Osnove pronalaženja informacija

Dr Branislava B. Šandrih Todorović

Beleške sa nastave

Sadržaj

Čuvanje i pronalaženje informacija

1. Uvod

1.1 Podatak ili informacija

Zadatak ove knjige je da čitaoca upozna sa osnovnim konceptima pronalaženja informacija. Pre nego što se upustimo u zadatak koji je pred nama, prvo treba da se zapitamo zašto pronalazimo baš *informacije*, a ne *podatke*? Ponekad čujemo da govornici ova dva termina koriste misleći na isti pojam, ali takva upotreba nije precizna. Ipak, iako se radi o dva potpuno drugačija koncepta, oni jedno bez drugog ne mogu. U narednim pasusima objasnićemo ukratko šta su podaci, a šta informacije, kao i kakva je to čvrsta međusobna povezanost između njih.

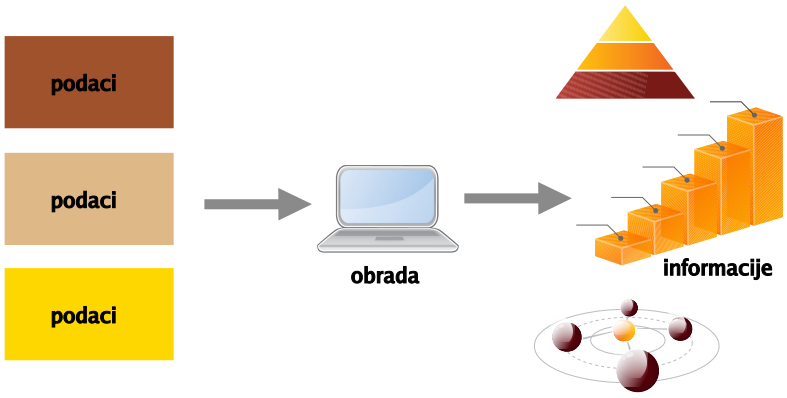
Podaci su sirove, uglavnom neorganizovane činjenice, koje za bilo kakvo dalje tumačenje iziskuju obradu. Takve činjenice mogu biti predstavljene u vidu iskaza, brojeva, karaktera, simbola, slika, itd. Mogu biti i struktuirani, u vidu tabela, grafova ili drveta, pa čak i teksta. U načelu, podaci se tumače od strane čoveka ili računara, kako bi se iz njih moglo izvesti nekakvo značenje. Podaci su, sami po sebi, beskorisni bez konteksta; *de facto* postoje, nezavisno od informacija.

Z Etimološki, reč potiče od latinske reči *datum*, što predstavlja **nešto dato**, dok je odgovarajuća reč u množini *data*. Uprkos tome što predstavlja oblik množine, na engleskom se reč **data** uvek koristi kao imenica u jednini. Na primer, rekli bismo *Data is displayed in the following table*.

Informacije predstavljaju izlaz nakon obrade podataka (videti sliku 1.1). Za razliku od podataka, informacije su obrađene, struktuirane ili predstavljene u odgovarajućem kontekstu, tako da imaju značaj i upotrebnu vrednost.

Z Etimološki, reč potiče od latinskog glagola *informare*, koji predstavlja **davanje forme** ili **oblikovanje**. Odgovarajuća reč na engleskom, **information**, u ovom slučaju je *singularia tantum*, dakle uvek je u jednini. Stoga, rekli bismo *Information is derived from the data*, čak i ako se odnosi na više raznih informacija.

Informacije se mogu izraziti u vidu izjava, ideja, zaključaka, opisa i slično. Predstavljaju se najčešće nekim značećim jedinicama, kao što su vreme ili količina. Informacije *daju značaj* podacima, te je njihovo postojanje uvek zavisno od njih. Sadrže kontekst, relevantnost i svrhu.



Slika 1.1: Informacije su rezultat obrade podataka.

■ **Primer 1.1** Neka su dati sledeći brojevi: 29.7, 29.8, 30.2, 31.3, 28.5, 25.7, 26.9, 33.4, 36.7, 35.6. Ovi brojevi sami po sebi nemaju nikakvo značenje, zato što uz njih nije dat nikakav kontekst. Međutim, ukoliko bismo iste brojeve predstavili sa sve mernom jedinicom, bilo bi jasnije da se radi izmerenim temperaturama u nekim razmacima: 29.7°, 29.8°, 30.2°, 31.3°, 28.5°, 25.7°, 26.9°, 33.4°, 36.7°, 35.6°. Ukoliko bi izmerene temperature predstavljale, na primer, prosečne dnevne temperature u prvoj polovini avgusta, izmerene u periodu od 10 dana, mogli bismo da zaključimo da je prosečna dnevna temperatura u ovom periodu bila oko 30.8°, što je još jedna informacija. ■

1.2 Rudarenje podataka

Iako su razni podaci svugde oko nas, u ovoj knjizi ograničićemo se na one koji se zapisuju u računaru. Postavlja se pitanje koliko ima raspoloživih podataka? Odgovor na ovo pitanje je u najmanju ruku fascinantan.

Erik Šmit (orig. Eric Emerson Schmidt), izvršni direktor kompanije Google duže od decenije, 2010. godine pokušao je da odgovori na ovo pitanje. Njegov odgovor glasio je otprilike ovako: ako bismo mogli da izmerimo količinu podataka koju smo kao čovečanstvo napravili od početka civilizacije, pa sve do 2003. godine – to je količina podataka koju danas stvorimo za dva dana! Iako je ova informacija sada zastarela, pouzdano svedoči o eksponencijalnom rastu podataka na internetu. Teško je reći koliko zaista podataka ima na vebu danas, na primer izraženo u bajtovima, ali čak i kad bismo mogli da uhvatimo tako promenljiv podatak u nekom trenutku, pitanje je koliko bismo bili sposobni da ga zaista i razumemo.

Slično kao što su pojmovi podatak i informacija, pojmovi *internet* i *veb* u svakodnevnom govoru odnose se na isti pojam. Ipak, ne treba mešati ove dve stvari: **internet** predstavlja samu infrastrukturu mreže, dok su **veb** podaci koji se skladište i razmenjuju preko te mreže.

Prema kompaniji Cisco, koja se bavi razvojem, proizvodnjom i prodajom mrežne infrastrukture, 2019. godine podaci na internetu zauzimaju između 30–40 zetabajtova. Jedan zetabajt je 10^{21}

bajtova, odnosno 1 000 000 000 000 000 000 000 bajtova. Na papiru ove nule ne znače puno, ali pokušajmo da napravimo analogiju kojom bismo mogli da bolje percepiramo grandioznost ovog broja.

Primera radi, neka video snimak koji želimo da preuzmemo sa veba iznosi 1 zetabajt. Koliko vremena je potrebno da naš računar učita takav snimak, pod pretpostavkom da imamo najveću brzinu internet veze koja danas postoji (100 Gbps)? Odgovor je: 2,535 godina!

Još jedno zanimljivo pitanje koje radoznalog čitaoca može nasvesti na razmišljanje je sledeće: koliko je veb *težak*? Imaju li podaci uopšte masu? Odgovor na drugo pitanje je potvrdan, a na prvo još impresivniji: ceo veb teži koliko i jedna jagoda! Odogovor o masi veba daje nam osnovna fizika. Podaci se u računaru zapisuju kao sekvence nula i jedinica. Fizički, jedinica se realizuje protokom elektrona kroz tranzistor, što je osnovna fizička gradivna komponenta računara. S obzirom na to da elektroni imaju masu, iako infinitezimalnu, dolazimo do zaključka da određen deo mase našeg računara čini i njegov operativni sistem, aplikacije koje koristi, kao i naše slike, muzika, filmovi, itd. Ilustracije radi, profesor sa Berklija Džon Kubiatovicz (orig. John Kubiatowicz) procenio je da se masa kindl uređaja[1] uveća za 10^{-18} grama kada je iskorišćen maksimalni memorijski kapacitet. Kada bismo ovaj broj raspisali, to je 0.000000000000000001 grama!

Možda je najpreciznije reći da je odgovor o veličini interneta *beskonačnost*. Naše poimanje o beskonačnosti je kompleksno; beskonačnost je nedostižna, nezamisliva, ponekad zastrašujuća. Ipak, baš ona pruža radoznalim korisnicima neograničen broj mogućnosti za analizu i istraživanje. Postoji ogromna praznina između trenda rasta podataka i našeg razumevanja istih. Kada tek malo zagrebemo na površini veba, pred nama se otvaraju vrli novi horizonti informacija, pa i znanja. Da bismo makar pokušali da ispratimo korak rasta podataka, neophodno je da primenimo računarsku obradu kako bismo ekstrahovali različite informacije. Upravo to je zadatak discipline *rudarenja podataka*. Onaj koji rudari pred sobom ima čitavo prirodno bogatstvo – veb. Tradicionalni rudar kao alat koristi kramp, dok savremeni rudar ima računar. Ishod rudarenja su informacije, a skupovi informacija organizovani su u još veću jedinicu – *znanje*. Drugi naziv za rudarenje podataka je i *otkrivanje znanja iz podataka*.

U engleskoj literaturi, odgovarajući termin za rudarenje podataka je *data mining*.

Zadatak rudarenja podataka je multi-disciplinaran, što znači da je usko povezan sa drugim naučnim oblastima, kao što su statistika, mašinsko učenje, pronalaženje informacija, distribuirano programiranje, pa čak i sa sistemima baza podataka. Stručnjaci u ovim različitim oblastima prilaze rudarenju podataka na različite načine; na primer, statističare će zanimati trend rasta korisnika veba, dok je za stručnjake za baze podataka glavni zadatak optimizacija organizacije skladištenja. U ovoj knjizi, zauzećemo ugao gledišta jednog pronalazača informacija.

U engleskoj literaturi, odgovarajući termin za pronalaženje informacija je *information retrieval*.

1.3 Struktuirani i nestruktuirani podaci

Da bismo objasnili šta je osnovni zadatak naučne oblasti pronalaženja informacija, neophodno je da prvo objasnimo razliku između struktuiranih i nestruktuiranih podataka. Upravo od (ne)struktuiranosti

1. Kindl uređaj (engl. kindle) je mini računar koji se koristi isključivo za čitanje elektronskih tekstova.

podataka zavisi i vrsta informacija koja se može dobiti.

Uvedimo prvo nekoliko pojmova:

Informacioni izvor Skladište velikog broja dokumenata (tabela, teksta, slika, multimedijalnih dokumenata, i sl.) na nekom računaru; kaže se još i *sistem dokumenata*;

Informaciona potreba Kriterijum prema kom korisnik želi da dobije određene dokumente iz pomenutog skladišta;

Relevantnost Skladišteni dokumenti koja odgovaraju datom kriterijumu;

Mašina za pronalaženje Računarski program koji pronalazi relevantne dokumente na osnovu datih kriterijuma;

Upit Zahtev kojim korisnik formuliše svoju informacionu potrebu mašini za pronalaženje.

Pitanje 1.1 Šta je potrebno da korisnik uradi, kako bi zadovoljio svoju *informacionu potrebu*, tako da iz *informacionog izvora* dobije samo ona dokumenta koja su *relevantna* za njegove potrebe? U ovakvoj situaciji, korisnik će svoju informacionu potrebu izraziti odgovarajućim *upitom* koji će uputiti *mašini za pronalaženje*.

Iako ćemo sve ove pojmove detaljnije obraditi u narednim odeljcima, sada ćemo se najviše zadržati na tome šta je **upit** i kako se on može formulisati. Prvo, da bi korisnik formulisao svoj zahtev, potrebno je da ima predstavu o tome kakva je struktura skladištenih dokumenata. Radi ilustracije, neka jednostavni informacioni izvor sadrži samo tri dokumenta, tabele 1.1 i 1.2 i 1.3.

Tabela 1.1: Tabela studenata

Puno ime	**Broj indeksa**	**Status**
Jovan Todorović	20220456	budžet
Milena Aleksić	20100231	samofinansirajući
Dragana Stanković	20150642	budžet

Tabela 1.2: Tabela predmeta

Šifra predmeta	**Naziv predmeta**	**Izbornost**
PJT12	Jezičke tehnologije	izborni
PBIB5	Bibliotekarstvo	obavezni
PPI56	Pronalaženje informacija	obavezni

Tabela 1.3: Tabela studenata i predmeta

Šifra predmeta	**Broj indeksa studenta**	**Slušanje**
PJT12	20220456	1
PJT12	20150642	2
PBIB5	20220456	2
PBIB5	20100231	1
PPI56	20150642	3

Ovakav sistem dokumenata sadrži **struktuirane podatke**, zato što je značenje nekog polja potpuno određeno pripadnosti odgovarajućoj koloni. Konkretno, radi se o relacionoj bazi podataka.Kod ovakvih sistema, sadržaj upita zavisi od struktura tabela i veza (relacija) među njima. Na primer, tabela 1.1 sadrži podatke o troje studenata: njihovo puno ime, broj indeksa i status studija. Tabela 1.2

čuva tri sloga koja se odnose na predmete, date svojim nazivom, šifrom i statusom izbornosti. Na kraju, tabela 1.3 je tabela koja povezuje studente i predmete, te se vidi da, primera radi, student sa indeksom 20220456 pohađa predmet sa šifrom PJT12, i to prvi put, dok je student sa brojem indeksa 20150642 isti predmet već jednom pohađao. Iz tabele studenata vidimo da se radi redom o studentima Todorović Jovanu i Stanković Dragani, a da je u pitanju predmet Jezičke tehnologije.

Upiti koje korisnik može postaviti nad ovakvim sistemom dokumenata mogu, na primer, biti:

- Koja su puna imena studenata koji su na budžetu?
- Koje su šifre izbornih predmeta?
- Koliko je samofinansirajućih studenata koji slušaju predmet Bibliotekarstvo po prvi put?

Iako ovakvi sistemi dokumenata nose sa sobom mnoštvo izazova (optimizacija fizičke organizacije tabela, brzina izvršavanja upita, redukcija redudantnosti i mnogi drugi), oni nisu od interesa u ovoj knjizi. U ovoj knjizi bavićemo se sistemima čiji je zadatak da zadovolje informacionu potrebu korisnika koja se odnosi na **nestruktuirane** sisteme podataka, poput veb stranica ili PDF dokumenata, kao što je ilustrovano u sledećem primeru.

■ **Primer 1.2** Neka sistem dokumenata sadrži sledeća dva tekstualna dokumenta (označimo ih redom D_1 i D_2, s leva na desno):

Skladištena dokumenta koja odgovaraju informacionoj potrebi smatraju se relevantnim. Mašina za pronalaženje je računarski program koji pronalazi relevantne dokumente na osnovu datih kriterijuma.	Upit je zahtev kojim korisnik formuliše svoju informacionu potrebu mašini za pronalaženje. Kod nestruktuiranih podataka, upiti mogu biti logički i vektorski, što je sledeća tema.

Upiti koje korisnik može postaviti nad ovakvom kolekcijom imaju donekle ograničene mogućnosti, poput:

- Vrati one dokumente koji sadrže termin *pronalaženje*;
- Vrati one dokumente koje sadrže istovremeno termine *pronalaženje* i *upit*;
- Vrati one dokumente koji sadrže termine *mašina* ili *mašini*, a ne sadrže termin *korisnik*.

Kako je u pitanju kolekcija koja sadrži samo dva dokumenta, moguće je vrlo jednostavno dati odgovore na postavljena pitanja. Kako oba dokumenta sadrže termin *pronalaženje* (uokvireno, videti ispod), oba bi bila sadržana i u odgovoru.

Skladištena dokumenta koja odgovaraju informacionoj potrebi smatraju se relevantnim. Mašina za pronalaženje je računarski program koji pronalazi relevantne dokumente na osnovu datih kriterijuma.	Upit je zahtev kojim korisnik formuliše svoju informacionu potrebu mašini za pronalaženje. Kod nestruktuiranih podataka, upiti mogu biti logički i vektorski, što je sledeća tema.

U slučaju dokumenata koji istovremeno sadrže i termin *pronalaženje* i termin *upit*, u odgovoru bi se našao samo dokument D_2.

Upit je zahtev kojim korisnik formuliše svoju informacionu potrebu mašini za pronalaženje.

Kod nestruktuiranih podataka, upit i mogu biti logički i vektorski, što je sledeća tema.

Što se tiče dokumenata koji sadrže neki od oblika termina *mašina*, a ne sadrže termin *korisnik*, odgovarajući dokument je samo D_1.

Skladištena dokumenta koja odgovaraju informacionoj potrebi smatraju se relevantnim. Mašina za pronalaženje je računarski program koji pronalazi relevantne dokumente na osnovu datih kriterijuma.	Upit je zahtev kojim **korisnik** formuliše svoju informacionu potrebu mašini za pronalaženje. Kod nestruktuiranih podataka, upiti mogu biti logički i vektorski, što je sledeća tema.

■

Ipak, ne postoje potpuno nestruktuirani podaci. Čak i sadržaj veb stranice ima izvesnu strukturu: naslov, zaglavlja, pasuse; pasusi se, pak, sastoje iz rečenica. Stoga je moguće ovakve upite i proširiti, na primer:

- Vrati one dokumente koji u naslovu sadrže termin *pronalaženje*;
- Vrati one dokumente koji u nekoj rečenici sadrže istovremeno i termin *pronalaženje* i termin *upit*, ali tako da postoji najviše pet drugih reči između njih;
- Vrati one dokumente koji ne sadrže termine *mašina* i *korisnik* u istom pasusu.

Stoga, preciznije je reći da je tekstualni dokument ipak **delimično-struktuirani** tip podatka. Prva nevolja leži već u iole većim sistemima dokumenata: da li to znači da je potrebno, prilikom svakog upita, pročitati sve dokumente kolekcije i ispitivati pripadnost, odnosno odsutnost termina navedenih u upitu? Čak i za moderne računare, ovakav naivni pristup daleko je od efikasnog.

U nastavku knjige bavićemo se isključivo polu-struktuiranim sistemima dokumenata. Diskutovaćemo razne izazove koji oni nose, ali i postojeće tehnike za izgradnju brzih i efikasnih mašina za pronalaženje informacija kakve danas poznajemo. Na osnovu datih primera, možemo zaključiti da su glavni zadaci ovakvih sistema za pronalaženje informacija sledeći:

- analiza sadržaja sistema dokumenata,
- pronalazak odgovarajuće reprezentacije dokumenata, tako da se pripadnost, odnosno odsutnost termina može efikasno utvrditi,
- razumevanje korisničkih upita,
- identifikacija relevantnih informacionih izvora i
- pronalaženje relevantnih informacija.

2. Invertovani indeksi dokumenata

2.1 Izgradnja invertovanog indeksa dokumenata

Neka sistem dokumenata čini sledeći podskup diskografije muzičke grupe Bijelo Dugme (radi lakšeg referisanja u daljem tekstu, dodelićemo svakoj pesmi kraću oznaku, $D_1, \ldots, D_7$):

D1 Ipak poželim neko pismo;
D2 Loše vino;
D3 Bitanga i princeza;
D4 U vrijeme otkazanih letova;
D5 Evo zakleću se;
D6 Sanjao sam noćas da te nemam;
D7 Lipe cvatu.

Neka je informaciona potreba sledeća: želimo da nađemo sve tekstove koje sadrže reči *kraj* i *pjesma*, ali ne sadrže termin *noćas*. Ovakav zahtev predstavlja korisnički upit. Naivni pristup bi bio sledeći: iščitali bismo svaki tekst posebno, tražeći ove termine. Ukoliko tekst ispunjava postavljeni kriterijum, onda on zadovoljava i našu informacionu potrebu. Ovakav pristup je intuitivan i potpuno izvodljiv za tako skromnu kolekciju kratkih tekstova. Šta se, međutim, dešava kada kolekciju čine hiljade ili desetine hiljada tekstova, sa nekoliko desetina hiljada reči? Ovakvi informacioni izvori se uglavnom i sreću u praksi. U takvoj situaciji, ponovno čitanje dugačkih tekstova iznova i iznova, za svaki upit, daleko je od efikasnog. Resurs koji se u takvoj prilici najviše troši je *vreme* – a korisnici su najmanje tolerantni kada je utrošak vremena u pitanju. Danas želimo sisteme koji daju trenutni odziv, a svako čekanje predstavlja potencijalni razlog da korisnik odustane od upotrebe i potraži bolje rešenje. Stoga je potrebno potražiti efikasnije tehnike.

Izgradnja matrice dokumenata

Prvi korak ka rešenju bio bi izgradnja matrice pripadnosti termina dokumentima. Svaki dokument iz kolekcije (u ovom slučaju, tekst pesme) pre stavljanja na raspolaganje korisnicima, potrebno je

unapred obraditi. Obrada podrazumeva iščitavanje teksta i pravljenje skupa svih različitih termina od interesa koji se javljaju u tekstu; ovaj korak je potom potrebno ponoviti za svaki dokument kolekcije, beležeći na isti način termine koji se pojavljuju. Zatim se konstriše matrica: kolone matrice čine ranije zabeleženi termini (tako da se ne ponavljaju), dok matrica ima onoliko vrsta koliko ima dokumenata u kolekciji. Za svaki dokument, vrednost odgovarajućeg polja je 1, ukoliko se termin predstavljen tom kolonom pojavljuje bar jednom u dokumentu; inače, vrednost polja je 0.

Na primer, u prvom tekstu D_1 javljaju se termini *noćas* i *pjesma*, te se u odgovarajućim kolonama beleže jedinice, a u ostalim kolonama su nule. Slično, u drugoj pesmi D_2, javljaju se termini *noćas* i *srce*, ali ne i termini *grad*, *pjesma* i *kraj*. Ovakva matrica naziva se **matricom dokumenata**, a njen izgled dat je u tabeli 2.1.

Tabela 2.1: Matrica dokumenata

	noćas	grad	pjesma	srce	kraj
D_1	1	0	1	0	1
D_2	1	0	0	1	0
D_3	0	1	0	0	0
D_4	0	1	0	0	0
D_5	0	0	1	0	1
D_6	0	0	1	0	0
D_7	0	0	0	1	0

Kako bi odgovaranje na upit izgledalo sada? Bilo bi za nijansu efikasnije. Ponovo, za svaki dokument (odnosno za svaki red matrice), proveravale bi se vrednosti polja u odgovarajućim kolonama. Podsetimo se, upit je glasio: pronaći sve tekstove koje sadrže reči *kraj* i *pjesma*, ali ne sadrže reč *noćas*. U slučaju teksta D_1, kriterijum nije ispunjen, jer tekst sadrži sva tri termina (jedinice u kolonama *noćas*, *kraj* i *pjesma*); za drugi tekst, prvi deo upita nije ispunjen (ne sadrži ni reč *kraj* ni reč *pjesma*), itd. Konačno, dolazimo da jedini tekst kolekcije koji ispunjava uslov je dokument D_5.

Šta smo dobili, a šta izgubili ovim pristupom? Prvo, pristup jeste nešto brži, u smislu da se sada ne iščitavaju celoviti tekstovi, već samo odabrane, značeće reči koje mu pripadaju. Dakle, u neku ruku, dobitak je na brzini. Da li je brzina odziva zadovoljavajuća, odnosno trenutna? Ipak ne odgovor odričan u opštem slučaju, kada se radi o kolekcijama iz prakse.

Šta se gubi? Vreme uloženu za obradu unapred (tzv. za pre-procesiranje) se može zanemariti, zato što se takva obrada obavi jednom, a dobici koje ona pruža se uživaju neograničeno vremena kasnije. Jedini, uslovno rečeno, gubitak, jeste memorijski prostor potreban za smeštanje matrice dokumenata. Slično kao i za vreme pre-procesiranja, ne smatra se gubitkom u pravom smislu, već prvenstveno ulaganjem. Štaviše, u praksi je poželjnije uložiti u memorijski prostor, po cenu efikasnosti, nego obrnuto. Svakako, svi računarski algoritmi teže da budu štedljivi, kako po pitanju brzine izvršavanja, tako i po pitanju korišćenja memorije za skladištenje.

Kako bi onda moglo bolje? Prva intervencija bila bi u vidu transponovanja matrice, kada kolone postaju vrste, a vrste kolone. Tada bi matrica imala onoliko vrsta koliko je bilo ukupno različitih termina, a broj dokumenata predstavljao bi broj kolona. Transponovanu matricu pogledajmo kroz tabelu 2.2. Ako je prva vrsta (koja se odnosila na D_1) redom sadržala vrednosti $1,0,1,0,1$, sada će to biti vrednosti prve kolone, slično i za ostale vrste i kolone.

Tabela 2.2: Invertovana matrica dokumenata

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
noćas	1	1	0	0	0	0	0
grad	0	0	1	1	0	0	0
pjesma	1	0	0	0	1	1	0
srce	0	1	0	0	0	0	1
kraj	1	0	0	0	1	0	0

Kako bi sada izgledalo izvršavanje upita? Posmatrajmo samo vrste za termine *kraj*, *pjesma* i *noćas*:

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
kraj	1	0	0	0	1	0	0
pjesma	1	0	0	0	1	1	0
noćas	1	1	0	0	0	0	0

Tražimo kolone u kojima su vrednosti redom 1, 1, 0 (prisutnost prva dva termina, a odsustvo trećeg). Do odgovora sada dolazimo znatno brže (to je tekst D_5).

Štaviše, dovoljno je izvršiti po kolonama sledeće logičke operacije:

$$(\text{kraj} \wedge \text{pjesma}) \wedge \neg(\text{noćas})$$

Tada bi, redom (kraj $\wedge$ pjesma) bilo:

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
kraj	1	0	0	0	1	0	0
pjesma	1	0	0	0	1	1	0
$a = \text{kraj} \wedge \text{pjesma}$	1	0	0	0	1	0	0

Označimo dobijeni međurezultat sa a. Komplementirajmo zatim vrstu za termin noćas (operacija negacija):

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
noćas	1	1	0	0	0	0	0
$b = \neg$ noćas	0	0	1	1	1	1	0

Označimo dobijeni međurezultat sa b. Uradimo na kraju konjukciju $a \wedge b$:

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
a	1	0	0	0	1	0	0
b	0	0	1	1	1	1	0
$a \wedge b$	0	0	0	0	1	0	0

Konačno, vidimo da je vrednost 1 upravo u koloni koja odgovara dokumentu D_5. Bitovske operacije su vrlo efikasne i njihovo izvršavanje je trenutno.

Iako deluje da je rešenje sada tu, uvek se postavlja pitanje može li bolje. Prva, sasvim sitna modifikacija, ali sa ogromnim pozitivnim učinkom jeste **sortiranje termina**. Podsetimo se, ukoliko niz elemenata nije sortiran, za pretragu elementa u tom nizu možemo koristiti samo linearnu pretragu, koja nije efikasna. Na primer, ako se traženi element nalazi na kraju niza, potrebno je ispitati sve termine u nizu, što dovodi do vraćanja na početak. Dakle, prilikom smeštanja termina u matricu, potrebno je sortirati ih, kako bi se prilikom ispitivanja termina mogao primeniti algoritam *binarne pretrage*. Pogledajmo kako bi izgledala invertovana matrica dokumenata sa sortiranim terminima u tabeli 2.3. Poredak među terminima je takozvani **leksikografski poredak**.

Tabela 2.3: Invertovana matrica dokumenata gde su termini sortirani leksikografski

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
grad	0	0	1	1	0	0	0
kraj	1	0	0	0	1	0	0
noćas	1	1	0	0	0	0	0
pjesma	1	0	0	0	1	1	0
srce	0	1	0	0	0	0	1

Podsetimo se leksikografskog poretka. Reč a manja je od reči b ($a < b$) ukoliko se na prvoj poziciji na kojoj se ove reči razlikuju, u reči a nalazi slovo koje je leksikografski manje od slova na istoj poziciji u reči b.

Primera radi, pro c editi < pro k uvati, jer su slova C i K prva slova po kojima se ove reči razlikuju, a u abecedi se slovo C nalazi pre slova K.

Evo i poretka za sva slova srpske abecede:

A < B < C < Č < Ć < D < DŽ < Đ < E < F < G < H < I < J < K < L < LJ
< M < N < NJ < O < P < R < S < Š < T < U < V < Z < Ž

Konačno, ono što veštom oku možda upada u oko je tzv. *retkost matrice*. I u posmatranom primeru, a posebno u praksi, ovakve matrice sadrže mnogo više nula nego jedinica, te se za takve matrice kaže da su retke. One nepotrebno zauzimaju dosta prostora i moguće je prevazići ovu manu upotrebom druge strukture podataka za čuvanje informacija o pripadnosti termina dokumentima. Kako nam je bitno da znamo u kojim dokumentima ja reč sadržana, možemo svakom terminu dodeliti *listu identifikatora dokumenata* u kojima se nalazi.

Elem, dokument se nalazi negde u memoriji, a sistem ima informaciju o tome. Terminu se onda dodeli lista, čiji su elementi samo oznake tih dokumenata, a ako je potrebno dokumentu i *pristupiti*, onda će se sistem naknadno pobrinuti za to. Na kraju dolazimo do optimalne reprezentacije, a to je **invertovani indeks dokumenata**. Za dati primer, invertovani indeks elemenata ima sledeći, jednostavan izgled:

grad $\rightarrow$ [D_3, D_4]

kraj $\rightarrow$ [D_1, D_5]

noćas $\rightarrow$ [D_1, D_2]

pjesma $\rightarrow$ [D_1, D_5, D_6]

srce $\rightarrow$ [D_2, D_7]

Kako je reč *grad* imala vrednost 1 u kolonoma koje su odgovarale dokumentima D_3 i D_4, pridružuje joj se lista [D_3, D_4], slično i za ostale termine.

Ovakav pristup je izuzetno memorijski štedljiv, ali se ne mogu primeniti bitovski operatori na način koji je iznad ilustrovan. Ipak, videćemo da postoji još efikasniji način da se izvršavaju najrazličitije vrste upita sa ovakvom reprezentacijom koja se oslanja na liste i operaciju spajanja listi.

Invertovano-indeksne operacije koje su neophodne da bi se dobili odgovori na upite zasnivaju se na procesu spajanja listi koji uzima dve ili više vrsta iz invertovane matrice termina i zapisa i proizvodi jednu kombinovanu listu identifikatora dokumenata.

Kada su identifikatori dokumenata iz invertovane matrice uređeni u rastućem redosledu identifikatora dokumenata, kakav je slučaj u ovom primeru (npr. za termin *pjesma*, D_1, D_5, D_6), onda se operacija spajanja lista može izvršiti jednim prolazom kroz ulazne liste. Sledećim pseudo-kodom opisan je algoritam operacije spajanja listi u kom se najmanji element, a to je dokument čiji je identifikator najmanji, redom prenosi u spojenu listu sve dok obe ulazne liste nisu iscrpljene.

Algoritam spajanja listi

ako je kraj obe liste **onda** kraj;
inače ako je kraj jedne liste
onda prebaci u izlaznu listu sve preostale elemente druge liste i završi;
inače Uzmi tekući element E_1^i iz prve liste i tekući element E_2^j iz druge liste
    ako je $E_1^i < E_2^j$
    onda prebaci E_1^i element u izlaznu listu
        element E_1^{i+1} postaje novi tekući element
        ponovi proces
    inače ako je $E_1^i > E_2^j$
    onda prebaci E_2^j element u izlaznu listu
        element E_2^{j+1} postaje novi tekući element
        ponovi proces
    inače prebaci E_1^i i E_2^j elemente u izlaznu listu
        element E_1^{i+1} postaje novi tekući element
        element E_2^{j+1} postaje novi tekući element
        ponovi proces

2.2 Logički upiti

Pogledajmo tri vrste logičkih upita koje možemo izvršavati nad invertovanim indeksima koji se oslanjaju na operacije spajanji listi, kao i kako se oni izvršavaju.

AND-upiti

Neka je postavljen sledeći upit:

$$T_1 \text{ AND } T_2$$

PRONAĆI SVE DOKUMENTE KOJI SADRŽE ISTOVREMENO TERMINE T_1 I T_2.

Algoritam je sledeći:

- Spojiti liste za termine T_1 i T_2, dodavanjem identifikatora dokumenata u rastućem poretku;
- Napraviti novu listu koja sadrži samo one dokumente koji se pojavljuju dva puta, a ostale dokumente izbaciti iz liste (odnosno, zadržati samo *duplikate*).

Zašto se zadržavaju baš duplikati, odnosno dokumenti koji se pojavljuju dva puta? Ako se u listi nastaloj spajanjem neki dokument javlja dva puta, to znači da se i termin T_1 nalazi u tom dokumentu (otud jedno pojavljivanje), ali i T_2 se javlja u tom dokumentu (stoga drugo pojavljivanje), što je upravo informaciona potreba u ovom slučaju.

■ **Primer 2.1** Posmatrajmo invertovani indeks iz uvodnog odeljka, a neka je upit *kraj* AND *pjesma*.

grad $\rightarrow$ [D_3, D_4]

kraj $\rightarrow$ [D_1, D_5]

noćas $\rightarrow$ [D_1, D_2]

pjesma $\rightarrow$ [D_1, D_5, D_6]

srce $\rightarrow$ [D_2, D_7]

Prvi korak je spajanje liste za ova dva termina (obeležimo operaciju spajanja sa $sp()$); ne zaboravimo, nova lista treba da sadrži deksriptore dokumenata sortirane u rastućem poretku.

$sp(kraj, pjesma)$ = [D_1, D_1, D_5, D_5, D_6]

Sada je dovoljno da prođemo kroz listu i da zadržimo samo one koji se ponavljaju, čime se dobija lista: [D_1, D_5].

■

OR-upiti

Neka je postavljen sledeći upit:

$$T_1 \text{ OR } T_2$$

PRONAĆI SVE DOKUMENTE KOJI SADRŽE T_1 ILI T_2.

Svi dokumenti koji sadrže bar jedan od ova dva termina, ali i dokumenti koji sadrže oba termina, ispunjavaju postavljeni kriterijum.

Algoritam je sledeći:

- Spojiti liste za termine T_1 i T_2, dodavanjem identifikatora dokumenata u rastućem poretku;
- Napraviti novu listu koja sadrži sve one dokumente koji se bar jednom pojavljuju u dobijenoj listi.

Zašto se zadržavaju baš svi dokumenti koji su se pojavili u listi nastaloj spajanjem? Ako se u listi neki dokument javlja dva puta, to znači da se i termin T_1 nalazi u tom dokumentu (otud jedno pojavljivanje), ali i T_2 se javlja u tom dokumentu (stoga drugo pojavljivanje), što odgovara postavljenom kriterijumu; ako se dokument pojavljuje samo jednom, to znači da se samo jedan od termina (ili T_1 ili T_2) javlja u tom dokumentu, što jednako zadovoljava informacionu potrebu.

■ **Primer 2.2** Posmatrajmo invertovani indeks iz uvodnog odeljka, a neka je upit *noćas* OR *pjesma*.

Prvi korak je spajanje liste za ova dva termina; ne zaboravimo, kao i ranije, nova lista treba da sadrži deksriptore dokumenata sortirane u rastućem poretku.

$sp(noćas, pjesma)$ = [D_1, D_1, D_2, D_5, D_6]

Sada je dovoljno da prođemo kroz listu i da zadržimo sve koji se javljaju bar jednom, čime se dobija lista: [D_1, D_2, D_5, D_6].

■

AND NOT-upiti

Neka je postavljen sledeći upit:

$$T_1 \text{ AND NOT } T_2$$

PRONAĆI SVE DOKUMENTE KOJI SADRŽE T_1 I NE SADRŽE T_2.

Svi dokumenti koji sadrže termin T_1 i ne sadrže T_2 ispunjavaju postavljeni kriterijum.

Algoritam je sledeći:

- Spojiti liste za termine T_1 i T_2, dodavanjem identifikatora dokumenata u rastućem poretku;
- Napraviti novu listu a koja sadrži samo one dokumente koji se pojavljuju dva puta, a ostale dokumente izbaciti iz liste;
- Napraviti novu listu b spajanjem liste a i liste za termin T_1;
- Rezultujuća lista sadrži samo one dokumente iz b koji se pojavljuju tačno jednom.

Zašto je sled koraka baš ovakav? Drugi korak, formiranje liste a, je u stvari rešavanje AND upita. Lista a u tom slučaju evidentira dokumente koji sadrže oba termina. Sledeći korak, formiranje liste b je bitan pomoćni korak. Kada spojimo listu a i listu dokumenata za termin T_1, ako se u toj dobijenoj listi neki dokument javlja dva puta, to znači da se u njemu pojavljuju i termin T_2 (poreklo vodi iz liste a) i termin T_1 (otud se dokument ponovio), što znači da taj dokument ne želimo u rezultatu.

■ **Primer 2.3** Posmatrajmo invertovani indeks iz uvodnog odeljka, a neka je upit *noćas* AND NOT *pjesma*.

Prvi korak je spajanje listi za ova dva termina:

$sp(noćas, pjesma)$ = [D_1, D_1, D_2, D_5, D_6]

Onda formiramo listu a prolazeći kroz listu i da zadržavajući sve dokumente koji se javljaju tačno dva puta, čime se dobija lista: a = [D_1].

Potom spajamo dobijenu listu a sa listom dokumenata za prvi termin:

$b = sp(a, noćas)$ = [D_1, D_1, D_2]

Konačno, formiramo rezultujuću listu zadržavajući samo one dokumente koji se javljaju tačno jednom, čime se dobija lista: [D_2]

■

2.3 Vektorski upiti

Za vektorske upite, koji se sastoje od termina bez logičkih operatora, kakav je, na primer (T_i, T_j, T_k), konstruiše se spojena lista dokumenata za termine T_i, T_j i T_k invertovanog indeksa, i dokumenti se prenose na izlaz u opadajućem redosledu u odnosu na broj pojavljivanja u spojenoj listi. Tako se pronalaze stavke koje imaju n upitnih termina pre onih koje imaju $n-1$ upitni termin i tako dalje, sve do onih stavki koje sadrže samo jedan upitni termin.

■ **Primer 2.4** Posmatrajmo invertovani indeks iz uvodnog odeljka, a neka je upit (*kraj, noćas, pjesma*).

Prvi korak je spajanje listi za ova tri termina: $[D_1, D_1, D_1, D_2, D_5, D_5, D_6]$.

Rangirajmo sada dokumente po učestalosti: D_1 (3), D_5 (2), D_2 (1) i D_6 (1). U ovom redosledu termini treba da se prikažu u rezultatu upita.

■

2.4 Zadaci

1. Dati su dokumenti D_1, D_2, D_3, D_4 i njihove ključne reči:

   D_1 `(mačka, puma, lav)`
   D_2 `(aligator, puma, noj)`
   D_3 `(slon, lav, lemur, noj)`
   D_4 `(aligator, slon, mačka, noj)`

   a) Formirati invertovani indeks, tako da ključne reči budu sortirane leksikografski.
   b) Navesti rezultat pretraživanja za vektorski upit (lav, lemur, noj).
   c) Navesti rezultat pretraživanja za logički upit lav AND noj.
   d) Navesti rezultat pretraživanja za logički upit aligator OR mačka.
   e) Navesti rezultat pretraživanja za logički upit noj AND NOT aligator.
2. Dati su dokumenti D_1, D_2, D_3, D_4 i njihove ključne reči:

   D_1 `(tenis, Đoković, teren, Rim)`
   D_2 `(tenis, reket, gladijator, "traka za trčanje")`
   D_3 `(Rim, gladijator, Cezar, Brut, istorija)`
   D_4 `(Rim, vučica, Romul, Rem)`

   a) Formirati invertovani indeks, tako da ključne reči budu sortirane rastuće.
   b) Navesti rezultat pretraživanja za vektorski upit (Rim, gladijator, Cezar).
   c) Navesti rezultat pretraživanja za logički upit gladijator AND "traka za trčanje".
   d) Navesti rezultat pretraživanja za logički upit tenis OR Rim.
   e) Navesti rezultat pretraživanja za logički upit Rim AND NOT tenis.
3. Dati su dokumenti D_1, D_2, D_3, D_4, D_5 i invertovan indeks nad tim dokumentima, sortiran leksikografski:

aronija	D_1, D_2, D_3, D_5
borovnice	D_2, D_4
cikorija	D_1, D_2, D_3, D_5
dinstanje	D_3, D_4
flambirana banana	D_1, D_2
garnirung	D_1, D_4
gibanica	D_2, D_3, D_5
hleb	D_1, D_2, D_4
iseckano meso	D_1, D_2, D_3, D_4
jabuke	D_2, D_4
kompot od šljiva	D_3, D_4, D_5
limunov sok	D_3, D_4, D_5
mleveno meso	D_1, D_2
nektarine	D_2, D_4
orasi	D_1, D_2, D_5
piletina	D_2, D_4
riba	D_1, D_5
som	D_2, D_3, D_4
tunjevina	D_1, D_2
uva	D_1, D_3, D_5
uvijanje sarme	D_1, D_2, D_5
vađenje semenki	D_2, D_3, D_4, D_5
zapraška	D_1, D_4, D_5

a) Navesti rezultat pretraživanja za vektorski upit (dinstanje, "iseckano meso", piletina).
b) Navesti rezultat pretraživanja za logički upit "flambirana banana" AND "jabuke" AND "nektarine".
c) Navesti rezultat pretraživanja za logički upit aronija AND cikorija AND NOT orasi.
d) Navesti rezultat pretraživanja za logički upit limunov sok AND (tunjevina OR som OR riba).
e) Navesti rezultat pretraživanja za logički upit (jabuke OR orasi) AND "kompot od šljiva" AND NOT borovnice.

2.5 Rešenja

1. a) Invertovani indeks:

aligator	D_2, D_4
lav	D_1, D_3
lemur	D_3
mačka	D_1, D_4
noj	D_2, D_3, D_4
puma	D_1, D_2
slon	D_3, D_4

b) (lav, lemur, noj)

$$a = sp(lav, lemur) = \{D_1, D_3, D_3\}$$
$$b = sp(a, noj) = \{D_1, D_2, D_3, D_3, D_3, D_4\}$$
$$D_3(3), D_1(1), D_2(1), D_4(1)$$

c) lav AND noj

$$a = sp(lav, noj) = \{D_1, D_2, D_3, D_3, D_4\} = \underbrace{\{D_3\}}_{\text{samo duplikati}}$$

d) aligator OR mačka

$$a = sp(aligator, macka) = \{D_1, D_2, D_2, D_4\} = \underbrace{\{D_1, D_2, D_4\}}_{\text{svi koji se javljaju}}$$

e) noj AND NOT aligator

$$a = sp(aligator, noj) = \{D_2, D_2, D_3, D_4, D_4\} = \underbrace{\{D_2, D_4\}}_{\text{samo duplikati}}$$

$$b = sp(noj, a) = \{D_2, D_2, D_3, D_4, D_4\} = \underbrace{\{D_3\}}_{\text{eliminacija duplikata}}$$

2. a) Invertovani indeks:

brut	D_3
cezar	D_3
đoković	D_1
gladijator	D_2, D_3
istorija	D_3
reket	D_2
rem	D_4
rim	D_1, D_3, D_4
romul	D_4
tenis	D_1, D_2
teren	D_1
traka za trčanje	D_2
vučica	D_4

b) (Rim, gladijator, Cezar)

$$a = sp(Rim, gladijator) = \{D_1, D_2, D_3, D_3, D_4\}$$
$$b = sp(a, Cezar) = \{D_1, D_2, D_3, D_3, D_4\}$$
$$D_3(3), D_1(1), D_2(1), D_4(1)$$

c) gladijator AND traka za trčanje

$$a = sp(gladijator, traka) = \{D_2, D_2, D_3\} = \underbrace{\{D_2\}}_{\text{samo duplikati}}$$

d) Rim AND gladijator

$$a = sp(gladijator, Rim) = \{D_1, D_2, D_3, D_3, D_4\} = \underbrace{\{D_3\}}_{\text{samo duplikati}}$$

e) tenis OR Rim

$$a = sp(tenis, Rim) = \{D_1, D_1, D_2, D_3, D_4\} = \underbrace{\{D_1, D_2, D_3, D_4\}}_{\text{svi koji se javljaju}}$$

f) Rim AND NOT tenis

$$a = sp(tenis, Rim) = \{D_1, D_1, D_2, D_3, D_4\} = \underbrace{\{D_1\}}_{\text{samo duplikati}}$$

$$b = sp(Rim, a) = \{D_1, D_1, D_3, D_4\} = \underbrace{\{D_3, D_4\}}_{\text{eliminacija duplikata}}$$

3. a) (dinstanje, iseckano meso, piletina) = (T_1, T_2, T_3)

$$a = sp(T_1, T_2) = \{D_1, D_2, D_3, D_3, D_4, D_4\}$$
$$b = sp(a, T_3) = \{D_1, D_2, D_2, D_3, D_3, D_4, D_4, D_4\}$$
$$D_2(3), D_3(2), D_2(2), D_1(1)$$

b) flambirana banana AND jabuke AND nektarine = T_1 AND T_2 AND T_3

$$a = sp(T_1, T_2) = \{D_1, D_2, D_2, D_4\} = \underbrace{\{D_2\}}_{\text{samo duplikati}}$$

$$b = sp(T_3, a) = \{D_2, D_2, D_4\} = \underbrace{\{D_2\}}_{\text{samo duplikati}}$$

c) aronija AND cikorija AND NOT orasi = T_1 AND T_2 AND NOT T_3

$$a = sp(T_1, T_2) = \{D_1, D_1, D_2, D_2, D_3, D_3, D_5, D_5\} = \underbrace{\{D_1, D_2, D_3, D_5\}}_{\text{samo duplikati}}$$

$$b = sp(a, T_3) = \{D_1, D_1, D_2, D_2, D_3, D_5, D_5\} = \underbrace{\{D_1, D_2, D_5\}}_{\text{samo duplikati}}$$

$$c = sp(a, b) = \{D_1, D_1, D_2, D_2, D_3, D_5, D_5\} = \underbrace{\{D_3\}}_{\text{eliminacija duplikata}}$$

d) limunov sok AND (tunjevina OR som OR riba) = T_1 AND (T_2 OR T_3 OR T_4)

$$a = sp(T_2, T_3) = \{D_1, D_2, D_2, D_3, D_4\} = \underbrace{\{D_1, D_2, D_3, D_4\}}_{\text{svi koji se javljaju}}$$

$$b = sp(a, T_4) = \{D_1, D_1, D_2, D_3, D_4, D_5\} = \underbrace{\{D_1, D_2, D_3, D_4, D_5\}}_{\text{svi koji se javljaju}}$$

$$c = sp(T_1, b) = \{D_1, D_2, D_3, D_3, D_4, D_4, D_5, D_5\} = \underbrace{\{D_3, D_4, D_5\}}_{\text{samo duplikati}}$$

e) (jabuke OR orasi) AND kompot od šljiva AND NOT borovnice
= (T_1 OR T_2) AND T_3 AND NOT T_4

$$a = sp(T_2, T_3) = \{D_1, D_2, D_2, D_3, D_4\} = \underbrace{\{D_1, D_2, D_3, D_4\}}_{\text{svi koji se javljaju}}$$

$$b = sp(a, T_4) = \{D_1, D_1, D_2, D_3, D_4, D_5\} = \underbrace{\{D_1, D_2, D_3, D_4, D_5\}}_{\text{svi koji se javljaju}}$$

$$c = sp(T_1, b) = \{D_1, D_2, D_3, D_3, D_4, D_4, D_5, D_5\} = \underbrace{\{D_3, D_4, D_5\}}_{\text{samo duplikati}}$$

3. Proširenja invertovano-indeksnih operacija

Sa ciljem da se pojednostave invertovano-indeksne operacije, ili kako bi se samo pronalaženje poboljšalo, predložena su i implementirana različita proširenja standardnih invertovano-indeksnih tehnika. Kada dokumenti kolekcije sadrže tekstove i dokumente na prirodnom jeziku, termini koji se koriste za formiranje indeksa uglavnom potiču iz samog teksta dokumenata. To je slučaj i kod ručnog i kod automatskog indeksiranja gde, u prvom slučaju, obučeni indekser bira termine, a u drugom, bira ih neki za to predviđen računarski program. Termini se obično biraju nezavisno jedan od drugog, ali veze između termina se uvode u formulacijama upita dodavanjem odgovarajućih logičkih operatora.

Tako, se recimo upit *natural AND language AND processing* može upotrebiti za pronalaženje tekstova koji se bave opštom temom obrade prirodnog jezika (engl. natural language processing, NLP). Takav upit *natural AND language AND processing* pronalazi dokumente koji su indeksirani terminima i *natural* i *language* i *processing*, što u slučaju ekstrahovanja termina iz teksta znači da su se sva tri termina pojavila u tekstu, ali ne nužno i zajedno. Na primer, i dokument koji sadrži rečenicu *Processing of the Python programming language is different than processing any natural language.* bi zadovoljio ovakav upit. Ipak, ukoliko nas zanimaju dokumenti koji se bave oblašću NLP, ovakav dokument ne treba da bude relevantan.

■ **Primer 3.1** Pretraga *beli AND luk* na Google daje, između ostalog:

- U narodu **beli luk** je lek za sve.
- Predsjednik Hrvatskog sabora **Luka** Bebić sastao se u srijedu u Budafi s predsjednikom Narodne skupštine Mađarske **Belom** ...
- **Beli** je morska **luka** otvorena za javni promet lokalnog značaja, a njezino proširenje je moguće uz prethodno obavljene istražne radove. ...
- **Beli** je jedno od najstarijih i u prošlosti najvažnijih creskih naselja. ... Nastao je kao **luka** starog naselja Bucevciji su ostaci nadeni kod crkve Sv. Marka ...
- **Luk** čelični **beli**. Prodaje se na komad. 360.00RSD.

■

3.1 Parametri blizine u specifikaciji upita

Specifikovana veza između dva ili više termina može da se ojača dodavanjem *parametara blizine* specifikaciji upita. Dva moguća parametra ove vrste su:

- unutar rečenice *(T_1 unutar iste rečenice kao T_2)*: implicira da termini T_1 i T_2 moraju zajedno da se pojave u istoj rečenici;
- specifikacija susedstva *(T_1 susedno sa T_2)*: implicira da se termini T_1 i T_2 moraju pojaviti kao susedni u tekstu. Kada se parametri blizine uključe u specifikaciju upita, tema se bliže definiše, a verovatnoća relevantnosti svake pronađene stavke je veća.

Prethodni primer se sada može preformulisati tako da preciznije specifikuje zahtev: *(beli susedno sa luk)*. Na ovaj način, od pet primera odgovora koje nudi Google samo prvi bi bio pronađen.

Kako se implementiraju upiti sa specifikacijom parametra blizine? Konceptualno, najjednostavniji način je obrada šireg AND-upita (T_1 AND T_2), a zatim odbacivanje pronađenih stavki koje ne zadovoljavaju dodatni uslov. Odmah se vidi da to zahteva operaciju iščitavanja originalnog, polaznog, teksta svih stavki koje je pronašao upit (T_1 AND T_2), sa ciljem da se odbace one u kojima dva termina nisu onoliko blizu koliko traži upit.

Sistemi za sravnjivanje teksta (o kojima će biti reči u narednim poglavljima) se mogu koristiti da bi se efikasno otkrilo prisustvo ili odsustvo određenih karaktera ili reči u tekstu dokumenta. Ipak, operacije skaniranja teksta su vremenski zahtevne i obično prespore za opštu upotrebu. Osim toga, njihova primena narušava opšti princip sistema za pronalaženje da se dokumentima pristupa tek kada je utvrđeno da su dovoljno relevantni za korisnika.

Poželjnije rešenje za zadovoljavanje uslova udaljenosti sastoji se u *uključivanju informacija o lokaciji termina u invertovani indeks*. To omogućava da se provere uslovi blizine, a da se ne pristupa sadržaju samog dokumenta i da se ne sravnjuje tekst dokumenta sa terminima iz upita.

Na specifikaciju *unutar rečenice* može se odgovoriti proširivanjem liste za svaki termin u invertovanom indeksu tako što se svakom pojavljivanju termina pridruži i podatak o rednom broju rečenice u kojoj se termin pojavljuje.

■ **Primer 3.2** Posmatrajmo dve klasične liste za termine *beli* i *luk* koje se pojavljuju u nekom invertovanom indeksu:

beli $\rightarrow$ [D_3, D_4, D_5]

luk $\rightarrow$ [D_1, D_2, D_3, D_4]

Upit (*beli* AND *luk*) pronalazi dokumenta D_3 i D_4.

Ako se dodaju podaci o lokaciji termina unutar rečenice, liste se proširuju na sledeći način:

beli $\rightarrow$ [$D_{3,12}, D_{3,14}, D_{4,28}, D_{5,26}, D_{5,33}$]

luk $\rightarrow$ [$D_{1,28}, D_{2,44}, D_{3,12}, D_{3,15}, D_{4,39}$]

Ove liste sada ukazuju da se termin *beli* nalazi u dokumentu D_3 dva puta i to u 12. i 14. rečenici, u dokumentu D_4 jednom i to u 28. rečenici, u dokumentu D_5 dva puta i to u rečenicama 26 i 33; s druge strane, termin *luk* nalazi se u D_1 jednom, u rečenici 28, D_2 u rečenici 44, u dokumentu D_3 dva puta i to u rečenicama 12 i 15, a na kraju i u D_4 u rečenici 39. Sada vidimo da se jedino u dokumentu D_3 ova dva termina nalaze u istoj, 12. rečenici, te je rezultat ovog upita samo dokument D_3. ■

Dalja proširenja invertovanih indeksa koja mogu da pokriju, na primer, brojeve pasusa, brojeve rečenica unutar pasusa, brojeve reči unutar rečenica, i slično, mogu da pruže odgovore na upite kakvi su:

- (*beli* susedno sa *luk*) ili
- (*beli* najviše 5 reči udaljeno od *luk*).

Tako bi dve liste u novom invertovanom indeksu bile:

beli $\rightarrow$ [$D_3, 2, 3, 5$; $D_3, 3, 8, 7$;]

luk $\rightarrow$ [$D_{1,5,1,3}$; $D_{2,2,5,12}$; $D_{3,2,3,6}$;...]

Upit (*beli* susedno sa *luk*) bi pronašao kao relevantan dokument D_3 jer on zadovoljava ograničenje susedstva - prvi termin pojavljuje se u 2 pasusu, 3. rečenici, kao 5. reč u zapisu D_3, dok se drugi termin pojavljuje u 2. pasusu, 3. rečenica, kao 6. reč u istom dokumentu.

Koji su nedostaci modifikovanog invertovanog indeksa koji zadovoljava uslove udaljenosti? Uključivanje informacija o lokaciji termina u invertovano-indeksne liste značajno uvećava veličinu indeksa dvostruki ili čak trostruko, jer svako pojavljivanje termina u dokumentu mora zasebno da bude zabeleženo, a ne samo jednom, kakav je slučaj sa jednostavnim invertovano-indeksnim listama. Međutim, ove informacije omogućavaju da se dobiju mnogo brži odgovori na upite sa uslovima udaljenosti nego što to izgleda moguće sa metodama sravnjivanja teksta.

U nekim sistemima mogu se nametati i dodatni lokacijski uslovi koji ukazuju na to da se termin iz upita mora ograničiti na specifični deo dokumenta. Takva specifikacija lokacije može da bude oblika **unutar-naslova**, **unutar-bibliografskog-citata** ili **unutar apstrakta**, a one se mogu zadovoljiti koristeći slične metode kao one koje se koriste za termine unutar rečenice.

■ **Primer 3.3** Pretpostavimo da su termini dodeljeni kuvarskim receptima izvađeni iz samog teksta. Recimo da su termini sve imenice i pridevi iz teksta u svom rečničkom obliku (a ne u obliku u kome se javljaju u tekstu).

Neka je D_1:

1 Juneće šnicle posolite, utrljajte vegetu i ostavite da odstoje. 2 Za to vreme ispržite na ulju seckan crni luk. 3 Slaninu isecite na rezance. 4 U vatrostalnu posudu poređajte meso, luk i odozgo slaninu, biber i lovorov list. 5 Prelijte belim vinom. 6 Kuvajte na umerenoj vatri.

Indeksni termini bi bili (pridevi i imenice): beo, biber, crn, juneći, list, lovorov, luk, meso, posuda, rezanac, seckan, slanina, šnicla, ulje, umeren, vatra, vatrostalan, vegeta, vino, vreme.

Neka je D_2:

1 Teleće šnicle špikovati slaninom i belim lukom, uvaljati u rastopljeni puter i peći, dok ne porumene na 230 stepeni. 2 Tada dodati pavlaku i vino pa vratiti u rernu, na 180 stepeni, oko sat vremena. 3 Napraviti bešamel i dodati mu seckane orahe..

Indeksni termini bi bili: beo, bešamel, luk, orah, rastopljen, pavlaka, puter, rerna, sat, seckan, slanina, stepen, šnicla, teleći, vino, vreme.

Invertovani indeks ove kolekcije bi bio:

beo → $[D_1, D_2]$	pavlaka → $[D_2]$	šnicla → $[D_1, D_2]$
bešamel → $[D_2]$	posuda → $[D_1]$	teleći → $[D_2]$
biber → $[D_1]$	puter → $[D_2]$	ulje → $[D_1]$
crn → $[D_1]$	rastopljen → $[D_2]$	umeren → $[D_1]$
juneći → $[D_1]$	rerna → $[D_2]$	vatra → $[D_1]$
list → $[D_1]$	rezanac → $[D_1]$	vatrostalan → $[D_1]$
lovorov → $[D_1]$	sat → $[D_2]$	vegeta → $[D_1]$
luk → $[D_1, D_2]$	seckan → $[D_1, D_2]$	vino → $[D_1, D_2]$
meso → $[D_1]$	slanina → $[D_1, D_2]$	vreme → $[D_1, D_2]$
orah → $[D_2]$	stepen → $[D_2]$	

Šta vraća upit (*beo* AND *luk*)? Rezultat je $[D_1, D_2]$.

Šta vraća upit (*seckan* AND *slanina*)? Slično, rezultat bi bio $[D_1, D_2]$.

Pogledajmo sada prošireni indeks koji zadovoljava uslove udaljenosti:

beo → $[D_{1,5}; D_{2,1}]$	pavlaka → $[D_{2,2}]$	šnicla → $[D_{1,1}; D_{2,1}]$
bešamel → $D_{2,3}]$	posuda → $[D_{1,4}]$	teleći → $[D_{2,1}]$
biber → $[D_{1,4}]$	puter → $[D_{2,1}]$	ulje → $[D_{1,2}]$
crn → $[D_{1,2}]$	rastopljen → $[D_{2,1}]$	umeren → $[D_{1,6}]$
juneći → $[D_{1,1}]$	rerna → $[D_{2,2}]$	vatra → $[D_{1,6}]$
list → $[D_{1,4}]$	rezanac → $[D_{1,3}]$	vatrostalan → $[D_{1,4}]$
lovorov → $[D_{1,4}]$	sat → $[D_{2,2}]$	vegeta → $[D_{1,1}]$
luk → $[D_{1,2}; D_{1,4}; D_{2,1}]$	seckan → $[D_{1,2}; D_{2,3}]$	vino → $[D_{1,5}; D_{2,2}]$
meso → $[D_{1,4}]$	slanina → $[D_{1,3}; D_{1,4}; D_{2,1}]$	vreme → $[D_{1,2}; D_{2,2}]$
orah → $[D_{2,3}]$	stepen → $[D_{2,1}; D_{2,2}]$	

Šta vraća upit (*beli* u istoj rečenici *luk*)? Rezultat je samo $[D_2]$.

Šta vraća upit (*seckan* u istoj rečenici *slanina*)? Rezultat je prazna lista $\emptyset$.

■

3.2 Težine termina

Uvođenje pojma težine termina kako za termine iz dokumenata tako i za termine iz upita, u smislu njihovog značaja, omogućava da se vrednuju termini koji su značajniji za potrebe pretraživanja od drugih, manje značajnih. Ako se težine dodeljuju terminima iz dokumenata onda surogat nekog zapisa može da izgleda ovako:

$$D_i = \{T_{i1,0.2}; T_{i2,0.5}; T_{i3,0.6}\}$$

što znači da termin T_{i3} dokumenta D_i ima težinu (značaj) 0.6, dok termin T_{i1} istog dokumenta ima mnogo manju težinu (značaj) 0.2. Kada se težina termina doda u invertovani indeks, pronađeni dokumenti se mogu rangirati po opadajućim težinama sravnjenih termina između upita i dokumenata.

Kako se termini sa težinama dodeljuju dokumentima? Strategija za dodeljivanje težine terminima dokumenta mora biti jednostavna za generisanje i laka za primenu. Postoji više metoda za (au-

tomatsko) dodeljivanje težina terminima koje će biti detaljno pokrivene u drugom delu knjige (u okviru teme automatskog indeksiranja).

Vektorski upiti i težine termina

Kada svi termini iz dokumenta nose oznaku težine, pronađene dokumente je lako rangirati u opadajućem redosledu težina termina. Ako imamo vektorski upit, pomoću kojih se specifikacija upita izražava jednostavno kao skup termina, onda se težina pronađenog dokumenta može definisati jednostavno kao **zbir težina svih termina tog dokumenta**.

Tako, ako se upit q sastoji od termina (T_i, T_j, T_k), težina pronađenog dokumenta D_n koji sadrži sva tri termina iz upita bi se izračunala jednostavno kao

$$W_{ni} + W_{nj} + W_{nk}$$

ili, normalizovano

$$\frac{W_{ni} + W_{nj} + W_{nk}}{3}$$

gde je W_{np} težina termina T_p u dokumentu D_n.

Logički upiti i težine termina

Određivanje težine pronađenog dokumenta kod logičkih upita je kompleksnije jer postoje mnoge ekvivalentne formulacije logičkih izraza. Jedna mogućnost sastoji se u transformisanju svakog upita u oblik zbira proizvoda (ili disjunkcija konjunkcija), koji je takođe poznat kao **disjunktivna normalna forma** (DNF), gde se upit pojavljuje kao skup **ili**-veza jednog ili više terma, a to su termini povezani **i**-operatorom kao u upitu:

$$(T_i \text{ AND } T_j \text{ AND } Tk)$$

a svi termi (konjunkti, klauze) se povezuju jedan s drugim operatorom **OR**.

Na primer, logički upit: $(T_1 \wedge T_2) \vee T_3$ je već u disjunktivnoj normalnoj formi, dok je disjunktivna normalna forma bulovskog upita

$$T_1 \wedge (T_2 \vee T_3) = (T_1 \wedge T_2) \vee (T_1 \wedge T_3)$$

Kako se logički izraz prevodi u disjunktivnu normalnu formu? Za svaki logički izraz postoji disjunktivna normalna forma. Neka je dat logički izraz $m(p,r,q)$ u čijoj je istinitosnoj tablici samo jedna vrednost $\top$ u poslednjoj koloni:

p	q	r	m
$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$
$\boxed{\bot}$	$\boxed{\top}$	$\boxed{\top}$	$\boxed{\top}$
$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\bot$

Jasno je da je izraz tačan samo kada je p netačno, a q i r tačno. Dakle, izraz m se svodi na sledeću disjunktivnu normalnu formu koja ima samo jedan term: $(\neg p \wedge q \wedge r)$.

Dalje, neka je dat logički izraz $m(p,r,q)$ u čijoj je istinitosnoj tablici ima više vrednosti $\top$ u poslednjoj koloni:

p	q	r	m
$\bot$	$\bot$	$\bot$	$\bot$
$\boxed{\bot}$	$\boxed{\bot}$	$\boxed{\top}$	$\boxed{\top}$
$\bot$	$\top$	$\bot$	$\bot$
$\boxed{\bot}$	$\boxed{\top}$	$\boxed{\top}$	$\boxed{\top}$
$\boxed{\top}$	$\boxed{\bot}$	$\boxed{\bot}$	$\boxed{\top}$
$\top$	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\bot$

Jasno je da je izraz tačan za tri kombinacije tačno/netačno iskaza p, q i r. Dakle, izraz m se svodi na sledeću disjunktivnu normalnu formu koja ima tri konjukta:

$$(\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge \neg q \wedge \neg r)$$

■ **Primer 3.4** Kako glasi disjunktivna normalna forma bulovskog izraza $m = (\neg p \vee r) \wedge (\neg r \vee q)$?

Treba sačiniti istinitosnu tablicu koristeći istinitosne tablice osnovnih iskaza:

p	q	r	$\neg p$	$\neg p \vee r$	$\neg r$	$\neg r \vee q$	m
$\boxed{\bot}$	$\boxed{\bot}$	$\boxed{\bot}$	$\top$	$\top$	$\top$	$\top$	$\boxed{\top}$
$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\boxed{\bot}$	$\boxed{\top}$	$\boxed{\bot}$	$\top$	$\top$	$\top$	$\top$	$\boxed{\top}$
$\boxed{\bot}$	$\boxed{\top}$	$\boxed{\top}$	$\top$	$\top$	$\bot$	$\top$	$\boxed{\top}$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\boxed{\top}$	$\boxed{\top}$	$\boxed{\top}$	$\bot$	$\top$	$\bot$	$\top$	$\boxed{\top}$

DNF datog iskaza sadrži četiri konjukta:

$$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge r)$$

■

■ **Primer 3.5** Kako glasi disjunktivna normalna forma bulovskog izraza $m = (p \vee r) \wedge (\neg r \vee \neg q)$?

Treba sačiniti istinitosnu tablicu koristeći istinitosne tablice osnovnih iskaza:

p	q	r	$p \vee r$	$\neg q$	$\neg r$	$\neg r \vee \neg q$	m
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$
$\boxed{\bot}$	$\boxed{\bot}$	$\boxed{\top}$	$\top$	$\top$	$\bot$	$\top$	$\boxed{\top}$
$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\boxed{\top}$	$\boxed{\bot}$	$\boxed{\bot}$	$\top$	$\top$	$\top$	$\top$	$\boxed{\top}$
$\boxed{\top}$	$\boxed{\bot}$	$\boxed{\top}$	$\top$	$\top$	$\bot$	$\top$	$\boxed{\top}$
$\boxed{\top}$	$\boxed{\top}$	$\boxed{\bot}$	$\top$	$\bot$	$\top$	$\top$	$\boxed{\top}$
$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$

DNF datog iskaza sadrži četiri konjukta:

$$(\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r)$$

■

Kada je jasno da se svaki logički izraz može automatski prevesti u disjunktivnu normalnu formu, pitanje stoji i dalje: kako se računa relevantnost dokumenata indeksiranih terminima sa težinama?

Za logički upit preveden u disjunktivnu normalnu formu, težina (relevantnost) pronađenog dokumenta može se izračunati u dva koraka:

- Težina dokumenta se računa u odnosu na svaki term (konjunkt) u upitu kao najmanja težina svakog termina u dokumentu iz tog konjunkta;
- Težina dokumenta konačnog upita (disjunkcija svih konjunkta) se računa kao najveća od težina konjunkata izračunata u prvom koraku.

■ **Primer 3.6** U tabeli je prikazana tipična situacija, u kojoj postoje dva dokumenta, D_1 i D_2 čiji sadržaji su reprezentovani istim terminima T_1, T_2 i T_3 od kojih svaki ima težinu specifičnu za dokument. Neka je upit $m = ((T1 \wedge T2) \vee T3)$

Vektori dokumenata	**Težine konjukata**		**Težine upita**
	$T_1 \wedge T_2$	T_3	$(T_1 \wedge T_2) \vee T_3$
D_1=$(T_{1,0.2}; T_{2,0.5}; T_{3,0.6})$	0.2	0.6	0.6
D_2=$(T_{1,0.7}; T_{2,0.2}; T_{3,0.1})$	0.2	0.1	0.2

Težina pronađenog odgovora D_1 je 0.6, dok je težina pronađenog odgovora D_2 0.2, što znači da će dokument D_1 biti više rangiran. Treba uočiti da težina pronađenog dokumenta zavisi i od težine termina koji su dodeljeni dokumentu i od formulacije upita.

■

3.3 Specifikacija sinonima

U nekim sistemima za pronalaženje informacija može se dodati specifikacija sinonima bilo kom upitnom terminu u formulaciji upita. Svaki takav sinonim zatim automatski menja originalni termin, a dokumenta se pretražuju i originalnim i dodatim terminima. Kada se koriste Bulovske upitne formulacije, sinonimne specifikacije se mogu jednostavno spojiti sa originalnim terminima pomoću OR operatora pre nego što se pretraživanje obavi.

Tako se upit

$$((T_1 \wedge T_2) \vee T_3)$$

transformiše u širi iskaz

$$(((T_1 \vee S_1) \wedge T_2) \vee (T_3 \vee S_3)$$

kada korisnik doda S_1 kao sinonim za T_1 i S_3 kao sinonim za T_3. Kada se sinonimi dodaju određenim upitima, broj relevantnih pronađenih stavki u pronalaženju može da se uveća (ali i smanji, zavisi od formulacije upita).

■ **Primer 3.7** Neka kolekcja sadrži sledeće dokumente sa odgovarajućim indeksnim terminima:

D_1: juneće meso, crni luk, slanina, biber, lovorov list, vino

D_2: teleće meso, beli luk, slanina, puter, pavlaka, vino, orasi

D_3: teleće meso, so, jaja, brašno, mrvice, maslac

D_4: teleće meso, crni luk, češnjak, ulje, biber, soda bikarbona

D_5: svinjsko meso, crni luk, ulje, so, biber, ljuta paprika

Za ovakvu kolekciju, odgovarajući invertovani indeks je:

beli luk $\rightarrow$ [D_2]	orasi $\rightarrow$ [D_2]
biber $\rightarrow$ [D_1, D_4, D_5]	pavlaka $\rightarrow$ [D_2]
brašno $\rightarrow$ [D_3]	puter $\rightarrow$ [D_2]
crni luk $\rightarrow$ [D_1, D_4, D_5]	slanina $\rightarrow$[D_1, D_2]
češnjak $\rightarrow$ [D_4]	so $\rightarrow$ [D_3, D_5]
jaja $\rightarrow$ [D_3]	soda bikarbona $\rightarrow$ [D_4]
juneće meso $\rightarrow$ [D_1]	svinjsko meso $\rightarrow$ [D_5]
lovorov list $\rightarrow$ [D_1]	teleće meso $\rightarrow$ [D_2, D_3, D_4]
ljuta paprika $\rightarrow$ [D_5]	ulje $\rightarrow$ [D_4, D_5]
maslac $\rightarrow$ [D_3]	vino $\rightarrow$ [D_1, D_2]
mrvice $\rightarrow$ [D_3]	

Neka je dat rečnik sinonima:

beli luk $\rightarrow$ češnjak	pavlaka $\rightarrow$ mileram
biber $\rightarrow$ papar	svinjsko meso $\rightarrow$ svinjetina
juneće meso $\rightarrow$ junetina	teleće meso $\rightarrow$ teletina
ljuta paprika $\rightarrow$ aleva paprika	ulje $\rightarrow$ zejtin
maslac $\rightarrow$ puter	
mrvice $\rightarrow$ prezle	

Izvršavanje upita (*teleće meso* $\wedge\neg$ (*beli luk* $\vee$ *puter*)) bi izgledalo:

teleće meso $\rightarrow [D_2, D_3, D_4]$

beli luk $\rightarrow [D_2]$

puter $\rightarrow [D_2]$

sp(*beli luk*, *puter*) = $[D_2, D_2]$

dakle, (*beli luk* $\vee$ *puter*) = $[D_2]$

sp(*teleće meso*, (*beli luk* $\vee$ puter)) = $[D_2, D_2, D_3, D_4]$

dakle, (*teleće meso* $\wedge$ (*beli luk* $\vee$ *puter*)) = $[D_2]$

sp((*teleće meso* $\wedge$ (*beli luk* $\vee$ *puter*)), *teleće meso*) = $[D_2, D_2, D_3, D_4]$

teleće meso $\wedge\neg$ (*beli luk* $\vee$ *puter*)) = $[D_3, D_4]$

Ipak, kada se koristi rečnik sinonima upit postaje:

$$((\textit{teleće meso} \vee \textit{teletina}) \wedge\neg ((\textit{beli luk} \vee \textit{češnjak}) \vee (\textit{puter} \vee \textit{maslac})))$$

U tom slučaju:

teleće meso $\rightarrow [D_2, D_3, D_4]$

teletina $\rightarrow \emptyset$

beli luk $\rightarrow [D_2]$

češnjak $\rightarrow [D_4]$

puter $\rightarrow [D_2]$

maslac $\rightarrow [D_3]$

Odgovor: $\emptyset$

■

3.4 Odsecanje termina

Još jedan način da se prošire funkcionalnosti inverovatno-indeksnih operacija je postavljanje upita u odsečenom obliku, odnosno tako da se terminima u upitu uklone prefiksi, sufiksni, ali čak i infiksi. Tako podsečeni oblik predstavlja širi pojam od originalnog, nepodsečenog oblika, jer uključuje raznovrsne nepodsečene termine.

Ilustracije radi, podsečeni termin PSYCH*, gde je * specijalni znak koji predstavlja **završetak promenljive dužine**, predstavlja *psychiatrist, psychiatry, psychiatric, psychology, psychologist, psychological*, i mnoge druge.

Mogu se koristiti različite metode za podsecanje termina:

- uklanjanje određenih sufiksa i prefiksa,
- uklanjanje fiksnog broja završnih karaktera u reči,
- svođenje svake reči na njen koren fiksne dužine, itd.

Najčešće se u praksi određeni krajevi reči ili prefiksi uklanjaju samo ako pri tome ostaje koren koji je dovoljan da identifikuje reč. Tako bi, na primer, sufiks -ING bio uklonjen iz termina kakav je *indexing* ali ne i iz termina *king*.

Podsecanje sufiksa

Kada je jasno šta je željena funkcionalnost, postavlja se pitanje kako se ona implementira u okviru strukture invertovatnog indeksa koju smo do sada uveli. U slučaju da je u formulaciji upita dopušteno samo podsecanje sufiksa prilikom postavljanja upita, onda se metodologija invertovanih indeksa koja je do sada primenjivana može zadržati. Razlog je sledeći: liste dokumenata koje pokrivaju određeni podsečeni termini su susedni u indeksu (ako je on uređen u alfabetskom poretku), pa je dovoljno naći sve one susedne termine u indeksu koje počinju datim početkom.

■ **Primer 3.8** Takav primer je upit PSYCH*. Ovaj upit ekvivalentan je upitu

PSYCHIATRIST OR PSYCHIATRY OR PSYCHIATRIC OR PSYCHOLOGY OR PSYCHOLOGIST OR PSYCHOLOGICAL. . . .

Odsečak invertovanog indeksa koji sadrži ove termine mogao bi da izgleda, npr:

pseudoscience → $[D_1]$
psittacosis → $[D_2]$
psychiatry → $[D_3]$
psychoactive → $[D_2]$
psychoanalysis → $[D_3]$
psychogeriatrics → $[D_4]$
psychology → $[D_3]$
psychometric → $[D_5]$
psychoneuroimmunology → $[D_4]$
psychopathic → $[D_2]$
psychosis → $[D_3]$
psychosomatic → $[D_5]$
psychotherapy → $[D_2]$
puberty → $[D_6]$

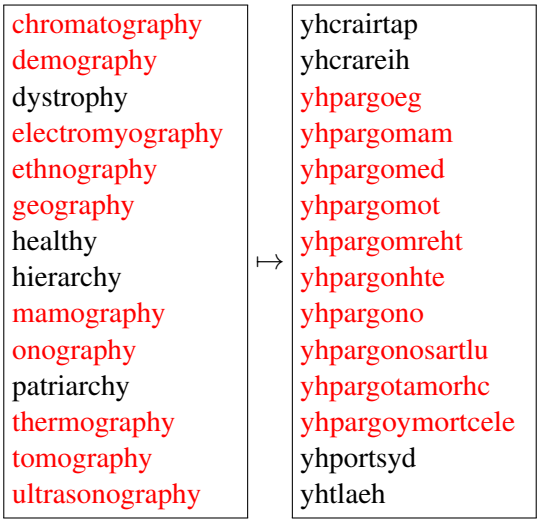
Lista koja predstavlja rezultat ovog upita je $[D_2, D_3, D_4, D_5]$, odnosno u slučaju vektorskog upita $D_3(4), D_2(3), D_4(2), D_5(2)$. ■

Podsecanje prefiksa

Podsecanje prefiksa upitnog termina, kao npr. u upitu *SYMMETRY, gde je * specijalni znak koji predstavlja **početak promenljive dužine**, može se uklopiti na analogan način korišćenjem invertovanih indeksa u kojima su termini alfabetski uređeni **u obrnutom poretku (s desna na levo)**.

Na taj način će se reč *antisymmetry* u indeksu pojaviti kao *yrtemmysitna*, dok će se reč *asymmetry* slično pojaviti u indeksu kao *yrtemmysa*. Kada se zada upit *SYMMETRY, indeks uređen u alfabetskom poretku s kraja termina se pretražuje sa YRTEMMYS*, pa se proces pronalaženja svodi na slučaj podsecanja sufiksa.

■ **Primer 3.9** Za upit *GRAPHY, odsečak invertovanog indeksa koji sadrži termine zapisane s desna na levo, sortirano alfabetski, izgledao bi:



■

Jasno je kako se funkcionalnost postavljanja upita sa odsecanjem termina integriše u invertovani indeks. Istovremena integracija mogla bi se postići kombinovanjem ova dva indeksa, tako da se originalnom terminu kojem je pridružena lista dokumenata u kojima se nalazi, pridruži i odgovarajući termin u obrnutom poretku iz paralelnog invertovanog indeksa koji sadrži termine u redosledu s desna na levo.

Takav sistem omogućio bi i postavljanje upita čijim se terminima istovremeno podsecaju i prefiks i sufiks, kao što je oblik *SYMM*. Ovaj upit odgovara i terminu *antisymmetric* i terminu *asymmetry*. Možemo čitati ovaj upit kao *svi oblici koji imaju bilo šta na početku ili na kraju, ali se u sredini mora naći sekvenca* SYMM. Štaviše, znak * može označavati i praznu nisku, tako da upitu *SYMM* odgovaraju i termini koji samo počinju ili se završavaju na SYMM, poput *symmetrical*.

Opšte rešenje

Da bi se obradili slučajevi infiksnog podsecanja potrebna su još složenija rešenja. Na primer, upitni termin WOM*N bi obuhvatio i termin *woman* i termin *women*, dok bi upitni termin COL*R obuhvatio i termin *color* i termin *colour*.

Opšte rešenje problema podsecanja upitnih termina podrazumevalo bi da invertovani indeks sadrži sve moguće *rotirane* oblike reči. Takav indeks se može formirati na sledeći način:

- Svaki termin $s = a_1a_2\ldots a_n$ gde su a_i pojedinačni karakteri, u prvom koraku se proširuje dodavanjem specijalnog završnog karaktera, na primer kosa crta / (ili drugim unapred dogovorenim karakterom koji se sigurno neće pojaviti ni u jednoj reči);
- Svaki prošireni termin $a_1a_2\ldots a_n/$ se **ciklično rotira sam oko sebe** $n+1$ **puta**. To proizvodi $n+1$ različitih oblika reči: $a_1a_2\ldots a_n/$, $/a_1a_2\ldots a_n$, $a_n/a_1a_2\ldots a_{n-1}$, $a_{n-1},a_n/a_1a_2\ldots a_{n-2}$ i tako dalje.
- Na kraju se dobijena lista oblika reči sortira u alfabetskom poretku, pri čemu je poredak karaktera u rečima:

$$/ < \text{A} < \text{B} < \ldots < \text{Z}.$$

U rečniku koji se na taj način dobija postoji ulaz koji može da obradi svaki oblik odsecanja.

■ **Primer 3.10** Primera radi, neka je potrebno omogućiti sve varijante podsecanja termina za invertovani indeks koji sadrži samo tri termina: ABC, BABC i BCAB.

Svaki od termina se proširuje znakom /, te se dobijaju prošireni termini: ABC/, BABC/ i BCAB/.

Sada je potrebno odrediti sve rotacije za svaki od termina. Pošto termin ABC/ ima četiri karaktera, biće četiri rotacije. Slično, za BABC/ i BCAB/ biće po pet rotacija:

ABC $\mapsto$ ABC/ $\mapsto$ ABC/, /ABC, C/AB, BC/A

BABC $\mapsto$ BABC/ $\mapsto$ BABC/, /BABC, C/BAB, BC/BA, ABC/B

BCAB $\mapsto$ BCAB/ $\mapsto$ BCAB/, /BCAB, B/BCA, AB/BC, CAB/B

Invertovani indeks sa sortiranim terminima sada bi izgledao:

/ABC
/BABC
/BCAB
AB/BC
ABC/
ABC/B
B/BCA
BABC/
BC/A
BC/BA
BCAB/
C/AB
C/BAB
CAB/B

■

Kako sada izgleda pretraživanje invertovanog indeksa? Strategija je sledeća:

- Za upitni termin **X**, koji predstavlja **nepodsečenu karaktersku nisku** (traži se sam termin **X**), biraju se ulazi u invertovanom indeksu **/X** ili **X/**. Odgovarajući identifikatori dokumenata se svi odnose na termin **X** koji je proširen specijalnim karakterom **/**.
- Za upitni termin **X***, odnosno kada je **podsečen sufiks**, u indeksu treba tražiti **/X** koji odgovara svim ulazima koji počinju sa **/** (početak reči) iza koga sledi **X**, i možda još neki dodatni karakteri. Pronalazi se **/X, /X**Y_1**, /X**Y_2,... koji predstvaljuju termine **X, /X**Y_1**, /X**Y_2,...
- Za upitni termin ***X**, odnosno kada je **podsečen prefiks**, treba tražiti termine koji počinju sa **X/**; taj termin pronalazi listu ulaza: **X/, X/**Y_1**,...,X/**Y_n koja predstavlja originalni termin **X**, a zatim i termine Y_1**X,...,**Y_n**X** koji sadrže proizvoljan prefiks iza koga sledi **X**.
- Za upitni termin ***X***, odnosno kada su **podsečeni i prefiks i sufiks**, traže se svi termini koji počinju sa **X**; on pronalazi liste ulaza **X**Y_1**/**Z_1**,...,X**Y_n**/**Z_n koji odgovaraju terminima Z_1**X**Y_1**,...,**Z_n**X**Y_n u kojima je **X** traženi infiks.
- Za upitni termin **X*Y**, odnosno kada je **podsečen infiks**, treba tražiti termin **Y/X**, koji pronalazi ulaze Y**/X**Z_1**,...,**Y**/X**Z_n, koji odgovaraju terminima **X**Z_1Y**,...,X**Z_nY gde je Z_i infiks promenljive dužine.

■ **Primer 3.11** Kako bi za rotirani invertovani indeks dat u Primeru 3.10 izgledalo odgovaranje na upit ***B***? Tada je termin za traženje sam **B**, dakle **tražimo sve termine koji počinju sa B**. Kako su u invertovanom indeksu termini sortirani i ovi termini nalaze se jedan pored drugog:

/ABC
/BABC
/BCAB
AB/BC
ABC/
ABC/B
B/BCA
BABC/
BC/A
BC/BA
BCAB/
C/AB
C/BAB
CAB/B

Pronađenim terminima odgovaraju sledeći originalni termini, redom:

BCAB,

BABC,

ABC,

BABC,

BCAB.

Konačno, odgovor su sledeći originalni termini: **BCAB, BABC, ABC** (termini koji sadrže bar jedno slovo B).

■

Kao i svako rešenje koje smo do sada videli, i ovo ima svoje nedostatke. Mana ovog rešenja je što se neki termini javljaju više puta u izlaznoj listi (a samim time i dokumenti koji odgovaraju tim terminima) ako se traženi segment pojavljuje više puta u pronađenom terminu. Na primer, BABC i BCAB se javljaju dva puta u izlaznoj listi zbog dvostrukog pojavljivanja traženog B. U praksi ovo nije tako česta pojava.

Dakle, metod permutovanog rečnika omogućava da se podsečeni termini koriste u formulacijama upita, ali po cenu znatnog povećavanja broja stavki u indeksu.

3.5 Regulisanje veličine izlaza logičkih upita

Standardno korišćenje logičkih formulacija upita može da postavlja mnogo problema korisnicima jer je izlaz pretrage osetljiv na formulaciju upita, i može ozbiljno da varira sa veoma malim promenama u formulaciji upita.

Osim toga, nedostatak jednostavne kontrole nad veličinom izlaza i proizvodnja rezultata pretrage koji nisu ni na koji način rangirani prema nekoj pretpostavljenoj koristi za korisnika, komplikuje pretragu za mnoge neobučene korisnike.

Mnoge metode su razvijene koje pojednostavljuju operacije pretrage, a u isto vreme zadržavaju tehnogiju pretraživanja utemeljenu na logičkim formulacijama upita i na standardnim metodama objedinjavanja lista.

Kod ovakvog pronalaženja, originalni logički upit koji sadrži *n* termina se zamenjuje novim upitom koji je izabran iz liste logičkih upita od istih *n* termina, a koji variraju od vrlo uskih formulacija gde je svih *n* termina uključeno u jedan *and* iskaz

$$(T_1 \text{ and } T_2 \text{ and } ...)$$

do veoma širokih formulacija gde je svih n termina uključeno u jedan *or* iskaz

$$(T_1 \text{ or } T_2 \text{ or } ...)$$

Među-upiti se sastoje, prvo od disjunkcije *n and* iskaza, pri čemu je svaki od ovih *n* iskaza dobijen od najuže formulacije brisanjem iz nje jednog člana. Sledeća formulacija je još opuštenija i sastoji se od disjunkcije *and* iskaza koji su iz najuže formulacije dobijeni brisanjem dva člana, i tako redom, dok se ne dođe do *or* iskaza koji povezuju samo po jedan upitni termin.

■ **Primer 3.12** Primer hijerarhije upita sa četiri upitna termina: T_1 T_2, T_3, T_4.

Broj upita	Hijerarhija upita (od najužeg do najšireg)	Broj pronađenih	Broj pronađenih relevantnih
0	$(T_1 \wedge T_2 \wedge T_3 \wedge T_4)$	2	2
1	$(T_1 \wedge T_2 \wedge T_3) \vee$ $(T_1 \wedge T_2 \wedge T_4) \vee$ $(T_1 \wedge T_3 \wedge T_4) \vee$ $(T_2 \wedge T_3 \wedge T_4)$	6	5
2	$(T_1 \wedge T_2) \vee$ $(T_1 \wedge T_3) \vee$ $(T_1 \wedge T_4) \vee$ $(T_2 \wedge T_3) \vee$ $(T_2 \wedge T_4) \vee$ $(T_3 \wedge T_4)$	23	15
3	$(T_1 \vee T_2 \vee T_3 \vee T_4)$	86	25

U hijerarhiji upita kakva je prikazana u prethodnom primeru može se očekivati da broj pronađenih dokumenata raste sa širinom upita. Korisnik tada može da izabere formulaciju iz hijerahije upita koja izdvaja onoliko dokumenata koliko odgovara njegovim potrebama i mogućnostima.

Kada je izabrani upit veoma uzak, ukupan broj pronađenih stavki biće mali, ali će najveći broj njih biti relevantan. S druge starne, kod širokih upita izlaz je veliki, ali može se očekivati da će deo relevantnih dokumenata među svim izdvojenim biti mnogo manji.

■

■ **Primer 3.13** Isprobati princip regulisanja veličine izlaza logičkih upita na sajtu COBISS-a: `https://plus.sr.cobiss.net/opac7/bib/search/expert?db=cobib`.

Neka su parametri autor **AU="Andrić, Ivo"**, godina izdanja **PY="1988"**, naslov **TI="*avlija"** i izdavač je **PU="Prosveta"**.

Upiti bi bili:

```
AU="Andrić, Ivo" AND PY="1988" AND TI="*avlija" AND PU="Prosveta"
```

```
(AU="Andrić, Ivo" AND PY="1988" AND TI="*avlija") OR
(AU="Andrić, Ivo" AND PY="1988" AND PU="Prosveta") OR
(AU="Andrić, Ivo" AND TI="*avlija" AND PU="Prosveta") OR
(PY="1988" AND TI="*avlija" AND PU="Prosveta")


(AU="Andrić, Ivo" AND PY="1988") OR
(AU="Andrić, Ivo" AND TI="*avlija") OR
(PY="1988" AND TI="*avlija") OR
(AU="Andrić, Ivo" AND PU="Prosveta") OR
(PY="1988" AND PU="Prosveta") OR
(TI="*avlija" AND PU="Prosveta")


AU="Andrić, Ivo" OR PY="1988" OR TI="*avlija" OR PU="Prosveta"
```

■

3.6 Zadaci

1. Formirati invertovani indeks dopunjen informacijama o rednom broju reči u dokumentu u kom se ta reč javila, za indeksne termine *finale, rival, srpski, teniser, titula, trijumf* i *turnir* i dokumente:
   D_1:
   `Srpska teniserka Ana Ivanović objavila je kraj igračke karijere,`
   `tokom koje je osvojila ukupno 15 turnira od kojih je svakako`
   `najznačajniji trijumf iz 2008. na Rolan Garosu.`
   D_2:
   `Pobedom u finalu nad svojim najvećim rivalom,`
   `srpski teniser Novak Đoković je osvojio turnir u Dohi`
   `i istovremeno prekinuo seriju pobeda Endija Mareja`
   `koji je tako ostao na 28 trijumfa u nizu.`
   D_3:
   `Najnovijom titulom u Melburnu, američka teniserka Serena Vilijams`
   `postala je najtrijumfalnija teniserka sa 23 Gren Slem titule.`
2. Navesti rezultat logičkih upita za dati prošireni invertovani indeks informacijom o rednom broju rečenice u kojoj se termin javlja:
   $T_1 \rightarrow$ [D_1,3; D_1,8; D_1,11; D_2,4; D_2,15;]
   $T_2 \rightarrow$ [D_1,8; D_1,11; D_2,8; D_2,15;]
   $T_3 \rightarrow$ [D_1,3; D_1,6; D_1,13; D_2,7; D_2,12;]
   $T_4 \rightarrow$ [D_1,2; D_1,8; D_2,7; D_3,4; D_3,5;]
   $T_5 \rightarrow$ [D_2,1; D_2,7; D_2,11; D_3,5;]
   $T_6 \rightarrow$ [D_1,3; D_2,7; D_2,15; D_3,4; D_3,5;]
   (a) T_1 U ISTOJ REČENICI KAO T_2
   (b) T_2 U ISTOJ REČENICI KAO T_6
   (c) T_3 U ISTOJ REČENICI KAO T_4 U ISTOJ REČENICI KAO T_5
3. Navesti rezultat logičkih upita za dati prošireni invertovani indeks informacijama o rednom broju pasusa u dokumentu, rednom broju rečenice u pasusu i rednom broju reči u rečenici u kojoj se termin javlja:

$T_1 \rightarrow$ [D_1,1,3,5; D_1,8,1,5; D_1,11,2,3; D_2,4,4,4;]
$T_2 \rightarrow$ [D_1,1,3,6; D_1,8,5,1; D_2,8,7,1;]
$T_3 \rightarrow$ [D_1,11,2,6; D_1,11,2,8; D_2,4,4,5; D_2,8,7,5]

(a) T_1 SUSEDNO SA T_2
(b) T_1 SUSEDNO SA T_3
(c) T_1 U BLIZINI NAJVIŠE 6 T_3
(d) T_2 U BLIZINI NAJVIŠE 5 T_3

Napomena: za četvorku (D,i,j,k), D je dokument, i je redni broj pasusa, j je redni broj rečenice u tom pasusu, k je redni broj reči u toj rečenici.

4. Navesti rezultat vektorskog upita (T_1,T_2,T_3) u obliku $(dokument, relevantnost)$ za date dokumente i težine datih termina koji se u njima javljaju. Urediti dokumente po relevantnosti, opadajuće.

D_1 { T_1,0.3; T_2,0.8; T_3,0.1; }
D_2 { T_1,0.7; T_2,0.1; T_3,0.45; }
D_3 { T_1,0.2; T_2,0.4; }
D_4 { T_1,0.1; T_3,0.4; }

5. Odrediti disjunktivnu normalnu formu narednih logičkih iskaza:
   (a) $p \wedge (q \vee r)$
   (b) $(p \vee q) \wedge (q \vee r)$
   (c) $(\neg p \vee r) \wedge (\neg r \vee q)$
   (d) $(p \vee r) \wedge (\neg r \vee \neg q)$
6. Navesti rezultat sledećih logičkih upita u obliku $(dokument, relevantnost)$ za prošireni indeks iz 4. zadatka. Urediti dokumente po relevantnosti, opadajuće.
   (a) $T_1 \wedge T_2 \wedge T_3$
   (b) $T_1 \vee T_2 \vee T_3$
   (c) $(T_1 \wedge T_2) \vee T_3$
   (d) $(T_1 \wedge T_2) \vee (T_2 \wedge T_3)$
7. Kako se mogu proširiti sledeći logički upiti:
   (a) predviđanje AND NOT nagađanje
   pri čemu je poznato da su parovi sinonima
   (predviđanje, predikcija) i (nagađanje, špekulisanje)
   (b) (a AND b) OR (c AND NOT d)
   pri čemu je poznato da su parovi sinonima
   (a, A), (b, B), (c, C) i (d, D)
8. Neka originalni rečnik indeksiranja nekog dokumenta sadrži termine: KAP, KAPA, RAK, MAK. Sistem za pronalaženje predviđa pronalaženje po upitnim terminima koji dozvoljavaju sve vrste odsecanja.
   (a) Sastaviti prošireni indeks koji će ovo omogućiti.
   (b) Ako korisnik postavi navedene upite, kako ove upite treba preformulisati da bi se oni prosledili proširenom indeksu, šta oni pronalaze u proširenom indeksu i kojim to originalnim terminima odgovara:
      i. KAP
      ii. KA*
      iii. *AK
      iv. *AP*
      v. K*A
9. Neka originalni rečnik indeksiranja nekog dokumenta sadrži termine: RANA, MANA, MAK, RIMA. Sistem za pronalaženje predviđa pronalaženje po upitnim terminima koji dozvoljavaju sve vrste odsecanja.
   (a) Sastaviti prošireni indeks koji će ovo omogućiti.

(b) Ako korisnik postavi navedene upite, kako ove upite treba preformulisati da bi se oni prosledili proširenom indeksu, šta oni pronalaze u proširenom indeksu i kojim to originalnim terminima odgovara:
   i. MAK
   ii. MA*
   iii. *ANA
   iv. *A*
   v. R*A

10. Neka originalni rečnik indeksiranja nekog dokumenta sadrži termine: OKO, SOKO, ONA, SKVO. Sistem za pronalaženje predviđa pronalaženje po upitnim terminima koji dozvoljavaju sve vrste odsecanja.
   (a) Sastaviti prošireni indeks koji će ovo omogućiti.
   (b) Ako korisnik postavi navedene upite, kako ove upite treba preformulisati da bi se oni prosledili proširenom indeksu, šta oni pronalaze u proširenom indeksu i kojim to originalnim terminima odgovara:
      i. OKO
      ii. O*
      iii. *KO
      iv. *K*
      v. S*O

3.7 Rešenja

finale → [D_2,3;]
rival → [D_2,7;]
srpski → [D_1,1; D_2,8;]

1. teniser → [D_1,2; D_2,9; D_3,6; D_3,12;]
titula → [D_3,2; D_3,17;]
trijumf → [D_1,22; D_2,30; D_3,11;]
turnir → [D_1,16; D_2,14;]

2. (a) T_1 U ISTOJ REČENICI T_2

T_1 → [D_1,3; D_1,8; D_1,11; D_2,4; D_2,15;]
T_2 → [D_1,8; D_1,11; D_2,8; D_2,15;]

$$sp(T_1,T_2) = [D_1,3;D_1,8;D_1,8;D_1,11;D_1,11;D_2,4;D_2,8;D_2,15;D_2,15;] = \underbrace{[D_1,8;D_1,11;D_2,15;]}_{\text{samo duplikati}}$$

(b) T_2 U ISTOJ REČENICI T_6

T_2 → [D_1,8; D_1,11; D_2,8; D_2,15;]
T_6 → [D_1,3; D_2,7; D_2,15; D_3,4; D_3,5;]

$$sp(T_2,T_6) = [D_1,3;D_1,8;D_1,11;D_2,7;D_2,8;D_2,15;D_2,15;D_3,4;D_3,5;] = \underbrace{[D_2,15;]}_{\text{samo duplikati}}$$

(c) T_3 U ISTOJ REČENICI T_4 U ISTOJ REČENICI T_5

$T_3 \rightarrow$ [D_1,3; D_1,6; D_1,13; D_2,7; D_2,12;]
$T_4 \rightarrow$ [D_1,2; D_1,8; D_2,7; D_3,4; D_3,5;]
$T_5 \rightarrow$ [D_2,1; D_2,7; D_2,11; D_3,5;]

$$a = sp(T_3, T_4) = [D_1,2;D_1,3;D_1,6;D_1,8;D_1,13;D_2,7;D_2,7;D_2,12;D_3,4;D_3,5;] = \underbrace{[D_2,7;]}_{\text{samo duplikati}}$$

$$b = sp(a, T_5) = [D_2,1;D_2,7;D_2,7;D_2,11;D_3,5;] = \underbrace{[D_2,7;]}_{\text{samo duplikati}}$$

3. (a) T_1 SUSEDNO SA T_2
$T_1 \rightarrow$ [D_1,1,3,5; D_1,8,1,5; D_1,11,2,3; D_2,4,4,4;]
$T_2 \rightarrow$ [D_1,1,3,6; D_1,8,5,1; D_2,8,7,1;]

$$sp(T_1, T_2) = [D_1,1,3,5;D_1,1,3,6;D_1,8,1,5;D_1,8,5,1;D_1,11,2,6;D_2,4,4,4;D_2,8,7,1;] = \underbrace{[D_1,1,3,5;D_1,1,3,6;]}_{\text{uzastopne reči}}$$

(b) T_1 SUSEDNO SA T_3
$T_1 \rightarrow$ [D_1,1,3,5; D_1,8,1,5; D_1,11,2,3; D_2,4,4,4;]
$T_3 \rightarrow$ [D_1,11,2,6; D_1,11,2,8; D_2,4,4,5; D_2,8,7,5;]

$$sp(T_1, T_3) = [D_1,1,3,5;D_1,8,1,5;D_1,11,2,3;D_1,11,2,6;D_1,11,2,8;D_2,4,4,4;D_2,4,4,5;D_2,8,7,5;] = \underbrace{[D_2,4,4,4;D_2,4,4,5;]}_{\text{uzastopne reči}}$$

(c) T_1 U BLIZINI NAJVIŠE 6 T_3
$T_1 \rightarrow$ [D_1,1,3,5; D_1,8,1,5; D_1,11,2,3; D_2,4,4,4;]
$T_3 \rightarrow$ [D_1,11,2,6; D_1,11,2,8; D_2,4,4,5; D_2,8,7,5;]

$$a = sp(T_1, T_3) = [D_1,1,3,5;D_1,8,1,5;D_1,11,2,3;D_1,11,2,6;D_1,11,2,8;D_2,4,4,4;D_2,4,4,5;D_2,8,7,5;] = \underbrace{[(D_1,11,2,3;D_1,11,2,6;D_1,11,2,8);(D_2,4,4,4;D_2,4,4,5);]}_{\text{najviše 5 reči između}}$$

(d) T_2 U BLIZINI NAJVIŠE 5 T_3
$T_2 \rightarrow$ [D_1,1,3,6; D_1,8,5,1; D_2,8,7,1;]
$T_3 \rightarrow$ [D_1,11,2,6; D_1,11,2,8; D_2,4,4,5; D_2,8,7,5;]

$$a = sp(T_2, T_3) = [D_1,1,3,6;D_1,8,5,1;D_1,11,2,6;D_1,11,2,8;D_2,4,4,5;D_2,8,7,1;D_2,8,7,5;] = \underbrace{[(D_1,11,2,6;D_1,11,2,8);(D_2,8,7,1;D_2,8,7,5);]}_{\text{najviše 4 reči između}}$$

4.
$D_1 \rightarrow$ [T_1,0.3; T_2,0.8; T_3,0.1;]
$D_2 \rightarrow$ [T_1,0.7; T_2,0.1; T_3,0.45;]
$D_3 \rightarrow$ [T_1,0.2; T_2,0.4;]
$D_4 \rightarrow$ [T_1,0.1; T_3,0.4;]

$$w_1 = 0.3 + 0.8 + 0.1 = 1.2$$
$$w_2 = 0.7 + 0.1 + 0.45 = 1.25$$
$$w_3 = 0.2 + 0.4 + 0 = 0.6$$
$$w_4 = 0.1 + 0 + 0.4 = 0.5$$

$$\underbrace{[(D_2, 1.25), (D_1, 1.2), (D_3, 0.6), (D_4, 0.5),]}_{\text{prvo najrelevantniji}}$$

5. (a) $p \wedge (q \vee r)$

p	q	r	$q \vee r$	$p \wedge (q \vee r)$
$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$$p \wedge (q \vee r) \Longleftrightarrow (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r)$$

(b) $(p \vee q) \wedge (q \vee r)$

p	q	r	$p \vee q$	$q \vee r$	$(p \vee q) \wedge (q \vee r)$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$$(p \vee q) \wedge (q \vee r) \Longleftrightarrow (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r)$$

(c) $(\neg p \vee r) \wedge (\neg r \vee q)$

p	q	r	$\neg p$	$\neg p \vee r$	$\neg r$	$\neg r \vee q$	$(\neg p \vee r) \wedge (\neg r \vee q)$
$\top$	$\top$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$

$$(\neg p \vee r) \wedge (\neg r \vee q) \Longleftrightarrow (p \wedge q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge \neg r)$$

(d) $(p \vee r) \wedge (\neg r \vee \neg q)$

p	q	r	$p \vee r$	$\neg r$	$\neg q$	$\neg r \vee \neg q$	$(p \vee r) \wedge (\neg r \vee \neg q)$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$

$$(p \vee r) \wedge (\neg r \vee \neg q) \Longleftrightarrow (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$$

6. (a) $T_1 \wedge T_2 \wedge T_3$

	T_1	T_2	T_3	$T_1 \wedge T_2 \wedge T_3$
D_1	0.3	0.8	0.1	0.1
D_2	0.7	0.1	0.45	0.1
D_3	0.2	0.4	0	0
D_4	0.1	0	0.4	0

$$[(D_1, 0.1), (D_2, 0.1), (D_3, 0), (D_4, 0)]$$

(b) $T_1 \vee T_2 \vee T_3$

	T_1	T_2	T_3	$T_1 \vee T_2 \vee T_3$
D_1	0.3	0.8	0.1	0.8
D_2	0.7	0.1	0.45	0.7
D_3	0.2	0.4	0	0.4
D_4	0.1	0	0.4	0.4

$$[(D_1, 0.8), (D_2, 0.7), (D_3, 0.4), (D_4, 0.4)]$$

(c) $(T_1 \wedge T_2) \vee T_3$

	T_1	T_2	T_3	$T_1 \wedge T_2$	$(T_1 \wedge T_2) \vee T_3$
D_1	0.3	0.8	0.1	0.3	0.3
D_2	0.7	0.1	0.45	0.1	0.45
D_3	0.2	0.4	0	0.2	0.2
D_4	0.1	0	0.4	0	0.4

$$[(D_2, 0.45), (D_4, 0.4), (D_1, 0.3), (D_3, 0.2)]$$

(d) $(T_1 \wedge T_2) \vee (T_2 \wedge T_3)$

	T_1	T_2	T_3	$T_1 \wedge T_2$	$T_2 \wedge T_3$	$(T_1 \wedge T_2) \vee (T_2 \wedge T_3)$
D_1	0.3	0.8	0.1	0.3	0. 1	0.3
D_2	0.7	0.1	0.45	0.1	0.1	0.1
D_3	0.2	0.4	0	0.2	0	0.2
D_4	0.1	0	0.4	0	0	0

$$[(D_1, 0.3), (D_3, 0.2), (D_2, 0.1), (D_4, 0)]$$

7. (a) (predviđanje OR predikcija) AND NOT (nagađanje OR špekulisanje)
   (b) ((a OR A) AND (b OR B)) OR ((c OR C) AND NOT (d OR D))
8. Prošireni indeks:

```
/KAP, /KAPA, /MAK, /RAK
A/KAP, AK/M, AK/R, AP/K
APA/K, K/MA, K/RA, KAP/
KAPA/, MAK/, P/KA, PA/KA, RAK/
```

Tabela upita:

originalni upit	preformulisani upit	sistem pronalazi	originalni termini
KAP	KAP/, /KAP	KAP/, /KAP	KAP
KA*	/KA	/KAP, /KAPA	KAP, KAPA
*AK	AK/	AK/M, AK/R,	MAK, RAK
AP	AP	AP/K, APA/K	KAP, KAPA
K*A	A/K	A/KAP	KAPA

9. Prošireni indeks:

```
/MAK /MANA, /RANA, /RIMA
A/MAN, A/RAN, A/RIM, AK/M
ANA/M, ANA/R, IMA/R
K/MA, MA/RI, MAK/, MANA/
NA/MA, NA/RA, RANA/, RIMA/
```

Tabela upita:

originalni upit	preformulisani upit	sistem pronalazi	originalni termini
MAK	MAK/, /MAK	MAK/, /MAK	MAK
MA*	/MA	/MAK, /MANA	MAK, MANA
*ANA	ANA/	ANA/M, ANA/R,	MANA, RANA
A	A	A/MAN, A/RAN, A/RIM, AK/M, ANA/M, ANA/R	ANA, MANA, RAK, RIMA
R*A	A/R	A/RAN, A/RIM	RANA, RIMA

10. Prošireni indeks:

```
/OKO, /ONA, /SKVO, /SOKO
A/ON, KO/O, KO/SO, KVO/S
NA/O, O/OK, O/SKV, O/SOK
OKO/, OKO/S, ONA/, SKVO/
SOKO/, VO/SK
```

Tabela upita:

originalni upit	preformulisani upit	sistem pronalazi	originalni termini
OKO	OKO/, /OKO	OKO/, /OKO	OKO
O*	/O	/OKO, /ONA	OKO, ONA
*KO	KO/	KO/O, KO/SO,	SOKO, OKO
K	K	KO/O, KO/SO, KVO/S	OKO, SOKO, SKVO
S*O	O/S	O/SKV, O/SOK	SKVO, SOKO

4. Dodatni aspekti sistema za pronalaženje inform

4.1 Mere za ocenjivanje efikasnosti pronalaženja

U prethodnim odeljcima pokazali smo invertovano-indeksnu reprezentaciju dokumenata njihovim indeksnim terminima (o tome kako se ti termini biraju, biće kasnije reči). Potom smo videli osnovne, logičke i vektorske upite, a zatim i razna proširenja tih upita: uključivanjem parametara blizine, dodavanjem sinonima, podsecanjem upitnih termina i slično.

Videli smo i da svakim pristupom nešto dobijamo, a nešto gubimo. Na primer, u slučaju dodavanja sinonima, upiti postaju komplikovaniji, ali bi teorijski trebalo da daju više relevantnih, odnosno da isključe što više nerelevantnih dokumenata. Slično, kod podsecanja dolazi do *eksplozije* indeksa u smislu povećanja broja indeksnih termina (zbog svih različitih rotacija), ali smo s druge strane dobili mogućnost da pojednostavimo upit, pritom formulišući mnogo više. S obzirom na dobijenu korist, zauzeće memorije uopšte ni ne deluje kao velika mana.

Ipak, sve je to samo intuitivna procena. Postavlja se pitanje kako se formalno mogu oceniti i uporediti svi ovi različiti sistemi? Da bismo odgovorili na ovo pitanje, prvo se moramo zapitati šta je kranji cilj.

Neka je data kolekcija dokumenata po koju korisnik posredstvom sistema za pronalaženje pretražuje kako bi dobio relevantne dokumente koji zadovoljavaju njegovu informacionu potrebu. Ukratko, idealan sistem će korisniku vratiti sve relevatne dokumente u odnosu na formulisani upit, odnosno neće prikazati dokumente koji nisu relevantni. Ali, da li je granica tako jasna? Odnosno, da li su svi upiti dovoljno precizni i nedvosmisleni? Primera radi, neka je korisnički upit samo termin PYTHON.[1] Da li korisnika interesuje životinja (zmija, piton) ili programski jezik?

Cilj sistema je da zadovolji korisnika, tako da je krajnja ocena sistema ustvari koliko je korisnik zadovoljan rezultatom. U nekim slučajevima, korisnik želi samo odgovor, odnosno želi da mu se prikažu dokumenti koji u putpunosti odgovaraju upitu. Ipak, u nekim situacijama, kao što je

1. Piton (engl. *Python*) je vrsta zmije, ali i programski jezik.

istraživački rad, korisniku ne smeta ni da ima *malo više* dokumenata u rezultatu, potencijalno i nešto manje relevantnih, ali iz kojih će moći da dođe i do neke druge informacije, zaključka, povezanosti i slično. Očito, nije lako zadovoljiti korisnika i uzeti u obzir sve moguće aspekte pretrage.

Da bi se za konkretan sistem dala ocena efikasnosti, moramo imati skup upita za testiranje na osnovu kojih će se proveravati relevantnost rezultata. Osim toga, neophodno je imati posebnu kolekciju dokumenata koja se zove **zlatni standard**. To je kolekcija dokumenata koja služi za testiranje sistema. Svakom dokumentu te kolekcije pridružena je po jedna binarna vrednost za svaki upit koja govori o tome da li je dati dokument relevantan ili ne u odnosu na postavljeni upit i takav skup parova (upit, dokument) naziva se **skup procena relevantnosti**.

Koliki treba da bude skup upita za testiranje, a koliki zlatni skup? Veličina zlatnog standarda treba da bude dovoljno velika i raznovrsna da rezultati merenja efikasnosti ne bi suviše zavisili od samog izbora dokumenata, a dovoljno mala da se njihova relevantnost može proceniti za svaku informacionu potrebu. U praksi te evaluacije vrši čovek, odnosno češće i više ljudi. Što se tiče broja upita, praksa pokazuje da je 50 donja granica. Uspešnost sistema za pronalaženje se potom procenjuje poređenjem rezultata sistema za pronalaženje koji se ocenjuje i zlatnog standarda.

Relevantnost se određuje relativno u odnosu na informacionu potrebu, a ne na upit. Sistem je mašina koja nema razumevanja o upitu, već će bukvalno protumačiti upit i vratiti rezultat koji je određen internim algoritmima sistema. Na primer, ako korisnik želi da u rezultatu vidi *Information on whether drinking red wine is more effective at reducing your risk of heart attacks than drinking white wine*, može da formuliše logički upit WINE AND RED AND WHITE AND HEART AND ATTACK AND EFFECTIVE (izostavljajući funkcionalne, neznačeće reči, poput veznika i članova). Dokument je relevantan ako odgovara informacionim potrebama korisnika, a ne ako slučajno sadrži reči sadržane u upitu. Ukoliko bismo ovaj upit prosledili Google pretraživaču, među odgovorima više njih će se odnositi na kurseve Pronalaženja informacija sa raznih svetskih Univerziteta. Zbog čega je to tako? Zbog toga što je navedeni primer preuzet iz knjige **jackson2007natural**, koji su koristili i drugi predavači na srodnim kursevima.

U narednim odeljcima biće opisane neke standardne mere za za ocenu sistema za pronalaženje.

Preciznost i odziv

Dva glavna parametra za ocenjivanje efikasnosti pronalaženja koja su u upotrebi su:

Preciznost odnos izdvojenih relevantnih dokumenata i ukupno izdvojenih dokumenata (engl. *Precision*, oznaka *P*) i

Odziv odnos izdvojenih relevantnih dokumenata i ukupno relevantnih dokumenata (engl. *Recall*, oznaka *R*).

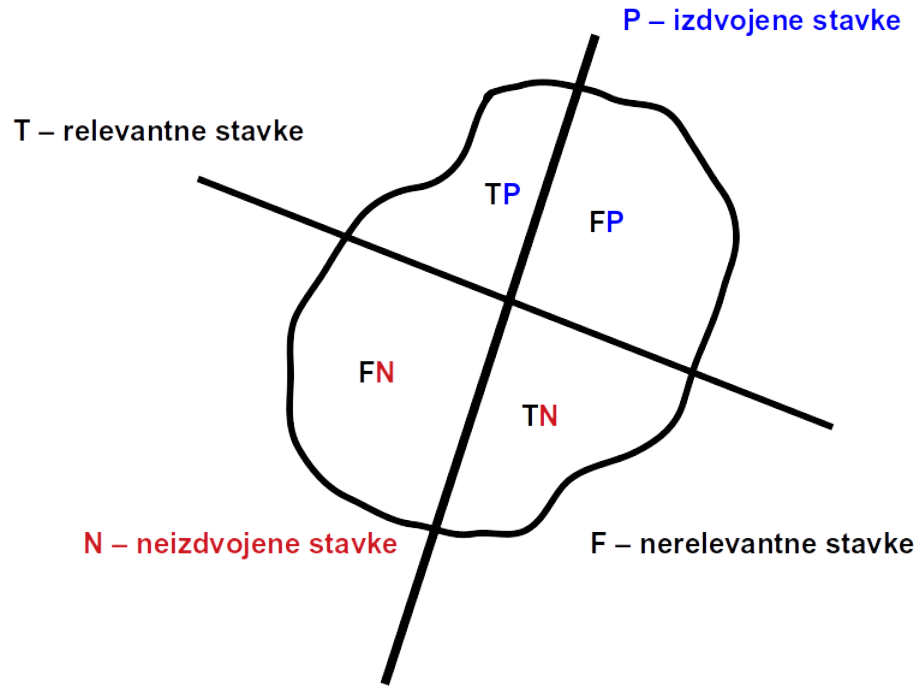
Ova dva parametra su **obrnuto proporcionalna**: za uske upite, preciznost je velika – skoro sve izdvojeno je i relevantno – ali je odziv mali jer je u stvari jako malo relevantnih stavki pronađeno. Kako se formulacije upita šire, ukupan broj pronađenih relevantnih dokumenata raste, što povećava odziv. Ali u isto vreme raste i broj izdvojenih nerelevantnih stavki, a to smanjuje preciznost. To jest, uski upiti daju visoku preciznost a mali odziv, dok široki upiti daju obrnuti rezultat - veliki odziv a malu preciznost.

Postavljanjem upita može se reći da se kolekcija dokumenata deli na četiri dela. Prva podela je na dokumente koji su izdvojeni i koji su se prijavili u rezultatu (obeleženi kao pozitivni, engl. *positive*, oznaka *P*) i koji nisu izdvojeni (obeleženi kao negativni, engl. *negative*, oznaka *N*). Druga podela je na dokumente koji jesu relevantni (engl. *True*, oznaka *T*) i na one koji nisu relevantni u odnosu na postavljeni upit (engl. *False, F*).

Među dokumentima koji su izdvojeni P, dakle, imamo one koji jesu relevantni – **relevantni izdvojeni dokumenti** odnosno **stvarno pozitivni** (engl. *True Positives*, oznaka TP) i one koji nisu relevantni – **nerelevantni izdvojeni dokumenti** odnosno **lažno pozitivni** (engl. *False Positives*, oznaka FP).

Slično, među neizdvojenim dokumentima N, potencijalno ima onih koji je trebalo da budu izdvojeni – **relevantni neizdvojeni dokumenti** odnosno **lažno negativni** (engl. *False Negatives*, oznaka FN) i onih koji zaista nisu relevantni i dobro je što nisu izdvojeni – **nerelevantni neizdvojeni dokumenti** odnosno **stvarno negativni** (engl. True Negatives).

Ova podela vizuelno je prikazana na slici 4.1.



Slika 4.1: Podela kolekcije dokumenata upitom na četiri dela.

Kako je preciznost odnos relevantnih izdvojenih dokumenata (TP) i ukupan broj izdvojenih dokumenata ($TP+FP$), preciznost se računa kao:

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

Odziv je odnos relevantnih izdvojenih dokumenata (TP) i ukupan broj relevantnih dokumenata ($TP+FN$), računa se kao:

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

Kad je upit uzak i precizan, biće mali broj izdvojenih dokumenata, ali će se među njima uglavnom naći oni koji jesu relevantni. Istovremeno, potencijalno će biti mnogo neizdvojenih relevantnih dokumenata: dakle, visoka preciznost, ali mali odziv. S druge strane, što je upit uopšteniji i širi, biće

više izdvojenih dokumenata, pa će se među njima naći i više relevantnih izdvojenih dokumenata, ali i više nerelevantnih izdvojenih. Dakle, što je veći odziv, manja je preciznost.

Odziv uvek može da bude visok, čak 1 (tj. 100%) ako ponudimo korisniku sva dokumenta kolekcije. Odziv je neopadajuća funkcija broja pronađenih dokumenata, tj. kako raste broj izdvojenih dokumenata, raste i odziv. Preciznost obično opada sa brojem pronađenih dokumenata, tj. kako raste broj izdvojenih dokumenata preciznost pada.

Tačnost

Tačnost (engl. *Accuracy*, oznaka *A*) je mera koja utvrđuje koliki deo klasifikovanih dokumenata je ispravno klasifikovan, tj. ispravno smešten u grupu relevantnih odnosno nerelevantnih dokumenata. To je odnos broja dobro klasifikovanih dokumenata (relevantni izdvojeni i nerelevantni izdvojeni) i ukupne veličine kolekcije $N = TP + TN + FP + FN$, pa računa se kao:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Ova mera nije naročito pouzdana. Razlog tome je što su podaci u realnoj situaciji najčešće vrlo iskrivljeni. Šta ovo znači? Naime, najčešće je najveći deo dokumenata kolekcije nerelevantan, i do 99,9%. Ako bismo želeli da podesimo sistem na maksimalnu tačnost dovoljno je da odbacimo sva dokumenta i dobićemo tačnost od skoro 100%. Za korisnike je to nepoželjno jer oni najčešće žele da dobiju bar neki odgovor, čak iako je on izmešan sa dosta nerelevantnih odgovora.

F mera

Radi se o parametru koji meri izbalansiranost odziva i preciznosti. U opštem slučaju, korisnik želi da dobije određen nivo odziva pri čemu će tolerisati određen procenat pogrešno određenih relevantnih odgovora.

U posebnom slučaju, korisnik želi da odgovor dobije odmah i to tako da mu svi rezultati na prvoj stranici budu relevantni – dakle, poželjna je visoka preciznost. Ipak, kao što je ranije pomenuto, istraživač ili naučnik želi da pronađe sve relevantne dokumente, bez obzira i ako će zato morati da pregleda i mnogo nerelevantnih ponuđenih – ovakvom korisniku važniji je odziv od prezicnosti.

Podsetimo se, aritmetička sredina dve vrednosti a i b računa se kao $A = \frac{a+b}{2}$, dok se geometrijska sredina računa kao $G = \sqrt{ab}$. Što se tiče harmonijske sredine vrednosti a i b, ona se izražava kao $H = \frac{1}{\frac{1}{a}+\frac{1}{b}}$.

Uvek važi da je harmonijska sredina manja od geometrijske sredine, dok je geometrijska još manja od aritmetičke sredine, neformalno: $H \leq G \leq A$. Svojstvo harmonijske sredine je da ona naginje manjoj od dve vrednosti i da teži da neutrališe uticaj veće, a potencira značaj manje vrednosti.

F-mera se zasniva se upravo na harmonijskoj sredini preciznosti i odziva, pa se računa kao:

$$F = \frac{1}{\frac{1}{P} + \frac{1}{R}}$$

Zašto ne koristimo aritmetičku sredinu za procenu? Kako uvek možemo da dobijemo odziv 100% ako izdvojimo sva dokumenta, preciznost će biti skoro 0, ali aritmetička sredina će biti 50%, tako da ispada da je rezultat pronalaženja skoro dobar.

Kako se može dati prednosti odzivu ili preciznosti? Može se posmatrati težinska harmonijska sredina odziva i preciznosti:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

pri čemu je $\alpha \in [0,1]$ parametar kojim se daje značaj preciznosti ili odzivu.

Na primer, za $\alpha = 0$:

$$F = \frac{1}{0\cdot\frac{1}{P} + (1-0)\frac{1}{R}} = \frac{1}{\frac{1}{R}} = R$$

Za $\alpha = 1$:

$$F = \frac{1}{1\cdot\frac{1}{P} + (1-1)\frac{1}{R}} = \frac{1}{\frac{1}{P}} = P$$

Dakle, ako je na primer $\alpha = \frac{2}{3}$, želimo da damo prednost preciznosti (npr. u slučaju prosečnog veb surfera). Ako bi pak bilo $\alpha = \frac{1}{3}$, prednost želimo da damo odzivu.

Slično, za $\alpha = \frac{1}{2}$:

$$F = \frac{1}{\frac{1}{2P} + \frac{1}{2R}} = \frac{1}{\frac{P+R}{2PR}}$$

što je upravo izraz za harmonijsku sredinu.

■ **Primer 4.1** Neka je 1 dokument kolekcije relevantan ($TP+FN=1$) i neka se on TP našao među $10,000$ ($TP+FP=10,000$) izdvojenih dokumenata. Tada je odziv

$$R = \frac{TP}{TP+FN} = \frac{1}{1} = 1$$

(dakle, 100%).

Slično, preciznost se dobija kao

$$P = \frac{TP}{TP+FP} = \frac{1}{10,000} = 0.0001$$

(ili 0.01%).

Aritmetička sredina je tada

$$\frac{P+R}{2} = \frac{1+0.0001}{2} = 0.50005$$

(ili 50.005%).

Harmonijska sredina je ipak nepristrasnija:

$$\frac{2PR}{P+R} = \frac{2\cdot 1\cdot 0.0001}{1+0.0001} = 0.0002$$

(ili 0.02%). ■

■ **Primer 4.2** Treba oceniti sistem za pronalaženje informacija. Neka zlatni standard sadrži 240,000 dokumenata. U odnosu na postavljeni upit q, u kolekciji je 24,600 dokumenata obeleženo kao relevantno, ali je u rezultatu od toga izdvojeno 12,800 dokumenata. Među izdvojenim dokumentima pojavljuje se još 3,600 dokumenata kolekcije koji nisu relevantni za upit q. Oceniti sistem izračunavanjem preciznosti, odziva, tačnosti i F-mere.

Označimo prvo date podatke adekvatnim oznakama. Ukupan broj dokumenata kolekcije za testiranje (zlatnog standarda) je $N = 240,000$.

Dokumenti koji su relevantni su $TP + FN = 24,600$, a izdvojeni relevantni su $TP = 12,800$. Odavde možemo odrediti $FN = 24,600 - 12,800 = 11,800$.

U rezultatu se pojavljuju dokumenti koji nisu relevantni, pa su oni ustvari $FP = 3,600$.

Sada možemo odrediti i četvrtu vrednost, TN. Ako je $N = FP + TP + FN + TN$, onda je $TN = N - (FP + TP + FN) = 240,000 - (3,600 + 12,800 + 11,800) = 211,800$.

Konačno, prezicnost je $P = \frac{TP}{TP+FP} = \frac{12,800}{12,800+3,600} \sim 0.78$.

Dalje, odziv računamo kao $R = \frac{TP}{TP+FN} = \frac{12,800}{12,800+11,800} \sim 0.52$.

Računamo F-meru $F = \frac{2PR}{P+R} \sim \frac{2 \cdot 0.78 \cdot 0.52}{0.78+0.52} \sim 0.62$.

Tačnost se dobija kao $A = \frac{TP+TN}{N} = \frac{12,800+211,800}{240,000} \sim 0.94$.

Ne bi bilo u redu dati ocenu sistema samo u odnosu na jedan upit (kao što smo rekli ranije, minimalni broj je makar 50). Ipak, na osnovu dobijenih vrednosti možemo prokomentarisati da sistem ima bolju preciznost nego odziv, pri čemu je preciznost solidna. Gledajući meru tačnosti, sistem ima odlične performanse. Ipak, ovo je samo zbog toga što ima dosta nerelevantnih dokumenata koji se *nisu* pojavili u rezultatu. F-mera je ipak nepristrasnija i ona nam daje merodavniju ocenu celokupnog sistema, za koji se može reći da ima prostora za poboljšanjem. ■

4.2 Organizacija invertovano-indeksnih datoteka

Kao što smo do sad videli, u invertovanim indeksima svakom terminu (ili ključu) je pridružena potencijalno dugačka lista identifikatora dokumenata. Mnogim ključevima su pridružene iste liste identifikatora dokumenata, npr. u slučaju sinonima, kao i kod ključeva u indeksima koji obrađuju upite sa mogućnošću podsecanja. U ovakvim slučajevima bi bilo efikasnije da se koriste indeksi u kojima je uz ključ zapisan samo pokazivač ka listama identifikatora dokumenata. Tako će npr. dva sinonima pokazivati na jedinstvenu listu u memoriji, odnosno sadržaće informaciju o memorijskoj lokaciji te liste.

Pitanje koje postavljamo u ovom odeljku je sledeće: kako u inverovatnom indeksu pronaći ključ koji se traži? Postoji više pristupa, od kojih ćemo izdvojiti neke koji su najkarakterističniji.

Liste i sekvencijalni pristup

Kako je na početku rečeno, ključevi u indeksu su na neki način uređeni, najčešće u leksikografskom poretku. U slučaju sekvencijalnog, odnosno linearnog upita, upitni termin ispituje poređenjem redom sa svakim ključem u indeksu sve dok se ne pronađe, dok se ne dođe do kraja liste ili sve dok se ne dođe do termina koji mu sledi u alfabetskom poretku. U poslednjem slučaju, pošto je indeks uređen, nema potrebe ispitivati naredne termine, jer se traženi termin sigurno neće naći u listi nakon aktuelne pozicije.

Vreme potrebno da se obavi sekvencijalna pretraga zavisi prvenstveno od samog upitnog termina, jer od njega zavisi pozicija u indeksu. Dalje, zavisi i od broja termina u indeksu N. Za sekvencijalno pretraživanje uređenog indeksa potrebno je otprilike $N/2$ poređenja i u slučaju da je termin prisutan u listi i u slučaju da nije prisutan.

■ **Primer 4.3** Neka se sekvencijalno pretražuje lista ključeva:

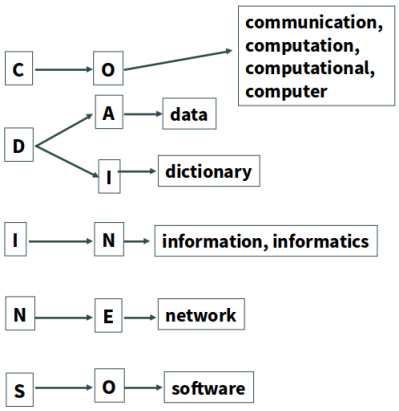
```
communication
computation
computational
computer
data
dictionary
information
informatics
network
software
```

U slučaju da se traži termin *dictionary*, pretraga se završava nakon 6 koraka (jer je termin prisutan i to kao šesti u listi). U slučaju da se traži termin *computing*, dovoljno je ispitati prvih 5 termina: kada se dođe do termina *data*, staje se sa pretragom, jer je reč *computing* leksikografski manja od reči *data*, te je sigurno nema smisla tražiti posle tog termina. ■

Alfabetski lanci

Pretraga ove strukture nalikuje pretrazi rečnika. Ako npr. tražimo termin *psychology* u rečniku ćemo prvo pronaći slovo *P*, a zatim bismo gledali u zaglavlja stranica dok ne nađemo *PS*, a dalje bismo tražili linearno. Kod alfabetskih lanaca, prvo pronalazimo kutijice po prvom slovu, a odatle sledimo pokazivač ka novim kutijicama, itd. dok ne nađemo šta tražimo, ili ne dođemo do liste koju ćemo linearno pretraživati.

■ **Primer 4.4** Ilustracija strukture alfabetskih lanaca za invertovani indeks dat u primeru 4.3 prikazana je na slici 4.2.



Slika 4.2: Ilustracija alfabetskih lanaca.

Ako bi se tražio termin *computational*, prvo se pronalazi slovo *C*, pa *O* i dolazimo do svih termina koji počinju na *CO-*. ■

Problem sa alfabetskim lancima je što su u realnim situacijama neki termini vrlo bliski jedni drugima, tj. neka slova su mnogo gušće popunjena od ostalih pa će od toga zavisiti koliko poređenja je potrebno dok ne nađemo termin (ili utvrdimo da ga nema). Na primer, *computation, computations, computational, computer, computers, computerisation*, etc.

Liste i binarno pretraživanje

Za velike indeksne datoteke sekvencijalno pretraživanje zahteva mnogo vremena. Jedno poboljšanje se sastoji u tome da se smanji broj poređenja potrebnih da se utvrdi prisutnost, odnosno odsutnost traženog termina. Primenom tehnike binarnog pretraživanja se broj poređenja može efikasno redukovati.

Kod binarne pretrage, osnovna ideja leži u tome da se u svakom koraku pretraživanja indeksa broj indeksnih termina koje potencijalno treba pretražiti prepolovi. Načelno, termin koji se traži se poredi sa terminom koji se nalazi u sredini intervala indeksnih termina preostalih za pretraživanje. Prilikom ispitivanja središnjeg termina moguće su tri situacije:

- Indeksni termin je jednak središnjem terminu – termin je pronađen;
- Indeksni termin je manji od središnjeg termina – interval sledeće pretrage je leva (odnosno donja) polovina tekućeg intervala;
- Indeksni termin je veći od središnjeg termina – interval sledeće pretrage je desna (odnosno gornja) polovina tekućeg intervala;

Algoritam binarne pretrage

Neka je n broj indeksnih termina u indeksu, a *ključ* je indeksni termin koji se traži.

```
početak
        levi ← 1
        desni ← n
        sve dok je levi ≤ desni ponavljaj
                srednji ← (levi + desni) div 2
                ako je ključ = indeks[srednji]
                        onda je ključ pronađen
                inače ako je ključ < indeks[srednji]
                        onda je desni ← srednji-1
                inače levi ← srednji+1
kraj
```

U opštem slučaju je broj poređenja kod binarnog pretraživanja $O(log_2 n)$, odnosno *reda veličine binarnog logaritma od broja indeksnih termina*. Logaritamska i eksponencijalna funkcija su inverzne funkcije. Kako nam ova informacija koristi u ovom slučaju? Na primer, ako je $2^3 = 8$, onda je $log_2 8 = log_2 2^3 = 3 \cdot log_2 2 = 3 \cdot 1 = 3$, što znači da će u slučaju indeksa koji ima 8 indeksnih termina biti potrebno najviše 3 poređenja da bi se utvrdilo da li je *ključ* prisutan ili nije prisutan u indeksu.

■ **Primer 4.5** Ilustracija binarnog pretraživanja za invertovani indeks dat u primeru 4.3 i ključ *network*.

■

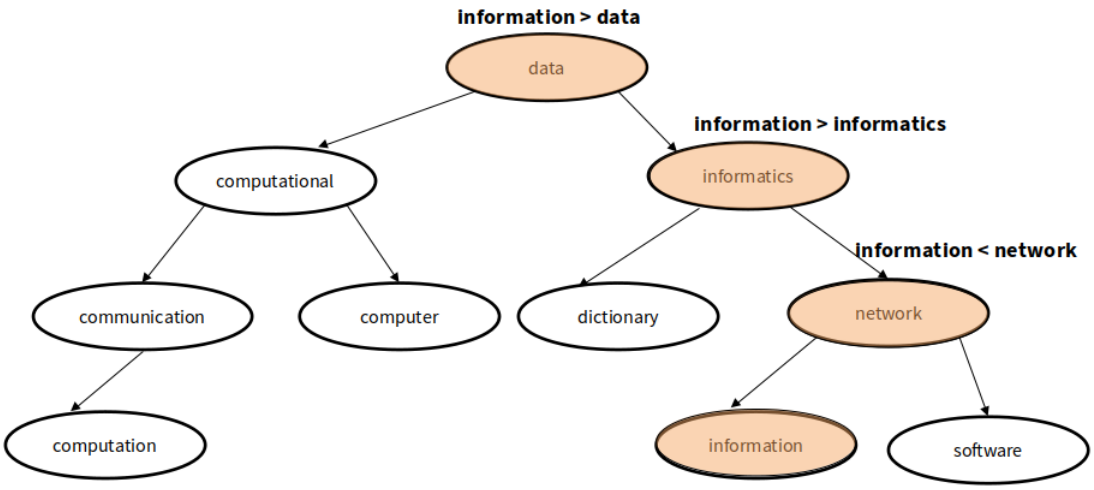
srednji	indeks[srednji]	poređenje	levi	desni
			1	10
5	data	network > data	6	10
8	informatics	network > informatics	9	10
9	network	pronađen		

Binarno pretraživačko stablo

Osnovni problemi binarnim pretraživanjem leži u otežanom dodavanju ili izbacivanju elemenata iz indeksa, jer pri svakom ažuriranju indeks treba ponovo sortirati. Struktura binarnog pretraživačkog stabla se sastoji od čvorova od kojih svaki sadrži indeksni termin i pokazivač ka levom i desnom podstablom koja redom sadrže indeksne termine koji mu prethode, odnosno koji ga slede u izabranom poretku.

Organizacija stabla podržava operacije kao što su pronalaženje ključa, umetanje indeksnog termina, brisanje indeksnog termina.

■ **Primer 4.6** Izgled binarnog pretraživačkog stabla za invertovani indeks dat u primeru 4.3 dat je na slici 4.3.



Slika 4.3: Binarno pretraživačko stablo.

Uopšteno govoreći, za svaki čvor stabla važi da se sa njegove leve strane nalaze potomci koji su *manji ili jednaki*, a sa desne strane potomci koji su *veći* od njega, u zavisnosti od vrste sadržaja i poretka među odgovarajućim tipom podataka koji se čuva u čvorovima. Kod indeksa, u odnosu na neki čvor, levi potomci su termini koji su leksikografski manji ili jednaki, a desni leksikografski veći termini. Ovo se pravilo primenjuje rekurzivno na svaki čvor, počevši od korena.

Pretraga teče isto kao i dodavanje čvora. Postupak se zaustavlja ili kada se termin pronađe (odnosno, sadržan je u korenu nekog podstabla) ili kada se dođe do lista (čvora koji nema potomke), kada se konstatuje da termin nije prisutan u stablu. Na slici 4.3 je ilustrovana pretraga termina *information*. Prvo se *information* poredi sa *data*, koje se nalazi u korenu stabla. Pošto je termin *information* leksikografski veći, ide se u desno podstablo. Koren desnog podstabla sadrži termin *informatics*. Kako je traženi termin *information* leksikografski veći od termina *informatics*, opet se ide u desno

podstablo. U korenu desnog podstabla je termin *network*. Kako je traženi termin leksikografski manji, ide se u levo podstablo. U korenu levog podstabla se nalazi traženi termin i pretraga se zaustavlja pozitivnim ishodom.

Kako se gradi ovakvo stablo? Prvo, termini ne treba da budu sortirani, već se dodaju u stablo onako kako se izdvajaju. Štaviše, ako je ulazna lista termina sortirana, stablo će biti neizbalansirano, odnosno svi termini će se stalno dodavati sa desne strane (u slučaju rastuće sortirane liste), odnosno sa leve strane (u slučaju opadajuće sortiranih termina). Prvi termin koji se dodaje postaje koren. Za sledeći termin se ispituje da li je leksikografski manji ili jednak u odnosu na koreni. Ako jeste, on postaje levi potomak, a inače se dodaje kao desni potomak. Za svaki sledeći čvor, postupak se ponavlja. Poređenje uvek kreće od korena. Ako je termin leksikografski manji ili jednak, ide se u levo podstablo, a u suprotnom se ide u desno podstablo. Tada se procedura ponavlja za odgovarajuće podstablo, tako što se poređenje vrši ponovo od korena podstabla. Postupak se zaustavlja dodavanjem čvora kao potomka korenu podstabla koje nema potomke, odnosno koje je list.

■

Šta bi bili nedostaci ovakve strukture? Izbacivanje u opštem slučaju nije jednostavno:

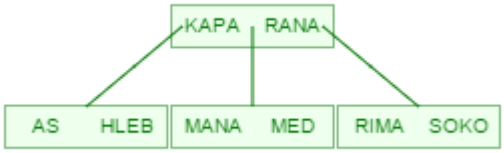
- Ako je čvor koji treba izbaciti list drveta, čvor se samo isključi;
- Ako čvor koji se izbacuje ima samo jedno dete, onda se taj čvor zamenjuje tim jedinim detetom;
- Ako čvor koji se izbacuje ima više dece, onda se on rekurzivnom procedurom zamenjuje čvorom koji ima najmanju vrednost u desnom podstablu.

Druga mana je, kako je već nagovešteno, neizbalansiranost: stablo raste u dubinu i ubrzo postaje nebalansirano – neke grane su veoma dugačke, a od dužine grane zavisi broj poređenja pre nego što se termin pronađe u indeksu ili se utvrdi da ga nema.

Balansirano B-stablo

Kod ovog stabla iz svakog čvora polazi više grana, a ne samo dve kao kod binarnog. Svaki čvor se sastoji iz uređene liste termina, a uz svaki termin ide i pokazivač. Kada se traži ključem pronalazi se u čvoru prvi termin koji sledi iza ključa u izabranom poretku – na tom mestu se prati pokazivač koji vodi ka sledećem čvoru.

Kapacitet B-stabla zavisi od broja termina u čvoru – na slici 4.4 kapacitet stabla je dva termina po čvoru, a broj pokazivača 3 (što upravo i predstavlja parametar B u nazivu). Ono nam omogućava da prilikom svakog ažuriranja stabla (dodavanja ili izbacivanja čvorova), stablo ostaje izbalansirano, odnosno da broj nivoa i u levom i u desnom podstablu bude jednak.



Slika 4.4: Balansirano 3-stablo.

Kako izgleda dodavanje ključeva u balansirano B-stablo? Umetanje počinje od korenog čvora, kao i kod binarnog pretraživačkog stabla. Da bi se umetnuo novi element treba prvo pretražiti

stablo kako bi se pronašao list u koji novi element treba umetnuti. Novi element se stablu dodaje u nekoliko koraka:

- Ako čvor sadrži manje elemenata od maksimalne dužine u čvoru, onda ima mesta za novi element pa ga treba umetnuti tako da elementi u čvoru ostanu uređeni.
- Ako je čvor popunjen treba ga podeliti na dva jednaka dela:
  - Izabere se središnji element među elementima iz lista koji se deli i novog elementa.
  - Vrednosti koje su manje od središnjeg elmenta idu u novi levi list, a vrednosti koje su veće idu u novi desni list, dok je središnji element onaj koji ih razdvaja.
  - Ubaciti središnju vrednost u nadređeni čvor ako u njemu ima mesta, a ako nema on mora da se deli. Ako čvor nema nadređeni čvor, onda se kreira novi koren i visina drveta raste.

Glavna prednost ovog stabla, osim što osigurava balansiranost, jeste i ta što je broj pristupa indeksu (tj. broj poređenja odnosno pokušaja) srazmeran broju nivoa drveta (slično kao i kod binarnih drveta). Ipak, pošto se blokovi uvek dele umesto da se dodaju nove grane na dno drveta, novi nivoi se ne prave sve dok to nije apsolutno neophodno. Na taj način se smanjuje srednje pristupno vreme indeksu.

Metode heširanja i heš tabele

Ovaj pristup se sastoji u transformaciji indeksa u heš-tabelu koja svodi oblike reči promenljivih dužina u kratke kodove (brojeve) fiksne dužine. Dobijeni kod se koristi kao adresa odgovarajućeg termina u invertovanom indeksu (npr. adresa termina u datoteci ili u nizu u unutrašnjoj memoriji).

Funkcija koja prevodi reč promenljive dužine u heš-kod, odnosno adresu, mora da bude takva da, u najvećem broju slučajeva, različiti termini promenljive dužine imaju različit heš-kod (u optimalnom slučaju, funkcija treba da bude 1-1).

Heš-funkcija se, primera radi, može realizovati na sledeći način: binarni zapis prvih osam karaktera termina, shvaćeni kao ceo broj deli se sa željenom veličinom rečnika (koja treba da bude prost broj), a ostatak pri deljenju je traženi heš-kod.

Druga mogućnost je da se dve četvorke binarno zapisanih karaktera termina shvaćene kao celi brojevi, pomnože međusobno i da se iz dobijenog rezulata izdvoji jedan deo kao heš-kod.

■ **Primer 4.7** Neka je dat tekst: *Now is the time to read some books.*

Neka je svaka reč u indeksu, a heš funkcija je sledeća: heš-kod se dobija sabiranjem za svaku reč pozicije u engleskom alfabetu svakog slova reči. Tada se dobijaju sledeći heš-kodovi za indeks: now $\mapsto$ 52, is $\mapsto$ 28, the $\mapsto$ 33, time $\mapsto$ 47, to $\mapsto$ 35, read $\mapsto$ 28, some $\mapsto$ 54, books $\mapsto$ 41. Kako? Primera radi, za *now* se sabiraju brojevi 14 (pozicija slova n), 15 (pozicija slova o) i 23 (w je dvadeset i treće slovo u abecedi), što daje 52.

Iako je jednostavan i efikasan metod u pitanju, pojavljuje se problem: za samo 8 reči potrebno je bar 54 lokacije za skladištenje, iako bi se koristile samo lokacije 28, 33, 35, 41, 47, 52 i 54. Metod je moguće dopuniti tako da se svaka dobijena vrednost deli sa maksimalnom veličinom indeksa koja je prost broj, npr. brojem 19, a heš-kod je onda ostatak pri deljenju. Pošto je broj ostataka pri deljenju sa brojem p upravo broj p, u opštem slulaju potrebno je p lokacija za skladištenje, odnosno 19 u našem slučaju.

Takvom modifikacijom dobijaju se sledeće vrednosti: books: 41 mod 19 $\mapsto$ 7, is, read: 28 mod 19 $\mapsto$ 9, time: 47 mod 19 $\mapsto$ 9, now: 52 mod 19 $\mapsto$ 14, the: 33 mod 19 $\mapsto$ 14, to: 35 mod 19 $\mapsto$ 16, some: 54 mod 19 $\mapsto$ 16. ■

Šta je mana ovog metoda? Iako nije potrebno više od p lokacija za skladištenje, problem je u tome što neki termini treba da se smeste na iste lokacije (npr. is, time i read kojima je heš-funkcija dodelila istu vrednost 9). Ova pojava se naziva *kolizija*.

Prost broj za deljenje se upravo zato i koristi da bi se kolizija svela na što je moguće manju meru, ali se ipak ne može izbeći. Jednostavno rešavanje kolizije se može obaviti tako da se uz svaku adresu vezuje povezana lista svih termina čija je to heš-funkcija. Ta povezana lista može lako da se vodi i kao uređena, što čini pretraživanje efikasnijim.

Postoje i drugi pristupi prevazilaženja kolizije, kao što je, na primer, smeštanje termina kojem je dodata već zauzeta lokacija na prvu slobodnu lokaciju, ali takve tehnike nisu dovoljno efikasne. Kod ovako implementiranih eaš-funkcija narušava se leksikografski poredak termina u invertovanom indeksu, tako da će se veoma slične reči, na primer *SAND* i *SANE* naći na sasvim različitim mestima u indeksu. Rezultat je da se pretraživanje unutar opsega (recimo pronalaženje svih termina koji počinju sa SAN*), veoma teško realizuje.

4.3 Zadaci

1. Neka indeks neke kolekcije dokumenata sadrži sledeće termine:
   `KAPA, RANA, SOKO, MANA, RIMA, HLEB, MED, AS`
   Kako izgleda sortirani indeks i
   (a) linearno
   (b) binarno pretraživanje termina KAPA, RANA, RAK, KAP
2. Za date indeksne termine formirati
   (a) binarno pretraživačko stablo
   (b) balansirano B-stablo (kapaciteta 3), a zatim prikazati i postupak pretrage navedenih termina.
   Indeksni termini su:
   (a) KAPA, RANA, SOKO, MANA, RIMA, HLEB, MED, AS [pretražuje se: AS, RANA, LUK]
   (b) LUK, VODA, SLANINA, MESO, SO, BIBER, TIGANJ, BELI [pretražuje se: BIBER, SO, MED]
   (c) MAČKA, ANA, PERA, BISA, RISTA, STOP, CICA, NOJ [pretražuje se: BISA, NOJ, LUK]
3. Napraviti heš-tabelu za termine: BABA, CACA, DEDA, CECA, DACA, ACA i datu heš-funkciju. Ilustrovati potom dodavanje reči BEBA, DECA. Ilustrovati zatim i pretragu reči DACA, CACA, AB.

slovo	A	B	C	D	E
kod	1	2	3	4	5

$$f(s) = zbir\_kodova\_slova\_u\_s \mod 5$$

4.4 Rešenja

1. Termini su
   `KAPA, RANA, SOKO, MANA, RIMA, HLEB, MED, AS`
   (a) Sortirani termini: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOKO
   (b) Linearna pretraga
   KAPA: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK [nađen, 3 poređenja]
   RANA: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK [nađen, 6 poređenja]

RAK: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK [nije nađen, 8 poređenja]
KAP: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK [nije nađen, 3 poređenja]

(c) Binarna pretraga
KAPA: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK
AS, HLEB, KAPA
KAPA [nađen, 3 poređenja]

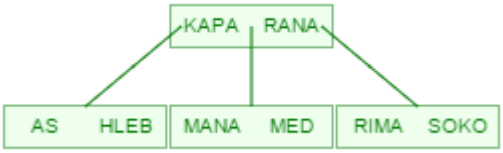
RANA: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK
MED, RANA, RIMA, SOK [nađen, 2 poređenja]

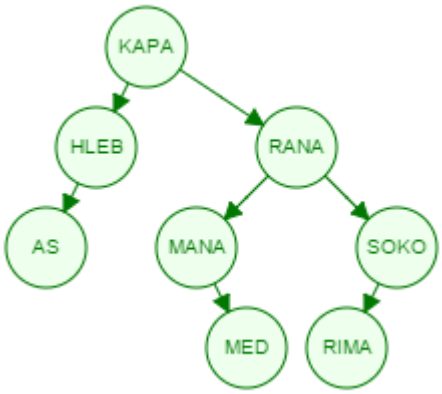
RAK: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK
MED, RANA, RIMA, SOK
MED [nije nađen, 3 poređenja]

KAP: AS, HLEB, KAPA, MANA, MED, RANA, RIMA, SOK
AS, HLEB, KAPA
KAPA [nije nađen, 3 poređenja]

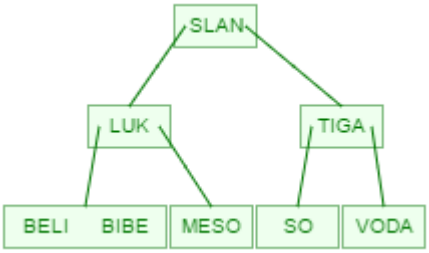
2. Termini su:

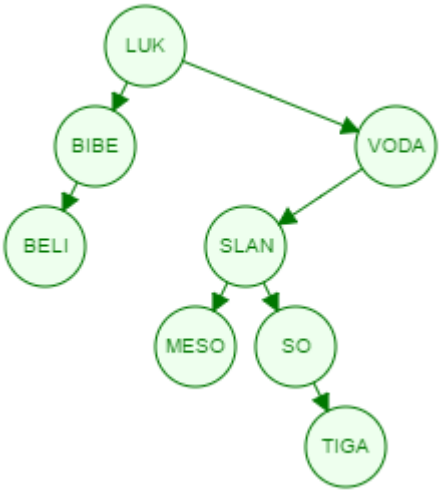
(a) `KAPA, RANA, SOKO, MANA, RIMA, HLEB, MED, AS`



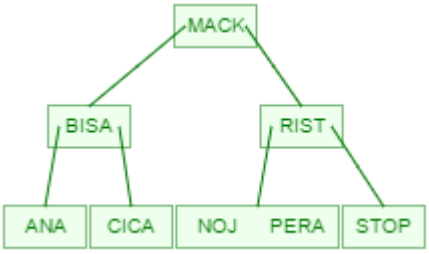


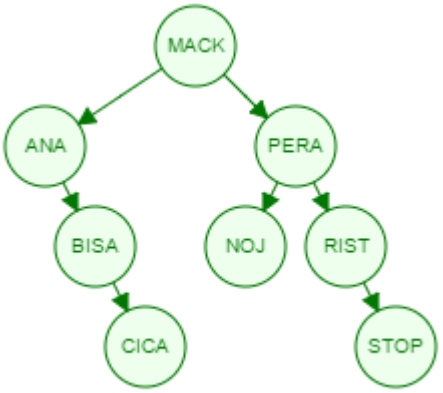
(b) `LUK, VODA, SLANINA, MESO, SO, BIBER, TIGANJ, BELI`





(c) `MAČKA, ANA, PERA, BISA, RISTA, STOP, CICA, NOJ`





3. Računanje vrednosti heš-funkcije:

$$
\begin{aligned}
f(BABA) &= (1+2+1+2) \mod 5 = 1 \\
f(CACA) &= (1+3+1+2+3) \mod 5 = 3 \\
f(DEDA) &= (4+5+4+1) \mod 5 = 4 \\
f(CECA) &= (3+5+3+1) \mod 5 = 2 \\
f(DACA) &= (4+1+3+1) \mod 5 = 4 \\
f(ACA) &= (1+3+1) \mod 5 = 0
\end{aligned}
$$

Heš-tabela:
$0 \rightarrow$ ACA
$1 \rightarrow$ BABA

$2 \rightarrow$ CECA
$3 \rightarrow$ CACA
$4 \rightarrow$ DEDA, DACA
Dodavanje novih termina:

$$f(BEBA) = (2+5+2+1) \mod 5 = 0$$
$$f(DECA) = (4+5+3+1) \mod 5 = 3$$

Heš-tabela nakon dodavanja:
$0 \rightarrow$ ACA, BEBA
$1 \rightarrow$ BABA
$2 \rightarrow$ CECA
$3 \rightarrow$ CACA, DECA
$4 \rightarrow$ DEDA, DACA

II Obrada teksta

5. Sistemi zasnovani na skaniranju teksta

Procedure za pronalaženje informacija koje smo videli u prethodnim poglavljima načelno se zasnivaju na korišćenju invertovanih indeksa za sve termine koji karakterišu dokumente kolekcije. Primena invertovanih indeksa je zgodna kod velikih datoteka, što je najčešće i slučaj, jer se brzo može dobiti odgovor koristeći informacije koje su dobijene prethodnom obradom i uskladištene u invertovanim indeksnim listama. Osnovni problem rada s invertovanim datotekama je njihovo kreiranje, ali još više održavanje. Na primer, kada se dodaju novi dokumenti, potrebno je ažurirati i listu termina (treba očuvati poredak termina), ali i listu dokumenata za svaki od postojećih termina.

Za manje datoteke, kada su dužine uskladištenih tekstova ograničene, odnosno kada su tekstovi ograničenog veka trajanja, ispostavlja se da je bolje prekršiti princip postavljen na početku, koji govori da se datoteci sme pristupiti tek onda kada se utvrdi da zadovoljava informacionu potrebu. Tada se odgovori na zahtev za informacijom mogu dobiti direktno pristupajući tekstu u glavnoj datoteci, bez konsultovanja nekog među-indeksa. U takvim slučajevima moraju se koristiti **operacije skaniranja teksta** koje sravnjuju reči i termine iz upita direktno sa tekstom dokumenta. Na primer, da bi se pronašli dokumenti koji se odnose na *računar*, svi dokumenti se moraju skanirati da bi se pronašli oni koja sadrže, recimo, reč (nisku) računar.

Operacije skaniranja teksta se zasnivaju na poređenju indvidualnih karaktera upita sa karakterima koji su uskladišteni u tekstu dokumenta. Kako raste broj karaktera u dokumentu, skaniranje postaje sve skuplje, i poređenje karakter-po-karakter tada postaje preskupo, dakle presporo, za praktično korišćenje. Tada se mora preći ili na tehnologiju invertovanih indeksa ili se moraju koristiti sistemi koji ubrzavaju proces skaniranja teksta.

Ipak, osnovni nedostatak ovakvih sistema je nefleksibilnost jer traže egzaktno slaganje. Na primer, ako je upitni termin (obrazac) *library* ne može sravniti sa pojavom niske *libraries* u tekstu. Kada se upiti formulišu kao karakterske niske od više reči, onda i redosled reči postaje dodatni problem. Primera radi, *information retrieval* se ne može direktno sravniti sa tekstom dokumenta koji sadrži frazu *retrieval of information*.

U narednom tekstu bavićemo se algoritmima skaniranja teksta, odnosno sravnjivanja upitnog

obrasca sa sadržajem dokumenta u cilju utvrđivanja prisutnosti termina koji predstavlja upitni obrazac u tekstu, imajući na umu njihovu efikasnost samo u slučajevima kraćih dokumenata.

5.1 Naivni algoritam skaniranja

Ovakav metod naziva se još i primenom **grube sile** (engl. *brute force*), a zbog čega je tako, govori sam postupak. Neka je obrazac $P = p_1 p_2 \dots p_m$, gde su $p_j, j \in \{1,2,\dots m\}$ individualni karakteri obrasca. Neka je tekst $S = s_1 s_2 \dots s_n$, gde su $s_i, i \in \{1,2,\dots n\}$ individualni karakteri obrasca i pritom je $n \gg m$ (dužina teksta je mnogo veća od dužine obrasca). Postupak se onda može opisati na sledeći način:

- Prolazi se kroz S i kroz P simultano, počevši od kranje levih karaktera obe niske.
- Dva pokazivača, i i j za niske S i P vode dalje računa o redosledu skaniranja.
- Na početku se oba pokazivača postavljaju na 1, pa su prva dva karaktera koja se porede s_1 i p_1.
- Kad god se karakteri koji se porede slažu, uvećava se pokazivač obrasca, pa se sledeći porede s_2 i p_2, zatim s_3 i p_3 i tako redom.
- Kad god se karakteri koji se porede ne slažu, obrazac se pomera udesno za jednu poziciju i poređenje se nastavlja od početka obrasca; pokazivač obrasca se vraća na 1, a pokazivač teksta se uvećava za 1.

Pseudo-kod glasio bi:

početak
$i \leftarrow 1, j \leftarrow 1$
sve dok $i \leq n - m + 1$
    inače ako je $p_j = s_{i+j-1}$ i $j = m$
        onda kraj programa sa rezultatom i
    inače ako je $p_j = s_{i+j-1}$ i $j < m$
        onda $j \leftarrow j + 1$
    inače ako je $p_j \neq s_{i+j-1}$
        onda $j \leftarrow 1, i \leftarrow i + 1$
kraj sve-dok
kraj programa sa rezultatom nula
(jer je obrazac P duži od teksta S ili je kraj P prešao preko kraja od S)
kraj

■ **Primer 5.1** Neka je tekst S = *a a a a a a a a b*, a obrazac koji se traži je *a a b*. U prvom koraku, porede se prvi karakter niske i prvi karakter obrasca. Pošto je u pitanju slaganje, prelazi se na poređenje drugog karaktera niske i drugog karaktera obrasca. Kako je i tu reč o slaganju, ide se na treći karakter niske i treći karakter obrasca.

a	a	a	a	a	a	a	a	b
a	a	b						
✓	✓	☹						

Kako se ovi karakteri ne slažu, u sledećem koraku vraća se na prvi karakter obrasca P, a prelazi se na drugi karakter niske S.

a	a	a	a	a	a	a	a	b
	a	a	b					
	✓	✓	☹					

Postupak se nastavlja. Kako je opet došlo do neslaganja na trećem karakteru obrasca, u sledećem koraku poredi se naredni (tek treći) karakter teksta i ponovo prvi karakter obrasca.

a	a	a	a	a	a	a	a	b
		a	a	b				
		✓	✓	☹				

I tako se postupak nastavlja sve dok ne utvrdimo slaganje, odnosno dok obrazac ne pređe tekst (što znači da je u pitanju neslaganje).

a	a	a	a	a	a	a	a	b
			a	a	b			
			✓	✓	☹			
				a	a	b		
				✓	✓	☹		
					a	a	b	
					✓	✓	☹	
						a	a	b
						✓	✓	✓

■

5.2 Knuth-Morris-Pratt algoritam

Za naivno sravnjivanje niski karakteristično je pomeranje obrasca uvek za jedno mesto kada se otkrije neslaganje između neka dva para karaktera u S i P. Ovakvo minimalno pomeranje pretpostavlja da ništa nije unapred poznato o sadržaju bilo teksta S bilo obrasca P. Međutim, neko znanje o tekstu S i obrascu P može se steći pre sravnjivanja analiziranjem obrasca P: ako je do neslaganja došlo na poziciji j obrasca, nešto se zna o prethodnih $j-1$ karaktera teksta – oni su jednaki sa prethodnih $j-1$ karaktera obrasca. Na osnovu ovog znanja može se izbeći iscrpno poređenje teksta S i obrasca P, već je potrebno porediti samo određene karaktere.

Nešto efikasniji algoritam za sravnjivanje je takozvani **KMP algoritam**. Kao i kod naivnog skaniranja, ovim postupkom takođe se skaniraju karakteri s leva na desno. Kada se naiđe na neslaganje, obrazac P se ne pomera automatski za 1 već za neki optimalni, unapred sračunati broj karaktera.

Kako odrediti šta je optimalni pomeraj? Pogledajmo sledeće sravnjivanje:

S	...	s_i	s_{i+1}	s_{i+2}	s_{i+3}	s_{i+4}	s_{i+5}	s_{i+6}	s_{i+7}	s_{i+8}	...
P				p_1	p_2	p_3	p_4	p_5			
				✓	✓	✓	✓	☹			

Početni deo obrasca P (drugi naziv je i *glava niske*) je sravnjen sa podniskom od S, a onda je došlo do neslaganja kod para s_{i+6} i p_5. Jasno je da do novog sravnjivanja ne može doći na prethodnim pozicijama ($i+3$, $i+4$, $i+5$) osim ako je neka glava sravnjenog dela od P identična sa repom (kranji deo niske) tog istog dela (što je i rep sravnjenog dela od S). Na primer, ako bi $s_{i+4}\ s_{i+5}$ bilo jednako $p_1\ p_2$, u sledećem koraku se obrazac može pomeriti tako da se p_1 pozicionira odmah ispod s_{i+4} — dakle, da se preskoči s_{i+3} — a poređenje se onda nastavlja od s_{i+6}, jer znamo da je prethodni deo obrasca sravnjen (jednak) sa prethodnim karakterima teksta:

S	...	s_i	s_{i+1}	s_{i+2}	s_{i+3}	s_{i+4}	s_{i+5}	s_{i+6}	s_{i+7}	s_{i+8}	...
P						p_1	p_2	p_3	p_4	p_5	
						✓	✓	?			

Ali, ako su p_1 i p_2 identični sa s_{i+4} i s_{i+5}, onda su p_1 i p_2 identični i sa p_3 i p_4 (zbog ranijeg sravnjivanja) i ovakva pravilnost se može unapred utvrditi.

Ako se nijedna glava sravnjenog dela od P ne može sravniti sa repom sravnjenog dela od S, onda tu više nema svrhe da se obrazac postavlja, i on se pomera na poziciju neslaganja (u našem primeru s_{i+6}).

Ako je, na primer, sravnjena glava podniske S igračka sa glavom obrasca igran (sravnjeni deo je *igra*), nema smisla da se u sledećem koraku obrazac pomera za jednu poziciju, jer je poznato da je slovo na sledećoj poziciji u tekstu **g** – jer je jednako drugoj poziciji obrasca. Znamo da prva dva slova obrasca nisu ista, tj. obrazac ne počinje sa *gg* – jer smo unapred istražili obazac.

S	...	i	g	r	a	č	k	a	...
P		i	g	r	a	n			
		✓	✓	✓	✓	☹			

Ilustrujmo logiku na sravnjivanju niske S = *b a b c b a b c a b c a a b c a* ($n = 16$) sa obrascem P = *a b c a b c a c a b* ($m = 10$). Ukoliko je do neslaganja došlo odmah na prvom karakteru, onda obrazac treba pomeriti za 1 mesto udesno:

S	b	a	b	c	b	a	b	c	a	b	c	a	a	b	c	a
P	a	b	c	a	b	c	a	c	a	b						
	☹															
		a	b	c	a	b	c	a	c	a	b					

Do neslaganja dolazi posle tri sravnjena karaktera. Kako u sravnjenom delu *abc* nema delova koji se ponavljaju (nijedna glava od *abc* nije njen rep), treba poravnati prvi karakter od P sa prvim nesravnjenim karakterom od S.

S	b	a	b	c	b	a	b	c	a	b	c	a	a	b	c	a
P		a	b	c	a	b	c	a	c	a	b					
		✓	✓	✓	☹											
					a	b	c	a	b	c	a	c	a	b		

Sravnjena je podniska *abcabca*. Najduža glava ove podniske koja je i njen rep je *abca*. Treba pomeriti obrazac P za tri mesta da bi se ostvarilo poklapanje sa već sravnjenim delom (to je najmanje pomeranje kojim se neće propustiti nijedna potencijalna mogućnost):

S	b	a	b	c	b	a	b	c	a	b	c	a	a	b	c	a
P						a	b	c	a	b	c	a	c	a	b	
						✓	✓	✓	✓	✓	✓	✓	☹			
									a	b	c	a	b	c	a	c	a	b

Dostignut je kraj teksta S jer je kraj obrasca je prešao preko kraja teksta. Kako je dužina teksta $n = 16$, a dužina obrasca $m = 10$, pozicija obrasca u odnosu na tekst $i = 9$, to je $i = 9 > n - m + 1 = 7$, konstatujemo da obrazac nije sravnjen s tekstom.

Broj pozicija za koje treba pomeriti obrazac u slučaju neslaganja se izračunava unapred.

neslaganje na poziciji obrasca P (p)	nesravnjen karakter	sravnjeni deo obrasca	dužina najduže glave koja je identična sa repom (d)	broj pozicija pomeranja obrasca ($f = p - d - 1$)
1	a	a	0	1
2	b	ab	0	1
3	c	abc	0	2
4	a	abca	0	3
5	b	abcab	1	3
6	c	abcabc	2	3
7	a	abcabca	3	3
8	c	abcabcac	4	3
9	a	abcabcaca	0	8
10	b	abcabcacab	1	8

Primera radi, kada je do neslaganja došlo na sedmoj poziciji ($p = 7$), sravnjeni deo je abcabc. Odredimo sve *glave* ove niske: a, ab, abc, abca, abcab. Odredimo sada i sve *repove*: c, bc, abc, cabc, bcabc. Vidimo da postoji rep obrasca koji je jednak glavi obrasca (abc) i da je on dužine $d = 3$, pa računamo $f = p - d - 1 = 7 - 3 - 1 = 3$, dakle u tom slučaju obrazac treba pomeriti za 3 mesta. Ukoliko postoji više istih glava i repova jednake dužine, uzimamo par najveće dužine.

5.3 Boyer-Moore algoritam

Karakteristika **BM algoritma** je da skanira tekst s desna u levo. Ovim algoritmom se još više smanjuje broj poređenja i time sravnjivanje postaje još efikasnije. Algoritam funkcioniše na sledeći način:

- Ako se krajnje desni karakter obrasca ne sravnjuje sa odgovarajućim karakterom teksta, i taj karakter teksta se uopšte ne pojavljuje u obrascu, obrazac se pomera za celu svoju dužinu u desno.
- Ako se krajnje desni karakter obrasca ne sravnjuje sa odgovarajućim karakterom teksta, ali se taj karakter teksta pojavljuje negde u obrascu, onda se obrazac pomera u desno tako da se taj karakter u obrascu koji je najbliži njegovom desnom kraju (ako ih ima više) poravna sa odgovarajućim karakterom teksta (najmanje pomeranje kojim se ostvaruje slaganje).
- Ako se krajnje desni karakter obrasca sravnjuje sa odgovarajućim karakterom teksta, onda se poređenje nastavlja ulevo dok:
  - se ne dođe do početka obrasca (sravnjivanje);
  - se ne ustanovi neslaganje na nekoj pozici, kada se ceo postupak ponavlja.

Moguće je odrediti unapred tablicu pomeranja, ali se broj pomeraja može utvrditi i *ad-hoc*. Neka je tekst *S* = PREDRAG PREPREDENO PRELAZI ULICU, a obrazac *P* = PRELAZ. Sravnjivanje BM algoritmom počinjemo poređenjem poslednjeg slova obrasca sa odgovarajućim slovom teksta. Kako je neslaganje odmah pri prvom poređenju (A $\neq$ Z), a slovo teksta kod kog je došlo do neslaganja je prisutno u obrascu, pomeramo obrazac tako da se sravne slova A (samo za jednu poziciju, jer je A odmah ispred slova Z):

```
P R E D R A G   P R E P R E D E N O   P R E L A Z I   U L I C U
P R E L A Z
          ☹
-------------------------------------------------------------------
  P R E L A Z
```

Opet poredimo poslednje slovo obrasca i odgovarajuće slovo teksta. Kako je neslaganje i u ovom slučaju (G $\neq$ Z), ali slovo G nije prisutno u obrascu, obrazac se pomera celom svojom dužinom (dakle, za 6 pozicija), tako da dođe *posle* slova G u tekstu:

```
P R E D R A G   P R E P R E D E N O   P R E L A Z I   U L I C U
  P R E L A Z
            ☹
-------------------------------------------------------------------
              P R E L A Z
```

Opet poredimo poslednje slovo obrasca (Z) i odgovarajuće slovo u tekstu (R). Kako je opet reč o neslaganju, a slovo R jeste prisutno u obrascu, obrazac pomerimo tako da se slovo R teksta i slovo R obrasca poravnaju (odnosno, ukupna dužina obrasca - poslednja pozicija slova R = 6 - 2 = 4):

```
P R E D R A G   P R E P R E D E N O   P R E L A Z I   U L I C U
            P R E L A Z
                      ☹
-------------------------------------------------------------------
                    P R E L A Z
```

Postupak se ponavlja. Kako je N $\neq$ Z, a slovo N nije prisutno u obrascu, obrazac pomeramo celom dužinom:

```
P R E D R A G   P R E P R E D E N O   P R E L A Z I   U L I C U
                    P R E L A Z
                              ☹
-------------------------------------------------------------------
                              P R E L A Z
```

Kako je L $\neq$ Z, ali slovo L jeste prisutno u obrascu, obrazac pomeramo za 6 - 4 = 2 pozicije (jer je slovo L na četvrtoj poziciji u obrascu):

```
P R E D R A G   P R E P R E D E N O   P R E L A Z I   U L I C U
                              P R E L A Z
                                        ☹
-------------------------------------------------------------------
                                    P R E L A Z
```

Nastavljamo poređenje od kraja obrasca, s desna u levo. U ovom slučaju dolazi do sravnjivanja:

P R E D R A G	P R E P R E D E N O	P R E L A Z I P R E L A Z ✓ ✓ ✓ ✓ ✓ ✓	U L I C U

Vidimo da smo imali ukupno 5 pomeranja i 11 poređenja.

5.4 Zadaci

1. Za sledeće obrasce formirati tablice pomeranja tih obrazaca, a zatim demonstrirati sravnjivanje sa tekstom KMP algoritmom:
   (a) Obrazac: POPODNE
   Tekst: POPOVI PONEKAD POPODNE POPRIČAJU
   (b) Obrazac: KOKOŠKA
   Tekst: KOKOŠ PREDVODI KOKA KOKOŠKA
   (c) Obrazac: PREPREDENI
   Tekst: PREPREDENJAK PREVARIO PRESTONICU
   (d) Obrazac: GUGUTANJE
   Tekst: GUGUTAVO GUGUČU GUSKE
   (e) Obrazac: ŠIŠTANJE
   Tekst: ŠIŠTAVI ŠIŠMIŠ ŠIŠTI U ŠUMI
2. Za sledeće obrasce demonstrirati rad BM algoritma nad datim tekstovima:
   (a) Obrazac: EKSPLOZIJU
   Tekst: BUS JE VEROVAO DA BI EKSPLOZIJA INFORMACIJA MOGLA PRERASTI U EKSPLOZIJU ZNANJA
   (b) Obrazac: INFORMACIJA
   Tekst: POSTOJI NEKOLIKO MERA ZA RAD SISTEMA ZA PRETRAZIVANJE INFORMACIJA
   (c) Obrazac: BANANA
   Tekst: PRECIZNOST I ODZIV ZASNIVAJU SE NA LISTI DOKUMENATA KOJE SISTEM VRACA
   (d) Obrazac: KRAK
   Tekst: SVAKA SVRAKA SKAKALA NA DVA KRAKA
   (e) Obrazac: KUK
   Tekst: KRALJ KARLO I KRALJICA KLARA KRALI KLARINET
   (f) Obrazac: ARK
   Tekst: KRALJ KARLO I KRALJICA KLARA KRALI KLARINET

5.5 Rešenja

1. (a) Potrebno je 25 poređenja, obrazac je pronađen u tekstu.

```
P O P O V I   P O N E K A D   P O P O D N E   P O P R I Č A J U
P O P O D N E
. . P O P O D N E
. . . . P O P O D N E
. . . . . . . . . P O P O D N E
. . . . . . . . . . . . . . . . P O P O D N E
```

p	karakter	neslaganje na...	d	f = p - d - 1
1	P	P	0	1
2	O	PO	0	1
3	P	POP	0	2
4	O	POPO	1	2
5	D	POPOD	2	2
6	N	POPODN	0	5
7	E	POPODNE	0	6

(b) Potrebno je 30 poređenja, obrazac je pronađen u tekstu.

p	karakter	neslaganje na...	d	f = p - d - 1
1	K	K	0	1
2	O	KO	0	1
3	K	KOK	0	2
4	O	KOKO	1	2
5	Š	KOKOŠ	2	2
6	K	KOKOŠK	0	5
7	A	KOKOŠKA	1	5

```
K O K O Š   P R E D V O D I   K O K A   K O K O Š K A
K O K O Š K A
. . . . . K O K O Š K A
. . . . . . . . . . . . . . . . K O K O Š K A
. . . . . . . . . . . . . . . . . . . . K O K O Š K A
. . . . . . . . . . . . . . . . . . . . . . . . K O K O Š K A
```

(c) Potrebno je 35 poređenja, obrazac nije pronađen u tekstu.

p	karakter	neslaganje na...	d	f = p - d - 1
1	P	P	0	1
2	R	PR	0	1
3	E	PRE	0	2
4	P	PREP	0	3
5	R	PREPR	1	3
6	E	PREPRE	2	3
7	D	PREPRED	3	3
8	E	PREPREDE	0	7
9	N	PREPREDEN	0	8
10	I	PREPREDENI	0	9

```
P R E P R E D E N J A K   P R E V A R I O   P R E S T O N I C U
P R E P R E D E N I
. . . . . P R E P R E D E N I
. . . . . P R E P R E D E N I
. . . . . P R E P R E D E N I
. . . . . P R E P R E D E N I
. . . . . P R E P R E D E N I
. . . . . P R E P R E D E N I
```

(d) Potrebno je 25 poređenja, obrazac nije pronađen u tekstu.

p	karakter	neslaganje na...	d	f = p - d - 1
1	G	G	0	1
2	U	GU	0	1
3	G	GUG	0	2
4	U	GUGU	1	2
5	T	GUGUT	2	2
6	A	GUGUTA	0	5
7	N	GUGUTAN	0	6
8	J	GUGUTANJ	0	7
9	E	GUGUTANJE	0	8

```
G U G U T A V O   G U G U Č U   G U S K E
G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
. . . . . G U G U T A N J E
```

(e) Potrebno je 33 poređenja, obrazac nije pronađen u tekstu.

p	karakter	neslaganje na...	d	f = p - d - 1
1	Š	Š	0	1
2	I	ŠI	0	1
3	Š	ŠIŠ	0	2
4	T	ŠIŠT	1	2
5	A	ŠIŠTA	0	4
6	N	ŠIŠTAN	0	5
7	J	ŠIŠTANJ	0	6
8	E	ŠIŠTANJE	0	7

```
Š I Š T A V I   Š I Š M I Š   Š I Š T I   U   Š U M I
Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
. . . . . Š I Š T A N J E
```

2. (a) Potrebno je 18 poređenja, 8 pomeranja, obrazac je pronađen u tekstu.

```
B U S   J E   V E R O V A O   D A   B I   E K S P L O Z I J A   I N F O R M A C I J A
E K S P L O Z I J U
. . . . . E K S P L O Z I J U
. . . . . E K S P L O Z I J U
. . . . . E K S P L O Z I J U
. . . . . E K S P L O Z I J U
. . . . . E K S P L O Z I J U
```

```
  M O G L A   P R E R A S T I   U   E K S P L O Z I J U   Z N A N J A
E K S P L O Z I J U
. . . . . E K S P L O Z I J U
. . . . . . . . . . . . . . . . . . E K S P L O Z I J U
```

(b) Potrebno je 17 poređenja, 6 pomeranja, obrazac je pronađen u tekstu.

```
P O S T O J I   N E K O L I K O   M E R A   Z A   R A D   S I S T
I N F O R M A C I J A
. . . . . I N F O R M A C I J A
. . . . . . . . . . . . . . . . . . . . . . . I N F O R M A C I J A

E M A   Z A   P R E T R A Z I V A N J E   I N F O R M A C I J A
I N F O R M A C I J A
. . . . . I N F O R M A C I J A
. . . . . I N F O R M A C I J A
. . . . . . . . . . . . . . . . . . . . . . . I N F O R M A C I J A
```

(c) Potrebno je 14 poređenja, 12 pomeranja, obrazac nije pronađen u tekstu.

```
P R E C I Z N O S T   I   O D Z I V   Z A S N I V A J U   S E   N A   L
B A N A N A
. . . . . B A N A N A
. . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B A N A N A

I S T I   D O K U M E N A T A   K O J E   S I S T E M   V R A C A
B A N A N A
. . . . . B A N A N A
. . . . . . . . B A N A N A
. . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . B A N A N A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B A N A N A
```

(d) Potrebno je 23 poređenja, 12 pomeranja, obrazac je pronađen u tekstu.

```
S V A K A   S V R A K A   S K A K A L A   N A   D V A   K R A K A
K R A K
. . K R A K
. . . . . K R A K
. . . . . . . . K R A K
. . . . . . . . . . K R A K
. . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . . . . . . . . . . . . . . . K R A K
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . K R A K
```

(e) Potrebno je 15 poređenja, 14 pomeranja, obrazac nije pronađen u tekstu.

```
K R A L J   K A R L O   I   K R A L J I C A   K L A R A   K R A L I
K U K
. . . K U K
. . . . . K U K
. . . . . . . . . . K U K
. . . . . . . . . . . . . K U K
```

```
. . . . . K U K
. . . . . K U K
. . . . . K U K
. . . . . K U K
. . . . . K U K
. . . . . K U K
. . . . . K U K

K L A R I N E T
K U K
. . . K U K
. . . . . K U K
```

(f) Potrebno je 20 poređenja, 18 pomeranja, obrazac nije pronađen u tekstu.

```
K R A L J  K A R L O  I  K R A L J I C A  K L A R A  K R A L I  K L A R I N E T
A R K
. . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
. . . . . A R K
```

6. Priprema teksta za dalju obradu

U prethodnim poglavljima pretpostavljali smo da invertovani indeks imamo kao strukturu koja prvenstveno sadrži nekakve indeksne termine koji su deskriptori dokumenata, a da uz svaki takav termin postoji uvezana lista identifikatora dokumenata kolekcije koja je raspoloživa za pretraživanje. Pitanje koje sada postavljamo je sledeće: *odakle nam takva lista termina?*

Postoji nekoliko načina da se do takvih termina dođe. Prvo, to treba da budu termini koji će na neki način opisivati sadržaj dokumenta. Najbolji, ali i najskuplji i najmanje efikasan način je ručno indeksiranje. To podrazumeva da se tekst prosledi stručnjacima iz odgovarajuće oblasti, koji će onda tekst pažljivo pročitati i izdvojiti termine i fraze koje najbolje opisuju njegov sadržaj. Takva lista termina obično se dobija poštovanjem određenih pravila o strukturi i dužini, što proizvodi uniformni i čisti indeks. U takvim slučajevima često se konsultuju i terminološki rečnici koji propisuju šta je sve dozvoljeno da se pojavi kao indeksni termin, ali je tada sadržaj indeksa ograničen sadržajem rečnika.

Najjednostavniji način je *ekstrakcija termina* direktno iz teksta koja podrazumeva da tekst bude unapred pročitan od strane nekog programa koji prepoznaje obrasce potencijalnih termina. Na primer, ako bi tekst prethodno bio obeležen vrstama reči (engl. Part-of-Speech Tagging, a takvi programi se zovu PoS-tageri) onda bi ekstraktori termina tražili uzastopne sekvence prideva koji prethode imenici, što bi potencijalno predstavljalo imeničku frazu. Jedan od problema ovog pristupa je nefleksibilnost. Podsetimo se, rezultat pronalaženja puno zavisi od formulacije upita, odnosno od nivoa slaganja upitnih termina sa indeksnim. Ako indeks sadrži doslovno ekstrahovane sekvence reči iz teksta, onda jedan sistem koji je npr. zasnovan na skaniranju sadržaja može totalno da podbaci ako upit ne sadrži oblik termina koji je sadržan u indeksu.

U ovom poglavlju bavićemo se automatskim indeksiranjem, odnosno analiziraćemo korake rada programa koji do indeksnih termina dolaze nekim algoritmom. Kao što je već rečeno, svaki tekst kolekcije potrebno je unapred obraditi. U narednim odeljcima posmatraćemo prvo korake obrade teksta, a na kraju i izračunavanje važnosti termina.

Podsetimo se prvo primera 3.3, gde je D_1 sadržaj dokumenta bio sledeći:

Juneće šnicle posolite, utrljajte vegetu i ostavite da odstoje.
Za to vreme ispržite na ulju seckan crni luk.
Slaninu isecite na rezance.
U vatrostalnu posudu poređajte meso, luk i odozgo slaninu, biber i lovorov list.
Prelijte belim vinom.
Kuvajte na umerenoj vatri.

Tada smo rekli da se kao indeksni termini uzimaju samo pridevi i imenice, i to njihovi rečnički oblici, tako da bi to bile sledeće reči: beo, biber, crn, juneći, list, lovorov, luk, meso, posuda, rezanac, seckan, slanina, šnicla, ulje, umeren, vatra, vatrostalan, vegeta, vino, vreme.

Ovo je jedan način da se ekstrahuju termini direktno iz teksta, pod uslovom da sistem podržava obeležavanje teksta vrstama reči, odnosno lematizacije teksta (svođenje reči na kanonski, rečnički oblik). Pod resursa koji se nameću kao preduslov, postavlja se i pitanje da li su imenice i pridevi dovoljni diskriminatori. Šta je, na primer, sa glagolima? Ukoliko domaćica želi ručak koji se prži, a ne na primer peče, uključivanje reči *isprženi* neće biti od značaja, iako je sadržana u tekstu. Dalje, da li su sve imenice i svi pridevi jednako dobri? Primera radi, ako se posmatra kolekcija kuvarskih recepata, da li su reči *beo*, *biber* ili *posuda* zaista nužno specifični za neki recept? Ako i jesu, u kolikoj meri? Upravo time ćemo se baviti u narednim odeljcima.

6.1 Tokenizacija teksta

Kako se na ovom predmetu bavimo isključivo kolekcijama digitalnih tekstova, smatraćemo da je tekst linearna sekvenca simbola (karaktera, reči ili fraza). Pre bilo kakve obrade, elektronski tekst potrebno je raščlaniti na lingvističke jedinice kao što su reči, brojevi, rečenice, znaci interpunkcije itd, u zavisnosti od potrebe. Prema semantici, odnosno prema ulozi u sintaksi jezika, razlikujemo sledeće tekstualne segmente:

- *morfeme*: predstavljaju značeće, nedeljive morfološke jedinice jezika (npr. morfeme *od* i *baciti* koje se ne mogu dalje raščlanjivati, a zajedno formiraju reč *odbaciti*);
- *tokene*: sekvence karaktera grupisane tako da čine korisne semantičke jedinice za obradu;
- *tipove*: klase tokena koje se sastoje iz istih sekvenci karaktera.

Za neku sekvencu karaktera, *tokenizacija* predstavlja zadatak rašlaljivanja teksta na tokene.

■ **Primer 6.1** Posmatrajmo rečenicu iz primera 3.3:

Juneće šnicle posolite, utrljajte vegetu i ostavite da odstoje.

U rečenici se uočavaju sledeći tokeni:

Juneće	šnicle	posolite	,	utrljajte	vegetu	i	ostavite	da	odstoje	.

■

Iako je naizgled jednostavan zadatak u pitanju, tokenizacija zapravo nosi razne izazove sa sobom. Za razliku od jezika poput srpskog, engleskog i španskog, čije se reči u pisanju razdvajaju razmakom, u jezicima kao što su japanski i kineski, reči nemaju jasne granice. Tokenizacija rečenica ovakvih jezika zahteva dodatne leksičke i morfološke resurse, ali to nije u domenu ovog predmeta.

6.2 Model vreće reči

Model vreće reči (engl. Bag of Words) predstavlja surogat prezentaciju dokumenta i ovu prezentaciju smo u nekom smislu već imali prilike da vidimo u prethodnim odeljcima. Ona podrazumeva da se za tekst napravi skup svih različitih reči – dakle, ne tokena – koji se u njemu pojavljuju, bez obraćanja pažnje na redosled ili broj pojavljivanja, pri čemu se taj skup dalje može profinjavati.

Za primer 3.3, model vreće reči bio bi jednostavno skup: D_1^{BoW} = { Juneće, šnicle, posolite, utrljajte, vegetu, i, ostavite, da, odstoje, Za, to, vreme, ispržite, na, ulju, seckan, crni, luk, Slaninu, isecite, rezance, U, vatrostalnu, posudu, poređajte, meso, odozgo, slaninu, biber, lovorov, list, Prelijte, belim, vinom, Kuvajte, umerenoj, vatri }. Primetimo prvo da smo eliminisali sve znake interpunkcije. Primetimo dalje i da se reči *i*, *na*, *luk* pojavljuju samo jednom. To je zato što u ovoj prezentaciji pravimo uniju svih različitih reči koje su se pojavile, a ne nabrajamo redom reči koje se pojavljuju.

Pošto indeks čine sortirani indeksni termini, sortirana lista termina bi glasila (jer redosled nije bitan): { Juneće, Kuvajte, Prelijte, Slaninu, U, Za, belim, biber, crni, da, i, isecite, ispržite, list, lovorov, luk, meso, na, odozgo, odstoje, ostavite, poređajte, posolite, posudu, rezance, seckan, slaninu, to, ulju, umerenoj, utrljajte, vatri, vatrostalnu, vegetu, vinom, vreme, šnicle }. Vidimo da se reči mogu ponavljati u varijanti sa velikim i malim slovom, pa bi jedno moguće profinjavanje bilo da se estrakcija učini neosetljivom na veličinu slova, što bi smanjilo nepotrebno ponavljanje indeksnih termina (reči *Slaninu* i *slaninu* su predstavljene samo u drugoj varijanti): { juneće, poređajte, utrljajte, umerenoj, belim, lovorov, crni, u, rezance, list, da, isecite, odstoje, luk, seckan, prelijte, ostavite, posudu, biber, za, meso, vatri, vegetu, ulju, na, vatrostalnu, posolite, i, ispržite, šnicle, to, vreme, slaninu, vinom, kuvajte, odozgo }.

6.3 Normalizacija teksta

Kako se dalje ova lista može profinjavati? Već je pominjana normalizacija reči. U ovom odeljku objasnićemo moguće načine normalizacije teksta.

Uobičajno je da proizvoljan tekst sadrži različite oblike iste reči, na primer *student*, *studenti*, *studentkinje*, *studentima* itd. Ne samo za potrebe kreiranja indeksnih termina, već u mnogim primenama obrade teksta, često je korisno *normalizovati* tekst redukovanjem različitih oblika reči na neki kanonski oblik. Kao što smo rekli u prethodnom odeljku, morfeme su značeće nedeljive jedinice teksta. Tako se, na primer, reč *student* sastoji od jedne morfeme (sama reč *student*) dok se reč *studentima* sastoji od dve: morfema *student* i morfema *-ima*. Ono što je nama interesantno u ovom slučaju je prva, ovde *glavna* morfema, koja nosi značenje ove reči i predstavlja njen koren. Morfema *-ima* je sufiks i ona samo dodaje dodatno značenje korenu reči.

Stemeri teksta su programi koji implementiraju neku heuristiku koja ima za cilj da u rečima eliminiše sve takve dodatne morfeme, poput sufiksa, prefiksa i infiksa, zadržavajući samo stem. Stem nije koren u pravom smislu reči, već sekvenca karaktera koja ostane nakon eliminacije afiksa. Primeri za engleski jezik dati su u narednoj tabeli.

Oblik	Sufiks	Stem
flies	-es	fli
traditional	-ional	tradit
skiing	-ing	ski

Primera radi, jedno od pravila može glasiti: *ako se reč engleskog jezika završava na -ing, eliminisati taj nastavak*. Ovo pravilo je primenljivo za primer iz tabele 6.3, jer reč *skiing* svodi na reč *ski*, ali već za reč *king* neće dati zadovoljavajući rezultat (jedno slovo ne može biti stem). Uz to pravilo može se dodati i uslov dužine reči: na primer, *primeniti uklanjanje ako reč ima nastavak -ing i sadrži pet ili više karaktera*. Najpoznatiji stemeri su Porterov (1980. godina), Lovinsov (1968. godina) i Pajs-Haskov stemer (1990. godina).

Lematizacija se obično odnosi na redukciju koja se zasniva na rečničkoj i morfološkoj analizi reči koja za cilj ima svođenje reči na njen rečnički oblik, poznatiji kao *lema*. Lematizacija je finiji, ali i složeniji proces, jer zahteva dodatne jezičke resurse poput rečnika (stemeri samo implementiraju neku heuristiku). Primer lematizacije za reči iz tabele 6.3 naveden je u narednoj tabeli.

Oblik	Morfološka informacija	lema
flies	množina imenice **fly**	fly
traditional	pridev izveden iz imenice **tradition**	traditional
skiing	gerund glagola **to ski**	ski

Da sumiramo:

stem je osnovni oblik reči, bez sufiksa i prefiksa;
stemer je program koji uklanja afikse i zadržava samo stem;
lema je reč predstavljena odgovarajućom rečničkom odrednicom, odnosno kanonski oblik neke reči;
lematizacija je proces zamene reči svojom lemom.

■ **Primer 6.2** Za tekst pesme *El Cóndor Pasa* izvođača Simon & Garfunkel, u narednoj tabeli prikazan je rad dve varijante Porterovog stemera.

I'd rather be a sparrow than a snail, yes I would, if I could, I surely would. Away, I'd rather sail away like a swan that's here and gone. A man gets tied up to the ground, he gives the world its saddest sound, its saddest sound. I'd rather be a hammer than a nail, yes I would, if I only could, I surely would.
i'd rather be a sparrow than a snail, ye I would, if I could, I sure would. away, i'd rather sail away like a swan that' here and gone. A man get tie up to the ground, he give the world it saddest sound, it saddest sound. i'd rather be a hammer than a nail, ye I would, if I onli could, I sure would.
i'd rather be a sparrow than a snail, yes i would, if i could, i sure would. away, i'd rather sail away like a swan that here and gone. a man get tie up to the ground, he give the world it saddest sound, it saddest sound. i'd rather be a hammer than a nail, yes i would, if i onli could, i sure would.

■

Lista iz primera 3.3, nakon normalizacije lematizacijom, izgledala bi ovako: { juneće, poređati, utrljati, umeren, beo, lovor, crni, u, rezanci, list, da, iseći, odstojati, luk, seckan, preliti, ostaviti, posuda, biber, za, meso, vatra, vegeta, ulje, na, vatrostalna, posoliti, i, ispržiti, šnicla, to, vreme, slanina, vino, kuvajti, odozgo }. Ovaj korak će potencijalno opet smanjiti broj kandidata za indeksne termine.

6.4 Označavanje vrstama reči

Još jedan mogući korak profinjavanja liste kandidata je eliminacija funkcionalnih reči – zamenica, uzvika, veznika, rečca i predloga – odnosno reči koje malo ili uopšte ne doprinose semantici teksta. To se može uraditi tako što se unapred pripremi lista takvih reči. Drugi način je obeležavanje teksta vrstom reči.

Već je bilo reči o PoS-tagerima, odnosno programima koji tokenima teksta dodeljuju oznaku vrste reči. Objasnimo ovaj postupak malo detaljnije.

PoS-tagiranje je proces automatskog dodeljivanja deskriptora vrsta reči tokenima teksta, odnosno *PoS-tagova*. *PoS-tageri* su računarski programi koji vrše ovakvu dodelu. U primeru 6.3 dat je primer teksta čijim tokenima su dodeljene odgovarajuće kategorije.

■ **Primer 6.3** Can_MD you_PRP water_VB the_DT plants_NNS well_RB ,_X using_VBG the_DT water_NN from_IN the_DT well_NN ?_X

Sledi objašnjenje kategorija za primer 6.3.

POS-tag	Objašnjenje
MD	modalni glagol
PRP	lična zamenica
VB	infinitiv glagola
DT	određeni član
NNS	imenica u množini
RB	prilog
VBG	glagol, gerund ili sadašnji particip
IN	predlog ili veznik
NN	zajednička imenica
X	interpunkcija

Ovaj primer ilustruje i homonimiju, rečima *water* (koji može biti i imenica i glagol) i *well* (kao prilog i kao imenica). Za PoS-tagere, ovaj femonem je poseban izazov: dobri PoS-tageri treba da budu u mogućnosti da dodele adekvatnu kategoriju reči, u zavisnosti od njenog konteksta u rečenici.

■

Skup PoS-tagova može za različite jezike biti različito definisan, a nivo detaljnosti, odnosno granularnosti, može se definisati u skladu sa potrebom. Skup tagova može, ali ne mora biti specifičan za jezik. Jedan od najpoznatijih skupova PoS-tagova je Univerzalni skup[1]. Među prvim PoS-tagerima su tager *CLAWS* i Brilov tagger.

Kada je tekst obeležen vrstama reči, moguće je, na primer, eliminisati sve modalne glagole, lične zamenice, članove, predloge, veznike i slično, tako da u tekstu ostanu samo značeće reči. Prvo, takav postupak zahteva postojanje sofisticiranih PoS-tagera za jezik tekstova. Drugo, da li to znači da je rezultujuća lista reči konačna lista diskriminatora dokumenata?

Odogovor na to pitanje daćemo u narednom odeljku, uvođenjem mere koja rečima dokumenata kolekcije dodeljuje njihove težine u skladu sa tim koliko su dobri deskriptori.

1. Universal POS tags, `https://universaldependencies.org/u/pos/`

7. Automatsko indeksiranje

7.1 TF-IDF mera reči

Uobičajna tehnika kojom se eliminišu funkcionalne reči (neznačeće reči, reči najmanjeg značaja za semantiku teksta) iz teksta je upotrebom *TF-IDF* mere. Prvenstvena namena ove mere ipak nije eliminacija funkcionalnih reči, već je to samo jedna od primena, a šira namena je daleko opštija. Naime, ova mera rečima dodeljuje *težine* na osnovu njihovog značaja i ona predstavlja proizvod dve vrednosti: frekvencije termina (engl. Term Frequency, TF) i inverzne frekvencije dokumenata (engl. Inverse Document Frequency, IDF). Na kakav se značaj misli, ilustrovaćemo na jednom konkretnom slučaju.

U narednom tekstu, podrazumevaćemo da su izvršeni prethodno opisani koraci preprocesiranja teksta: obavezno tokenizacija i eliminacija interpunkcije, a zatim i potencijalno stemovanje ili lematizacija.

Neka kolekcija dokumenata $\mathscr{D}$ sadrži $N = 806,791$ dokumenata. Posmatrajmo neka tri dokumenta te kolekcije, D_1, D_2 i D_3 i termine *vozilo*, *automobil*, *osiguranje* i *najbolji*. Neka je poznato da se termin *vozilo* javlja u prvom dokumentu $n^1_{vozilo} = 27$ puta, u drugom dokumentu $n^2_{vozilo} = 4$ puta i u trećem dokumentu $n^3_{vozilo} = 24$ puta, termin *automobil* 3, 33 i 0 puta, termin *osiguranje* 0, 33 i 29 puta, a termin *najbolji* 14, 0 i 17 puta u D_1, D_2 i D_3, respektivno. Radi preglednosti, predstavimo ove podatke tabelarno:

t	n^1_t	n^2_t	n^3_t
vozilo	27	4	24
automobil	3	33	0
osiguranje	0	33	29
najbolji	14	0	17

Neka je poznato da dokumenti D_1, D_2 i D_3 redom sadrže ukupno $|D_1| = 4,237$, $|D_2| = 11,298$ i

$|D_3| = 2,011$ termina.

Frekvencija termina (engl. Term Frequency, TF)

Za dati termin t u tekstualnom dokumentu D_i, TF predstavlja količnik apsolutnog broja pojavljivanja termina n_t^i u dokumentu D_i i ukupnog broja termina sadržanih u dokumentu $|D_i|$: $tf_t^i = \frac{n_t^i}{|D_i|}$.

Dakle, za termin *vozilo*, koji se u dokumentu D_1 javlja $n_{vozilo,1} = 27$ puta, a koji sadrži $|D_1| = 4,237$ termina, TF mera je praktično procenat pojavljivanja te reči u tekstu u odnosu na ukupan broj reči dokumenta, odnosno $tf_{vozilo} = \frac{n_{vozilo,1}}{|D_1|} = \frac{27}{4,237} \sim 0.006372$.

Uradimo isto i za druge dokumente, ali i preostale termine. Pokažimo postupak ponovo tabelarno:

t	tf_t^1	tf_t^2	tf_t^3
vozilo	$\frac{27}{4,237} \sim 0.006372$	$\frac{4}{11,298} \sim 0.000354$	$\frac{24}{2,011} \sim 0.011934$
automobil	$\frac{3}{4,237} \sim 0.000708$	$\frac{33}{11,298} \sim 0.002920$	$\frac{0}{2,011} \sim 0$
osiguranje	$\frac{0}{4,237} \sim 0$	$\frac{33}{11,298} \sim 0.002920$	$\frac{29}{2,011} \sim 0.014420$
najbolji	$\frac{14}{4,237} \sim 0.003304$	$\frac{0}{11,298} \sim 0$	$\frac{17}{2,011} \sim 0.008453$

Neka je dalje poznato da se termin *vozilo* javlja u $df_{vozilo} = 18,165$ različitih dokumenata kolekcije $\mathscr{D}$. Slično, $df_{automobil} = 6,723$, $df_{osiguranje} = 19,241$ i $df_{najbolji} = 25,235$ dokumenata kolekcije. Prikažimo i to tablerno, radi preglednosti:

t	df_t
vozilo	18,165
automobil	6,723
osiguranje	19,241
najbolji	25,235

Inverzna frekvencija dokumenata – indeks značaja (engl. Inverse Document Frequency, IDF)

Neka je broj dokumenata u kolekciji $\mathscr{D}$ koji sadrže termin t označen kao df_t. IDF se zatim određuje kao: $idf_t = \log_{10} \frac{N}{df_t}$, pri čemu N predstavlja ukupan broj dokumenata kolekcije.

Odredimo onda IDF za svaki od termina, redom. Podsetimo se, veličina kolekcije dokumenata je $N = 806,791$.

t	idf_t
vozilo	$\log \frac{806,791}{18,165} \sim \log_{10}(44.414588) \sim 1.647525$
automobil	$\log \frac{806,791}{6,723} \sim \log_{10}(120.004611) \sim 2.079197$
osiguranje	$\log \frac{806,791}{19,241} \sim \log_{10}(41.930824) \sim 1.622533$
najbolji	$\log \frac{806,791}{25,235} \sim \log_{10}(31.971111) \sim 1.504757$

Dodeljivanje TF-IDF mere – združeni indeks

TF-IDF mera dodeljuje težinu terminu t u dokumentu D_i na sledeći način:

$$\text{tf-idf}_t^i = tf_t^i \cdot idf_t = \frac{n_t^i}{|D_i|} \cdot \log \frac{N}{df_t}$$

TF-IDF mera dodeljuje težinu termina t sadržanog u dokumentu D koja:

- ima najveću vrednost kada se termin pojavljuje puno puta u malom broju dokumenata, što daje visoku diskriminatornu moć tim dokumentima;
- ima nižu vrednost kada se termin pojavljuje malo puta u dokumentu ili ako se termin pojavljuje u više različitih dokumenata kolekcije;
- ima najnižu vrednost kada se termin pojavljuje u gotovo svim dokumentima kolekcije, što znači da termin nije naročito diskriminatoran ni za jedan dokument posebno.

Dakle, TF-IDF mera daje vrednost blisku 0 funkcionalnim rečima, što je čini zgodnom za eliminaciju te klase reči.

Konačno, odredimo TF-IDF meru za svaki od posmatranih termina u odnosu na sva tri dokumenta kolekcije. Potrebno je, za svaki termin t, pomnožiti njegovu normalizovanu frekvenciju tf_t sa odgovarajućom inverznom frekvencijom dokumenata idf_t. Prikažimo postupak ponovo tablerno:

t	tf-idf_t^1	tf-idf_t^2	tf-idf_t^3
vozilo	$0.006372 \cdot 1.647525$ $= 0.010498$	$0.000354 \cdot 1.647525$ $= 0.000583$	$0.011934 \cdot 1.647525$ $= \boxed{0.019661}$
automobil	$0.000708 \cdot 2.079197$ $= 0.001472$	$0.00292 \cdot 2.079197$ $= 0.006071$	$0 \cdot 2.079197$ $= \boxed{0}$
osiguranje	$0 \cdot 1.622533$ $= 0$	$0.00292 \cdot 1.622533$ $= 0.004737$	$0.01442 \cdot 1.622533$ $= \boxed{0.023396}$
najbolji	$0.003304 \cdot 1.504757$ $= 0.004972$	$0 \cdot 1.504757$ $= 0$	$0.008453 \cdot 1.504757$ $= \boxed{0.012719}$

Prokomentarišimo dobijene vrednosti. Od datih termina, za D_1 najuticajniji je termin *vozilo* jer ima najveću tf-idf meru. Za dokument D_2, to je termin *automobil*, a za D_3 je to termin *osiguranje*.

Kako se izračunava relevantnost dokumenta za upit? Možemo da posmatramo svaki dokument kao vektor u kome svaka komponenta odgovara jednom terminu iz rečnika kolekcije, i ta komponenta je upravo težina termina za dokument izračunata kao tf-idf. Za one termine iz rečnika koji se ne pojavljuju u dokumentu ta težina će biti 0. Ovaj vektor je od suštinskog značaja za procenjivanje i rangiranje. Procena dokumenta u odnosu na upit može da bude zbir kombinovanih indeksa svih termina iz vektora upita:

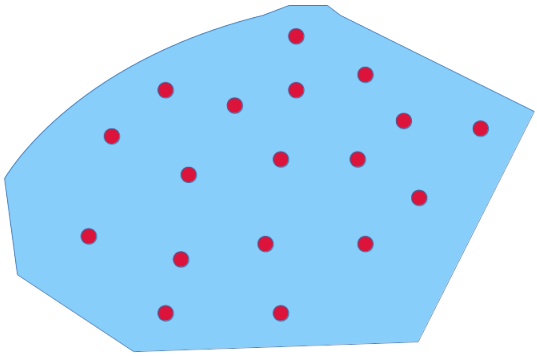
$$S(Q, D_i) = \sum_{t \in Q} \text{tf-idf}_t^i$$

Ako bi upit bio, na primer, Q = (*vozilo, automobil, osiguranje, najbolji*), najrelevantniji dokument za ovaj upit je D_3 jer je zbir težina za taj dokument $S(Q, D_i) = 0.019661 + 0 + 0.023396 + 0.012719 = 0.055776$ najveći.

7.2 Diskriminatorna vrednost termina i indeksiranje frazama

Jedno od poželjnih svojstava indeksnih termina je da pružaju mogućnost razlikovanja dokumenata kolekcije jednih od drugih, odnosno da budu što bolji diskriminatori. Diskriminatorna vrednost termina u prethodnom odeljku aproksimisana je korišćenjem specifičnosti termina koja se izračunavala kao inverzna vrednost frekvencije dokumenata. U narednom tekstu pokazaćemo metod koji diskriminatornu vrednost termina određuje na osnovu stepena sličnosti dokumenata.

Zamislimo tekstualne dokumente kao tačke u prostoru i predstavimo takvu kolekciju vizuelno, kao na slici 7.1. Pretpostavimo da je rastojanje između dve takve tačke, tj. dokumenata, inverzno proporcionalno sličnosti dokumenata: manje rastojanje – veća sličnost; veće rastojanje – manja sličnost. Sličnost (blizina) dva dokumenta zavisi od sličnosti njihovog sadržaja, odnosno od broja zajedničkih termina.



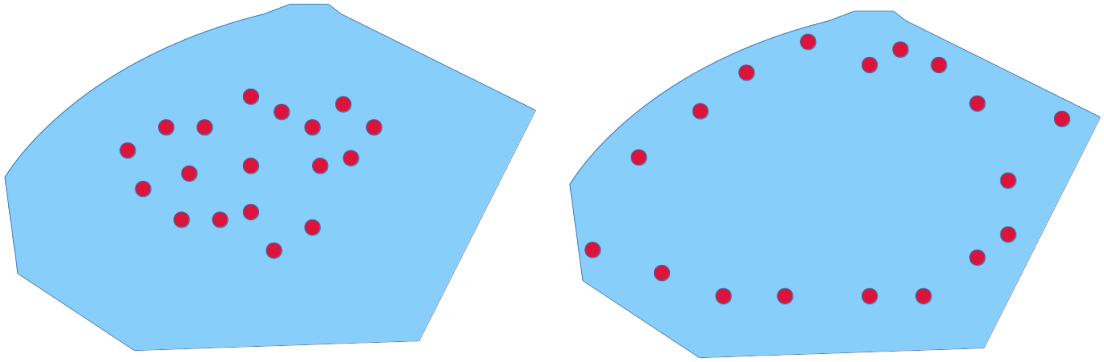
Slika 7.1: Prostor kolekcije dokumenata

Kada se dokumentima dodele vrlo slični skupovi termina, odgovarajuće tačke u ovakvom prostoru dokumenata se pojavljuju veoma blizu. Posledica je da prostor postaje gušći, što je ilustrovano na slici 7.2, levo. Kada se kolekciji dokumenata doda, na primer, novi termin koji nije dobar diskriminator, najčešće termin koji je visoko frekventan i koji ne pravi razliku između stavki kolekcije, takav termin će se pojaviti u mnogim dokumentima, i takva dodela će učiniti dokumente sličnijim, a prostor gušćim.

Obrnuto se događa kada se dokumentima dodele međusobno različiti skupovi termina: tada se sličnost između dokumenata smanjuje, odnosno rastojanje između dokumenata se povećava. Posledica ove promene je da prostor postaje *razuđeniji*, *rasutiji*, odnosno *ređi*, kao što je ilustrovano na slici 7.2, desno. Kada se, na primer, dobar diskriminator dodeli dokumentima kolekcije, one stavke kojima je termin dodeljen izdvojiće se od ostalog dela kolekcije. To bi trebalo da uveća srednje rastojanje između stavki kolekcije i prema tome da proizvede prostor dokumenata koji je manje gust.

Može se zaključiti da se diskriminatorna vrednost termina dv_j nekog termina T_j može izračunati kao razlika u gustini prostora pre (oznaka Q) i posle (oznaka Q_j) dodeljivanja termina T_j dokumentima kolekcije:

$$dv_j = Q - Q_j$$



Slika 7.2: Prostor kolekcije dokumenata nakon dodavanja visoko frekventnog termina postaje gušći (levo), dok nakon dodavanja dobrog diskriminatora postaje ređi (desno)

Gustina prostora Q_j i Q sa i bez dodeljenog termina T_j može se izračunati na različite načine. Najjednostavniji pristup bi bio određivanje srednje vrednosti sličnosti između svih parova različitih stavki:

$$Q = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} sim(D_i, D_k)$$

gde $sim(D_i, D_k)$ predstavlja koeficijent sličnosti između dokumenata D_i i D_k koji se zasniva na sličnosti indeksnih termina koji su im dodeljeni, a N je ukupan broj dokumenata kolekcije.

Šta predstavlja ova jednakost? Raspišimo je za $N = 4$. Spoljašnja suma govori da promenljiva i uzima redom vrednosti od 1 do 4, te ona sabira sličnost dokumenta D_i sa svim ostalim dokumentima kolekcije (osim sa samim sobom). Unutrašnja suma govori da za svako i, k uzima redom vrednosti od 1 do 4, koje su različite od tekućeg i, kako se ne bi računala sličnost dokumenta sa samim sobom:

$$\begin{aligned} Q = \frac{1}{4 \cdot (4-1)} \cdot (&sim(D_1, D_2) + sim(D_1, D_3) + sim(D_1, D_4) \\ &+ sim(D_2, D_1) + sim(D_2, D_3) + sim(D_2, D_4) \\ &+ sim(D_3, D_1) + sim(D_3, D_2) + sim(D_3, D_4) \\ &+ sim(D_4, D_1) + sim(D_4, D_2) + sim(D_4, D_3)) \end{aligned}$$

Kako se računa sličnost između dva dokumenta? Kao mera sličnosti dva dokumenta često se koristi takozvani **Dajsov indeks**. Ova vrednost može se definisati kao:

$$sim(D_i, D_j) = \frac{2 \cdot |S_i \cap S_j|}{|S_i| + |S_j|}$$

pri čemu je S skup termina dokumenta, a $|S|$ broj elemenata skupa (odnosno broj različitih termina).

Ako dokumenti nemaju zajedničkih termina, sličnost $sim(D_i, D_j) = 0$, jer je $S_i \cap S_j = \emptyset$. Ako pak imaju dodeljene potpuno iste skupove termina, tada je $sim(D_i, D_j) = 1$, jer je $S_i = S_j$, pa je

$S_i \cap S_j = S_i = S_j$. Iz jednakosti se vidi i da važi da je $sim(D_i, D_j) = sim(D_j, D_i)$, odnosno da je sličnost simetrična relacija.

Dajsov indeks može se još preciznije izračunati i na osnovu broja pojavljivanja ili čak težina zajedničkih termina, a ne samo informacije o tome koliko zajedničkih termina ima u dva dokumenta.

Neka je svaki dokument kolekcije predstavljen vektorom težina unije svih termina kolekcije dokumenata (tf-idf, apsolutan broj pojavljivanja ili neka treća vrednost), tzv. Bag-of-Words prezentacija:

	T_1	T_2	T_3	$\ldots$	T_t
D_1	d_{11}	d_{12}	d_{13}	$\ldots$	d_{1t}
D_2	d_{21}	d_{22}	d_{23}	$\ldots$	d_{2t}
D_3	d_{31}	d_{32}	d_{33}	$\ldots$	d_{3t}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
D_N	d_{N1}	d_{N2}	d_{N3}	$\ldots$	d_{Nt}

pri čemu je t ukupan broj termina koji se koriste za indeksiranje cele kolekcije, dok d_{ik} predstavlja težinu k-tog termina dokumenta D_i, što može biti i samo broj pojavljivanja tog termina.

Tada se Dajsov indeks može odrediti i pomoću izraza:

$$sim(D_i, D_j) = \frac{2 \cdot (\sum_{k=1}^{t} d_{ik} \cdot d_{jk})}{\sum_{k=1}^{t} d_{ik} + \sum_{k=1}^{t} d_{jk}}$$

Ceo izraz, dakle, predstavlja dvostruki broj zbira proizvoda težina termina dodeljenih jednom i drugom dokumentu prema zbiru težina termina u oba dokumenta. Proizvod termina koji nisu zajednički biće 0, jer će jedna od težina biti 0 (termin se ne pojavljuje u jednom dokumentu).

Postavlja se pitanje kada je termin dobar diskriminator. Dobar diskriminator razuđuje prostor, što znači da se smanjuje gustina, odnosno smanjuje se srednja sličnost dokumenata. Jednakost $dv_j = Q - Q_j$ pokazuje da se dobrim diskriminatorima, koji razuđuju prostor dokumenata, dodeljuje pozitivna diskriminatorna vrednost dv_j. Zbog čega? Zato što kad se doda dobar diskriminator T_j BoW modelu, Q_j se smanjuje, pa će biti $Q > Q_j$, te će razlika $dv_j = Q - Q_j$ biti pozitivna. Pozitivne diskriminatorne vrednosti se obično povezuju sa određenim srednje frekventnim terminima koji se ne pojavljuju ni previše retko ni previše često.

Slično, vrlo frekventnim terminima koji su dodeljeni mnogim dokumentima kolekcije se dodeljuju negativne diskriminatorne vrednosti. Nisko frekventni termini koji se pojavljuju u jednom ili dva dokumenta ne utiču na gustinu prostora ni na jednu stranu; tada su odgovarajuće diskriminatorne vrednosti termina približno jednake nuli.

■ **Primer 7.1** Neka su dva dokumenta predstavljena vektorima broja pojavljivanja termina:

$$D_i = (3,2,1,0,0,0,1,1)$$

$$D_j = (1,1,1,0,0,1,0,0)$$

Vidimo da su dokumenti u kolekciji indeksirani sa 8 termina ($t = 8$). Vidimo da dokumenta imaju 3 zajednička termina, da je 5 broj termina dodeljen prvom dokumentu, a 4 broj termina dodeljen drugom dokumentu. Tada je Dajsov indeks računat na prvi način:

$$sim(D_i,D_j) = \frac{2\cdot|S_i\cap S_j|}{|S_i|+|S_j|} = \frac{2\cdot 3}{5+4} = \frac{6}{9} = \frac{2}{3}$$

Dajsov indeks računat na drugi način je:

$$sim(D_i,D_j) = \frac{2\cdot(3\cdot 1+2\cdot 1+1\cdot 1+0\cdot 0+0\cdot 0+0\cdot 1+1\cdot 0+1\cdot 0)}{(3+2+1+1+1)+(1+1+1+1)} = \frac{2\cdot(3+2+1)}{8+4} = \frac{12}{12} = 1$$

pri čemu brojilac $2\cdot(3\cdot 1+2\cdot 1+1\cdot 1+0\cdot 0+0\cdot 0+0\cdot 1+1\cdot 0+1\cdot 0)$ predstavlja dvostrukti zbir proizvoda dodeljenih težina zajedničkih termina, 8 je zbir težina termina dodeljenih prvom dokumentu, a 4 je zbir težina termina dodeljenih drugom dokumentu.

■

Ako su dati n-dimenzioni vektori $a = (a_1,a_2,a_3,\ldots,a_n)$ i $b = (b_1,b_2,b_3,\ldots,b_n)$, onda se operacija **skalarnog proizvoda** vektora a i b definiše kao zbir proizvoda odgovarajućih koordinata, odnosno:

$$a\cdot b = a_1\cdot b_1 + a_2\cdot b_2 + a_3\cdot b_3 + \ldots + a_n\cdot b_n \in \mathbb{R}$$

(odnosno, vrednost je realni broj, to jest skalar).

■ **Primer 7.2** Neka su data tri dokumenta sa četiri pridružena termina. Izračunati diskriminatornu vrednost prvog termina T_1 preko gustine prostora kolekcije dokumenata. Dokumenti su predstavljeni vektorima tf-idf vrednosti pripadajućih termina.

T_i	t	D_1 tf-idf$_t^1$	D_2 tf-idf$_t^2$	D_3 tf-idf$_t^3$
T_1	*vozilo*	0.010498	0.000583	0.019661
T_2	*automobil*	0.001472	0.006071	0
T_3	*osiguranje*	0	0.004737	0.023396
T_4	*najbolji*	0.004972	0	0.012719

Radi jednostavnosti, prikažimo ove vrednosti na možda zgodniji način (uzimamo vrednosti po kolonama):

$D_1 = (0.010498, 0.001472, 0, 0.004972)$

$D_2 = (0.000583, 0.006071, 0.004737, 0)$

$D_3 = (0.019661, 0, 0.023396, 0.012719)$

Prvo izračunavamo gustinu prostora kada su pridruženi svi termini.

$$Q = \frac{1}{3\cdot(3-1)}\cdot\sum_{i=1}^{3}\sum_{\substack{k=1\\k\neq i}}^{3} sim(D_i,D_j)$$

Konkretno:

$Q = \frac{1}{6} \cdot (sim(D_1,D_2) + sim(D_1,D_3) + sim(D_2,D_1) + sim(D_2,D_3) + sim(D_3,D_1) + sim(D_3,D_2))$

Odredimo sličnost dokumenata po parovima.

Kod računanja vodićemo računa da je $sim(D_i,D_j) = sim(D_j,D_i)$:

$sim(D_1,D_2) = \frac{2\cdot(0.010498\cdot 0.000583+0.001472\cdot 0.006071)}{0.010498+0.001472+0.004972+0.000583+0.006071+0.004737} \sim 0.0010623$

$sim(D_1,D_3) = \frac{2\cdot(0.010498\cdot 0.019661+0.004972\cdot 0.012719)}{0.010498+0.001472+0.004972+0.019661+0.023396+0.012719} \sim 0.0074160$

$sim(D_2,D_3) = \frac{2\cdot(0.000583\cdot 0.019661+0.004737\cdot 0.023396)}{0.000583+0.006071+0.004737+0.019661+0.023396+0.012719} \sim 0.0036413$

Konačno, gustina nakon dodavanja termina T_1 je:

$$Q_1 \sim \frac{1}{6}(0.0010623+0.0074160+0.0010623+0.0036413+0.0074160+0.0036413) \sim 0.0040399$$

Odredimo sada gustinu prostora bez termina T_1. Smanjuje se dimenzionalnost vektora:

$D_1 = (0.001472, 0, 0.004972)$

$D_2 = (0.006071, 0.004737, 0)$

$D_3 = (0, 0.023396, 0.012719)$

Pa je:

$sim(D_1,D_2) = \frac{2\cdot(0.001472\cdot 0.006071)}{0.001472+0.004972+0.006071+0.004737} \sim 0.0010359$

$sim(D_1,D_3) = \frac{2\cdot(0.004972\cdot 0.012719)}{0.001472+0.004972+0.023396+0.012719} \sim 0.0029718$

$sim(D_2,D_3) = \frac{2\cdot(0.004737\cdot 0.023396)}{0.006071+0.004737+0.023396+0.012719} \sim 0.0047238$

Gustina pre dodavanja termina T_1 je:

$$Q \sim \frac{1}{6}(0.0010359+0.0029718+0.0010359+0.0047238+0.0029718+0.0047238) \sim 0.0029105$$

Znači da je promena gustine prostora $dv_1 = Q - Q_1 = 0.0029105 - 0.0040399 = -0.0011294 < 0$. Vidimo da je gustina prostora veća sa dodeljenim terminom T_1, nego bez njega, i zato je diskriminatorna vrednost termina negativna. To govori da je termin T_1 loš izbor (u ovom iskonstruisanom slučaju je to zato što je on dodeljen svim dokumentima kolekcije).

■

7.3 Zadaci

1. Za date dokumente, izdvojiti ključne reči koje identifikuju sadržaj dokumenta izdvajajući samo imenice i posmatrajući:
   (a) Apsolutan broj pojavljivanja (n_t^i, nenormalizovane vrednosti) termina u dokumentu. Posmatrati samo termine sa apsolutnom frekvencijom pojavljivanja većom ili jednakom od 2. Prokomentarisati rezultate upita APPLE, APPLE AND GADGET, APPLE AND SALAD. Pretpostavlja se da se koriste operacije spajanja listi.

(b) Broj pojavljivanja termina u dokumentu u odnosu na ukupan broj značajnih reči u dokumentu (term frequency, tf_t^i, normalizovane vrednosti). Posmatrati samo termine sa normalizovanom frekvencijom pojavljivanja većom ili jednakom od 0.25. Prokomentarisati rezultate upita APPLE, APPLE AND GADGET, APPLE AND PIECE, APPLE AND FRUIT.

(c) Indeks značaja (inverse document frequency, $idf_t = log_{10}\frac{N}{df_t}$, N - broj dokumenata u kolekciji)

(d) Združeni indeks (kombinovana mera, tf-idf$_t^i$, $w_t^i = tf_t^i \cdot \log_{10}\frac{N}{df_t}$). Za date upite proveriti u kojoj meri dokumenti zadovoljavaju informacionu potrebu: APPLE AND GADGET, FRUIT AND SALAD, APPLE AND IPHONE AND GADGET.

D_1: Apple has launched new gadget. Apple named it "apple".

D_2: In order to prepare fruit salad, cut apples in pieces. Mix apples and banana pieces.

2. U kolekciji od $N = 234,458$ dokumenata, date su frekvencije pojavljivanja sledećih reči (ukupan broj pojavljivanja reči u svim dokumentima):

termin	df_t
obrazovanje	40,186
prosveta	22,182
fakultet	14,684
katedra	36,240

Dat je i apsolutan broj pojavljivanja reči unutar dokumenata D_1 i D_2 iz kolekcije:

termin	D_1	D_2
obrazovanje	26	10
prosveta	34	5
fakultet	53	14
katedra	41	7

Broj reči po dokumentima: $|D_1| = 2,345, |D_2| = 864$

(a) Izračunati indeks značaja ovih termina za indeksiranje (idf). Koji je najznačajniji termin za indeksiranje prema ovoj meri?

(b) Izračunati združeni indeks (tf-idf) za ova tri dokumenta iz kolekcije. Koja reči su najznačajnije za indeksiranje svakog od tih dokumenata?

3. U kolekciji od $N = 123,456$ dokumenata, date su frekvencije pojavljivanja sledećih reči (ukupan broj pojavljivanja reči u svim dokumentima):

termin	df_t
računar	42,000
pc	18,360
mac	26,780
monitor	10,250

Dat je i apsolutan broj pojavljivanja reči unutar dokumenata D_1 i D_2 iz kolekcije:

termin	D_1	D_2
računar	64	22
pc	20	40
mac	5	32
monitor	8	21

Broj reči po dokumentima: $|D_1| = 630, |D_2| = 840$

(a) Izračunati indeks značaja ovih termina za indeksiranje (idf). Koji je najznačajniji termin za indeksiranje prema ovoj meri?

(b) Izračunati združeni indeks (tf-idf) za ova tri dokumenta iz kolekcije. Koja reči su najznačajnije za indeksiranje svakog od tih dokumenata?

4. Neka je data kolekcija dokumenata sa pridruženim terminima:

D_1 T_1, T_2
D_2 T_2, T_3, T_4
D_3 T_1, T_3, T_5
D_4 T_3, T_4, T_5

(a) Izračunati gustinu prostora Q_{pre_6} ove kolekcije dokumenata koristeći Dajsov indeks sličnosti između dokumenata.
(b) Neka se termin T_6 dodeli dokumentima D_2 i D_4. Odrediti gustinu prostora Q_{posle_6} posle ove dodele i diskriminatornu vrednost $dv_6 = Q_{pre_6} - Q_{posle_6}$ termina T_6.

5. Odrediti Dajsov indeks sličnosti za sledeće dokumente, predstavljene frekvencijom reči:
   (a) $D_1 = (1,3,2,0,5,1,0,0)$ $D_2 = (1,2,1,1,0,0,3,8)$
   (b) $D_1 = (1,2,0,3,1,0)$ $D_2 = (1,1,3,0,2,2)$
6. Neka je data kolekcija dokumenata sa pridruženim terminima:

D_1 T_2, T_3
D_2 T_1, T_3, T_5
D_3 T_1, T_2, T_3
D_4 T_2, T_4, T_5

(a) Izračunati gustinu prostora Q_{pre} ove kolekcije dokumenata koristeći Dajsov indeks sličnosti između dokumenata.
(b) Neka se termin T_6 dodeli dokumentima D_1 i D_3. Odrediti gustinu prostora Q_{posle_6} posle ove dodele i diskriminatornu vrednost $dv_6 = Q_{pre} - Q_{posle_6}$ termina T_6.

7.4 Rešenja

1. Posmatraju se samo imenice (funkcionalne reči se svakako eliminišu):

D_1: Apple launched new gadget. Apple named it “apple".
D_2: In order to prepare fruit salad, cut apples in pieces. Mix apples and banana pieces.

(a) Apsolutan broj pojavljivanja termina unutar dokumenata:
$n^1_{apple} = 3, n^1_{gadget} = 1$ (ukupno 4 značajnih reči)
$n^2_{apple} = 3, n^2_{fruit} = 1, n^2_{salad} = 1, n^2_{pieces} = 2, n^2_{banana} = 1$ (ukupno 8 značajnih reči)
Zadržavaju se samo reči koje se javljaju 2 ili više puta:
D_1 = {apple}, D_2 = {apple, pieces }
Invertovani indeks:
apple → { D_1, D_2 }
pieces → { D_2 }

(b) Broj pojavljivanja termina u odnosu na ukupan broj reči u dokumentu:
$tf^1_{apple} = \frac{3}{4} = 0.75, tf^1_{gadget} = \frac{1}{4} = 0.25$
$tf^2_{apple} = \frac{3}{8} = 0.375, tf^2_{fruit} = \frac{1}{8} = 0.125,$
$tf^2_{salad} = \frac{1}{8} = 0.125, tf^2_{pieces} = \frac{2}{8} = 0.25, tf^2_{banana} = \frac{1}{8} = 0.125$
Zadržavaju se samo reči sa normalizovanom frekvencijom iznad 0.25:
D_1 = {apple, gadget}, D_2 = {apple, pieces }
Invertovani indeks:
apple → { D_1, D_2 }
gadget → { D_1 }
pieces → { D_2 }

(c) Indeks značaja:
$N = 2$

termin	df_t	$idf_t = log_{10}\frac{N}{df_t}$
apple	2	0
gadget	1	0.3
fruit	1	0.3
salad	1	0.3
pieces	1	0.3
banana	1	0.3

(d) Združeni indeks:
$w_t^i = tf_{ij} \cdot log_{10}\frac{N}{df_t}$

termin	w_t^1	w_t^2
apple	0	0
gadget	0.75	0
fruit	0	0.375
salad	0	0.375
pieces	0	0.75
banana	0	0.375

2. Indeks značaja:

termin	df_t	idf_t
obrazovanje	40,186	0.765
prosveta	22,182	3.024
fakultet	14,684	4.203
katedra	36,240	0.810

Normalizovana frekvencija pojavljivanja:

termin	n_t^1	df_t^1	n_t^2	df_t^2
obrazovanje	26	0.011	10	0.011
prosveta	34	0.014	5	0.005
fakultet	53	0.022	14	0.016
katedra	41	0.017	7	0.008

Združeni indeks:

termin	D_1	D_2
obrazovanje	0.008	0.008
prosveta	0.042	0.015
fakultet	0.092	0.067
katedra	0.013	0.006

3. Indeks značaja:

termin	df_t	idf_t
računar	42,000	0.468
pc	18,360	0.827
mac	26,780	0.663
monitor	10,250	0.080

Normalizovana frekvencija pojavljivanja:

termin	n_t^1	df_t^1	n_t^2	df_t^2
računar	64	$\frac{64}{630}$	22	$\frac{22}{840}$
pc	20	$\frac{20}{630}$	40	$\frac{40}{840}$
mac	5	$\frac{5}{630}$	32	$\frac{32}{840}$
monitor	8	$\frac{8}{630}$	21	$\frac{21}{840}$

Združeni indeks:

termin	D_1	D_2
računar	0.041	0.012
pc	0.045	0.039
mac	0.005	0.025
monitor	0.001	0.002

4. (a) Broj zajedničkih reči: D_1 i D_2 imaju tri zajedničke reči. D_1 sadrži 5 reči, a D_2 6 reči:

$$sim(D_1,D_2) = \frac{2 \cdot 3}{5+6} \approx 0.55$$

Množi se broj pojavljivanja zajedničkih reči:

$$sim(D_1,D_2) = \frac{2 \cdot (1 \cdot 1 + 3 \cdot 2 + 2 \cdot 1)}{(1+3+2+5+1)+(1+2+1+1+3+8)} \approx 0.64$$

(b) Broj zajedničkih reči: D_1 i D_2 imaju tri zajedničke reči. D_1 sadrži 4 reči, a D_2 5 reči:

$$sim(D_1,D_2) = \frac{2 \cdot 3}{4+5} \approx 0.67$$

Množi se broj pojavljivanja zajedničkih reči:

$$sim(D_1,D_2) = \frac{2 \cdot (1 \cdot 1 + 2 \cdot 1 + 1 \cdot 2)}{(1+2+3+1)+(1+1+3+2+2)} \approx 0.625$$

5. (a) Gustina prostora dokumenata se računa kao:

$$\begin{aligned} Q = & \frac{1}{N(N-1)} \times \\ & (sim(D_1,D_2) + sim(D_1,D_3) + sim(D_1,D_4) \\ & + sim(D_2,D_1) + sim(D_2,D_3) + sim(D_2,D_4) \\ & + sim(D_3,D_1) + sim(D_3,D_2) + sim(D_3,D_4) \\ & + sim(D_4,D_1) + sim(D_4,D_2) + sim(D_4,D_3)) \end{aligned} \tag{7.1}$$

pri čemu je $N = 4$ broj dokumenata u kolekciji i važi $sim(D_i,D_j) = sim(D_j,D_i)$.

Bag-of-Words model dokumenata (T_1,T_2,T_3,T_4,T_5):
$D_1 = (1,1,0,0,0)$
$D_2 = (0,1,1,1,0)$
$D_3 = (1,0,1,0,1)$
$D_4 = (0,0,1,1,1)$

Računaju se sličnosti među različitim parovima dokumenata. Zbog svojstva simetričnosti mere sličnosti, ne računa se sličnost za $\frac{N*(N-1)}{2}$ sabiraka:
$sim(D_1,D_2) = \frac{2 \cdot 1}{2+3} = 0.4$
$sim(D_1,D_3) = \frac{2 \cdot 1}{2+3} = 0.4$
$sim(D_1,D_4) = \frac{2 \cdot 0}{2+3} = 0$
$sim(D_2,D_3) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_2,D_4) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_3,D_4) = \frac{2 \cdot 2}{3+3} \approx 0.67$

Gustina prostora:
$Q = \frac{2}{12} \cdot (0.4+0.4+0+0.33+0.33+0.67) \approx 0.355$

(b) Novi Bag-of-Words model dokumenata $(T_1, T_2, T_3, T_4, T_5, T_6)$:
$D_1 = (1,1,0,0,0,0)$
$D_2 = (0,1,1,1,0,1)$
$D_3 = (1,0,1,0,1,0)$
$D_4 = (0,0,1,1,1,1)$

Gustina prostora dokumenata se računa kao:
$sim(D_1, D_2) = \frac{2 \cdot 1}{2+4} \approx 0.33$
$sim(D_1, D_3) = 0.4$
$sim(D_1, D_4) = 0$
$sim(D_2, D_3) = \frac{2 \cdot 1}{4+3} \approx 0.29$
$sim(D_2, D_4) = \frac{2 \cdot 2}{4+4} = 0.5$
$sim(D_3, D_4) = \frac{2 \cdot 2}{3+4} \approx 0.57$

Gustina prostora:
$Q = \frac{2}{12} \cdot (0.33 + 0.4 + 0 + 0.29 + 0.5 + 0.57) \approx 0.348$

Diskriminatorna vrednost termina T_6 je $dv_6 = 0.355 - 0.348 = 0.007$.
Termin nije ni naročito loš ni naročito dobar diskriminator, zato što je vrednost veoma bliska nuli (gustina se nije značajno promenila).

6. (a) Bag-of-Words model dokumenata $(T_1, T_2, T_3, T_4, T_5)$:
$D_1 = (0,1,1,0,0)$
$D_2 = (1,0,1,0,1)$
$D_3 = (1,1,1,0,0)$
$D_4 = (0,1,0,1,1)$

Računaju se sličnosti među različitim parovima dokumenata. Zbog svojstva simetričnosti mere sličnosti, ne računa se sličnost za $\frac{N*(N-1)}{2}$ sabiraka:
$sim(D_1, D_2) = \frac{2 \cdot 1}{2+3} = 0.4$
$sim(D_1, D_3) = \frac{2 \cdot 2}{2+3} = 0.8$
$sim(D_1, D_4) = \frac{2 \cdot 1}{2+3} = 0.4$
$sim(D_2, D_3) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_2, D_4) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_3, D_4) = \frac{2 \cdot 1}{3+3} \approx 0.33$

Gustina prostora:
$Q = \frac{2}{12} \cdot (0.4 + 0.8 + 0.4 + 0.33 + 0.33 + 0.33) \approx 0.432$

(b) Novi Bag-of-Words model dokumenata $(T_1, T_2, T_3, T_4, T_5, T_6)$:
$D_1 = (0,1,1,0,0,1)$
$D_2 = (1,0,1,0,1,0)$
$D_3 = (1,1,1,0,0,1)$
$D_4 = (0,1,0,1,1,0)$

Gustina prostora dokumenata se računa kao:
$sim(D_1, D_2) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_1, D_3) = \frac{2 \cdot 3}{3+4} \approx 0.86$
$sim(D_1, D_4) = \frac{2 \cdot 1}{3+3} \approx 0.33$
$sim(D_2, D_3) = \frac{2 \cdot 2}{3+4} \approx 0.57$

$sim(D_2, D_4) = 0.33$
$sim(D_3, D_4) = \frac{2 \cdot 1}{4+3} \approx 0.29$

Gustina prostora:
$Q = \frac{2}{12} \cdot (0.33 + 0.86 + 0.33 + 0.57 + 0.33 + 0.29) \approx 0.452$

Diskriminatorna vrednost termina T_6 je $dv_6 = 0.432 - 0.452 = -0.02$.
Termin je loš diskriminator, zato što je diskriminatorna vrednost termina negativna (pre dodavanja termina, gustina dokumenata je bila manja).

Literatura

Knjige

Članci