TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Raigo Kodasmaa

Infootsingus kasutatavad loomuliku keele töötluse tehnikad

Bakalaureusetöö (*6* EAP)

Juhendaja: prof. Mare Koit

Autor: ……..…. "….." juuni 2011
Juhendaja: ……..... "….." juuni 2011

Lubada kaitsmisele
Professor .....…….... "….." juuni 2011

TARTU 2011

Sisukord

Sissejuhatus

Infootsing (*information retrieval*) on loomuliku keele töötluse (*natural language processing*) valdkond, mille ülesandeks on sobivate tekstidokumentide ja nendest sobiva informatsiooni (peamiselt teksti) otsimine ja säilitamine, et vastata kasutaja päringus esitatud teabevajadusele. Infootsingu süsteem (*information retrieval system*) suudab teha efektiivset otsingut, rakendades mitmeid loomuliku keele töötluse tehnikaid.

Valisin käesoleva töö teema, et tutvustada infootsingu süsteemis toimuvaid protsesse: päringutöötlus, otsing ja tulemuste järjestamine. Miljonid inimesed kasutavad infootsingut iga päev oma informatsioonivajaduse rahuldamiseks. Seetõttu leidsin, et eestikeelne ülevaade infootsingust koos eesti keele näidetega võiks valdkonna arengule kaasa aidata. Bakalaureusetöö eesmärgiks ongi anda põhjalikum ülevaade infootsingu tehnikatest ning esitada näiteid, rakendades mõningaid skripte eesti keelsetele dokumentidele. Töö käigus koostasin lisaks skriptidele ka eestikeelsete stoppsõnade näidisnimekirja, mille aluseks on võetud sarnased ingliskeelsed nimekirjad.

Töö ülesehitus on järgmine. Esimeses peatükis tutvustan infootsingu lähiajalugu ja infootsingu süsteemi ülesehitust. Teises peatükis kirjeldan dokumendi eeltöötluse (*document pre-processing*) protsesse: leksikaalset analüüsi, stoppsõnade (*stop words*) elimineerimist ja lemmatiseerimist (*stemming*). Kolmandas peatükis räägin terminite ja dokumentide indekseerimise põhimõtetest ning neljandas peatükis kirjeldan päringute formuleerimisel kasutatavaid tehnikaid: asjakohast tagasisidet (*relevance feedback*) ja päringu laiendamist (*query expansion*). Lisas *1* toon töös kasutatavate mõistete sõnastiku. Lisasse *2* paigutasin koostatud eestikeelsete stoppsõnade loendi. Lisades *3*, *4* ja *5* asuvad skriptid, mida kasutasin töös eestikeelsete näidete saamiseks.

1. Infootsingu tutvustus

1.1 Infootsingu lähiajalugu

1990. aastatel valitses olukord, kus enamik inimesi eelistas küsida informatsiooni pigem teistelt inimestelt, kui saada seda infootsingu süsteemidelt. Seda ilmestab ka asjaolu, et näiteks oma puhkuse- ja ärireisid broneeriti reisiagentuuri töötajate vahendusel. Tänapäeval aga kasutatakse reiside broneerimiseks laialdaselt infosüsteeme. *Pew Internet Survey* poolt koostatud uuringu kohaselt kasutas aastaks 2004 juba ligi 92% inimestest veebi oma igapäevase informatsiooni saamiseks [10:1].

Infootsing ei tekkinud aga Internetis. Esimesed infootsingu süsteemid juurutati hoopis suurtes raamatukogudes, kuid levisid kiirelt ka juristide, arstide ja ajakirjanike hulgas [10:1]. Raamatukogude niinimetatud esimese põlvkonna infootsingu süsteemid võimaldasid otsida kataloogidest autori ja pealkirja järgi. Teise põlvkonna süsteemid lisasid juurde otsingu võtmesõnade (*keyword*) järgi ning päringuoperaatorite kasutusvõimaluse. Kolmanda generatsiooni infootsingusüsteemid - need, mis ka tänapäeval kasutusel – keskendusid graafilistele kasutajaliidestele, elektroonilisel kujul olevatele andmetele ja hüperteksti (*hypertext*) kasutusele [2:3]. Kuna ainuüksi raamatukogude andmebaas kasvas juba tohutult suureks, siis tuli seda organiseerida. Selleks hakati andmeid indekseerima, et kiirendada otsingut. 1989. aastal veebi (*Web*) loomine oli üks murrangulisi samme infootsingu ajaloos. Veebi standardne kasutajaliidese mudel lubas igal kasutajal luua ise dokumente ning seetõttu hakkas ka veebi suurus kasvama. Indekseerimine sai andmete hulga tõttu populaarseks tehnikaks ning seda hakati kasutama ka veebi otsingusüsteemides [2:2]. Siiski, veel 1990ndate lõpus olid paljud inimesed kindlad, et kogu veebi indekseerimine on võimatu, kuna selle suurus kasvas eksponentsiaalselt [10:1].

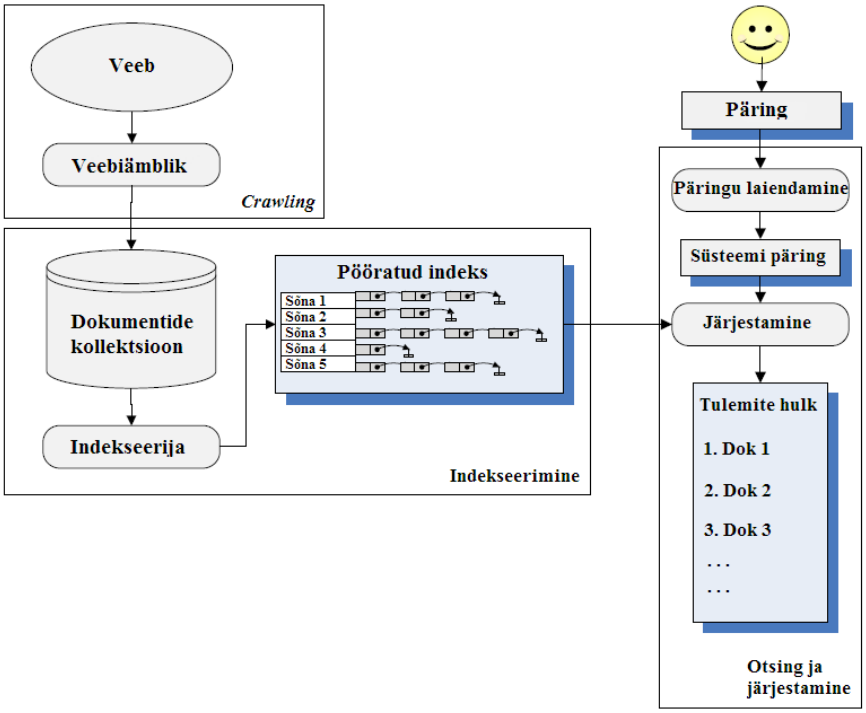
1.2 Infootsing tänapäeval

Tänapäevased otsimootorid suudavad üldiselt pakkuda väga kõrgetasemelist otsingut ning rohkelt asjakohaseid tulemusi vaatamata sellele, et iga päev sooritatakse ligi sada miljonit otsingut miljarditel veebisaitidel [10:1]. Infootsingu süsteemi kasutajate informatsioonivajadused võivad olla väga erinevad. Lihtsamatel juhtudel soovitakse otsida

mõne ettevõtte või ameti kodulehekülge, keerulisematel juhtudel aga näiteks Tartu raudteejaama platvormi renoveerimisplaani ja rahastamist [2:3]. Kasutaja infoteabe täielikku kirjeldust üldiselt infootsingu süsteemile ei sisestata. Selle asemel esitatakse päringuid, mis sisaldavad võtmesõnu ja indekseeritud termineid. Infootsingu süsteemi peamine eesmärk on väljastada kasutajale võimalikult palju asjakohaseid dokumente ning võimalikult vähe mitteasjakohaseid [2:4].

1.3 Infootsingu süsteem

Joonisel *1* on kujutatud infootsingu süsteemi arhitektuurimudel, kus süsteemi esimeseks ülesandeks on koostada dokumentide kollektsioon (*document collection*), mis võib olla privaatne või siis otsitud veebist veebiämbliku (*web crawler*) abil . Seejärel indekseeritakse kollektsioonis olevad dokumendid, et teostada kiiremat otsingut ja järjestada dokumente indeksite alusel [2:5]. Indekseerimisel kasutatakse peamiselt dokumendi pööratud esinemissagedust (*inverse document frequency*), mis seotakse iga erineva terminiga dokumendis. Kui dokumentide kollektsioon on indekseeritud, võib alata otsinguprotsess, mille ülesandeks võib olla kas kasutaja päringule otsese vastuse leidmine või uue veebisaidi avamine klikkimisel hüperlingile. Kasutaja päringule vastuse leidmiseks kasutatakse asjakohase tagasiside ja päringu laiendamise tehnikaid. Päringut modifitseeritakse ja esitatakse uuesti süsteemile, mis väljastab seejärel lõpptulemused - nendeks on n esimest dokumenti järjestatud dokumentide hulgas. [2:6]. Klikkimine võimaldab aga tuvastada populaarsemaid veebisaite, mida kasutajad tihedamini on külastanud, ning seada need järjestamisel ettepoole [2:7].



Joonis *1*. Infootsingu süsteemi arhitektuurimudel [2:6] (Autor modifitseeris ja tõlkis eesti keelde.)

2. Dokumendi eeltöötlus

Dokumendi eeltöötluseks võib nimetada protsessi, kus dokument valmistatakse ette temas sisalduvate terminite indekseerimiseks. See protsess hõlmab teksti leksikaalset analüüsi, stoppsõnade elimineerimist ja terminite lemmatiseerimist [2:223-224].

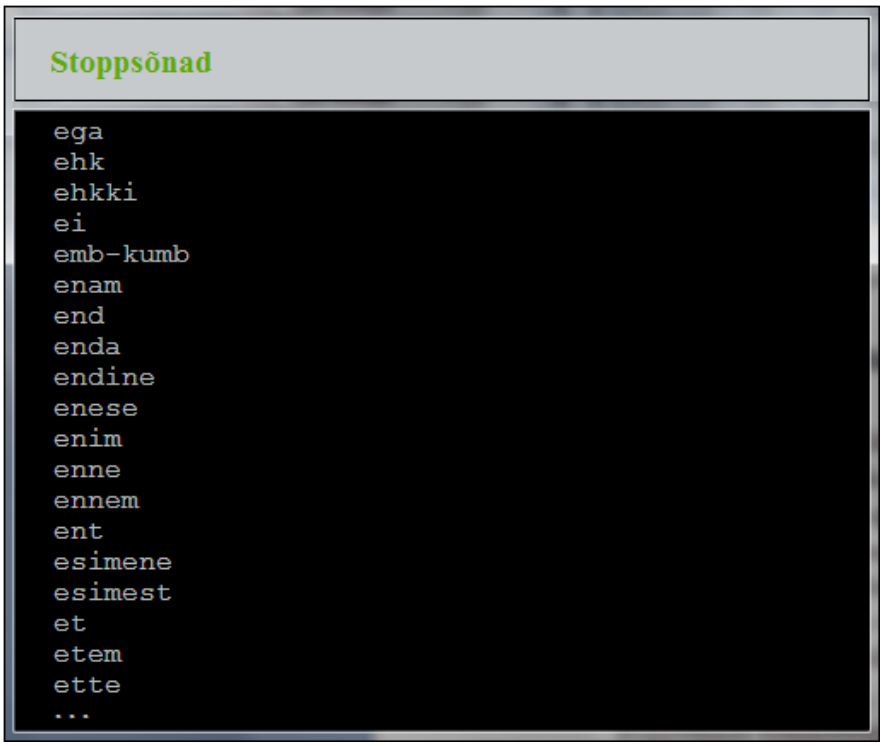
2.1 Teksti leksikaalne analüüs

Kõige olulisem ülesanne dokumendi eelanalüüsil on leida dokumendi sisust üles sõnad. Selleks kasutatakse enamasti teadmust, et tekstis sõna algab ja lõpeb tühikuga. Sellist meetodit kasutades jääb aga sõnade „külge“ ka enamus kirjavahemärke ja muid sümboleid, mis tuleb eraldi kustutada. Lisaks sõnadele ja erisümbolitele, sisaldavad dokumendid veel numbreid, mida ei ole samuti mõistlik indekseerida ning mis tuleks eelnevalt loendist elimineerida [2:224]. Numbrid aga ei pruugi tekstis esineda alati iseseisvalt, vaid võivad olla osa mõnest sõnast. Näiteks võib tekstis esineda termin *„802.11n“*, mis sisaldab nii numbrit, tähte kui ka kirjavahemärki. Sellist terminit ei tohiks loendist elimineerida, küll

aga võib jätta ära kirjavahemärgi nii päringus kui ka dokumendis sisalduvast terminist, mis ei mõjuta otsingu lõpptulemust (näiteks terminit „*www.google.ee*“ saab esitada kujul „*wwwgoogleee*“) [2:225]. Oluline on veel muuta tekstis kõik sõnad läbivalt väiketäheliseks, et vältida olukorda, kus sõnu *„Televiisor“* ning *„televiisor“* loetakse erinevateks [10:29].

2.2 Stoppsõnade elimineerimine

Dokumendist on mõistlik enne indekseerimist elimineerida hulk sõnu, mis raskendaksid päringule asjakohase vaste leidmist [2:226]. Nendeks sõnadeks võivad olla väga kõrge esinemissagedusega sõnad, samuti ase-, kaas-, määr- ja sidesõnad ning paljud muud semantilist informatsiooni mittekandvad sõnad [16] [9:806]. Sellist sõnade loendit nimetatakse stoppsõnade nimekirjaks. Kuna terminite esinemissagedus (*term frequency*) sõltub väga palju kontekstist, kus nad esinevad, dokumentide hulgast ja dokumentide suurustest, siis võib see nimekiri varieeruda. Siiski on võimalik koostada universaalne loend, mida saab kasutada stoppsõnade elimineerimiseks enamike dokumentide puhul. Järgneval joonisel *2* on väljavõte eesti keeles esinevatest stoppsõnadest.

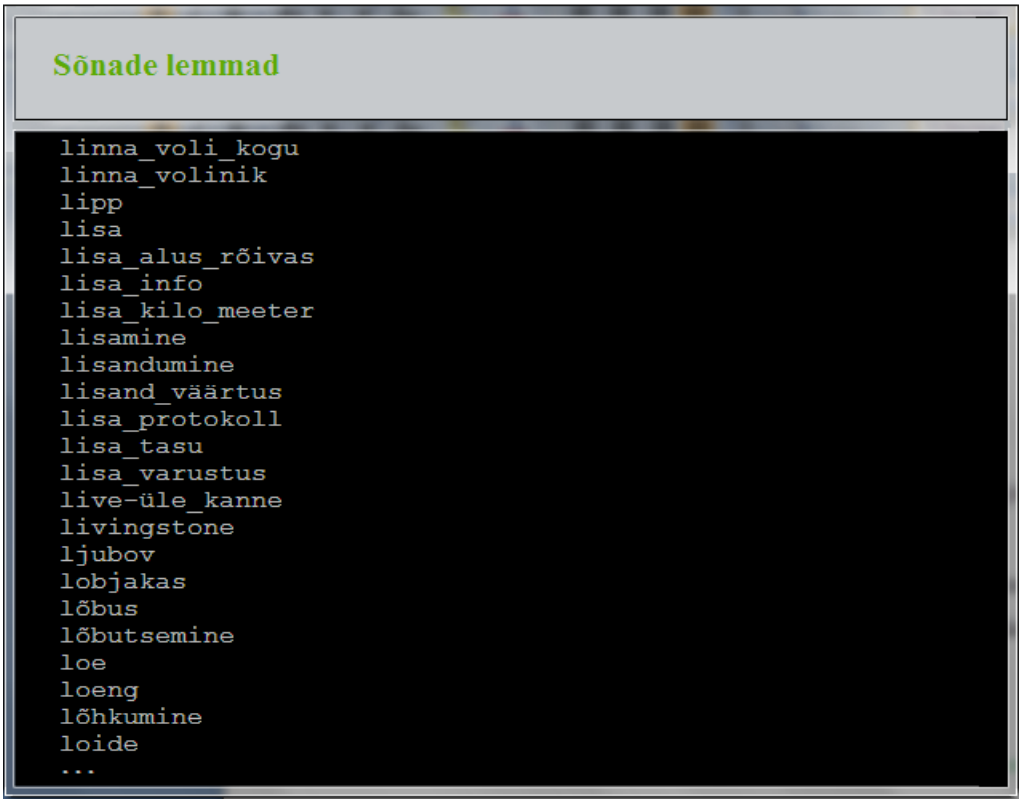


Joonis *2*. Stoppsõnade näidisloend [16]. (Kogupikkuses nimekiri lisas *2*.)

Lisaks vähest semantilist informatsiooni kandvate sõnade kaotamisele tekstist vähendab stoppsõnade elimineerimine ka indekseeritavate terminite hulka ning kiirendab indekseerimise protsessi. Stoppsõnade nimekirja koostamisel tuleks siiski olla ettevaatlik. Tekstist liialt palju sõnu elimineerides väheneb ka kasutajale kuvatavate tulemite koguhulk [2:226].

2.3 Terminite lemmatiseerimine

Kasutaja sisestab päringusse iga termini tavaliselt ühekordselt ning mingis kindlas vormis. Selleks, et vältida olukorda, kus otsisüsteem väljastab kasutajale vaid sellised tulemused, mis sisaldavad terminit täpselt sellel kujul, nagu päringus esitatud, kasutatakse sõnade algvormide leidmist ehk lemmatiseerimist [2:226]. Näiteks sõnadel *„müüja“*, *„müüma“* ning *„müük“* on ühine tüvi *„müü“*, kuhu ühtlustamiseks lisatakse *mine*-lõpp (*„müümine“*) [9:806]. Eesti keele jaoks on koostatud programm *t3mesta*, mis suudab leida igale terminile tekstist morfoloogilise analüüsi [12]. Seda tulemust edasi töödeldes on võimalik saada sõnade algvormid. Järgneval joonisel *3* on näide eesti keele sõnade lemmadest.



Joonis *3*. Lemmade näidisloend. (Loendi loomisel kasutatud lähtekood koos selgitustega lisas *4*.)

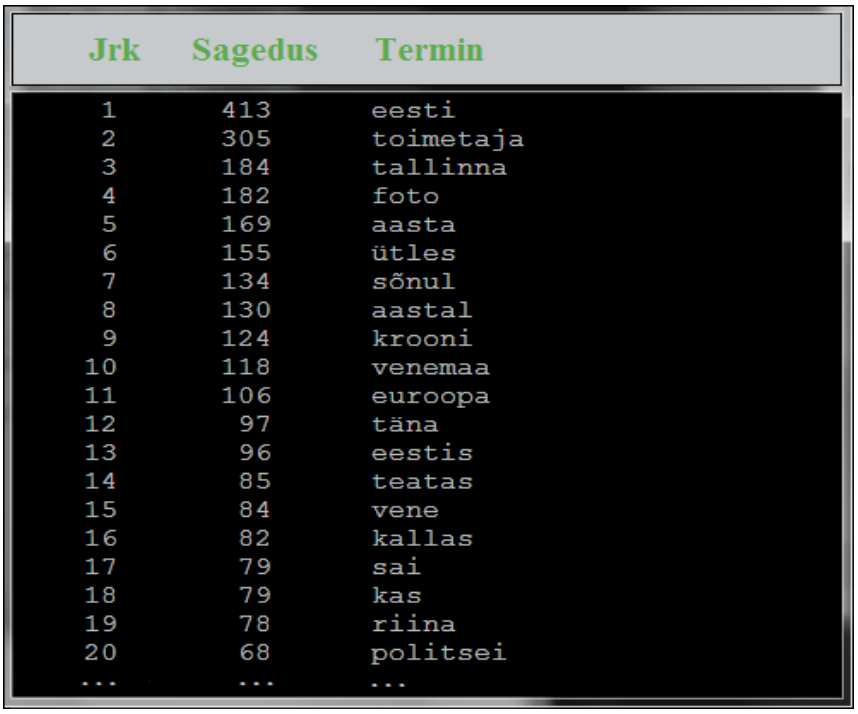
Sarnaselt stoppsõnade elimineerimisele vähendab ka lemmatiseerimine indekseeritavate terminite hulka, kuna loendist kõrvaldatakse sõnade käände- ja pöördevormid [2:226].

3. Terminite ja dokumentide indekseerimine

Iga tekstidokumenti on võimalik kirjeldada kui võtmesõnade kogumit, milles kõik sõnad on indekseeritud [2:61]. Siiski, tavaliselt ei indekseerita kõiki sõnu, vaid valitakse terminid, mis on teatava valdkonna esindamisel tähtsamad [2:66]. Lisaks terminitele sisaldavad dokumendid veel struktuurset informatsiooni ehk metaandmeid (*metadata*). Sel juhul jaotatakse dokument tsoonideks, kus saab eristada osi [10:110]. Kasutades kahendmudeli meetodit (*boolean model*), siis termin või dokument kas sobib või mitte kasutaja päringuga. Suurte dokumendikollektsioonide puhul võib aga sobivuste arv minna nii suureks, et kasutaja ei jõua kogu seda nimekirja läbi vaadata. Seetõttu on vajalik dokumente kuidagi järjestada, et kasutajal oleks mugavam tulemitega tegeleda, säästes ühtlasi aega [10:109]. Järjestamisel kasutatakse terminitele ja dokumentidele omistatavate kaalude arvutamisel peamiselt vektoresituse meetodit [2:61].

3.1 Terminite esinemissagedus

Päringus olevale terminile, mis esineb dokumendis või mõnes selle tsoonis, omistatakse väärtus, mis eristab teda teistest terminitest. Selle väärtuse arvutamisel võetakse arvesse, mitu korda esineb termin t dokumendis d. Sellist meetodit nimetatakse terminile esinemissageduse leidmiseks. Edaspidi tähistame sellist esinemissagedust *TE* [10:117]. Võime kujutleda dokumentide kollektsiooni, milles on sada tuhat dokumenti. Termin, mis sisaldub igas dokumendis sellest kollektsioonist, ei ole kasulik määramaks, milline dokument on kasutaja päringule vastuse leidmiseks asjakohane. Termin, mis esineb näiteks viis korda, on palju suurema tähtsusega. Selleks, et neid termineid tähtsuse poolest teineteisest eristada, omistatakse neile kaalud $w_{i,j}$ (kus i tähistab terminit ja j dokumenti), mis sõltuvad terminite esinemissagedustest dokumentides [2:66]. Järgneval joonisel *4* on dokument eelnevalt tükeldatud terminiteks, elimineeritud stoppsõnad ja leitud terminite esinemissagedused.



Jrk	Sagedus	Termin
1	413	eesti
2	305	toimetaja
3	184	tallinna
4	182	foto
5	169	aasta
6	155	ütles
7	134	sõnul
8	130	aastal
9	124	krooni
10	118	venemaa
11	106	euroopa
12	97	täna
13	96	eestis
14	85	teatas
15	84	vene
16	82	kallas
17	79	sai
18	79	kas
19	78	riina
20	68	politsei
...	...	...

Joonis *4.* Terminite esinemissageduste loend. (Loendi loomisel kasutatud lähtekood koos selgitustega lisas *3.*)

3.1.1 Terminitele kaalude omistamine

Olgu $e_{i,j}$ termini k_i esinemissagedus dokumendis d_j. Kogu esinemissagedus E_i termini k_i jaoks arvutatakse kõikide esinemissageduste summana järgmise valemiga:

$$E_i = \sum_{j=1}^{N} e_{i,j} \,, \qquad (1)$$

kus *N* on dokumentide hulk kollektsioonis. Dokumendi esinemissagedus (*document frequency*) termini k_i jaoks on nende dokumentide hulk, kus termin esineb, ning seda tähistatakse n_i, kusjuures $n_i \leq E_i$ [2:67].

Dokumendis d_j esineva termini k_i kaal on võrdne tema esinemissagedusega $e_{i,j}$. Seega $te_{i,j} = e_{i,j}$.

Kuna järgmises alajaotuses vaadeldav dokumendi pööratud esinemissagedus on logaritmilisel kujul, siis tuleb ka *TE* teisendada sellele kujule:

$$te_{i,j} = \begin{cases} 1 + log_2\, e_{i,j}, & kui\ e_{i,j} > 0 \\ 0, & teistel\ juhtudel \end{cases}, \quad (2)$$

kus on taas kasutusel kahendmudel ehk $te_{i,j} \in [0, 1]$ [2:68-69].

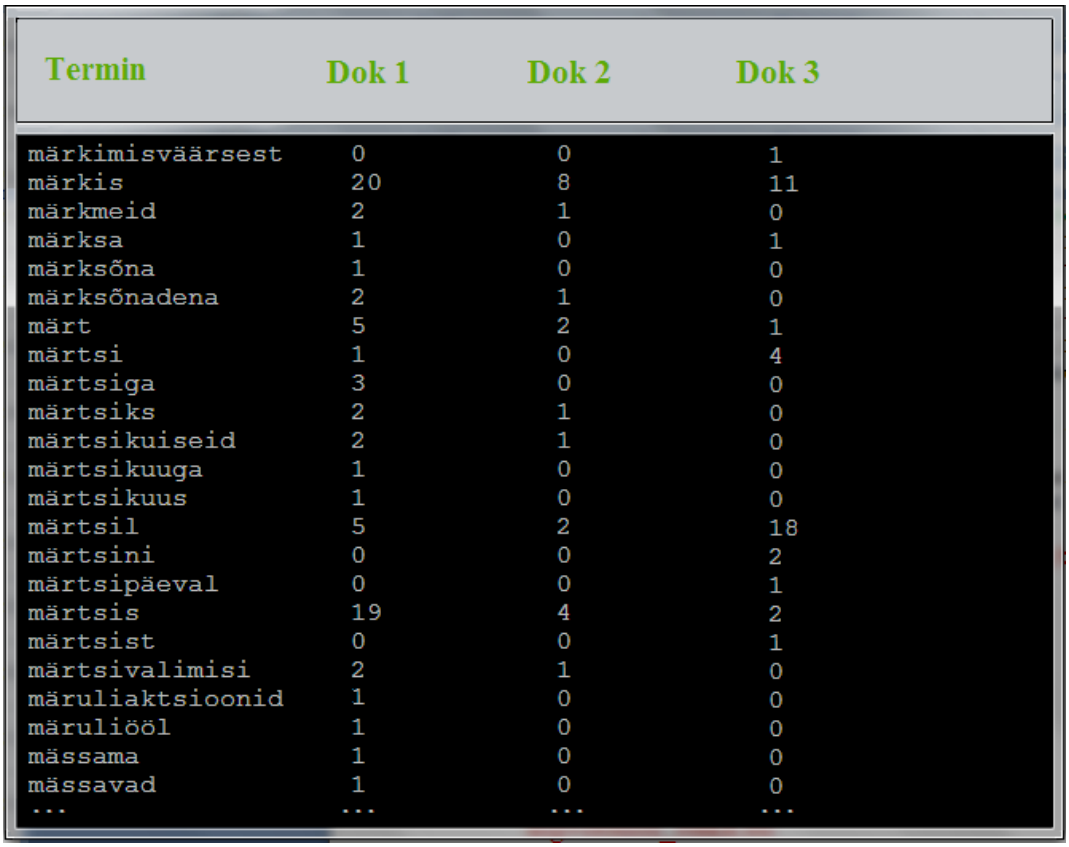
Olgu $V = \{k_2, k_2, .., k_t\}$ dokumentide kollektsiooni sõnastik. Oletame, et kolm terminit k_m, k_n ja k_l esinevad samas dokumendis d_j. Siis esitatakse neid kujul $[k_m, k_n, k_l]$. Sõnastiku V jaoks, mille suurus on t, leidub 2^t erinevat esitust nende kolme termini jaoks. Illustreerides eelnevat esitusviisi, vaatame näidet, kus esimene termin esineb dokumendis ning teised kaks mitte. Sel juhul näeb esitus välja vektormudelina (1, 0, 0) ning kahendmudelina $k_m \wedge \neg k_n \wedge \neg k_l$. Sellist esitusviisi nimetatakse ka konjunktiivseks normaalkujuks. Iga termini konjunkt esindab päringut q otsingusüsteemile. Antud meetodi puhul esitatakse päringuid ja dokumente terminite konjunktidena ning seda nimetatakse „kotitäis sõnu“ (*bag of words*) mudeliks. [2:62].

Termin, mis esineb dokumendis, on seoses selle dokumendiga. Termini ja dokumendi vahelist seost on võimalik kirja panna järgmise termini-dokumendi maatriksina (*term-document matrix*):

$$\begin{array}{c} k_1 \\ k_2 \\ k_3 \end{array} \begin{bmatrix} e_{1,1} & e_{1,1} \\ e_{1,1} & e_{2,1} \\ e_{1,2} & e_{2,2} \end{bmatrix}, \quad (3)$$

kus iga element $e_{i,j}$ tähistab termini k_i esinemissagedust dokumendis d_j.

Selline maatriks annab rohkem informatsiooni, kui kahendmudelil põhinev esitusviis ning on aluseks terminitele kaalude omistamisel [2:62-63]. Järgneval joonisel *5* on näide termini-dokumendi maatriksist.



Termin	Dok 1	Dok 2	Dok 3
märkimisväärsest	0	0	1
märkis	20	8	11
märkmeid	2	1	0
märksa	1	0	1
märksõna	1	0	0
märksõnadena	2	1	0
märt	5	2	1
märtsi	1	0	4
märtsiga	3	0	0
märtsiks	2	1	0
märtsikuiseid	2	1	0
märtsikuuga	1	0	0
märtsikuus	1	0	0
märtsil	5	2	18
märtsini	0	0	2
märtsipäeval	0	0	1
märtsis	19	4	2
märtsist	0	0	1
märtsivalimisi	2	1	0
märuliaktsioonid	1	0	0
märuliööl	1	0	0
mässama	1	0	0
mässavad	1	0	0
...	...	...	...

Joonis *5*. Termini-dokumendi maatriks. (Maatriksi loomisel kasutatud lähtekood koos selgitustega lisas *5*.)

3.1.2 Dokumendi pööratud esinemissagedus ja kaalude omistamine

Dokumendi esinemissagedus, mida tähistame edaspidi *DE*, näitab termini k_i jaoks, mitu dokumenti kollektsioonist sisaldab antud terminit [10:118]. Sorteerime kõik *DE*-d kahanevas järjekorras. Olgu r termini järjekorranumber, mille esinemissagedus on $\mathrm{n(r)}$, ja tähistagu *N* dokumentide hulka kollektsioonis, siis:

$$\log r \sim \log N - \log \mathrm{n(r)}. \quad (4)$$

Olgu k_i termin, mille kõrgeima esinemissagedusega dokumendi number on r, siis $\mathrm{n(r)} = \mathrm{n(i)}$:

$$DPE_i = \log \frac{N}{n_i}, \quad (5)$$

kus *DPE* esitab dokumendi pööratud esinemissagedust termini k_i jaoks [2:71-72]. Vähe-esinev termin saab selle valemi kohaselt kõrgema *DPE* ja rohkelt esinev termin väiksema *DPE* väärtuse [10:118]. G. Salton ja C. S. Yang kombineerisid kaks meetodit ning lõid uue kaalude arvutamise meetodi, mida nimetatakse termini esinemissageduseks- dokumendi pööratud esinemissagedusel (*term frequency-inverse document frequency*, tähistame edaspidi *TE-DPE*). Olgu $w_{i,j}$ termini kaal vastavalt termini-dokumendi paarile (k_i, d_j), siis:

$$w_{i,j} = \begin{cases} 1 + log_2\, e_{i,j} \cdot \log \frac{N}{n_i}, & kui\ e_{i,j} > 0 \\ 0, & teistel\ juhtudel \end{cases}, \qquad (6)$$

mis näitabki, kuidas arvutada kaale *TE-DPE* meetodil [2:72].

3.2 Terminite positsioonide arvestamine dokumendis

Yue-Heng Sun'i pakutud meetod jagab dokumendi tsoonideks, kus on võimalik eristada formaati, pealkirja, autorit, loomiskuupäeva, keelt, sisu, kokkuvõtet ja valdkonda [11]. Iga sellise metaandmete tsooni jaoks on olemas parameetriline indeks (*parametric index*), mis võimaldab kasutajal piirata otsingut ainult konkreetse tunnuse järgi. Joonisel *6* on näide tsoonilisest otsimisest, kus kasutaja otsib andmebaasist dokumente, mille autoriks on W. Shakespeare.

Joonis *6*. Parameetrilise otsingu näide [10:111]. (Autor modifitseeris ja tõlkis eesti keelde.)

Terminid, mis esinevad pealkirjas, on dokumendi sisu kirjeldamisel tähtsamad kui need, mis esinevad näiteks sisus või kokkuvõttes [10:110] [11]. Kasutades termini esinemissagedust $te_{i,j}$, mis on kirjeldatud valemis *2*, lisatakse valemisse juurde muutujad, mis iseloomustavad termini asukohta dokumendis:

$$te_{i,j} = \alpha \cdot te_{ij1} + \beta \cdot te_{ij2} + \gamma \cdot te_{ij3}\,, \qquad (7)$$

kus te_{ijk} on termini esinemissagedus k-ndas piirkonnas ning α, β ja γ on faktorid, mida saab kohandada vastavalt vajadusele, arvestades tingimust, et $\alpha > \beta > \gamma \geq 1$. Selleks, et terminile vastavalt tema positsioonile dokumendis veel rohkem tähtsust omistada, kasutatakse informatsioonikogumiskordajat IK_t termini t jaoks. Valemis *7* esinevatele terminitele kaalude arvutamiseks kasutatakse *TE-DPE* sarnast valemit:

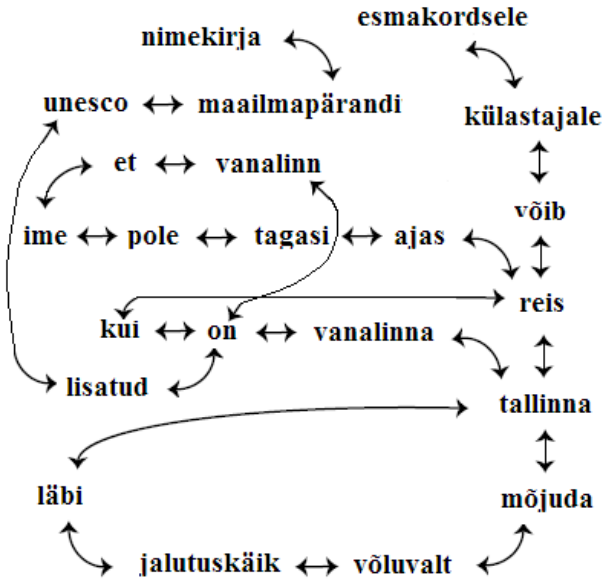
$$te_{i,j} \cdot dpe_i = \frac{te_{i,j} \cdot \log\left(\frac{N}{n_i} + 0{,}01\right) \cdot IK_t}{\sqrt{\sum_{t \in d}\left[te_{i,j} \cdot \log\left(\frac{N}{n_i} + 0{,}01\right) \cdot IK_t\right]^2}}\,, \qquad (8)$$

kus $te_{i,j}$ on termini i esinemissagedus dokumendis j ja dpe_i dokumendi pööratud esinemissagedus, n_i on terminit i sisaldavate dokumentide hulk ja *N* kogu dokumentide hulk kollektsioonis. See meetod põhineb aga „*kotitäis sõnu*“ mudelil ega ole parim viis kaalude omistamisel. Meetod on küll efektiivne lokaalse dokumendi sisu töötlemisel, kuid mitte globaalsel dokumentide kollektsioonil, kuna ei arvesta terminite vahelisi seoseid [11].

3.3 Juhusliku jalutuskäigu mudel

Hassan pakkus välja juhusliku jalutuskäigu mudeli (*random walk model*), mida võib kujutleda kui lugejat, kes liigub tekstis sammhaaval terminilt terminile. Selle „jalutamise“ ajal hinnatakse termini tähtsust selle järgi, kui suure tõenäosusega „jalutaja“ teda tekstis

kohtab. Hassan modelleeris oma süsteemi, kasutamaks terminite koos-esinemise relatsiooni, kus terminid, mis esinevad koos kindlas kontekstis, omavad suurema tõenäosusega tähtsust paarina, kui üksi analüüsides. Selleks, et modelleerida seda rekursiivset relatsiooni, kasutas ta graafi, mis põhines järjestamisalgoritmil [1]. Järgneval joonisel *7* on näide dokumendist, mis on esitatud graafina.



Joonis *7*. Dokumendi graafiline esitus [1]. Dokumendi sisu: „*Esmakordsele külastajale võib reis Tallinna mõjuda võluvalt. Jalutuskäik läbi Tallinna vanalinna on kui reis ajas tagasi. Pole ime, et vanalinn on lisatud UNESCO maailmapärandi nimekirja.*“ Kirjavahemärgid on elimineeritud ning sõnad muudetud läbivalt väiketähelisteks.

Graafi tippudeks on terminid ning graafi servadele omistati kaalud, arvestades ka niinimetatud sumbuvustegurit (*damping factor*), mis hindab servaga ühendatud tippude koosesinemise sagedust. Kui näiteks sõnapaar esineb tekstis 4 korda, siis neid kahte sõna ühendava serva kaaluks saab 4. Lisaks sumbuvustegurile, arvutatakse servadele kaalusid järgneva valemiga:

$$S_{v_1, v_2} = te \cdot dpev_1 \text{ x } te \cdot dpev_2 \,, \qquad (9)$$

kus S_{v_1, v_2} on serv tipu v_1 ja v_2 vahel ning $te \cdot dpev_1$ ja $te \cdot dpev_2$ esindavad vastavalt tippude v_1 ja v_2 *TE-DPE*-sid. Sumbuvustegurit võib $d_{S_{v_1, v_2}}$ kirjeldada kui funktsiooni tippu sisenevate servade kaaludest. Sumbuvustegurit arvutatakse valemiga:

$$d_{S_{v_1, v_2}} = \frac{S_{v_1, v_2}}{S_{max}}, \tag{10}$$

kus S_{max} on kõrgeima kaaluga serv graafis [11]. Tippude järjestamise puhul kasutatakse põhimõtet, et kui üks tipp on ühendatud teisega serva abil, siis saab see teine tipp juurde ühe „hääle“. Mida rohkem hääli ühel tipul on, seda suurema tähtsusega see tipp on. Olgu meil graaf $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ ning $\mathrm{S}(v_a)$ nende tippude hulk, mis viitavad tipule v_a ja $\mathrm{V}(v_a)$ nende tippude hulk, millele tipp v_a ise viitab. Tipu skoori saab arvutada järgneva valemi abil:

$$\mathrm{P'(v_a)} = \frac{(1-d)}{|N|} + \sum_{v_b \in S(v_a)} C \cdot \frac{d_{S_{v_b, v_a}} \cdot P(v_b)}{|V(v_b)|}, \tag{11}$$

kus *N* on kogu tippude hulk graafis, d on sumbuvustegur, mis saab väärtuseks 0,85 ning C skaalakonstant (*scaling constant*), mis saab väärtuseks 0,95. $\mathrm{P}(v_b)$ on tipu V_a skoor, kusjuures $V_a \in \mathrm{S}(v_a)$:

$$\mathrm{P}(v_b) = (1 - d) + \mathrm{d} \cdot \sum_{v_b \in S(v_a)} \frac{\mathrm{P}(v_b)}{|V(v_b)|}. \tag{12}$$

Valemit:

$$V_{min} = \frac{(1-d)}{N}, \tag{13}$$

kasutatakse tipule skoori määramiseks juhul, kui tippu sisenevate kaarte arv on 0. [6]

4. Päringute formuleerimisel kasutatavad tehnikad

4.1 Asjakohane tagasiside kasutajalt

Asjakohane tagasiside on infootsingus kasutatav tehnika, mille eesmärk on muuta esialgset kasutaja poolt sisestatud päringut nii, et see vastaks rohkem tema informatsioonivajadusele [10:178]. Kasutaja kaasatakse aktiivselt tegevusse hindamaks tagastatud tulemite sobivust esialgse päringuga. Protsessi saab kirjeldada järgmise eeskirjaga:

1) Kasutaja sisestab infovajadusega kooskõlas oleva päringu otsingusüsteemile.

2) Otsingusüsteem väljastab kasutajale leitud tulemused. Kasutaja märgib dokumendid, mis on tema arust sobivad, ning dokumendid, mis on mittesobivad.

3) Süsteem töötleb kasutajalt saadud informatsiooni ja väljastab uued tulemused arvestades kasutaja tehtud valikuid.

4) Kasutaja hindab jällegi dokumentide sobivust ning sisestab uue päringu otsingusüsteemile. Protsess jätkub vajadusel sammul 2.

Selline protsess võib läbi töötada üks kuni mitu iteratsiooni [18]. Suurimat efektiivsust on tüüpiliselt märgata peale esimese iteratsiooni läbimist, sest kasutaja on teinud oma valikud ning süsteem väljastab uuesti otsitud dokumendid koos esimeses iteratsioonis juba sobivateks tunnistatud dokumentidega [9:811]. Algoritmi korduv läbimine võib olla tingitud näiteks asjaolust, et kasutajale väljastatakse esimese iteratsiooni lõpuks tulemused ning neid uurides võib ta saada parema ülevaate otsitavast valdkonnast. Seejärel teeb kasutaja muudatusi ning esitab uue päringu juba suuremalt taustateadmiste baasilt [10:178]. Asjakohases tagasisides kasutatakse peamiselt kahendmudeli põhimõtet ehk kasutaja hindab kahte valikut: dokument sobib või mitte [18].

4.1.1 Asjakohane tagasiside vektorruumi mudelil ning Rocchio algoritm

Enamik praegu kasutusel olevatest asjakohase tagasiside algoritmidest kasutab vektorruumi mudelit (*vector space model*), mida saab esitada järgneva võrrandiga:

$$\vec{q}(t) = \vec{q}(t-1) + \alpha \sum_{d_p \in D_p} \overrightarrow{d_p} - \beta \sum_{d_n \in D_n} \overrightarrow{d_n}, \quad (14)$$

kus $\vec{q}(t)$ tähistab päringuvektorit q t- ndal otsingul, $\vec{d}$ dokumendi d vektorit ning α ja β on koefitsiendid, kus α näitab, kui palju tuleb uut vektorit nihutada sobivatele dokumentidele lähemale ja β näitab, kui palju nihutada mittesobivatest dokumentidest kaugemale. D_p tähistab sobivate dokumentide hulka ning D_n mittesobivate dokumentide hulka [18]. Muutujad α ja β varieeruvad vahemikus [0, 1] ning kehtib valem $\alpha + \beta = 1$. Arvutiteaduse professorid G. Salton ning C. Buckley on pakkunud välja väärtused $\alpha =$ 0,75 ning β = 0,25, sel juhul annab algoritm efektiivseid tulemusi [9:811].

Tuntuim vektorruumi mudelil põhinev võrrand on *Rocchio* algoritm, mida saab esitada järgneval kujul:

$$\vec{q}(t) = \vec{q}(t-1) + \frac{1}{|D_p|} \sum_{d_p \in D_p} \overrightarrow{d_p} - \frac{1}{|D_n|} \sum_{d_n \in D_n} \overrightarrow{d_n}, \quad (15)$$

mis kasutab sobivate ning mittesobivate dokumentide pöördväärtuseid ($\frac{1}{|D_p|}$ ja $\frac{1}{|D_n|}$). Päringuvektorit $\vec{q}_{opt}$ kasutatakse sobivate dokumentide hulga maksimeerimisel ning mittesobivate dokumentide hulga minimeerimisel:

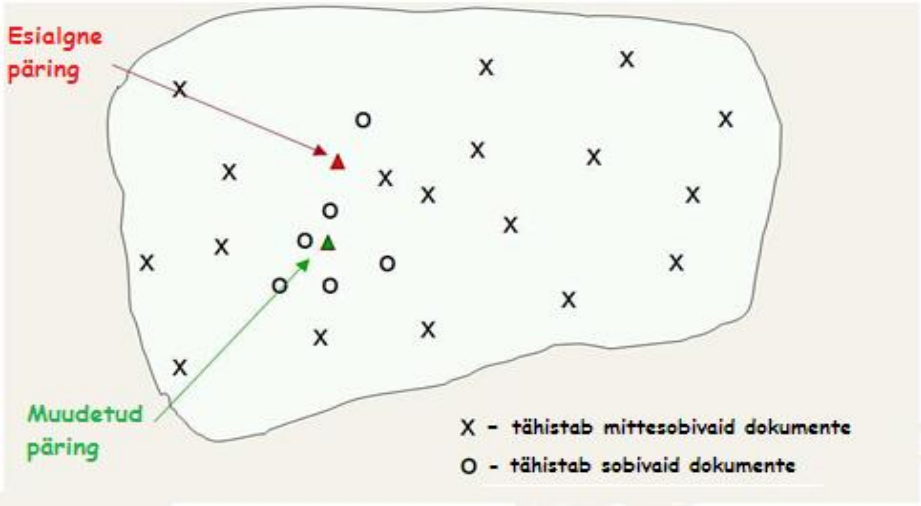
$$\vec{q}_{opt} = \max[\mathrm{sim}\ (\vec{q},\ D_p) - \mathrm{sim}\ (\vec{q},\ D_n)], \quad (16)$$

kus sarnasusfunktsiooni *sim* arvutatakse järgneva valemiga:

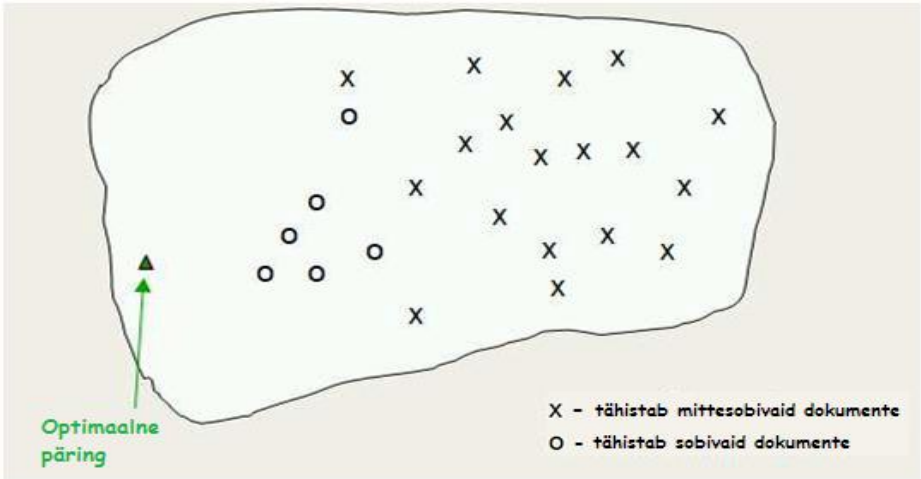
$$\mathrm{sim}(d_1,\ d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| \cdot |\vec{V}(d_2)|}\ , \qquad (17)$$

kus d_1 ja d_2 on kaks dokumenti, mille vahel sarnasust leitakse, ning $\vec{V}(d_1)$ ja $\vec{V}(d_2)$ nende dokumentide esitused vektorkujul.

Järgneval joonisel *8* on *Rocchio* algoritm esitatud graafiliselt, kus osad dokumendid on esitatud sobivatena ja osad mittesobivatena ning esialgne päringuvektor on nihutatud vastavalt kasutaja tagasisidele.



Joonis *8*. *Rocchio* algoritmi graafiline esitus [10: 181]. (Autor tõlkis eesti keelde.)



Joonis *9*. *Rocchio* optimaalse päringu graafiline esitus [10:182]. (Autor tõlkis eesti keelde.)

Joonisel *9* on *Rocchio* algoritmiga leitud optimaalne päring (*optimal query*), mis eraldab sobivad dokumendid mittesobivatest, kasutades valemit *16* [10:182].

4.1.2 Asjakohase tagasiside hinnang

Asjakohane tagasiside on efektiivne tehnika, kui ta suudab tagada, et kommunikatsioon kasutaja ning veebi vahel hõlmab ka olemasolevaid laialdaselt kasutatavaid otsingumootoreid (näiteks *Google*'it). Selleks tuleb päringuvektor $\vec{q}$ konverteerida võtmesõnade hulgaks, mida on võimalik esitada päringuna otsimootoritele. Üks võimalus seda teha on valida paar võtmesõna, mis omavad suuremat kaalu päringuvektoris kui teised. Siiski on päringuvektori konverteerimine võtmesõnadeks üpris kaudne lähenemine järeldamaks kasutaja eelistusi, kuna kasutaja peab valima dokumentide hulgast sobivaid dokumente, konstrueerides samal ajal efektiivset päringut, mis sisaldaks võtmesõnu. Samuti võib probleemiks saada asjaolu, et kasutaja teadmised spetsiifilistest valdkondadest ei ole piisavad, et otsustada dokumentide sobivuse üle. Vektorruumi mudelil põhineva asjakohase tagasiside miinuseks võib veel pidada seda, et uuesti genereeritavad päringud on seotud vaid hetkel aktiivsete dokumentide hulgaga ehk teisisõnu ei ole võimalik genereerida päringuid teemade kohta, mida aktiivsete dokumentide hulgas ei ole. Võtame näitena sõnad *A* ja *B*, mis esinevad tihti koos ja kuuluvad samasse valdkonda. Kui kasutaja soovib antud valdkonnas minna sügavamale, otsides ainult sõna *A* järgi ning välistades *B*, siis on raske genereerida sellist päringut, mis sisaldab sobivat dokumenti hetkel kasutuses olevast dokumentide hulgast [18].

4.1.3 Asjakohane tagasiside kasutades võtmesõnade kaarti

Määrates kindlaks võtmesõnade asukohad ja sagedasti koosesinevad võtmesõnade paarid, on võimalik koostada kahemõõtmeline võtmesõnade kaart (*keyword map*), kus ühes osas on päringus sisalduvatest võtmesõnast erinevad, kuid nendega relatsioonis olevad sõnad, ning teises osas võtmesõnad, mis sisalduvad päringus. Kasutajale kuvatakse võtmesõnade kaart ja ta saab otsustada, kas võtmesõnade paigutus kaardil vastab tema taustateadmistele või mitte. Kujutleme kahte stsenaariumit:

1) kasutaja paigutab lähestikku võtmesõnad *A* ja *B*, mis esialgselt olid paigutatud teineteisest eemale.

2) Kasutaja asetab võtmesõnad *A* ja *B* teineteisest kaugemale, kuigi esialgu olid nad kaardil lähestikku.

Esimesel juhul paigutatakse kasutajale kuvatavate dokumentide järjekord ümber vastavalt kasutaja tagasisidele. Neid võtmesõnade paare, mida kasutaja arvates tuli lähendada, nimetatakse kaugemalt-lähemale võtmesõnade paarideks (*Far2Near keyword pairs*), mida tähistame edaspidi *k-l*. Teisel juhul võib tekkida kaks võimalust. Kasutaja võib soovida dokumente, mis sisaldavad võtmesõna *A*, kuid ei sisalda võtmesõna *B*. Samuti võib tekkida olukord, kus tuleb jagada valdkond kaheks: teema sisaldab kas võtmesõna *A* või sisaldab võtmesõna *B*. Neid võtmesõnade paare, mida tuleks teineteisest eraldi asetada, nimetatakse lähemalt-kaugemale võtmesõnade paarideks (*Near2Far keyword pairs*), mida edaspidi tähistame *l-k*. Võtmesõnade paarid *k-l* ning *l-k* kuvatakse uuel võtmekaardil vastavalt kasutaja tehtud muudatustele. Neid paare arvutatakse järgneva algoritmi abil:

1) Eristatakse sisendfaili (*S-andmed*) ja modifitseeritud faili, kus iga võtmesõna on esitatud koordinaatidena (*XY-andmed*). *S-andmetesse* salvestatakse sarnasus S_{Ki} (=R_{lm} ∈ [0, 1]) iga võtmesõnapaari p_i (= w_l, w_m) vahel. *XY-andmetesse* salvestatakse koordinaadid x_i, y_i vastavalt kasutaja tehtud muudatustele.

2) Arvutatakse sarnasus S_{Xi} iga võtmesõnapaari p_i vahel, kus d_i on kaugus kahe võtmesõna vahel failis *XY-andmed*. d_M on maksimaalne kaugus kõigi võtmesõnade paaride vahel.

$$S_{Xi} = 1 - \left(\frac{d_i}{d_M}\right). \quad (18)$$

3) Arvutatakse iga võtmesõnapaari p_i jaoks maksimaalne kaugus Kaugus(S_{Ki}) ning lähedus Lähedus(S_{Ki}), kasutades faili *S-andmed*. Samuti arvutatakse kaugus ja lähedus ka *XY-andmete* jaoks. Kasutatakse järgnevaid valemeid:

$$\text{Kaugus}(x) = \max\left(-\frac{x}{t} + 1,\ 0\right), \quad (19)$$

$$\text{Lähedus}(x) = \max\left(\frac{x-t}{1-t},\ 0\right). \quad (20)$$

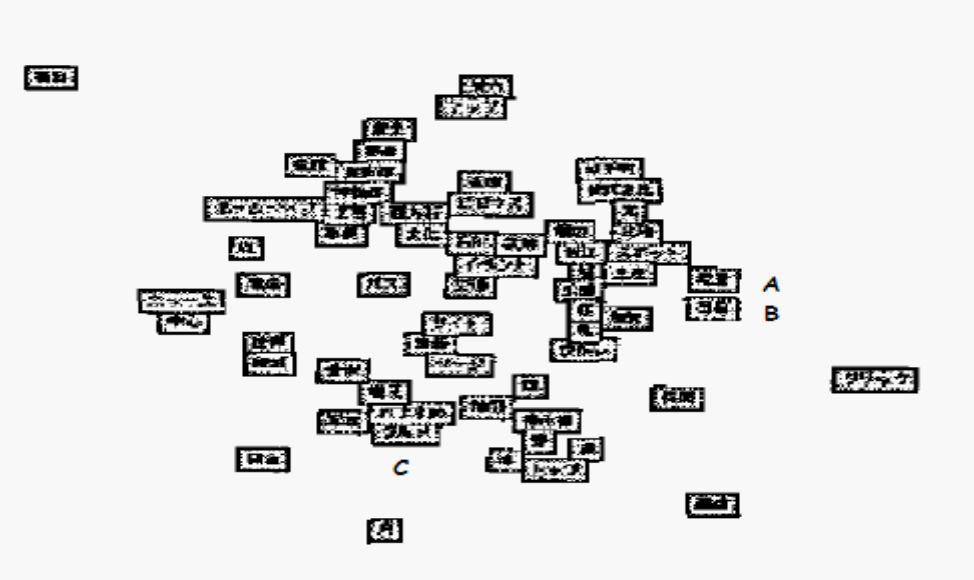
4) Võtmesõnade paare eristatakse selle järgi, kui kõrge väärtuse nad järgneva võrratuse abil saavad. Neid nimetatakse vastavalt *k-l* ning *l-k* võtmesõnade paarideks.

$$V_i^{k-l} = \max(\text{Lähedus}(S_{Xi}),\ (\text{Kaugus}(S_{Ki}))\ \ldots\ \text{Lähedus}(S_{Xi}),\ (\text{Kaugus}(S_{Ki}) > 0. \quad (21)$$

$$V_i^{l-k} = \max(\text{Lähedus}(S_{Ki}),\ (\text{Kaugus}(S_{Xi}))\ \ldots\ \text{Lähedus}(S_{Ki}),\ (\text{Kaugus}(S_{Xi}) > 0. \quad (22)$$

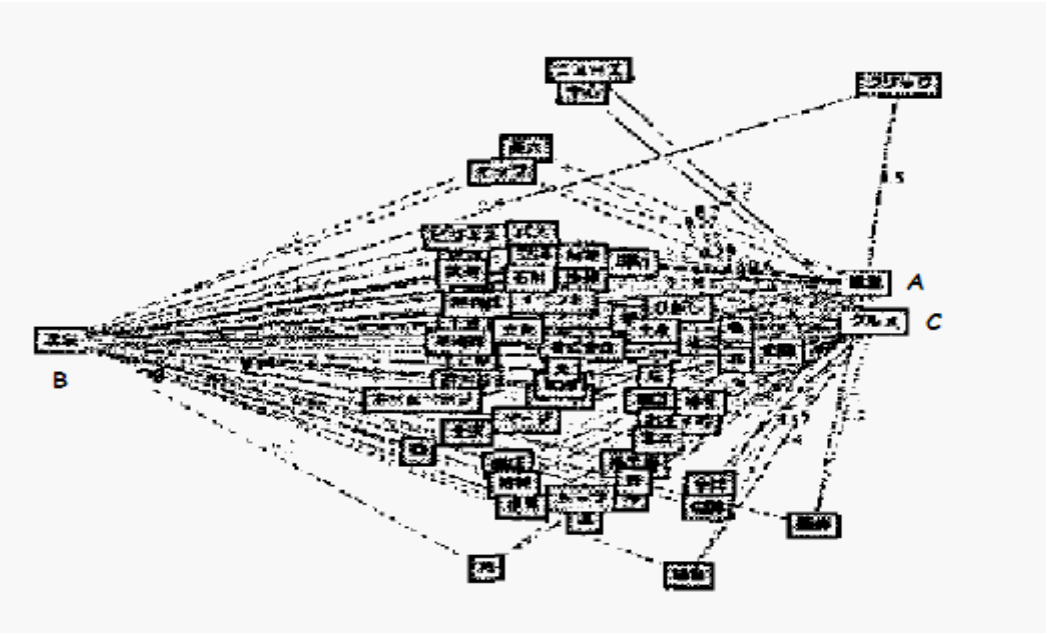
Selleks, et algoritmi võtmesõnade kaardil realiseerida, tuleb parameetrile t valemites *19* ja *20* anda väärtus. *S-andmete* mõõtmisel kasutatakse kauguse jaoks väärtust t = 0,25 ja läheduse jaoks t = 0,5. Eeldatakse, et võtmesõnad on tihedalt seotud, kui sarnasus R_{lm} ületab väärtuse 0,5. *XY-andmete* puhul võib anda kaugusele väärtuse t = 0,5 ja lähedusele t = 0,9. Läheduse väärtus on kõrge, sest eeldatakse, et kasutaja asetab vähem seoses olevad sõnad kaugemale kui 0,5 maksimaalsest.

Oletame, et kasutaja soovib otsida kolme võtmesõna (*A*, *B* ja *C*) järgi ning esitab päringu otsingusüsteemile. Järgnev joonis *10* illustreerib näidet võtmesõnade kaardist, mis on koostatud esialgsete otsitulemuste põhjal.



Joonis *10*. Esialgsete dokumentide hulgast genereeritud võtmesõnade kaart (sisaldab sõnu *A*, *B* ja *C*) [18]. (Autor modifitseeris joonist.)

Sõnad *A* ja *B* on lähestikku, kuna nad esinesid dokumentides sageli koos, sõna *C* on kaugemal, kuna esines vaid ühes dokumendis koos sõnadega *A* ja *B*.



Joonis *11*. Kasutajalt saadud informatsiooni põhjal modifitseeritud võtmesõnade kaart (sisaldab sõnu *A*, *B* ja *C*) [18]. (Autor modifitseeris joonist.)

Joonisel *11* on sõnad *A* ja *C* on lähestikku ning sõna *B* kaugemal. Tulemuse saamiseks arvutati $k-l$ ning $l-k$ võtmesõnade paarid. Sõnade paigutused joonisel *10* ja joonisel *11*

on erinevad, kuna esialgsed ning kasutaja poolt muudetud võtmesõnade paarid on erinevad ja sõltuvad palju kontekstist ning ka asjaolust, et osadel sõnadel võib olla mitu tähendust [18].

4.2 Päringu laiendamine

Keskmine tavakasutaja sisestab oma päringusse enamasti 2-3 võtmesõna. See ei ole aga piisav hulk informatsiooni infootsingu süsteemi jaoks ning kuvatavates tulemites võib olla palju ebasobivaid dokumente. Kasutajad omakorda ei saa lühikese päringu tõttu asjakohaseid tulemusi, mis rahuldaksid nende informatsioonivajadust. Selle probleemi lahendamiseks laiendatakse päringut ja formuleeritakse see siis otsisüsteemile uuesti [20].

Päringu laiendamiseks nimetatakse tehnikat, kus kasutaja esialgset päringut laiendatakse, lisades juurde termineid, mis on lähedalt seotud päringus olevatega [8]. Näiteks „*kotitäis sõnu*“ mudeli puhul ei kuvata kasutajale dokumenti, kui selles ei esine päringu termineid täpsel kujul. Päringu laiendamise tehnika peamine eesmärk ongi vähendada mittesobivust päringu ja dokumendi vahel, lisades päringusse juurde sõnu või fraase [14]. Protsessi efektiivsus sõltub suuresti sellest, milliseid sõnade vahelisi seoseid kasutatakse. Võtame näitena termini „*Java*“, mis on tihedalt seotud programmeerimisega. Kui lisame päringu laiendamise meetodit kasutades terminile „*Java*“ sõna „*reisid*“, siis tagastatud dokumentides tõenäoliselt väga palju informatsiooni programmeerimise kohta ei ole. Põhjuseks on see, et terminil „*Java*“ on mitu tähendust. Ühes kontekstis võib tegemist olla tõepoolest programmeerimiskeelega, kuid teisalt kannab nime Java ka Indoneesiale kuuluv saar, kuhu korraldatakse turismireise [7] [17]. Kasutusel on meetodid, mis võtavad vajaliku informatsiooni päringu käigus tagastatavatest dokumentidest, kuid enamjaolt võetakse need sõnad, fraasid ja sõnadevahelised seosed tesaurustest (*thesaurus*), mida on võimalik koostada nii käsitsi kui ka automaatselt.

4.2.1 Tesaurused

Tesauruseks nimetatakse mõistelist sõnaraamatut, mis sisaldab sünonüüme ja omavahel tihedalt seotuid sõnu, kusjuures sõnad ja väljendid pole järjestatud mitte tähestikuliselt, vaid semantilisi seoseid pidi. [10:190] [5]. Enamjaolt on terminid esitatud üksikute sõna-

dega, kuid kui seda pole mõnes loomulikus keeles võimalik teha, siis kasutatakse mitmest sõnast koosnevat gruppi ühe tervikuna [2:229]. Näiteks eestikeelne väljend „kivi kotti“ esineb tervikuna. Tesauruseid on erinevaid ning nad põhinevad tihti mingil kindlal valdkonnal. On olemas näiteks tehnika ja tehnoloogia terminite tesaurus, samuti ka loodusteaduste termineid sisaldavad tesaurused. Tuginedes D. Foskett'ile saab tesauruste kasutusvaldkondi jagada kolmeks: pakkuda standardset sõnavara indekseerimisel ja otsimisel, abistada kasutajaid otsimaks termineid efektiivse päringu formuleerimisel, pakkuda klassifitseeritud hierarhiaid, mis lubaksid laiendada või kitsendada konkreetset päringut, et vastata kasutaja informatsioonivajadusele.

Tesauruste kolm peamist komponenti on: indekseeritud terminid, seosed nende terminite vahel ja nende terminite vaheliste seoste kujundus ning konstruktsioon. Indekseeritud terminid on enamjaolt nimisõnad või siis ma-lõpuliseks teisendatud tegusõnad (näiteks sõna „helistas“ on teisendatud kujule „helistama“) [2:229]. Eesti keele jaoks on loodud *TEKSaurus*, mis koos viidetega ingliskeelsele tesaurusele *WordNet* moodustab eesti *WordNet*'i (*EstWN*) [5]. Joonisel *12* on näide eestikeelsest *TEKSaurus*'est, kus päringuna on sisestatud sõna „helistama“. Tulemuseks on 2 vastet erineva tähendusega: üks on heliga märku andma ja teine suhtlema telefonitsi. Lisaks on märgitud sõnale sõnaliik ning see, kuidas ta on lähedalt seotud mõne teise sõnaga. Antud näites on sõna „helistama“ seotud tihedalt sõnadega „kõlama“, „suhtlema“ ja „tilistama“.

Joonis *12*. *TEKSaurus*'e otsingu näide [13]. (Autor modifitseeris joonist.)

Tesauruste kasutamine on küll lihtne ja kiire viis päringut laiendada, kuid siiski on sellel ka miinuseid. Need tesaurused, mis on koostatud käsitsi enamjaolt lingvistide poolt, on tihti liiga laialivalguvad või siis vastupidi liiga väikese andmebaasiga ning võivad olla subjektiivsed. Samuti ei ole nende koostamisel rangelt ettekirjutatud nõudmisi, mistõttu neid võivad koostada väga erineva taustaga inimesed. Tesauruseid on tarvis ka pidevalt uuendada ja töökorras hoida. Käsitsi on seda teha keeruline, sest uut informatsiooni tuleb pidevalt juurde. On olemas ka meetodid, mis suudavad tesaurust automaatselt uuendada, säilitades samal ajal positiivsed omadused: lihtsuse ja kiiruse. Tesauruse struktuuri uuendatakse pidevalt, kasutades selleks tesauruse kasutajate eelistusi ning dokumentidest leitud termineid, mistõttu on see loogilisem [8].

4.2.2 Päringu laiendamine globaalsel analüüsil

Kõige levinum päringu laiendamise meetod põhineb globaalsel analüüsil (*global analysis*), kus kasutatakse tesaurusi. Igale terminile, mis esineb päringus, saab tesaurusest automaatselt lisada sünonüüme ja terminiga tihedalt seotud sõnu. Tesaurusel baseeruva päringu laiendamise suureks eeliseks on see, et ta ei nõua kasutaja osalemist protsessis [10:190].

Üks automaatselt koostatud tesauruste liik on sarnasus-tesaurus (*similarity thesaurus*). Sarnasus arvutatakse iga kahe termini vahel, mis esinevad dokumentide kollektsioonis. Olgu t terminite hulk kogu dokumentide kollektsioonis, *N* tähistagu dokumentide hulka ning $e_{i,j}$ termini k_i, esinemissagedust dokumendis d_j. Olgu t_j erinevate terminite hulk dokumendis d_j ning TPE_j termini pööratud esinemissagedus, mida saab esitada järgneva valemiga [2:195]:

$$TPE_j = \log\left(\frac{t}{t_j}\right) \tag{23}$$

Igale terminile k_i omistatakse vektor $\overrightarrow{k_i}$ nii, et

$$\overrightarrow{k_i} = (w_{i,1},\ w_{i,2}, \ldots ,\ w_{i,N}), \tag{24}$$

kus $w_{i,j}$, on termini ja dokumendi paarile (k_i, d_j) omistatud kaal. Dokumentide kaalusid kasutatakse selleks, et indekseerida terminivektoreid. Neid kaale arvutatakse järgmise valemiga:

$$w_{i,j} = \frac{\left(0.5+0.5\,\frac{e_{i,j}}{max_j(e_{i,j})}\right)\cdot TPE_j}{\sqrt{\sum_{l=1}^{N}\left(0.5+0.5\,\frac{e_{i,l}}{max_k(e_{i,k})}\right)^2\cdot TPE_l^2}}, \tag{25}$$

kus funktsioon $max_j\,(e_{i,j})$ arvutab iga i-nda termini esinemissageduse $e_{i,j}$ maksimaalväärtuse üle dokumentide kollektsiooni. Tegemist on *TE-DPE* kaalude arvutamisele analoogse valemiga, erinevuseks on vaid termini pööratud sageduse arvestamine.

Terminite k_u ja k_v vahelist seost on võimalik arvutada korrelatsiooniteguriga $r_{u,v}$ järgmiselt:

$$r_{u,v} = \overrightarrow{k_u} \bullet \overrightarrow{k_v} = \sum_{\forall d_j} w_{u,j} \cdot w_{v,j}\,. \qquad (26)$$

Sellise korrelatsioonimaatriksi loomine on küll kallis, kuid seda tuleb teha vaid kord ning seejärel on seda võimalik regulaarselt uuendada.

Kasutades sarnasus-tesaurust, on võimalik päringut laiendada järgneva kolme sammuga:

1) Esitada päring, kasutades sama vektorruumi mudelit mis indekseeritud terminitegi jaoks:

$$\vec{q} = \sum_{k_i \in q} w_{i,q}\,\overrightarrow{k_i}\ , \qquad (27)$$

kus $w_{i,q}$ on indekseeritud termini ja päringu paarile $[k_i, q]$ omistatud kaal, mida saab arvutada võrrandist *25*, asendades dokumendi d_j päringuga q [2:196].

2) Sarnasus-tesauruse baasil saab arvutada sarnasus-funktsiooni $\mathrm{sim}(q, k_v)$ iga termini k_i jaoks, mis on korrelatsioonis päringu terminite ja kasutaja päringuga q:

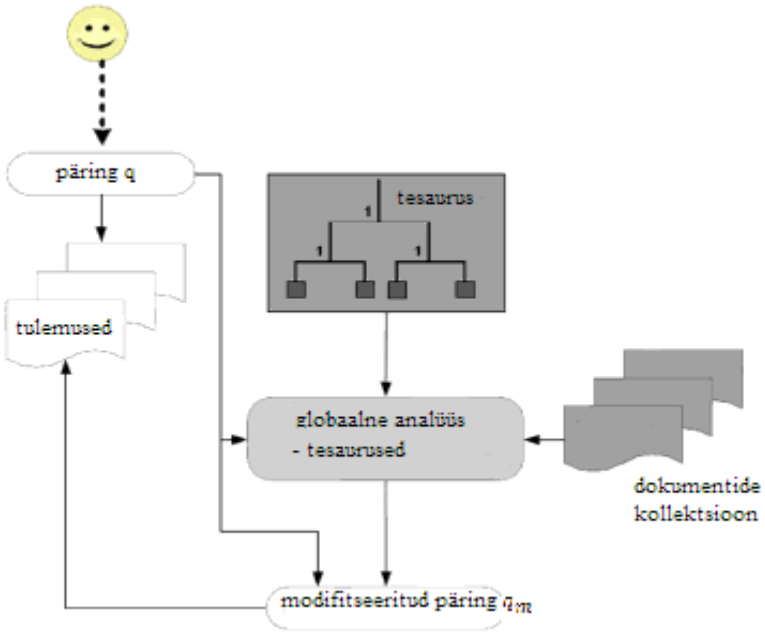
$$\mathrm{sim}(q, k_v) = \vec{q} \bullet \overrightarrow{k_v} = \sum_{k_i \in q} w_{i,q} \cdot r_{i,v}\ , \qquad (28)$$

kus $r_{i,v}$ on korrelatsioonitegur, mis arvutati valemist *26*.

3) Laiendada päringut n järjestuses eespool oleva terminiga, kasutades sarnasusfunktsiooni, mis arvutati sammul 2. Need terminid lisatakse esialgsesse päringusse q laiendatud päringust q_m. Igale laiendatud terminile k_v, mis sisaldus päringus q_m, omistatakse kaal w_{v,q_m} järgmise valemi abil:

$$w_{v,q_m} = \frac{\mathrm{sim}(q, k_v)}{\sum_{k_i \in q} w_{i,q}} \quad . \tag{29}$$

Modifitseeritud päringut q_m (vaata joonis *13*) kasutatakse seejärel uute dokumentide otsimisel [2:197].

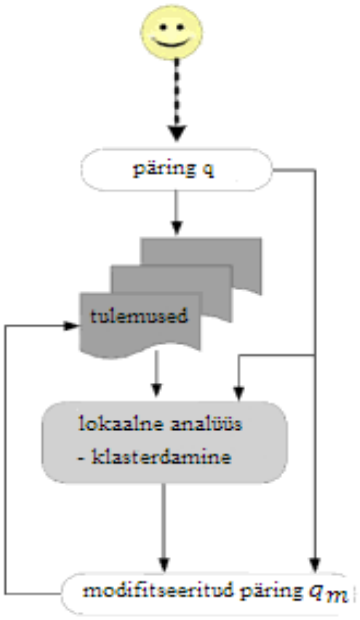


Joonis *13*. Päringu laiendamine globaalsel analüüsil [2:180]. (Autor modifitseeris ja tõlkis eesti keelde.)

4.2.3 Päringu laiendamine lokaalsel analüüsil

Saab kasutada ka lokaalsel analüüsil (*local analysis*) baseeruvat päringu laiendamist. Sel juhul analüüsitakse dokumente tulemite kogust. Globaalse meetodi suurimaks probleemiks on asjaolu, et see ei suuda alati parandada otsingu efektiivsust, kasutades üldiseid kollektsioone (näiteks tesaurusi). Efektiivsuse tõstmiseks kasutatakse lokaalset

klasterdamist. Päringu q jaoks otsitud dokumentide hulka D_l nimetatakse lokaalsete dokumentide kollektsiooniks. Olgu N_l dokumentide hulk sellest kollektsioonist. Tähistagu V_l kõikide erinevate terminite hulka lokaalsete dokumentide kollektsioonis. Ühe termini k_i esinemissagedust dokumendis $d_j \in D_l$ võib käsitleda kui esinemissagedust $e_{i,j}$. Olgu M_l termini ja dokumendi maatriks, mille ridadeks on V_l ning veergudeks N_l. Olgu M_l^T pöörd-maatriks. Siis maatriks $R_l = M_l$ x M_l^T on lokaalne kahe erineva termini vaheline korrelatsioonimaatriks. Iga element $r_{u,v} \in R_l$ tähistab korrelatsiooni terminite k_u ning k_v vahel, mis tagastatud dokumendis esinesid koos vastusena päringule q. Mida rohkemates dokumentides need terminid koos esinesid, seda tugevam on korrelatsioon. Kui korrelats-iooni tugevus on arvutatud, saab seda kasutada klastrite loomisel naabruses olevate terminite vahel, mis on üldjuhul sünonüümid, kuid sõltuvad palju lokaalse päringu kontekstist. Neid termineid, mis esinevad ühes klastris päringu terminitega, saab kasutada päringu laiendamisel.



Joonis *14*. Päringu laiendamine lokaalsel analüüsil [2:180]. (Autor modifitseeris ja tõlkis eesti keelde.)

Modifitseeritud päringut q_m, mis on näidatud eelneval joonisel *14*, kasutatakse seejärel uute dokumentide otsimisel [2:190].

Kokkuvõte

Töös anti kirjanduse põhjal ülevaade infootsingust, sealhulgas loomuliku keele automaattöötluse meetodite kasutamisest infootsingu erinevatel etappidel.

Infootsing toimub järgmiselt. Infootsingu süsteemi esimese ülesandena koostatakse dokumentide kollektsioon, milleks võib olla näiteks veebilehtede kogum. Seejärel teostatakse kollektsioonis olevatele dokumentidele leksikaalne analüüs, stoppsõnade elimineerimine ja lemmatiseerimine. Nende protsesside tulemusena väheneb terminite hulk dokumentides. Järelejäänud terminite alusel dokumendid indekseeritakse. Indekseerimise peamine ülesanne on dokumentide järjestamine selle alusel, kui tihti mingi termin neis esineb.

Kasutaja sisestab otsingusüsteemile päringu, mis väljendab tema teabevajadust. Päringust leitakse võtmesõnad ning nende esinemissageduste alusel on võimalik indekseeritud dokumente otsingutulemusena väljastada. Kasutaja teadmised valdkonnast ei pruugi alati olla piisavad, et konstrueerida sobivat päringut. Selle probleemi lahendamiseks kasutatakse asjakohast tagasisidet, mis tähendab, et kasutaja kaasatakse protsessi, kus otsingusüsteem väljastab tulemusi ja kasutaja hindab, millised dokumendid on sobivad tema informatsiooninõudega ja millised mitte. Päringute formuleerimise probleemiks on veel ka asjaolu, et päringusse sisestatakse tavaliselt 2-3 sõna, mis on liiga vähe, et otsingusüsteem suudaks leida sobivaid dokumente. Päringut laiendatakse automaatselt, kasutades sünonüümisõnastikke ja sõnade vahelisi relatsioone, seejärel koostatakse uus päring ning väljastatakse kasutajale tulemused.

Töö käigus loodi ka eesti keele stoppsõnade näidisloend ja koostati skriptid, mis suudavad teha eestikeelsete sõnade lemmatiseerimist ning terminite esinemissageduste ja termini-dokumendi maatriksi leidmist.

Natural language processing in information retrieval

Summary

Information retrieval is a field of natural language processing, which main task is to search, find and retain relevant text documents to match user's query. Information retrieval system can create effective search using many natural language processing tehniques. This document contains four bigger chapters: the introdution to information retrieval, document pre-processing, indexing terms and documents, tehniques used in formulating queries.

The first chapter gives an overview of information retrieval recent history and information retrieval systems' arhitecture. The second chapter describes the processes made before sending terms and documents to indexing, including lexical analysis, stop words removal and stemming. Lexical analysis identifies words from text. Stop words are the words, that carry a little semantic information and stemming find the words stems. The main purpose of document pre-processing is to reduce the set of words to accelerate indexing. The next bigger process introduced is indexing, where term and document frequencies are involved in weighting schemes. In addition to frequencies, term positions in documents are also considered. The fourth and the longest chapter shows how relevance feedback and query expansion are used in query formulation. In relevance feedback, users are involved to judge if the results are relevant or not and then information retrieval system creates new query based on users' feedback. Results are re-printed to user. Query expansion does not need users activity in query processing. This tehnique uses thesauri and relations between words to expand the query automatically and show final results to user.

The aim of this document is to introduce techniques used in information retrieval and create scripts to illustrate some of these tehniques in Estonian. In the end of the document, there are some additional parts, like terms vocabulary, Estonian stop words list and scripts, that can find term frequency, term-document matrix and process stemming to an Estonian words.

Kirjandus

[1] Arif, A. S. M., Rahman, M., Mukta, S. Y. (2009). *Information Retrieval by Modified Term Weighting Method Using Random Walk Model With Query Term Position Raking*: *International Conference on Signal Processing Systems, lk 526-530*. [Online] CS Digital Library (02.05.2011)

[2] Baeza-Yates, R., Ribeiro-Neto, B. 2011, *Modern Information Retrieval*: Pearson Education Limited 2011.

[3] *Bash* kooriku skript. [WWW] http://kuutorvaja.eenet.ee/programmeerimine/bash/shellscripts.html (24.05.2011)

[4] *Bash*'i skriptimise manuaal. [WWW] http://linuxconfig.org/Bash_scripting_Tutorial (24.05.2011)

[5] Eesti *Wordnet* ehk *TEKsaurus*. [WWW] http://www.cl.ut.ee/ressursid/teksaurus/ (01.05.2011)

[6] Hassan, S., Mihalcea, R., Banea, C. (2007). *Random-Walk Term Weighting for Improved Text Classification: Proceedings of the International Conference on Semantic Computing, 2007. p 242-249.* [Online] ACM Digital Library (01.05.2011)

[7] *Java* mitmetähenduslikkus. [WWW] http://www.thefreedictionary.com/Java (01.05.2011)

[8] Jian-Fu Li, Mao-Zu Guo, Shu-Hong Tian (2005). *A New Approach to Query Expansion*: *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 18-21th August. Guangzhou, China, lk 2302-2306*. [Online] IEEE Xplore (01.05.2011)

[9] Jurafsky, D., Martin, J. 2009, *Speech and Language Processing*: Pearson Education, Inc.

[10] Manning, C. D., Raghavan, P., Schütze, H. 2009. *An Introduction to Information Retrieval*: Cambridge University Press.

[11] Md. Rafiqul Islam, Buddha Dev Sarker, Md. Rakibul Islam (2008). *An Effective Term Weighting Method Using Random Walk Model for Information Retrieval: Proceedings of the International Conference on Computer and Communication Engineering, 13-15th May. Kuala Lumpur, Malaysia, lk 1357-1362.* [Online] IEEE Xplore (01.05.2011)

[12] Programm *t3mesta* OÜ Filosoft'ilt [WWW] http://www.filosoft.ee/ (11.05.2011)

[13] Päring Eesti *Wordnet*'ist ehk *TEKsaurus*'est. [WWW] http://www.cl.ut.ee/ressursid/teksaurus/teksaurus.cgi.et (01.05.2011)

[14] Päringu laiendamine. [WWW] http://homepages.inf.ed.ac.uk/srenals/pubs/1999/esca99-thisl/node6.html (01.05.2011)

[15] *Sed*-käskude näited. [WWW] http://www.grymoire.com/Unix/Sed.html (07.05.2011)

[16] Stoppsõnade näidisloend. [WWW] http://www.lextek.com/manuals/onix/stopwords1.html (10.05.2011)

[17] Zhou, L. (2009). *Using Term Relation in Context Sensitive Information Retrieval*: *Sixth International Conference on Fuzzy Systems and Knowledge Discovery, lk 354-358.* [Online] IEEE Xplore (01.05.2011)

[18] Takama, Y. (2001). *Consideration of Relevance Feedback on Keyword Space for Interactive Information Retrieval*: *Conference on Cybernetics and Intelligent Systems, 1-3rdDecember. Singapore, lk 324-328*. [Online] IEEE Xplore (01.05.2011)

[19] *Unix*'i manuaal. [WWW] http://bama.ua.edu/cgi-bin/man-cgi (07.05.2011)

[20] Yousef, N., Al-Bidewi, I., Fayoumi, M. (2010). *Evaluation of Different Query Expansion Techniques and using Different Similarity Measures in Arabic Document*. in *European Journal of Scientific Research: lk 156-166*. [E-ajakiri] http://www.eurojournals.com/ejsr_43_1_15.pdf (01.05.2011)

Lisad

Lisa 1.

Töös kasutatud mõisted

Definitsioonid on võetud allikatest [2] [6] [10] [18].

- Asjakohane tagasiside (*Relevance feedback*) – tehnika, kus kaasatakse kasutaja hindamaks otsingusüsteemi poolt väljastatud tulemuste sobivust.

- Dokumendi eeltöötlus (*Document pre-processing*) – protsess, mida rakendatakse dokumentides esinevatele terminitele enne dokumentide indekseerimist.

- Dokumendi esinemissagedus (*Document frequency*) – indekseerimisel kasutatav meetod, kus leitakse iga erineva dokumendi esinemissagedus dokumentide kollektsioonis.

- Dokumendi pööratud esinemissagedus (*Inverse document frequency*) – indekseerimisel kasutatav tehnika, mis seob termini esinemissageduse dokumendi pööratud esinemissagedusega ning mille alusel otsustatakse termini tähtsus dokumendis.

- Dokumentide kollektsioon (*Document collection*) – elektrooniline dokumentide kogum, mida infootsingu süsteem kasutab otsingul.

- Globaalne analüüs (*Global analysis*) – päringu laiendamise meetod, kus kasutatakse globaalseid tesaurusi.

- Hüpertekst (*Hypertext*) – elektrooniline tekst, kus viidatakse teistele tekstidele linkidega, mille aktiveerimisel saab kasutaja juurdepääsu viidatavale tekstile.

- Infootsing (*Information retrieval*) – loomuliku keele töötluse valdkond, mille ülesandeks on informatsiooni otsimine ja säilitamine.

- Infootsingu süsteem/otsingusüsteem/otsimootor/otsingumootor (*Information retrieval system*) – süsteem, mis tegeleb informatsiooni otsimise ja väljastamisega.

- Juhusliku jalutuskäigu mudel (*Random walk model*) – mudel, kus kujutletav „jalutaja“ astub dokumendis juhuslikus järjekorras samme terminilt terminile.

- Kahendmudel (*Boolean model*) – mudel, milles dokumendid on terminite kogumid (termin kas sisaldub või mitte dokumendis) ja päringud on kahendmudeli väljendused (operaatorid „ja“, „või“ ja „ei“) terminite vahel.

- Kaugemalt-lähemale võtmesõnade paar (*Far2Near keyword pair*) – võtmesõnade paar, mida kasutaja arvates tuleb teineteisele lähendada.

- Kotitäis sõnu mudel (*Bag of words model*) – mudel, mille puhul tekst on esitatud terminite kogumina, kusjuures terminite vahelised seosed ega ka nende järjekord dokumendis pole olulised.

- Lemmatiseerimine (*Stemming*) – sõnade algvormide ehk lemmade leidmine.

- Lokaalne analüüs (*Local analysis*) – päringu laiendamisel kasutatav meetod, kus analüüsitakse dokumente otsingusüsteemi poolt väljastatud tulemite kogust.

- Loomuliku keele töötlus (*Natural language processing*) – arvutiteaduse valdkond, mis tegeleb loomulike keele töötlemisega arvutis.

- Lähemalt-kaugemale võtmesõnade paar (*Near2Far keyword pair*) – võtmesõnade paar, mida kasutaja arvates tuleb teineteisest eraldada.

- Metaandmed (*Metadata*) – struktuurne informatsioon ehk andmed dokumentides olevate andmete kohta, mille alusel saab dokumente klassifitseerida.

- Optimaalne päring (*Optimal query*) – vektor, mis eraldab sobivad dokumendid mittesobivatest.

- Parameetriline indeks (*Parametric index*) – indeks, mis määratakse igale samalaadsele objektile dokumendis. Nendeks objektideks võivad olla pealkirjad, autorid, kuupäevad, jne.

- Päringu laiendamine (*Query expansion*) – tehnika, kus muudetakse kasutaja esialgset päringut, lisades juurde termineid, mis on lähedalt seotud päringus olevatega.

- Sarnasus-tesaurus (*Similarity thesaurus*) – tesaurus, mille puhul arvutatakse sarnasus iga kahe termini vahel, mis dokumentide kollektsioonis esinevad.

- Skaala konstant (*Scaling constant*) – konstant, mida kasutatakse juhusliku jalutuskäigu mudelis.

- Stoppsõnad (*Stop words*) – sõnad, mis kannavad vähe semantilist informatsiooni.

- Sumbuvustegur (*Damping factor*) – arv, mis hindab graafis omavahel servaga ühendatud tippude koosesinemise sagedust.

- Termini esinemissagedus (*Term frequency*) – indekseerimisel kasutatav meetod, kus leitakse iga erineva termini esinemissagedus dokumendis.

- Termini esinemissagedus-dokumendi pööratud esinemissagedusel (*Term frequency–inverse document frequency*) – indekseerimisel kasutatav meetod, mille alusel omistatakse terminitele kaalud, mis tähistavad nende tähtsust dokumentide kollektsioonis.

- Termini-dokumendi maatriks (*Term-document matrix*) – maatriks, mille ridadeks on terminite esinemissagedused ja veergudeks dokumendid, kus need terminid esinevad.

- Tesaurus (*Thesaurus*) – sarnaste sõnade (näiteks sünonüümide) ja nende vaheliste relatsioonide sõnastik.

- Veeb (*Web, World Wide Web*) - elektrooniliste tekstide, piltide, videote ja heliklippide kogumik koos nende vahele paigutatud hüpertekstilinkidega.

- Veebiämblik (*Web crawler*) – arvutiprogramm, mis otsib informatsiooni (dokumente) *World Wide Web*'ist automaatselt.

- Vektorruumi mudel (*Vector space model*) – mudel, kus dokument ja päring viiakse vektorkujule ning võtmesõnadele omistatakse kaalud.

- Võtmesõna (*Keyword*) – päringus olev termin, mille alusel otsingusüsteem alustab otsingu koostamist.

- Võtmesõnade kaart (*Keyword map*) – visuaalne graaf, mis genereeritakse otsingusüsteemi poolt otsingutulemuste põhjal, ning kus tähistatakse võtmesõnu ja nende vahelisi relatsioone.

Lisa 2.

Stoppsõnade list

Stoppsõnade näidisloend `stoppsõnad.txt` eesti keele sõnadest [16].

a, abil, aegu, aga, ainult, alalt, alates, alati, all, alla, alles, alt, asemel, b, c, d, e, eales, ealeski, edasi, eelmine, eemal, ees, eest, ega, ehk, ehkki, ei, emb-kumb, enam, end, enda, endine, enese, enim, enne, ennem, ent, esimene, esimest, et, etem, ette, f, g, h, halb, halvasti, halvim, hea, hiljem, hoopis, hulga, hulgas, hulk, hästi, i, iga, igal, igamees, iganes, igavesti, igaüks, iial, iialgi, ikka, ilma, ilmaski, ise, iseenda, iseenese, isegi, j, ja, jooksul, ju, juba, juhul, just, justkui, juures, järelikult, järgi, järgmine, k, ka, kahe, kaheksa, kaheksandat, kaheksat, kahte, kaks, kallal, kaudselt, kaudu, kaugel, keda, keegi, kehva, kehvem, kelle, kellega, kelleks, kellel, kellele, kellelt, kellena, kelleni, kelles, kellesse, kellest, kelleta, kes, kestahes, kestel, kogu, kohta, kokku, kolm, kolmandat, kolmas, kolme, koos, kord, kuhu, kui, kuid, kuidagi, kuidas, kuigi, kuivõrd, kumb, kumbki, kuna, kuni, kus, kust, kuue, kuuendat, kuus, kuute, kõige, kõik, kõrval, käes, küll, kümme, kümmet, kümne, kümnendat, küüsis, l, ligi, ligidal, lisaks, läbi, lähedal, m, ma, me, meelest, meid, meie, meiega, meieks, meiena, meieni, meieta, meil, meile, meilt, meis, meisse, meist, mida, midagi, mihuke, mihukene, mil, millalgi, mille, millega, milleks, millel, millele, millelt, millena, milleni, milles, millesse, millest, milleta, milline, mina, mind, mingi, mingisugune, minu, minuga, minuks, minul, minule, minult, minuna, minuni, minus, minusse, minust, minuta, mis, miski, miskisugune, mispärast, missugune, mistahes, misuke, mitmendik, mitmes, mitte, mitu, mu, mulle, mult, muu, muudkui, mõlema, mõlemad, mõnda, mõne, mõnelt, mõnes, mõni, mõningane, määrane, mööda, n, naasugune, nad, nagu, najal, natuke, natukese, natukeseks, need, neid, neiks, neil, neile, neilt, neis, neisse, neist, neli, nelja, neljandat, nemad, nende, nendega, nendeks, nendele, nendelt, nendena, nendeni, nendes, nendesse, nendest, nendeta, nendel, nigu, nihuke, nii, niikaua, niisamasugune, niisugune, ning, nõnda, nüüd, o, ole, oleks, olgu, olgugi, oli, olla, olnud, oma, on, osa, otsa, otsekui, p, paar, paarkümmend, paarsada, palju, parem, paremal, parim, peaks, peal, peale, peamiselt, pigem, pihta, pisut, pole, poleks, poole, poolest, praegu, praegune, puhul, päralt, pärast, päris, r, ringis, rohkem, s, sa, saa, saadik, saatel, sama, samasugune, samuti, seal, seda, see, seega, sees, seesama, seesamane, seesamune, seesugune, seitse, seitset, seitsme,

seitsmendat, sel, selle, sellepärast, selletaoline, selline, seltsi, seni, sest, sestap, sihuke, sihukene, siin, siis, siiski, sina, sind, sinu, sinuga, sinuks, sinule, sinult, sinuna, sinuni, sinus, sinusse, sinust, sinuta, sisse, siuke, siukene, suht, suhtes, sulle, sult, suur, säherdune, säärane, š, z, ž, t, ta, taga, tagant, tagasi, taha, tal, talle, talt, taoline, te, tea, teda, teel, teid, teie, teiega, teiena, teieni, teieta, teiks, teil, teile, teilt, teine, teineteise, teis, teisse, teist, teistsugune, tema, temaga, temaks, temal, temale, temalt, temana, temani, temas, temasse, temast, temata, terve, toda, toetub, toetudes, tohutu, tol, tolle, too, toosama, toosamane, tugineb, tuginedes, tõhusam, tõttu, täiesti, tänu, u, umbes, uue, uuega, uuelt, uuema, uuemaga, uuemal, uuemale, uuemalt, uuemana, uuemas, uuemasse, uuemast, uuena, uues, uuest, uus, v, w, vahel, vaid, valla, vana, vanaga, vanalt, vanana, vanas, vanast, vanema, vanemaga, vanemal, vanemale, vanemalt, vanemana, vanemas, vanemasse, vanemast, varal, varem, vasak, vasakul, vastu, veel, veidi, viie, viiendat, viis, viisi, viite, või, võib, võib-olla, võidu, võiks, võrra, väel, väga, vähe, vähem, vähemalt, väike, välja, väljas, vältel, õ, ä, ära, äärde, ääres, ö, ü, ühe, üheksa, üheksandat, üheksat, ühte, üks, üksi, ükski, üksteist, ülal, ülale, ülalt, üle, üles, ülesse, üleval, ülevalt, ülimalt, ümber, x, y

Lisa 3.

Terminite esinemissageduste loend

Skript `TE.sh` koostamaks terminite esinemissageduste loendit dokumendist (kasutab ka stoppsõnade loendit `stoppsõnad.txt` lisast *2*) [3] [4] [15] [19].

Terminite esinemissageduste loendi leidmine:

1. Vajalik teksti sisaldava faili olemasolu (näiteks ajalehest `artikkel1.txt`) ning stoppsõnade loend (näiteks `stoppsõnad.txt`).

2. Skripti käivitamine ning terminite esinemissageduste loendi kuvamine ekraanile: **`bash TE.sh artikkel1.txt | uniq -c | sort -nr | nl | less`**

```
#!/bin/bash

export setenv LANG=et_EE.UTF-8 # sümbolid utf-8 kooditabelisse

fail="$1" # fail loetakse esimese parameetrina konsoolilt

# kontrollib kas parameeter on võrdne nulliga ning vajadusel annab
veateate ja väljub
[ $# -eq 0 ] && { echo -e "\033[31mVeateade:\033[0m" fail puudub."
Sisestada kujul:""\033[32m bash $0 failinimi.txt\033[0m"; exit 1;}

# kontrollib kas fail eksisteerib ning vajadusel annab veateate ja
väljub
[ ! -f "$fail" ] && { echo -e "\033[31mVeateade:\033[0m" faili $1
ei leitud." Sisestada kujul:""\033[32m bash $0
failinimi.txt\033[0m"; exit 2;}

if [ -s "$fail" ] # kontrollib kas fail sisaldab andmeid

   then # fail sisaldab andmeid

     cat $fail |
```

```
            # kustutab html-märgendid, kirjavahemärgid ning
            erisümbolid
            sed -e 's/<[^>]*>//g' | tr -d '[:punct:]' |

            # teisendab k6ik tühikud reavahetusteks (e. koostab
            sõnade loendi eeldusel, et sõna ees ning järel on
            tühik)
             tr ' ' '\n' |

            # muudab faili k6ik suured tähed väikesteks sh. ka
            täpitähed
            tr '[:upper:]' '[:lower:]' | tr 'Š' 'š' | tr 'Ž' 'ž' |
            tr 'Ö' 'ö' | tr 'Ä' 'ä' | tr 'Ü' 'ü' | tr 'Õ' 'õ' |

            # eemaldab loendist tühjad read, numbrid ning
            erisümbolid
            sed 's/^[ \t]*$//g' | sed 's/[\!!\”\…\§\·\;\:\%\0-9]//g'
            | sed '/^$/d' | grep -v '[0-9]' |

            # elimineerib stop sõnad loendist
            grep -v -w -f stoppsõnad.txt |

                 # sorteerib tähestiku järjekorras
             sort | less

   else # fail ei sisalda andmeid

            echo -e "\033[31mVeateade:\033[0m" fail "\033[32m$fail
            \033[0m" on tühi või pole tekstifail.

fi
```

Lisa 4.

Sõnade lemmad

Skript `Lemmad.sh` koostamaks tegu- ja nimisõnade lemmasid ehk algvorme [3] [4] [12] [15] [19].

Algvormide leidmine:

1. Vajalik teksti sisaldava faili olemasolu (näiteks ajalehest `artikkel1.txt`) ning stoppsõnade loend (näiteks `stoppsõnad.txt`).

2. Skripti käivitamine ning tegu- ja nimisõnade lemmade loendi kuvamine ekraanile: **`bash Lemmad.sh artikkel1.txt | sort | uniq | sort | less`**

```
#!/bin/bash

export setenv LANG=et_EE.UTF-8 # sümbolid utf-8 kooditabelisse

fail="$1" # fail loetakse esimese parameetrina konsoolilt

# kontrollib kas parameeter on võrdne nulliga ning vajadusel annab
veateate ja väljub
[ $# -eq 0 ] && { echo -e "\033[31mVeateade:\033[0m" fail puudub."
Sisestada kujul:""\033[32m bash $0 failinimi.txt\033[0m"; exit 1;}

# kontrollib kas fail eksisteerib ning vajadusel annab veateate ja
väljub
[ ! -f "$fail" ] && { echo -e "\033[31mVeateade:\033[0m" faili $1
ei leitud." Sisestada kujul:""\033[32m bash $0
failinimi.txt\033[0m"; exit 2;}

if [ -s "$fail" ] # kontrollib kas fail sisaldab andmeid

   then # fail sisaldab andmeid

     cat $fail |

          # koostab morfoloogilise analüüsi terminite loendile
```

```
      t3mesta -MO -cin utf8 -cout utf8 +1 > morf_analüüs.txt

cat morf_analüüs.txt |

      # kustutab erisümbolid ning numbrid
      sed 's/[\!!\”\…\§\·\;\:\%\0-9]//g' > morf_analüüs2.txt

cat morf_analüüs2.txt |

      # leiab algvormid sõnadest, mille esimeses analüüsis on
      nimisõna märgend
      sed -e 's/<[^>]*>//g'| sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep '_S_' |
      grep '=' |
      sed 's%^\([^/]*//[^/]*//\).*$%\1%g' |
      sed 's/^[^ ]*    \([^ =]*\)=\([^ +]*\)+.*$/\1\2/' |
      sed '/^$/d' | sort > subst_lemmad1.txt

cat subst_lemmad1.txt > lemmad.txt

cat morf_analüüs2.txt |

      # leiab algvormid sõnadest, mille esimeses analüüsis on
      tegusõna märgend
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep '_S_' |
      grep -v '=' |
      sed 's%^\([^/]*//[^/]*//\).*$%\1%g' |
      sed 's/^[^ ]*    \([^ +]*\)+[^ /]*.*$/\1/' |
      sed '/^$/d' | sort > subst_lemmad2.txt

cat subst_lemmad2.txt >> lemmad.txt

cat morf_analüüs2.txt |

      # leiab algvormid sõnadest, mille esimeses analüüsis on
      tegusõna märgend
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' |
      sed 's%^\([^/]*//[^/]*//\).*$%\1%g' | grep '_S_' |
      sed 's/^[^ ]*    \([^ +]*\)+[^ /]*.*$/\1/' |
      sed '/^$/d' | sort > subst_lemmad3.txt

cat subst_lemmad3.txt >> lemmad.txt

cat morf_analüüs2.txt |
```

```
      # leiab algvormid sõnadest, mille esimeses analüüsis on
      tegusõna märgend
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep -v '_S_' | grep -v '_Z_' | grep -v
      '_V_' |
      sed 's%^\([^/]*//[^/]*//\).*$%\1%g' |
      sed 's/^[^ ]*    \([^ +]*\)+[^ /]*.*$/\1/' |
      sed '/^$/d' | sort > teised_lemmad.txt

cat teised_lemmad.txt >> lemmad.txt

cat morf_analüüs2.txt |

      # leiab tegusõnad ning muudab nende algvormid mine-
      lõpulisteks
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep '_V_' |
      sed 's%^\([^/]*//[^/]*//\).*$%\1%g' |
      sed 's/^[Ee]i    \(ei\)+.*$/\1/' |
      sed 's/^[^ ]*    \([^ +]*\)+[^ +]* \/\/_V_.*$/\1mine/'
|
      sed 's/^[^ ]*    \([^ +]*\)+.*$/\1/' |
      sed 's/ei//g' | sed '/^$/d' | grep -v 'ldmine' |
      grep -v '=nud' | grep -v '=tud' | grep -v '=dud' | sort
      > verb_lemmad.txt

cat verb_lemmad.txt >> lemmad.txt

cat morf_analüüs2.txt |

      # leiab nud-lõpulised sõnad ning nende algvormid
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep '+nud ' |
      sed 's/^[^ ]*    [^/]*\/\/_A_ \/\/
      [^/]*\/\/[^/]*\/\/    [^/]*\/\/[^/]*\/\/    \([^
      +]*\)+[^ +]* \/\/[^/]*\/\//\1mine/' |
      sed 's/^[^ ]*    \([^ +]*\)+[^ +]* \/\/_V_ nud,
      \/\/.*$/\1mine/' |
      sed 's/^[^ ]*    \([^ +]*\)+.*$/\1/' |
      sed '/^$/d' | sed '/^ole$/d' | sort > nud_lemmad.txt

cat nud_lemmad.txt >> lemmad.txt

cat morf_analüüs2.txt |

      # leiab dud-lõpulised sõnad ning nende algvormid
      sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
      grep -v '&' | grep '+dud ' |
```

```
        sed 's/^[^ ]*    [^/]*\/\/_A_ \/\/
        [^/]*\/\/[^/]*\/\/    [^/]*\/\/[^/]*\/\/    \([^
        +]*\)+[^ +]* \/\/[^/]*\/\//\1mine/' |
        sed 's/^[^ ]*    \([^ +]*\)+[^ +]* \/\/_V_ tud,
        \/\/.*$/\1mine/' |
        sed 's/^[^ ]*    \([^ +]*\)+.*$/\1/' |
        sed '/^$/d' | sed '/^ole$/d' | sort > dud_lemmad.txt

    cat dud_lemmad.txt >> lemmad.txt

    cat morf_analüüs2.txt |

        # leiab tud-lõpulised sõnad ning nende algvormid
        sed -e 's/<[^>]*>//g' | sed '/_Z_/d' | grep -v '?' |
        grep -v '&' | grep '+tud ' |
        sed 's/^[^ ]*    [^/]*\/\/_A_ \/\/
        [^/]*\/\/[^/]*\/\/    [^/]*\/\/[^/]*\/\/    \([^
        +]*\)+[^ +]* \/\/[^/]*\/\//\1mine/' |
        sed 's/^[^ ]*    \([^ +]*\)+[^ +]* \/\/_V_ tud,
        \/\/.*$/\1mine/' |
        sed 's/^[^ ]*    \([^ +]*\)+.*$/\1/' |
        sed '/^$/d' | sed '/^ole$/d' | sort > tud_lemmad.txt

    cat tud_lemmad.txt >> lemmad.txt

    cat lemmad.txt |

        # kontrollib sõnade lemmasid enne lõpptulemuse
        kuvamist, muudab läbivalt väiketähelisteks ning kuvab
        loendina
        sed 's/^[^ ]*    \([^ +]*\)+[^ +]* \/\/_._.*$/\1mine/'
    |
        sed 's/^\([^=]*\)=\([^ ]*\) $/\1\2/' | sed
        's/^[\.\_\/\-].*//g' | grep -v '<' | grep -v '>' | grep
        -v '=' | grep -v '"' |
        tr '[:upper:]' '[:lower:]' | tr 'Š' 'š' | tr 'Ž' 'ž' |
        tr 'Ö' 'ö' | tr 'Ä' 'ä' | tr 'Ü' 'ü' | tr 'Õ' 'õ' | sed
        's/^\([^ ]* [^ ]*\) .*$/\1/' | grep -v -w -f
        stoppsõnad.txt |sed 's/^[ \t]*$//g' | sed '/^$/d' |
        less

else # fail ei sisalda andmeid

        echo -e "\033[31mVeateade:\033[0m" fail "\033[32m$fail
        \033[0m" on tühi või pole tekstifail.

fi
```

```
rm morf_analüüs.txt
rm morf_analüüs2.txt
rm subst_lemmad1.txt
rm subst_lemmad2.txt
rm subst_lemmad3.txt
rm teised_lemmad.txt
rm verb_lemmad.txt
rm nud_lemmad.txt
rm dud_lemmad.txt
rm tud_lemmad.txt
```

Lisa 5.

Termini-dokumendi maatriks

Skript `TD_maatriks.sh` koostamaks termini-dokumendi maatriksit kolmest teksti sisaldavast dokumendist (kasutab ka skripti `TE.sh` lisast *3*) [3] [4] [15] [19].

1. Vajalik kolm teksti sisaldavat faili (näiteks `artikkel1.txt artikkel2.txt artikkel3.txt`) ning stop sõnade loend (näiteks `stoppsõnad.txt`).

2. Skripti käivitamine: **`bash TD_maatriks.sh artikkel1.txt artikkel2.txt artikkel3.txt`**

3. Termini-dokumendi maatriksi kuvamine ekraanile: **`cat termin-dokument_maatriks.txt | tr ' ' '\t' | less`**

```
#!/bin/bash

export setenv LANG=et_EE.UTF-8 # sümbolid utf-8 kooditabelisse

fail="$1" fail="$2" fail="$3" # failid loetakse kolmanda
parameetrina konsoolilt

# kontrollib kas parameetrid on võrdsed nulliga ning vajadusel
annab veateate ja väljub
[ $# -eq 0 ] && { echo -e "\033[31mVeateade:\033[0m" failid
puuduvad." Sisestada kujul:""\033[32m bash $0 fail1.txt fail2.txt
fail3.txt\033[0m"; exit 1;}

# kontrollib kas failid eksisteerivad ning vajadusel annab
veateate ja väljub
[ ! -f "$fail" ] && { echo -e "\033[31mVeateade:\033[0m" faile ei
leitud." Sisestada kujul:""\033[32m bash $0 fail1.txt fail2.txt
fail3.txt\033[0m"; exit 2;}

if [ -s "$1" ] && [ -s "$2" ] && [ -s "$3" ] # kontrollib kas fail
sisaldab andmeid
```

```
then # fail sisaldab andmeid

    # kirjutab failides esinevad erinevad terminid ühte faili
    bash TE.sh $1 | uniq | sort > terminid.txt
    bash TE.sh $2 | uniq | sort >> terminid.txt
    bash TE.sh $3 | uniq | sort >> terminid.txt

    cat terminid.txt |

        # sorteerib ja elimineerib kordused
        sort | uniq | sort > terminid_sort.txt

    # koostab failidest terminite loendid
    bash TE.sh $1 | sort > te_loend1.txt
    bash TE.sh $2 | sort > te_loend2.txt
    bash TE.sh $3 | sort > te_loend3.txt

    cat te_loend1.txt |

        # kustutab terminite loendist liigsed tühikud ning
        leiab terminite esinemissageduste loendi
        sed 's/^[^0-9]*\([0-9].*\)$/\1/' | sort | uniq -c  >
        te_sagedusloend1.txt

    cat te_loend2.txt |

        # kustutab terminite loendist liigsed tühikud ning
        leiab terminite esinemissageduste loendi
        sed 's/^[^0-9]*\([0-9].*\)$/\1/' | sort | uniq -c  >
        te_sagedusloend2.txt

    cat te_loend3.txt |

        # kustutab terminite loendist liigsed tühikud ning
        leiab terminite esinemissageduste loendi
        sed 's/^[^0-9]*\([0-9].*\)$/\1/' | sort | uniq -c  >
        te_sagedusloend3.txt

  # ühildab terminite loendi terminite esinemissageduste
  loenditega ning koostab termini-dokumendi maatriksi
  join -1 1 -2 2 -a1 -a2 -e 0 -o 1.1 2.1 terminid_sort.txt
  te_sagedusloend1.txt > maatriks1.txt
  join -1 1 -2 2 -a1 -a2 -e 0 -o 1.1 1.2 2.1 maatriks1.txt
  te_sagedusloend2.txt > maatriks2.txt
  join -1 1 -2 2 -a1 -a2 -e 0 -o 1.1 1.2 1.3 2.1 maatriks2.txt
  te_sagedusloend3.txt > termin-dokument_maatriks.txt

else # fail ei sisalda andmeid
```

```
        echo -e "\033[31mVeateade:\033[0m" fail "\033[32m$1\033[0m"
        või "\033[32m$2\033[0m" või "\033[32m$3\033[0m" on tühi või
        pole tekstifail.

fi

# kustutab termini-dokumendi maatriksi loomisel kasutatavad
ajutised failid
rm terminid.txt
rm te_loend1.txt
rm te_loend2.txt
rm te_loend3.txt
rm te_sagedusloend1.txt
rm te_sagedusloend2.txt
rm te_sagedusloend3.txt
rm terminid_sort.txt
rm maatriks1.txt
rm maatriks2.txt
```