
 OAI-PMH, 42
 razmena podataka, 41

- SRU, 45
udaljeno pretraživanje, 44
Z39.50, 45
- reč, 63
relevance feedback, 206
pseudo relevance feedback,
213
relevantnost, 185
rezultati pretrage, 199
dinamički sažetak, 200
statički sažetak, 199
- Search Engine Optimization, 153
sistemi za upravljanje dokumen-
tima, 18
arhiviranje, 22
distribucija, 23
katalogizacija, 19
oporavak od katastrofe, 21
backup, 21
recovery, 22
pretraživanje, 20
skladištenje dokumenata, 19
upravljanje poslovnim pro-
cesom, 23
zaštita podataka, 20
- sistemi za upravljanje dokumet-
nima
distribucija, 10
- slika, 163
rasterska grafika, 163
vektorska grafika, 163
- standardne poslovne aplikacije,
18
strukturni term, 116
- tačnost, 187
- težinska matrica, 98
tekst linka, 146
tekstualni digitalni dokument,
62
term, 62
tezaurus, 214
automatski generisan teza-
urus, 215
ručni tezaurus, 215
tf-idf, 97
token, 63
tokenizacija, 64
transakciona potreba, 131
upiti fraze, 87
upravljanje digitalnim dokumen-
tima, 8
upravljanje digitalnim dokumen-
tima, 18
upravljanje verzijama dokumenta,
11
konkurentno važenje verzija,
12
period formiranja, 11
period važenja, 11
sekvencijalno važenje verzija,
12
- veb pretraživač, 128, 135
vektor incidencije, 77
Vektorski model, 57, 92
video, 177
- zadovoljstvo korisnika, 184
zivotni ciklus dokumenta, 7
arhiviranje, 13
inicijalizacija, 8
korišćenje, 10

- priprema, 9
- revizija, 11
- uklanjanje, 13
- uspostavljanje, 10
- zvuk, 174