T A R T U Ü L I K O O L
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Riivo Kikas

Veebisessioonide klasterdamine kasutaja käitumise alusel

Bakalaureusetöö (4 AP)

Juhendaja: Konstantin Tretjakov

Autor: ..... "....." mai 2009

Juhendaja: ..... "....." mai 2009

Lubada kaitsmisele

Professor: .... "....." 2009

TARTU 2009

Sisukord

Sissejuhatus **4**

1 Taustinfo **6**
1.1 Markovi ahel . . . 6
1.2 Klasterdamine . . . 8
1.2.1 K-keskmiste algoritm . . . 9
1.2.2 Klastrite valideerimine . . . 10
1.3 Peakomponentide analüüs . . . 12

2 Sessioonigruppide leidmine **14**
2.1 Probleemi püstitus . . . 14
2.1.1 Kasutaja- ja sessioonigruppide erinevused . . . 14
2.2 Lahendus . . . 15
2.2.1 Andmete esitus . . . 15
2.2.2 Tegevuste üldistamine . . . 15
2.2.3 Klasterdamine . . . 16
2.2.4 Tulemuste analüüs . . . 17
2.3 Olemasolevad lahendused . . . 18
2.3.1 Sessiooni esitus sõnekujul . . . 18
2.3.2 Sessiooni esitus vektorkujul . . . 19
2.3.3 Analoogsed lahendused teistes valdkondades . . . 20
2.4 Kokkuvõte . . . 20

3 Algoritm sessioongruppide leidmiseks **22**
3.1 Markovi mudelitel baseeruv meetod . . . 22
3.1.1 Sessiooni esitus Markovi ahela mudelina . . . 22
3.1.2 Kaugusmõõt . . . 23
3.1.3 Klasterdamine . . . 23
3.1.4 Klastrite arvu hindamine . . . 24
3.1.5 Näide . . . 24
3.2 Meetodi hindamine tehisandmetel . . . 25
3.2.1 Tehisveebisessioonide kogu . . . 25
3.2.2 Tulemused . . . 27
3.2.3 Alternatiivid . . . 29

4 Praktiline näide **31**
4.1 Tööjõujälgimise tarkvara sessioonigruppide analüüs . . . 31
4.2 Protsess . . . 31
4.3 Eeltöötlus ja statistika . . . 32

4.3.1 Tegevuste üldsitamine 32
4.4 Meetod klastrite iseloomustamiseks 32
4.5 Klasteradamine üle kõikide rollide 33
4.5.1 Peakomponentide analüüs 35
4.6 Klastrid ühe rolli sees 35
4.6.1 Klastrite iseloomulikud jooned 38
4.6.2 Sessioonide visuaalne inspekteerimine 38
4.7 Järeldused 39

Kokkuvõte **41**

Abstract (in english) **42**

Kirjandus **43**

Lisad **46**

Sissejuhatus

Viimasel ajal on järjest enam hakanud levima tarkvarateenuse osutamise mudel, mille korral klient kasutab tarkvara läbi internetilehitseja, seejuures tarkvara on installeeritud ainult teenusepakkuja serverites. Sellise lähenemise puhul ei pruugi teenusepakkuja kunagi kohtuda kliendiga, ning seetõttu väheneb ka võimalus saada reaalset tagasisidet tarkvara kohta. Eelkõige võib teenuseosutajat huvitada, missugune funktsionaalsus huvitab inimesi kõige enam, või kas kliendid kasutavad kõiki tarkvara poolt pakutavaid võimalusi nii, nagu arendajad seda planeerisid. Tarbija huvide mõistmiseks saab uurida kasutajate käitumist. Üks võimalik lahendus selleks on leida kasutajate tüüpprofiile, grupeerides selleks sarnaseid külastussessioone ning leida iga grupi iseloomulikke omadusi. Kasutajaprofiilide leidmiseks vajalik kasutajakäitumise ajalugu on salvestatud veebirakenduse logifailidesse. Veebiserverid salvestavad iga kasutaja tegevuse ning seeria kasutaja poolt sooritatud järjestikust tegevustest moodustab kasutajasessiooni.

Info sarnaste sessioonigruppide kohta võimaldab tarkvara haldajatel reorganiseerida funktsionaalsust nii, et suuremate gruppide poolt kasutatav funktsionaalsus oleks lihtsamini leitav. Lisaks annab see sisendi äriotsuste tegemiseks, võimaldades koostada pakutavast teenusest erinevad paketid vastavalt kasutajate harjumusele tarbitava funktsionaalsuse osas. Info kasutajaprofiilide kohta on ka esimeseks sammuks kohanduvate veebilehekülgede loomisel, mille abil saab kuvada kasutaja grupikuuluvuse põhjal tema käitumisest sõltuvat sisu, abiteavet või viiteid teistele lehekülgedele, mis võiksid kasutajat huvitada.

Töö eesmärgiks on uurida veebisessioonide grupeerimise võimalusi ja olemasolevaid lahendusi. Töö on ajendatud AS Regio vajadusest analüüsida nende poolt tarnitava veebipõhise tarkvara kasutajate käitumist. Uurimuse teostamise ajal tutvus autor paljude teadaolevate lahendustega ning jõudis veendumusele, et praktikas eksisteerib juba probleemile häid lahendusi. Autori panuseks on ülevaade olemasolevatest lahendustest, eelkõige Markovi ahelate mudelil ning k-keskmiste algoritmil baseeruva meetodi analüüs, eksperimentaalne hindamine ning praktiline rakendamine reaalsete logifailide analüüsil. Lisaks koostati meetod tehisveebisessioonide kogu loomiseks, mis sisaldab endas sarnase kasutajakäitumisega sessioonigruppe.

Töö koosneb neljast peatükist:

Esimene peatükk annab ülevaate mõistetest ja algoritmidest, mida kasutatakse töös sessioongruppide leidmiseks.

Teine peatükk tutvustab sessioonigruppide leidmise probleemi, kirjeldab lahenduses olevaid etappe ning annab ülevaate olemasolevatest lahendustest. Samuti tutvustatakse

võimalusi sessioonigruppide visuaalseks esituseks ja interpreteerimiseks.

Kolmandas peatükis tutvume lähemalt Markovi ahelate mudelitel baseeruva meetodiga ning hindame meetodi tööd eksperimentaalselt selleks kooostatud tehisandmestikul. Lisaks esitame võrdluse teiste meetoditega.

Neljandas peatükis rakendame uuritud meetodit ning analüüsime veebipõhise tööjõujälgimistarkvara kasutajate käitumist.

Peatükk 1

Taustinfo

Selles peatükis anname ülevaate töös vajaminevast terminoloogiast ja algoritmidest. Veebisessioonide modelleerimiseks kasutatame Markovi ahelaid ning nende grupeerimiseks k-keskmiste algoritmi. Lisaks tutvustatame selles peatükis lähemalt klasterdamist ning meetodeid selle hindamiseks: puhtust (siin töös kasutame ka sõna *täpsus* refeerimaks klasterduse puhtust) ja silueti koefitsienti. Veebisessioonide visuaalseks esituseks rakendame peakomponentide analüüsi Markovi mudelite üleminekumaatriksitel ning projitseerime tasandile kaks tähtsamat peakomponenti.

1.1 Markovi ahel

Olgu $X = (X_1, X_2, X_3, \ldots, X_T)$ juhuslike muutujate jada, mille väärtused on lõplikust hulgast S, mida nimetatakse olekuruumiks. Jada X nimetatakse Markovi ahelaks (ingl. k. *Markov chain*), kui tal on Markovi omadus:

$$\Pr(X_{t+1} = x | X_t = x_t, \ldots, X_1 = x_1) = \Pr(X_{t+1} = x | X_t = x_t).$$

Lahtiseletatuna tähendab Markovi omadus, et fikseeritud oleviku korral tulevik ei sõltu minevikust. Markovi omaduse kehtimise korral sisaldab olevikuseisundi kirjeldus kogu informatsiooni, mis võib tulevast arengut mõjutada ehk ahel võimaldab järgneva seisundi tuletamist eelnevate üleminekutõenäosuste kaudu[Bre01].

Markovi ahelat saab kirjeldada kasutades üleminekumaatriksit $P = p_{ij}$:

$$p_{ij} = \Pr(X_{t+1} = X_j | X_t = X_i)$$

kus p_{ij} väärtus tähistab tõenäosust ülemineku toimumisest olekust X_i olekusse X_j. Maatriksil on järgnevad omadused [DEKM99]:

$$\begin{aligned} p_{ij} &\geqslant 0, \forall i, j, \\ \sum_{j=1}^{T} p_{ij} &= 1, \forall i. \end{aligned}$$

Mudeli koostamine andmetest

Markovi ahela mudeleid saab kasutada järjestikuse tegevuse modellerimiseks. Mudeli parameetreid saab hinnata andemetest. Antud juhul oleks andmeteks Markovi omadust omavad jadad ning mudeli parmeetriteks üleminekute tõenäosused ühest olekust teise. Hindamiseks saab kasutada suurima tõepära meetodit, mille korral hinnatakse olekust X_i võimalikke üleminekuid teistesse olekutesse, kus tõenäosus olekust X_i üleminekuks olekusse X_j avaldub järgnevalt

$$\hat{p}_{ij} = \frac{\text{üleminekute arv olekust } X_i \text{ olekusse } X_j}{\text{olekust } X_i \text{ algavate kõikide üleminekute arv}}$$

Suurima tõepära hinnangu eeliseks on tema lihtsus, kuid probleemiks on tundlikkus puuduvatele andmetele. Oletame, et puuduvad andmed võimalike üleminekute kohta, mis algavad olekust X_i, siis suurima tõepära hinnangu puhul saavad kõik antud olekust algavad üleminekud tõenäosuseks 0. Praktikas ei ole selline lähenemine alati hea ning me võime oletada andmete kohta mingit eelnevat infot ja seda rakendada hindamises. Sellist hinnangut nimetatakse Bayesi hinnanguks:

$$\hat{p}_{ij} = \frac{(\text{üleminekute arv olekust } X_i \text{ olekusse } X_j) + prior_{ij}}{(\text{olekust } X_i \text{ algavate kõikide üleminekute arv}) + prior_i},$$

kus $prior_{ij}$ ja $prior_i$ on konstandid, mis määravad hinnangu väärtuse, kui andmed üldse puuduksid. Antud juhul on selge, et mida rohkem on andmeid, seda väiksem on konstantide $prior_{ij}$ ja $prior_i$ kaal lõpphinnangusse [DEKM99]. Konstandid peavad olema valitud nii, et on rahuldatud tingimus:

$$\sum_{j=1}^{T} prior_{ij} = prior_i,$$

selleks, et andmete puudumise korral oleks olekust X_i väljuvate tõenäosuste summa 1. Väga sageli valitakse $prior_i$ väärtus kõikide i korral samaks ning $prior_{ij} = \frac{prior_i}{T}$, kus T on olekuruumi suurus.

Esitus graafina

Markovi ahelat võib kujutada suunatud graafina, mille tippudeks on hulga S elemendid ning servadele kahe tipu vahel on antud kaalud, mis iseloomustavad üleminekutõenäosust ühest olekust teise. Serva ei kujutata, kui üleminekutõenäosus kahe oleku vahel on 0.

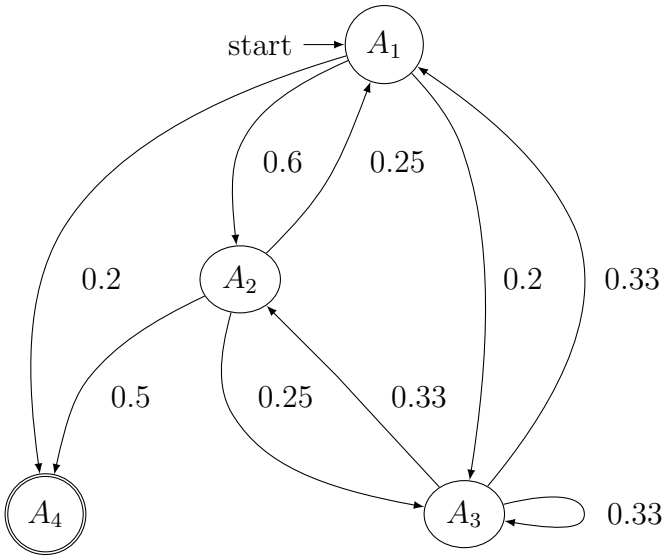
Näide. Olgu meil antud jadade komplekt ning koostame sellele vastava Markovi mudeli, kasutades suurima tõepära hinnangut. Olgu jadad järgnevad:

$$A_1, A_2, A_3, A_1, A_2, A_4$$
$$A_1, A_3, A_3, A_2, A_4$$
$$A_1, A_2, A_1, A_4$$

Sellisele jadade komplektile vastava mudeli üleminekumaatriksi hinnang on:

$$\hat{P} = \begin{pmatrix} 0 & 0.6 & 0.2 & 0.2 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

kus p_{ij} on tõenäosus, et sündmusele A_i järgnes A_j. Joonisel 1.1 on kujutatud antud mudeli esitus graafina.



Joonis 1.1: Markovi ahela mudeli esitus graafina.

1.2 Klasterdamine

Klasterdamine on objektide omavaheline grupeerimine sarnasuse alusel kasutades ainult objektide omadusi. Eesmärgiks on saada klastrite kogu nii, et igas klastris paiknevad kirjed oleks võimalikult sarnased üksteisele ning kirjed ühes klastris erinevad võimalikult palju kirjetest teistes klastrites. Klasterdamist võib käsitleda ka kui automaatset klassifitseerimist, mille korral klastrid esindavad andmete klasse ning protsessi käigus koostatakse klassijaotus andmetest.

Andmete grupeerimine sarnasuse alusel võimaldab edasi anda andmestikus olevat andmete loomulikku struktuuri. Mõned klasterdamise meetodid iseloomustavad klastreid prototüübi abil, milleks on klastri kõige iseloomulikum objekt. Prototüüpe võib kasutada andmete resümeerimiseks, esitades iga klastri ainult selle prototüübiga [TSK05].

Klasterdamisalgoritme saab klassifitseerida kaheks: eraldusmeetoditeks (ingl. k. *partitional methods*) ning hierarhilisteks meetoditeks (ingl. k. *hierarchical methods*).

Hierarhiliste meetodite väljundina tekib mitu hierarhilist andmete jaotust klastriteks, eraldusmeetodite tulemuseks aga vaid üks jaotus [Juh03]. Hierarhilise klasterdamise korral esitatakse andmestik puu kujul, kus lehtedeks on andmeobjektid. Puu on

genereeritud põhimõttel, et kahe objekti sarnasus on kaugus nende ühise hargnemiseni puus.

Eraldusmeetodite puhul jagatakse objektid etteantud arvu k grupiks, kusjuures gruppide arv ei tohi ületada kirjete arvu ja iga grupp peab sisaldama vähemalt ühte kirjet, kusjuures iga kirje kuulub vähemalt ühte gruppi. Hägusate meetodite (ingl. k. *fuzzy clustering*) korral võib iga objekt kuuluda mitmesse klastrisse samaaegselt, iseloomustades sellega objekti kuulumist kahe klastri vahele [TSK05].

Kaugusmõõt

Andmete klasteradamiseks on vajalik mingisugune mõõt, mis iseloomustab sarnasust kahe objekti vahel. Praktikas kasutatakse sarnasuse asemel erinevust ehk kaugust kahe objekti x_i ja x_j vahel, mida tähistatakse kui $d(x_i, x_j)$, kus $x_i = (x_{i1}, \ldots, x_{ip})$ ja $x_j = (x_{j1}, \ldots, x_{jp})$ on kaks objekti p-mõõtmelises ruumis.

Kaugusmõõt peab rahuldama järgmisi tingimusi[Juh03]:

1. $d(x_i, x_j) \geqslant 0$: Kaugus on mittenegatiivne arv.
2. $d(x_i, x_j) = 0$: Kaugus objektist iseendani on võrdne 0.
3. $d(x_i, x_j) = d(x_i, x_j)$: Kaugus on sümmeetriline funktsioon.
4. $d(x_i, x_j) \leqslant d(x_i, x_h) + d(x_h, x_j)$: Otsene tee x_i-st x_j-sse on lühem, kui suvaline muu tee, mis läbib suvalise objekti x_h.

Kõige sagedamini kasutatav kaugusmõõt on Eukleidiline kaugus:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + \ldots + (x_{ip} - x_{jp})^2},$$

1.2.1 K-keskmiste algoritm

K-keskmiste algoritm (ingl. k. *k-means algorithm*) on eraldusmeetodil põhinev algoritm, mis jagab n objekti k klastritesse nii, et iga objekt kuulub klastrisse, mille tsentroid on talle lähim. K-keskmiste meetodi puhul asetseb iga objekt ainult ühes klastris, mille tsentroidiks on temas olevate objektide keskväärtus.

K-keskmiste meetod töötab järgnevalt. Kõigepealt valitakse andmestikust juhuslikult k objekti, mis algselt esindavad klastrite tsentroide. Kõik ülejäänud objektid lisatakse klastrisse, mille tsentroid on talle lähemal. Sellega on esimene grupeerimine sooritatud.

Järgnevalt arvutatakse uued klastrite tsentroidid, milleks on kõigi klastri elementide keskmine. Nüüd, kui meil on k uut tsentroidi, leiame iga n elemendi jaoks talle kõige lähema tsentroidi ja lisame sellesse klastrisse. Tekib korduv tsükkel, mille tulemusena mingi hetk klastrite tsentroidid enam ei muutu ning seega on lõppenud algoritmi töö. Meetodi eesmärgiks on minimiseerida kriteeriumfunktsiooni, mis antud juhul on ruutvea funktsioon (ingl. k. *sum of squared errors*, edaspidi SSE)

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} d(\mathbf{c_i}, x)^2,$$

kus x tähistab andmeobjekti, C_i i-ndat klastrit, $\mathbf{c_i}$ selle klastri tsentroidi ning d eukleidilist kaugust. Eukleidilise kauguse korral võib kriteeriumfunktsiooni kohta öelda järgnevalt: iga objekti jaoks leitakse tema viga ehk kauguse ruut klastri tsentroidist, kuhu ta kuulub, ning seejärel summeeritakse kõik kauguste ruutväärtused, ning summa ongi SSE väärtus. Ruutvea funktsiooni väärtus on seda väiksem, mida lähemal on objektid oma klastri tsentroididele [HK00].

Algoritmi mäluvajadus on väike, kuna on vaja salvestada ainult andmeobjektid ning tsentroidid. Mälu keerukus on $O((m+k)n)$, kus k on klastrite arv, m on objektide arv andmestikus ning n on atribuutide arv. Meetodi ajaline keerukus on lineaarne andmeobjektide arvuga, täpsemalt $O(I*k*m*n)$, kus I on algoritmi tööjooksul sooritatud iteratsioonide arv. I väärtust on võimalik piirata ning on teada, et kõige suuremad muudatused objektide asetamisel klastritesse toimuvad paari esimese iteratsiooni jooksul [TSK05].

Algoritm annab häid tulemusi, kui klastrid eralduvad teineteisest kui sfäärilised objektide hulgad. Puuduseks on see, et klastrite arv k peab olema kasutajapoolt ette antud. Probleemiks on ka esialgsete klastrite keskpunktide valik, sest algoritm annab k punktide erineva valiku korral erinevad tulemused. Samas on tundlik erindite ning vigaste andmete suhtes, kuna üksik erakordselt suur väärtus mõjutab klastri keskväärtust ja seeläbi ka klastrite struktuuri [Juh03].

1.2.2 Klastrite valideerimine

Üks peamisi vajadusi klasterduse valideerimiseks on fakt, et algoritmid leiavad klastreid andmestikust ka siis, kui neid seal ei eksisteeri. See loob vajaduse meetodite järgi, mis võimaldavad hinnata algoritmi töö tulemust. Algoritmi disainimisel huvitab kindlasti millisel määral suudab algoritm tuvastada olmasolevaid klastreid andmestikus. Sellist lähenemist nimetatakse juhendatud valideerimiseks. Lisaks huvitab klasterduse korral tulemuste struktuur ehk kas klastrid on tugevalt eraldunud või kas nad kattuvad osaliselt.

Juhendatud valideerimine

Juhendatud(ingl. k. *supervised*) valideerimise korral on meil teada objektide mingi olemasolev klassimärgistus, mille abil saab võrrelda, kui hästi algoritm jagas objektid klastritesse. On selge, et klasterdamine toimus hästi kui iga klaster sisaldab ainult ühe klassi objekte, ning halvasti kui ühes klastris on võrdselt esitatud kõikide erinevate klasside objektid. Üldjuhul ei oma klasterdamine mõtet kui eelnevalt on teada objektide klassimärgistus, mis annab edasi objektide omavahelist sarnasust, küll aga annab see võimaluse hinnata erinevate algoritmide täpsust ning võrrelda neid [TSK05].

Üheks selliseks mõõduks on puhtus (ingl. k. *purity*, kasutame ka *täpsus*), mis iseloomustab kui suure osana iga klaster koosneb ainult ühe klassi elementidest. Iga klastri j jaoks arvutatakse p_{ij}, tõenäosus, et klastri C_i liige sisaldub klassis j. $p_{ij} = m_{ij}/m_i$, kus m_i on objektide arv klastris C_i ning m_{ij} on klassi j objektide arv klastris C_i. Kasutades

sellist jaotust, saame klastri C_i puhtuseks

$$p_i = \max_j p_{ij}$$

Üleüldine klasterduse puhtus avaldub:

$$\text{puhtus} = \sum_{i=1}^{k} \frac{m_i}{m} p_i,$$

kus m on objektide koguarv.

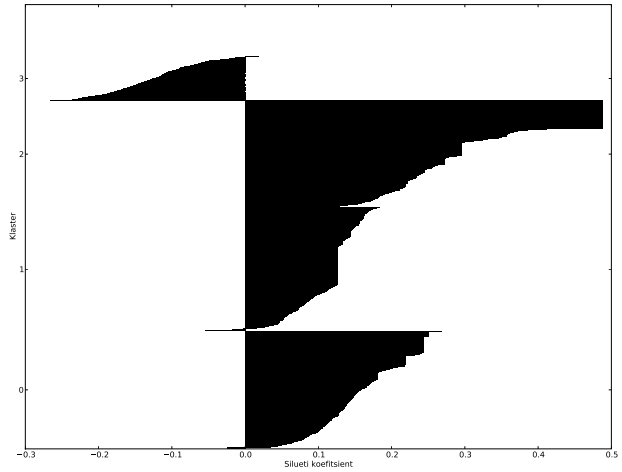
Juhendamata valideermine

Juhendamata (ingl. k. *unsupervised*) valideerimine annab vahendid klasterduse tulemuse hindamiseks kui puudub väline info klassimärgistuse või muu omaduste kohta. Silueti koefitsient (ingl. k. *Silhouette coefficent*) on mõõt, mis aitab hinnata kui tugevad on klastrid, kombineerides info klastrisisese tugevuse kohta ja info kui tugevalt klaster on eraldunud teistest klastritest [TSK05].

Silueti koefitsiendi arvutamine toimub järgneva algoritmi alusel. Iga objekti x_i korral leitakse tema keskmine kaugus kõigi teiste objektide suhtes, mis on temaga samas klastris, a_i. Lisaks leitakse iga objekti x_i korral ja iga klastri C_j korral, kuhu see objekt ei kuulu, tema keskmine kaugus kõigi klastri C_j objektide suhtes. Nendest kaugustest leitakse minimaalne keskmine kaugus, s.t. minimaalne keskmistest kaugustest C_j liikmete ja x_i vahel, b_i. Iga objekti x_i korral silueti koefitsient avaldub kujul:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)},$$

mille väärtus jääb vahemikku $[-1, 1]$. Negatiivne väärtus on halb, kuna see viitab olukorrale kui keskmine klastri sisene kaugus a_i on väiksem kui minimaalne keskmine kaugus mõne teise klastriga b_i, ehk teisisõnu objekt i asub kahe klastri piirimail ning klastrid ei ole koondunud. Silueti koefitsiendi hea väärtus oleks positiivne ja a_i nullile lähedal kuna s_i omab maksimum väärtust 1 kui $a_i = 0$. See iseloomustab punkti i paiknemist suhteliselt klastri keskmes. Klastri keskmist silueti koefitsienti saab arvutada kui klastri elementide silueti aritmeetilist keskmist. Kõikide klastrite silueti koefitsienti on defineeritud kui kõigi objektide silueti koefitsiendi keskmine [TSK05]. Praktikas on kasulik silueti koefitsiendi visuaalne kujutamine (Joonis 1.2), mille abil saab lihtsalt tuvastada tugevalt koondunud klastrid.



Joonis 1.2: Näide silueti koefitsiendist nelja klastri korral. Iga objekti jaoks klastris on kujutatud tema silueti koefitsiendi väärtus. Objektid on grupeeritud klastrikuuluvuse järgi ning sorteeritud kahaneva väärtuse järgi.

1.3 Peakomponentide analüüs

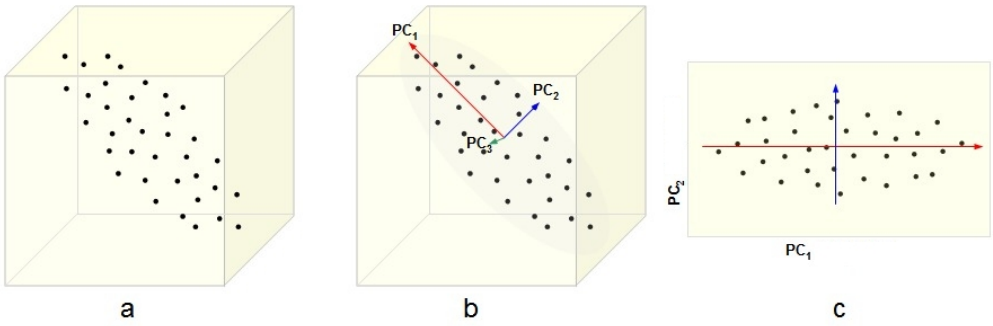
Peakomponentide analüüs (ingl. k. *principal component analysis*, edaspidi *PCA*) võimaldab vähendada andmestiku dimensionaalsust. Olgu andmestik esitatud maatriksi kujul, kus iga rida iseloomustab mingit andmeobjekti, ning veergudes on andmeobjektide atribuutide väärtused. Peakomponentide analüüsi abil saab vähendada sellise maatriksi mõõtmeid, kombineerides olemasolevate atribuutide abil uued vektorid, mida nimetatakse peakomponentideks. Nende abil saab esindada andmestiku vähema dimensionaalsusega kui algselt, sealjuures säilitades andmestikus leiduva dispersiooni. Teisisõnu on peakomponent vektor, millele andmeid projitseerides leidub kõige rohkem infot andmestikus oleva dispersiooni kohta.

Anname siinkohal lihtsustatud kirjelduse *PCA* tööpõhimõttest, täpsema kirjelduse võib leida tööst [Shl05].

Kõigepealt sisendandmestik normaliseeritakse, nii et atribuutide keskväärtus oleks 0 ja standardhälve 1. Seejärel leitakse andmetes selline ühikvektor, millele andmeid projitseerides on andmete hajuvus maksimaalne. Antud vektorit nimetatakse esimeseks peakomponendiks. Peale selle saab otsida ka teist peakomponenti - ühikvektorit, mis on ortogonaalne esimese peakomponendiga ning millele andmeid projitseerides on allesjäänud dispersiooni (andmestikus olev hajuvus mis ei ole seotud esimese peakomponendiga) maksimaalne. Analoogselt defineeritakse kolmandat, neljandat ja n-ndat peakomponenti[HK00].

Peakomponentide analüüsi saab kasutada suure dimensionaalsusega andmestiku visualiseerimiseks, projitseerides kaks tähtsamat komponenti tasandile. See annab võimaluse uurida andmeid ning leida mustreid. Joonisel 1.3 on näide peakomponentide analüüsist kolmemõõtmelises punktide ruumis, kus punktid projitseeritakse tasandile,

säilitades andmestiku põhijooned ehk punktide elliptilise paiknemise ruumis.



Joonis 1.3: *PCA* näide. *a*) Kolme atribuudiga andmestik. *b*) Kolm risti asetsevat peakomponenti PC_1, PC_2 ja PC_3. *c*) Andmestik projitseerituna kahele esimesele peakomponendile [Kav09].

Antud töös rakendame peakomponentide analüüsi Markovi ahelate mudelite üleminekumaatriksitel ning leiame kaks peakomponenti, mis annavad edasi kõige suurema osa dispersioonist. Nende peakomponentide projitseerimisel saame esmase ülevaate andmestikust. Kui punktid asuvad üksteisest kaugel kõrgemas dimenisonaalsusega ruumis, asuvad nad ka kaugel üksteisest kui nad on projitseeritud kahele peakomponendile. Selle abil saab visuaalselt uurida klastrite leidumist andmestikus. Samuti võimaldab analoogne visualiseerimine leida andmestikus olevaid erindeid.

Peatükk 2

Sessioonigruppide leidmine

Selles peatükis tutvustame lähemalt sessioonigruppide leidmise protsessi ja anname ülevaate olemasolevatest lahendustest. Ülevaade on koostatud üldises vaates, jättes võimaluse rakendada antud protsessi peaaegu kõikidele veebirakenduse logifailidele.

2.1 Probleemi püstitus

Sessioonigruppide tuvastamise eesmärgiks on leida veebirakenduse logifailide järgi sarnase kasutaja käitumisega sessioonid, mille käigus sooritati samasuguseid tegevusi ehk külastati samasid lehekülgi. Logifailide kasutamise tingimus põhineb teadmusel, et peaaegu kõik veebiserverid võimaldav kirjutada kasutajate poolt sooritatud tegevusi logidesse ning seega ei ole tarvidust muuta rakendust ega komponente, et saada vastavat infot.

Sessioonigruppide tuvastamiseks on vaja esiteks andmeid, ehk sessioone, mida grupeerida. Teiseks tuleb leida mingisugune mõõt, mis aitab mõõta erinevust kahe sessiooni vahel ehk kaugust. Peale selle toimub sessioonide grupeerimine, mis seisneb valitud algoritmi rakendamisel kasutades leitud mõõtu. Peale klastrite leidmist toimub klastrite analüüs, sageli kasutatakse selleks sessioonide visuaalset esitust. Antud probleemi juures peab meeles pidama, et tavaliselt on andmeid väga palju, ehk algoritm peab olema võimeline töötama ka suurte andmemahtude juures.

2.1.1 Kasutaja- ja sessioonigruppide erinevused

Kui veebirakenduse sessioonidele lisaks on teada ka infot kasutaja kohta, kes on sessiooni autor, saab sessiooni asemel leida kasutajate klastreid. Kasutajate grupperimiseks tuleb eelnevalt kõikides kasutaja sessioonide leiduv info grupeerida ning luua mudel, mis edastaks kasutajaprofiili. Seejärel saab võrrelda kasutajaprofiili teiste kasutajate omadega ning leida sarnased grupid.

2.2 Lahendus

2.2.1 Andmete esitus

Klastrite leidmise protsess algab logifailide eeltöötlusega, mille kõige tähtsam osa on sessioonide eraldamine logifailidest. Sessioonide eraldamisega saadakse sessioonide kogu, kus iga sessioon sisaldab tegevuste jada. Antud töös mõistame veebisessiooni all järjestatud kogumit kasutaja poolt sooritatud tegevustest. Tegevuseks loeme kasutaja enda poolt sooritatud tegevust, näiteks ühelt leheküljelt teisele liikumist. Iga tegevusega võib olla seotud lisainfot nagu tegevuse sooritamise aeg ja veebilehitseja, millega tegevus sooritati. Meid huvitab veebisessioonist ainult klikivoog (ingl. k. *clickstream*) ehk lehekülgede järjestus, mida kasutaja külastas.

```
foo.bar.edu - - [16/Nov/2008:18:50:04 -0800] \
"GET /$$87612/sigmod_record/ HTTP/1.0" 200 1252
foo.bar.edu - - [16/Nov/2008:18:50:14 -0800] \
"GET /$$87612/sigmod_record/issues.html HTTP/1.0" 200 653
foo.bar.edu - - [16/Nov/2008:18:50:23 -0800] \
"GET /$$87612/sigmod_record/9-95/ HTTP/1.0" 200 3565
foo.bar.edu - - [16/Nov/2008:18:50:29 -0800] \
"GET /$$87612/sigmod_record/issues.html HTTP/1.0" 200 653
```

Joonis 2.1: Näide logifailist.

Veebiserverid üldjuhul ei salvesta infot sessiooni kohta, vaid salvestavad iga kasutaja tegevuse kui iseseisva toimingu. Lisaks puudub sageli ühene tunnus, mille järgi paigutada kasutaja tegevused ühte sessiooni. Selleks võib kasutada lähenemist, et fikseeritud lühikese ajaintervalli jooksul tehtud päringud samalt *IP*-aadressilt on sama sessiooni osa. Joonisel 2.1 oleva logifaili tegevused kuuluvad tõenäoliselt sama sessiooni piiresse, kuna pärinevad samalt aadressilt, *foo.bar.edu* ning on sooritatud lühikese aja jooksul. Selles töös ei vaadelda sügavamalt sessioonide eraldamiseks kasutatavaid algoritme, seda on uuritud töös [BMMM05].

Sessioonide eraldamise juures on ka oluline, et sessiooni hulgas oleks ainult konkreetsed kasutaja sihipärased tegevused. Sageli on sessioonides ka info, millist pildifaili kasutaja laadis veebiserverist, mida antud juhul käsitleme mürana ning see eemaldatakse. Samuti eeldatakse töös, et kõik kasutaja lehekülgede külastused ja tegevused on salvestatud logifail ning puhverdamist ei ole kasutatud, ehk sessioon on terviklik.

Järgmise sammuna tuleb sessioonid viia kujule, mis oleks sobiv, et arvutada kaugust sessioonide vahel ning sooritada grupeerimist.

2.2.2 Tegevuste üldistamine

Veebisessioonides külastatav lehekülgede arv võib ulatuda väga suureks, seega on otstarbekas kuidagi vähendada erinevate veebilehekülgede arvu. Sageli on selleks kasutataud lehekülgede üldistamist [BG01, JJK00, FSyS99, WZ02], kus veebilehekülged

grupeeritakse eelnevalt juba gruppidesse ning sessioonides asendadatakse iga lehekülg vastavat gruppi tähistava leheküljega. Selleks kasutatakse veebilehekülgede hierarhilist paiknemist ning selle väljendumist lehekülgede aadressides. Vaatleme olukorda, kui on antud mingi järgnevad leheküljed järgnevate aadressidega:

- U_1 =*http://www.example.com/kategooria1/lehekülg1/parameeter1*
- U_2 =*http://www.example.com/kategooria1/lehekülg1/parameeter2*
- U_3 =*http://www.example.com/kategooria1/lehekülg2/parameeter2*
- U_4 =*http://www.example.com/kategooria2/lehekülg2/parameeter1.*
- U_5 =*http://www.example.com/kategooria2/lehekülg3/parameeter3.*

Kuigi kõik aadressid viitavad viitavad erinevatele lehekülgedele, võib aadresside struktuuri järgi öelda, et U_1 on sarnasem U_2-le, kui seda on U_4, sest U_1 ja U_2 omavad erinevust ainult viimasel tasemel, ehk erineb ainult üks parameeter. Tasemete eraldajaks loeme /. Selle idee abil saab näiteks asendada kõik aadressid tema esimese taseme aadressigaga. Antud juhul jääb meil alles ainult kaks erinevat lehekülge: /*kategooria*1 ja /*kategooria*2, millega saavutasime enam kui kahekordse vähendamise lehekülgede arvu suhtes.

Näide. Oletame, et meil olid antud ka sessioonid, mis sisaldavad eelpool kirjeldatud lehekülgi.

- $S_1 = U_1U_2U_1U_3U_5$
- $S_1 = U_2U_4U_4U_5U_4$

Siis üldistamise tulemusena kui asendasime U_1, U_2, U_3 üldistatud leheküljega U_{p1} ning ülejäänud U_{p2}, siis avalduvad vastavad sessiooni:

- $S_1 = U_{p1}U_{p1}U_{p1}U_{p1}U_{p2}$
- $S_1 = U_{p1}U_{p2}U_{p2}U_{p2}U_{p2}$

Tulemusena tekkinud sessioonid omavad sarnasusi oma üldistamata vastastega, säilitades tegevuste järjekorra, samuti iseloomustavad üldistatud leheküljed erinevate leheküljeklasside külastuste jaotust sessioonides. Vaadeldud meetod on lihtne vahend vähendamaks lehekülgede arvu, mis võimaldab omakorda vähendada klasterdamise keerukust, ning hiljem lihtsustab tulemuste analüüsi.

2.2.3 Klasterdamine

Järgmiseks sammuks on klasterdamise protsess, mille tulemusena leitakse sarnased sessiooniklastrid. Klasterdamiseks on vajalik:

- Kaugusmõõt, mis aitaks tuvastada sarnaseid veebisessioone. Kaugusmõõdu valikul tuleb arvestada veebisessioonide erinevaid omadusi - järjestikuseid tegevusi ja lehekülgedel viidetud aega. Järjestikuseid tegevusi arvestavad mõõdud sobivad, et leida enamlevinuid kasutajate liikumisteekondi veebirakenduses. Teisest küljest, kui eesmärgiks on leida ainult sarnaste huvidega kasutajaid, ei ole vaja arvestada tegevuste järjekorda, vaid pigem aega, kaua kasutaja veetis mingil leheküljel. Lisaks võimaldab aeg tuvastada ka kasutajate käitumises anomaaliaid - kui kasutaja veedab väga vähe aega paljudel erinevatel lehekülgedel, viitab see sellele, et kasutaja otsib midagi. Info selliste sessioonide kohta on vajalik, kui eesmäriks on veebirakenduse ülesehituse parandamine.

- Klasterdamisalgoritm, mis kasutab leitud kaugusmõõtu. Algoritmi valikul on tähtis, kas algoritm suudab töötada mõistliku aja piires oodatavate andmemahtude juures.

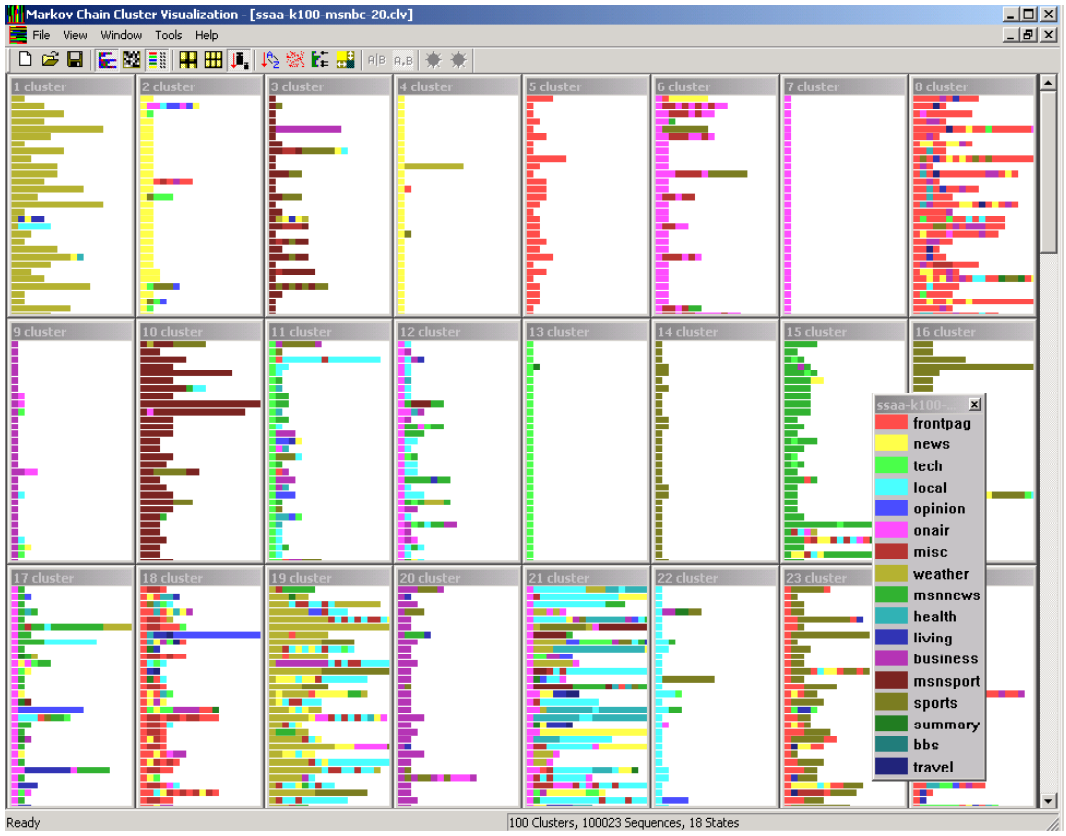
Võimalikke kaugusmõõte ja neid kasutavaid algoritme tutvustame jaotuses 2.3.

2.2.4 Tulemuste analüüs

Klastrite kogu olemasolul järgneb protsessi kõige tähtsam samm - tulemuste analüüs, interpreteerimine ja järelduste tegemine. Mahu tõttu võib olla kõikide sessioonide ja klastrite ükshaaval läbivaatamine keerukas, kui mitte võimatu. On loodud erinevaid lahendusi, mis aitavad lihtsustada seda tegevust. Enamlevinumateks on assotsiatsiooni reeglite avastamine klastrites ja korduvate järjestikuste mustrite ehk sagedamini kasutavate alamjadade koostamine [FL05], mida saab kasutada iga klastri iseloomustamiseks, rakendades meetodeid ainult klastris sisalduvatele andmetele.

Paljudes olukordades on klastrid esimeseks sammuks suuremal protsessil, näiteks linkide soovitamise juures vastavalt kasutaja külastusajaloole. Kõigepealt on teada mingisugused tüüpklastrid, ning iga klastri peal on arvutatud välja tüüpilised leheküljed, mis neid kasutajaid huvitab, kes kuuluvad sinna klastrisse. Kui kasutaja vaatab ringi veebilehel, saab teda pärast mõningaid lehel sooritatud külastusi asetada mõnda eelnevalt teadaolevasse klastrisse ning selle klastri info põhjal näiteks kuvada kasutajale abiteavet või linke [MCS00] [YJGMD96].

Visuaalseks tulemuste interpreteerimiseks saab kasutada külastatud lehekülgede histogrammi, mis toob esile kõige sagedamini külastatud lehed. On selge, et erinevatel klastrite peaks olema ka erinevad enamkülastatud leheküljed, eriti kui klasteradmine on toimunud arvestades kasutaja käitumist ehk külastatud lehekülgi [HNC+01].



Joonis 2.2: Iga aken tähistab klastrit. Iga rida aknas viitab ühele veebisessioonile. Erinevad värvid tähistavad üldistatud lehekülgede kategooriaid [CHM+00].

Praktikas on rakendatud sessioonide visualiseerimiseks meetodit [CHM+00], kus kuvatakse kõrvuti kõik sessiooniklastrid ning iga klastris on sessioonid esindatud tegevuste järjenditena, kusjuures iga lehekülg on tähistatud erinevate värvidega. See võimaldab peale vaadates tuvastada sessiooniklastrite iseärasusi. Joonisel 2.2 on näide antud meetodist, kus on esitatud klastrid uudistelehekülgelt.

2.3 Olemasolevad lahendused

Järgnevalt anname ülevaate olemasolevatest lahendusest. Olemasolevaid lahendusi võib liigitada kolmeks: lahendused mis käsitlevad veebisessioone kui sõnesid; lahendused mis esitavad veebisessiooni mingisugusel vektorkujul, tuvastades selleks huvitavad atribuudid ning lahendused, mis kasutavad sessioonide modelleerimiseks Markovi ahelaid. Viimasega tutvume lähemalt järgmises peatükis.

2.3.1 Sessiooni esitus sõnekujul

Veebisessiooni võib esitada kui sõne, kus iga sümbol tähistab külastatud lehekülge ning terve tähestiku moodustavad kõik tegevused. Selline lähenemine võimaldab rakendada kaugusmõõduna olemasolevaid sõnede erinevuse mõõte, nagu näiteks *Levenshteini* kaugus [Gus97].

Töös [BG01] kasutatakse kahe sessiooni sarnasuse leidmiseks pikimat ühist alamsõne ideed koos leheküljel veedetud ajaga. Nende kahe omaduse kombinatsioonina saadakse mõõt, mis tähtsustab kahe sessiooni pikimaid ühiseid alamosasid ning nendel veedetud aega. Kasutatakse ka tegevuste üldistamist. Klasterdamiseks esitatakse kõik sessioonid graafina, kus tipud on sessioonid ning servadele antud kaalud iseloomustavad sarnasust sessioonide vahel. Algoritmile on antud parameeter θ millega saab elimineerida graafist need servad (ja vastavalt ka servadeta tipud), mille kaal on väiksem kui θ. Klasterdamiseks partitsioneeritakse graaf kasutades algoritmi *Metis*, eesmärgiga jaotada graaf k osaks nii, et iga osa koosneks omavahel suhteliselt lähedal asetsevatest tippudest. Autor nimetab meetodi plussiks lehekülgedel viibitud aja arvestamist, samuti algoritmi võimet töötada pikemate sessioonidega. Tulemuste interpreteerimiseks leitakse iga klastri jaoks leheküljed, kus on veedetud kõige rohkem aega. Algoritmi ajaline keerukus on vähemalt $O(n^2)$

Analoogset ideed on kasutatud ka töös [WZ02], kus sarnasus on defineeritud kui sekventside joondamisel tehtav operatsioonide arv. Klasterdamiseks on kasutusel meetodid *ROCK*, *TURN* ja *CHAMELON*, mis kõik tuvastavad ise vajaliku klastrite arvu. Autor nimetab meetodi plussiks seda, et algoritm suudab tänu üldistamisele leida sarnaseid sessioone, millel ei pruugi olla omavahel ühtivaid lehekülgi. Töös tulemuste sisulist analüüsi ei sooritata.

2.3.2 Sessiooni esitus vektorkujul

Sessiooni saab esitada ka vektorkujul $v = (a_1, a_2, \ldots, a_n)$, kus iga atribuut a_i viitab mingile sessiooni tunnusele. Vektorkujul esituse korral saab rakendada palju olemasolevaid kaugusmõõte, näiteks Eukleidilist kaugust.

Sellist lähenemist on rakendatud töös [FSyS99], kus esiteks on üldistatud sessioonid, ning siis on koostatud igast sessioonist vektor kujul

$$(t_1, t_2, \ldots, t_n),$$

kus t_i viitab üldistatud i-ndal lehekülgedel viidetud aegade summat. Klasterdamiseks on kasutusel hierarhiline meetod *BIRCH*, mis on võimeline töötama hästi ka suurte andmehulkade korral, omades ajalist keerukust $O(n \log n)$, kus n on sessioonide arv. Tulemuste interpreteerimiseks uuritakse enam külastatud üldistatud lehekülgi klastrites.

Lisaks on lahendusi, kus sarnasusi sessioonide vahel mõõdetakse kui sarnasust hulkade vahel. Töös [SLdCT06] on rakendatud *Jaccard* mõõtu ning klasterdamiseks k-keskmistele analoogset meetodit, mida on optimeeritud töötama kategoorilistele andmetele, muutes kriteeriumfunktsiooni. Analüüsiks uuritakse enamkülastatuid lehekülgi klastrites.

On loodud ka spetsiifilise rakenduse keskseid lahendusi [mCC01], mis eeldavad rakenduse põhjalikku analüüsi. Töös [mCC01] uuritakse veebipõhist raamatukogu kataloogisüsteemi ning selle kasutajate käitumise iseärasusi. Analüüsi käigus defineeritakse 47 erinevat parameetrit, mis iseloomustaksid kasutaja sessiooni. Mõned näited parameetritest: sessiooni käigus sooritatud otsingute arv, sessiooni algusest esimese otsin-

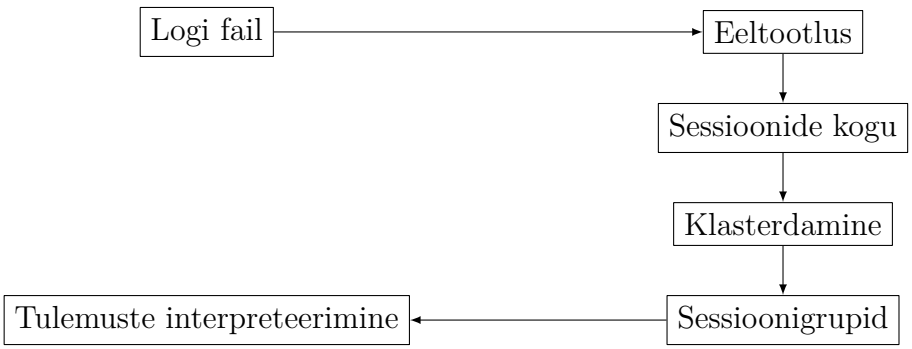
guni kulunud aeg, abiteabe kasutamise arv. Nende 47 numbrilise parameetri abil valiti peakomponentide analüüsi abil välja 16, mis moodustasid 75% dispersioonist. Klasterdamine sooritati kaheastmeliselt: esimesel sammul kasutati k-keskmiste algoritmi ja jagati 100 klastriks, teisel sammul pärast esimese sammu erindite eemaldamist hierarhilist meetodid, mis põhineb *Ward*i algoritmil. Erinditeks loeti klastrid, kus on ainult paar elementi.

Töös [WME04] uuriti erinevaid võimalusi kasutajagruppide tuvastamiseks. Üheks lahenduseks on esitada sessioon n-elemendilise vektorina, kus iga elemendi väärtus on kas 1 või 0, vastavalt kas lehekülge külastati sessiooni jooksul vähemalt üks kord või mitte. Klasterdamiseks on kasutusel k-keskmiste meetod.

2.3.3 Analoogsed lahendused teistes valdkondades

Kasutajasessioonide klasterdamist on uuritud ka andmebaasides [QYH05]. Töös uuritakse kuidas klasterada andmebaaside kasutajaid sooritatud *SQL* päringute järgi. Analoogset ideed saab lihtsalt üle kanda ka veebisessioonidele, kuna andmebaasi päringud saab veebi kontekstis ära asendada lehekülgede vaatamisega. Töös kasutataud kaugusmõõt koosneb kolmest komponendist: *Jaccard* sarnasusest, sekventside joondamise kaugusest ja heuristikast, mis hindab kahe sessiooni ühist naabrite arvu. Klasterdamiseks esitletakse partsioonilist meetodit, mis aitab tuvastada klastrite arvu.

2.4 Kokkuvõte



Joonis 2.3: Sessioonigruppide tuvastamise protsess.

Sessioonigruppide tuvastamise käigus (Joonis 2.4) eraldatakse veebirakenduse logifailist sarnase käitumisega ja huvidega sessioonide grupid. Kõige oluline selle protsessi juures on sessioonide sarnasusmõõdu ja klasterdamisalgoritmi valimine. Probleemi lahenduse jaoks on uuritud palju erinevaid meetodeid, milledest olemasolevatel lahendustel on erinevad probleemid:

- Vektoripõhised lahendused ei salvestad endas infot tegevuste järjekorra kohta, seejuures kaotades täpsust, ning ei sobi kasutaja teekondade analüüsiks.

- Sekventside järjestamisel loodud meetodite töö sõltub palju sessioonide pikkustest, andes erinevate pikkustega sessioonidele suure erinevuse ainult seetõttu, et nende pikkused erinevad. Selline meetod sobib ainult lühikeste sessioonide puhul ja siis kui kasutatakse üldistamist.
- Paljude lahenduste puhul ei ole head meetodit, kuidas anda edasi tulemusi. Tulemuste interpreteerimiseks kasutakse sageli lihtsalt sagedusanalüüsi, et tuua välja klastris esinevad leheküljed. Samuti on pööratud vähe tähelepanu järjestikuste mustrite eraldamisest klastritest.

Peatükk 3

Algoritm sessioongruppide leidmiseks

3.1 Markovi mudelitel baseeruv meetod

Lisaks eelmises peatükis vaadeldud meetoditele leidub veel klass meetodeid, mis kasutavad Markovi ahelaid sessioonide modelleerimiseks ja kauguse arvutamiseks sessioonide vahel. Selles peatükis vaatleme lähemalt meetodit, mille esialgne idee esitati töös [MAFM99] ning analoogne ka töös [HNC+01]. Illustreerimaks meetodi tööd on lisatud meetodi eksperimentaalne hindamine ja võrdlus teiste lahendustega.

Meetod kasutab sessioonide modelleerimiseks Markovi mudeleid ning klasterdamiseks k-keskmiste algoritmi. Edaspidi refereerime selles peatükis kirjeldatud meetodit kui *Markovi meetod sessioonide klasterdamiseks*.

3.1.1 Sessiooni esitus Markovi ahela mudelina

Markovi ahelaid on laialdaselt kasutatud veebisessiooni modelleerimiseks [CHM+00, LAV05, BL04]. Idee seisneb, et veebisessioon järgib Markovi omadust - järgmisena külastatav lehekülg sõltub ainult leheküljest, mida kasutaja hetkel vaatab. Eeliseks Markovi ahela kasutamises on eelkõige see, et ta salvestab tegevuste järjekorra. Praktikas on võimalik kasutada ka kõrgemat järku mudeleid sessiooni modelleerimiseks, mille korral sõltub külastatavale lehele sattumise tõenäosuse kahe või enama eelnevalt külastatud lehe põhjal, kuid sellisel juhul suureneb ka võimalik olekuteruum. Lisaks on mudel võimeline salvestama infot korraga mitme sessiooni kohta, mida saab rakendada klastrite iseloomustamisel, samuti kasutaja mudeli koostamiseks andmetest, kui on teada kõik kasutajamärgistusega sessioonid. Klastri iseloomustamiseks võib koostada mudeli kõikidest klastris olevatest sessioonidest.

Teisest küljest võimaldab mudeli generatiivne omadus koostada võimalikke veebisessioone mudeli põhjal, ning mudelist saab leida ka kõige tõenäolisemat tegevuste jada nendest sessioonidest, millest mudel on koostatud. See loob loomuliku vahendi mudeli interpreteerimiseks. Samuti saab mudeli abil ennetada kasutaja võimalikku järgmist külastavat lehekülge, mida saab rakendada kohanduvate veebisaitide loomisel.

Markovi ahela mudeli koostamisel võib kasutada suurima tõepära hinnangut, või vajadusel Bayesi hinnangut, kui on kahtlus puuduvate andmete kohta.

3.1.2 Kaugusmõõt

Sessioonide modelleerimisel Markovi ahelatega sõltub sarnasus sellest, kui sarnased on kahe mudelil vastavad üleminekutõenäosused iga olekute paari vahel. Antud esituses on kauguse arvutamiseks kaks võimalikku varianti: kasutada kauguse arvutamiseks Markovi mudelit graafina või leida erinevus kahe erineva üleminekumaatriksi vahel. Kasutades graafi, on võimalik rakendada erinevaid graafide teisenduskauguse variante (ingl. k. *graph edit distance*), kuid lihtsam on kasutada üleminekumaatriksit, mida on rakendatud ka varem [MAFM99, HNC+01], kasutades kahe üleminekumaatriksi vahelise erinevuse leidmiseks Eukleidilist kaugust.

Eukleidiline kaugus veebisessioonide $A1$ ja $A2$ vahel avaldub järgnevalt:

$$d(M(A_1), M(A_2)) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(M(A_1)_{ij} - M(A_1)_{ij}])^2},$$

kus $M(A)$ on sessiooni A Markovi mudeli üleminekumaatriks, $M(A)_{ij}$ on olekust i olekusse j ülemineku tõenäosus, ning n olekuruumi suurus, ehk võimalike tegevuste arv kõikide sessioonide peale kokku.

Kaugust kahe erineva sessiooni vahel mõjutab ülemineku ij tõenäosuse erinevus sessioonide $A1$ ja $A2$ vahel. Mida suurem on erinevus ülemineku ij vahel, seda erinevamad on sessioonid. Siit tuleneb ka kaugusmõõdu töötamise eeldus - mõõt käsitleb erinevusena neid sessioone, mille vastavad üleminekud on omavahel erinevad. Seega annab see erinevuse sessioonidele, kus tegevuste järjestus on teine, kuid külastatud leheküljed võivad olla samad.

3.1.3 Klasterdamine

K-kesmiste algoritm on sobiv algoritm sessioonide klasterdamiseks kirjeldatud meetodil. Eelkõige sobib algoritm oma hea võimega töötada suurte andmehulkadega. Antud juhul aga annab suurema lisaväärtuse algoritmi poolt tekitatud klastrite tsentroidid, mida saab käsitleda kui klastri prototüüpi. Tsentroid on Markovi mudel, mis on kõikide teiste mudelite keskmine, ning omab mingil määral kõikide klastriliikemete omadasi. Seega tekitab k-keskmiste algoritm korraga nii sessioonide jaotused klastritesse kui ka vahendid gruppide interpreerimiseks.

Töös [HNC+01] on rakendatud samadel tingimustel *k-modes* algoritmi, mis sarnaneb k-keskmiste algoritmile, kuid on mõeldud töötamaks kategoorilistel atribuutidel. Algoritm ei leia mitte tsentroide iga klastri jaoks, vaid klastrit iseloomustav element on klastri elementide mood ehk kõige sagedamine esinev objekt. Sellise rakendamisel on klastrid rohkem koondunud oma keskmiste ümber, kuna sageduse põhine prototüübi valik ei ole mõjutatud erinditest. Teisest küljest on veebisessioonide puhul raske rääkida erinditest, kuna kui kõik sessioonid on tekkinud legaalsel viisil, see tähendab kasutaja reaalsel rakenduse kasutamisel, siis on kõik sessioonid sama võrdsed. Erindid

pigem toovad välja kasutamise iseärasused. Selle koha pealt langeb ka eelis k-kesmiste algoritmile, kuna klastrites arvestatakse kõikide objektidega sealhulgas ka niiöelda erinditega.

3.1.4 Klastrite arvu hindamine

Klastrite arvu hindamine on keerukas protsess, mille jaoks ei ole ühtset universaalset lahendust[TSK05]. Praktikas on k-keskmiste algoritmi korral üheks võimaluseks klasterdada andmeid mitme erineva k väärtusega, ning iga tulemuse korral leida silueti koefitsiendi ja ruutveafunktsiooni väärtus. Kui kanda kõik väärtused graafikule võib märgata nende väärtuste muutusi. Eelkõige pakuvad huvi järsud langused kriteeriumfunktsiooni väärtuses klastrite arvu kasvades. Liiga suure klastrite arvu korral on klastrid hajusad, mitte tihedalt koondunud keskmise ümber. Sellise olukorral on ruutvea funktsiooni väärtus kindlasti palju suurem kui optimaalsete klastrite arvu korral. Teisestküljest muutub SSE väärtus klastrite arvu kasvades alati väiksemaks vastavalt oma definitsioonile, ning seega tulebki leida järsk väärtuse muutuse punkt. Silueti väärtuse korral tuleks otsida samasugust olukorda - hea klasterduse korral on klastrid rohkem koondunud, ehk silueti väärtus on suurem.

3.1.5 Näide

Järgnevalt illustreerime meetodi kaugusmõõdu tööd, kasutades selleks ideed tööst [HNC+01]. Olgu lehekülgede arv $n_p = 3$ ning leheküljed vastavalt P_1, P_2, P_3. Olgu meil antud ka kolm sessiooni
$A1 = P_1P_2P_3P_2P_2P_3P_3P_2P_3P_1P_2P_3$
$A2 = P_2P_2P_3P_1P_2P_3P_1P_2P_3P_3P_2P_3$
$A3 = P_1P_2P_3P_2P_2P_3$

Paneme tähele, et sessioon $A2$ on saadud sessioonist $A1$ esimese kolme lehekülje ja järgmise kolme lehekülje vahetamise abil ning samuti kolmanda ja neljanda lehekülgede kolmiku omavahelisel vahetamisel. Sessioon $A3$ moodustab sessioonist $A1$ pikkuselt pool.

Kasutades suurima tõepära hinnangut saame nende sessioonide esimest järku Markovi ahelate mudelite üleminekumaatriksid:

$$M(A_1) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{1}{5} & \frac{4}{5} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}, \; M(A_2) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{1}{5} & \frac{4}{5} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}, \; M(A_3) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 1 & 0 \end{pmatrix}$$

Kaugused järgnevate mudelite vahel on

$$\begin{aligned} d(M(A_1), M(A_2)) &= 0.35, \\ d(M(A_1), M(A_3)) &= 0.63, \\ d(M(A_2), M(A_3)) &= 0.95. \end{aligned}$$

Esimesel vaatlusel saab öelda, et antud kaugused sessioonide vahel iseloomustavad sessioonide erinevusi. A_1 ja A_2 omavad kõige rohkem sarnaseid üleminekuid, olenema-

ta sellest, et nende üleminekute järjekord pole sama mõlemas sessioonis. Samuti on A_3 sarnasem sessiooniga A_1, mis on põhjendatav, kuna A_3 on A_1 alamosa (A_3 moodustab pikkuselt pool sessioonist A_1). A_3 suurem erinevus A_2-st on tingitud eelkõige üleminekute, mis väljuvad leheküljelt P_3, erinevuses. Sessioonis A_2 on need üleminekud erinevad, samas kui A_3 puhul domineerib leheküljelt P_3 algavate üleminekute hulgas ainult üks. Samuti on A_1 korral üleminek olekust P_3 olekusse P_2 sarnasem A_3-ga kui A_2 puhul.

Oletame, et soovime klasterada antud kolme sessiooni kaheks grupiks. K-keskmiste algoritmi korral valitakse juhuslikud tsentroid, võime eeldada, et need olid A_2 ja A_3. Kuna A_1 on lähemal A_2-le, siis sattusid need ühte gruppi ning eeldame ka, et algoritmi töö lõpetati ühe iteratsiooniga. Sessioone A_2 ka A_3 sisaldava klastri tsentroidiks saab mudel:

$$(M(A_1) + M(A_2))/2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{1}{5} & \frac{4}{5} \\ \frac{3}{8} & \frac{3}{8} & \frac{1}{4} \end{pmatrix}$$

Antud tsentroid annab edasi mõlema sessiooni omadused, muutunud on ainult üleminekute tõenäosuste jaotus kolmandalt leheküljelt ülejäänud lehekülgedele.

3.2 Meetodi hindamine tehisandmetel

Praktikas huvitab, kas eelpool kirjeldatud meetod töötab ka reaaluses. Selleks sooritasime ekspermendid, kontrollimaks kas algoritm on võimeline tuvastama andmestikus leiduvad klastrid ning millise täpsusega algoritm leiab need. Kasutasime genereeritud tehisveebisessioonide kogu, kus erinevate paremeetriga sai muuta sessioonide omadusi, kõige olulisem on nendest kindlasti on andmestikus sisalduv klastrite arv.

3.2.1 Tehisveebisessioonide kogu

Veebisessioonide kogu loomisel lähtusime põhimõttest, et iga veebisessioon on genereeritud mingist Markovi mudelist [CHM+00]. Genereerimaks k klastriga veebisessioonide kogu, tuleb kõigepealt fikseerida k kasutajamudelit, millel on ära märgitud nii algus kui lõpplehekülg. Ühe mudeli järgi genereeritud sessioonid loeme samasse klastrisse kuuluvaks.

Juhusliku kasutajamudeli koostamine

Et genereeritud testandmestiks olevad veebisesioonid moodustaksid klastreid, peavad kasutatavad kasutajamudelid olema üksteisest erinevad, mistõttu genereerime need automaatselt.

Mudeli koostamine algab juhusliku arvu γ, valikuga *Beta* jaotusest parameetritega $B(1, 2)$, mille väärtus jääb vahemikku $(0, 1)$.

Järgnevalt antakse mudelis igalt leheküljelt üleminekutõenäosus igale teisele leheküljele nii, et ühest tipust väljuvad üleminekutõenäosused on valitud sõltumatult

Beta jaotusest $B(0.09, \gamma)$ ning normaliseeritakse, et tõenäosuste summa oleks 1. Antud parameetritega jaotus annab efekti, et ainult vähese tõenäosusega väljuvate kaarte väärtused oleksid suured. Selle tulemusel eristub igal lehel väljuvate kaarte hulgast paar üleminekut, mille kaal on suurem ning ülejäänud moodustavad ainult marginaalse osa. Samuti imiteerib antud parameetritega *Beta* jaotus *power-law* jaotust [BKM+00], millele alluvad veebitopoloogias olevate leheküljelt väljuvate linkide jaotus [FFF99]. Parameetri γ kasutamine aitab muuta erinevate mudelite üleminekute jaotust ning lisab rohkem juhuslikkust.

Järgnevalt valitakse ühtlase juhusliku valiku alusel, kas mingi üleminek ühelt leheküljelt teisele on üldse võimalik või mitte, ning seda tehakse iga võimaliku ülemineku puhul. Eesmärgiks on elimineerida üleminekuid, kuna reaalselt ei ole veebirakenduses võimalik navigeerida kasutades linke iga kahe võimaliku lehekülje vahel. Selleks, et fikseerida sessiooni keskmist pikkust lisame kõikidelt teistelt lehekülgedelt lõppleheküljele ülemineku tõenäouse $\frac{1}{\lambda}$, kus keskmine sessiooni pikkus on λ, mis aitab vältida liigselt pikki sessioone.

Sessiooni koostamine mudelist

Sessioonid koostatakse mudelitest kaalutud juhusliku valiku abil, mis toimub järgnevalt:

- Genereeri juhuslik arv a vahemikus $[0, 1]$ ühtlasest jaotusest. Olgu sessiooni algust tähistavalt leheküljelt väljuvate kaarte kaalud esitatud vektorina

  $$v = (p_1, p_2, \ldots, p_{n_p}),$$

  kus n_p on veebitopoloogias olevate lehekülgede arv. Vektori v väärtused sorteeritakse kasvavas järjekorras (indeksid jäävad samaks), ning valitakse selline element i, mis on esimene element kasvamise järjekorras, mille korral $a > p_i$. Valitud element i viitab järgmisele leheküljele.

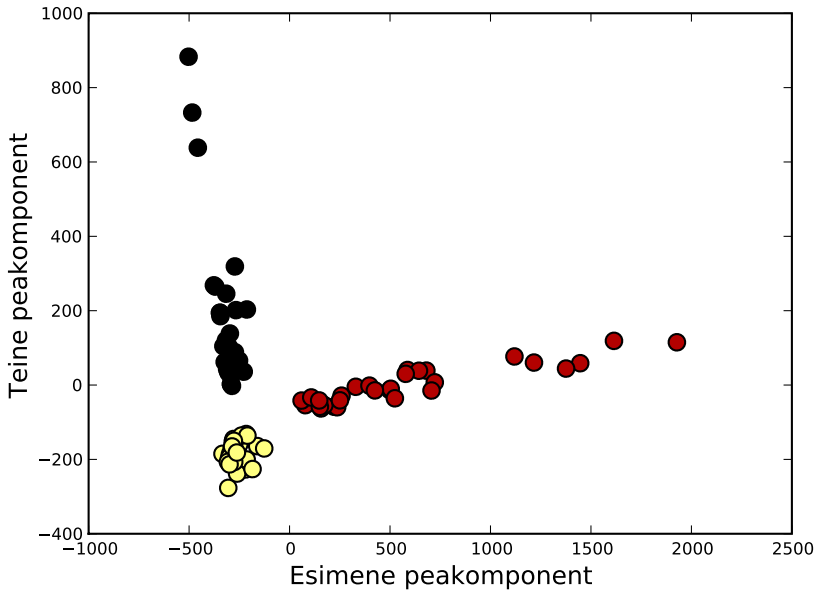
- Iga järgmise lehekülje puhul käitutakse analoogselt nagu eelmisel sammul. Sessioon lõppeb, kui jõutakse sessiooni lõppu tähistavale lehele.

Andmete genereerimisel rakendati veel kahte aspekti. Esiteks varieeriti keskmist sessiooni pikkust klastrite keskselt vahemikus $[0.5 * \lambda, 1.5 * \lambda]$, et erinevatel kasutajamudelitel oleks erinev keskmine sessiooni pikkus. Lisaks varieerisime minimaalset sessiooni pikkust vahemikus $[0.5 * \text{min}, 1.5 * \text{min}]$, kus min on etteantud parameeter minimaalseks sessiooni pikkuseks. Minimaalne sessiooni pikkuse varieerimine aitab eraldada kasutajamudeleid selgemalt üksteisest, vältides üksikuid, paari külastusega sessioone.

Näide

Illustreerimaks meetodi tööd, genereerisime kolme klastriga sessioonide kogu. Igast sessioonist ehitati Markovi mudel kasutades suurima tõepära hinnangut. Tulemuseks saadi maatriks M, kus iga rida esindas sessiooni ja veerud sessioonide üleminekuid. Sessioonid genereeriti kolme mudeli põhjal ehk klastrite arv oli 3, lehekülgede arv 100

ning iga mudeli põhjal genereeriti 20 sessiooni. Visualiseerimiseks sooritati maatriksi M põhjal peakomponentide analüüs ning projitseeriti tasandile kaks suurima dispersiooniga peakomponenti. Joonisel 3.1 on näha andmestiku projektsioon kahemõõtmelisele tasandile. On selgelt näha, et erinevad klastrid on eraldunud.



Joonis 3.1: Peakomponentide analüüs kolme klastriga tehisandmestikul. Sama värvi punktid tasandil tähistavad samasse klastrisse kuuluvaid sessioone.

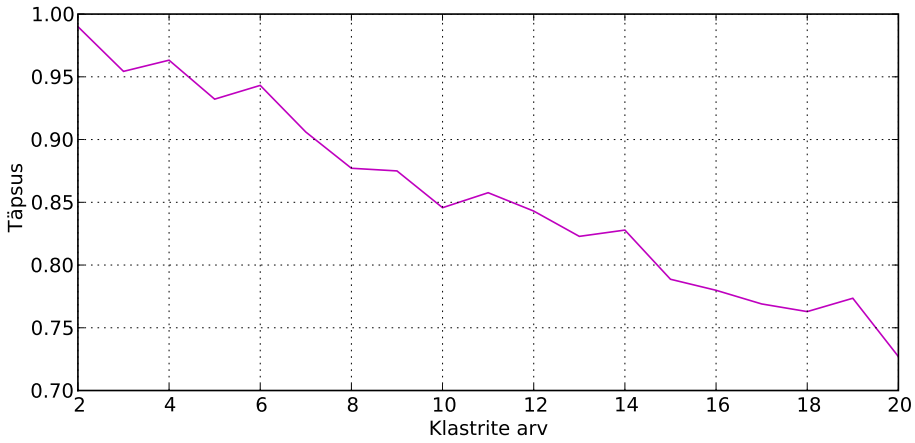
3.2.2 Tulemused

Markovi meetodil sessioonide grupeerimise algoritmi käitumise uurimiseks sooritasime eksperimente kasutades selleks genereeritud tehisandmestikku. Praktikas huvitab küsimus, kas algoritm suudab leida andmestikus esinevad klastrid ning kui suure täpsusega.

Andmed genereeriti järgnevatel parameetritega: sessioonide arv klastri kohta 50, lehekülgede arv 50, sessiooni keskmine pikkus 25 ning minimaalne pikkus 5. Klastrite arv varieerus vahemikus $2 \ldots 20$. Parameetritele valisime sellised väärtused, mis oleksid reaalselt esinevad. Klastrite arvu piiramine 20-ga, annab optimaalse klastrite arvu, mida inimene suudaks analüüsida ja hallata. Kuna k-keskmiste algoritm korral algsete tsentroidide valik on juhuslik ja seega ka terve algoritmi tulemus pole alati sama, sooritasime iga eksperimenti 10 korda. Eksperimendi protsessil genereeriti iga klastri arvu jaoks andmestik, ning siis klasterdati seda 10 korda ja leiti täpsuse aritmeetiline keskmine iga k väärtuse korral.

Kõikides järgmistes eksperimentides kasutasime mudeli hindamiseks Bayesi hinnangut koos parameetritega $prior_i = 1$, $prior_{ij} = 1/n_k$, et ühtlustada puuduvate andmetega tekkivaid jaotust, kus n_k on suvalisest tipust väljuvate üleminekute arv. Andmete puudumise korral saavad kõik ühest olekust väljuvad üleminekud võrdse tõenäosuste. Sellise hinnangu kasutamist saab põhjendada vähese sessioonide pikkusega

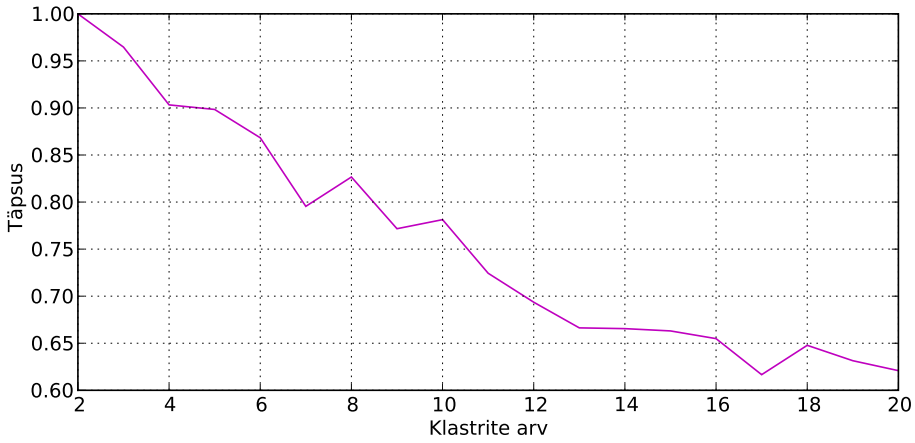
ning arvuga võrreldes lehekülgede arvuga, ning võimaliku olekuruumi suurusega.



Joonis 3.2: Klasterdamise täpsus klastrite arvu kasvades, $n_p = 50$.

Tulemustes (Joonis 3.2) on näha, et algoritm on võimeline eristama klastrid suhteliselt täpselt k väärtuste 2, 3 ja 4 korral. Sellel tuginedes saab väita, et antud meetod töötab ja leiab andmestikust klastrid. Klastrite arvu kasvades väheneb ka täpsus. Üheks põhjuseks võib olla, et suuremate klastrite korral ei suuda loodud tehisandmestik luua nii selgelt eralduvaid klastreid, teisest küljest on see ka meetodi iseärasus. Kuna sarnasus põhineb üleminekutõenäosustel, siis suuremate klastrite korral leidub selliseid sessioone rohkem, mis on erinevates klastrites, kuid omavad sarnaseid üleminekuid ning satuvad kahe klastri piirile.

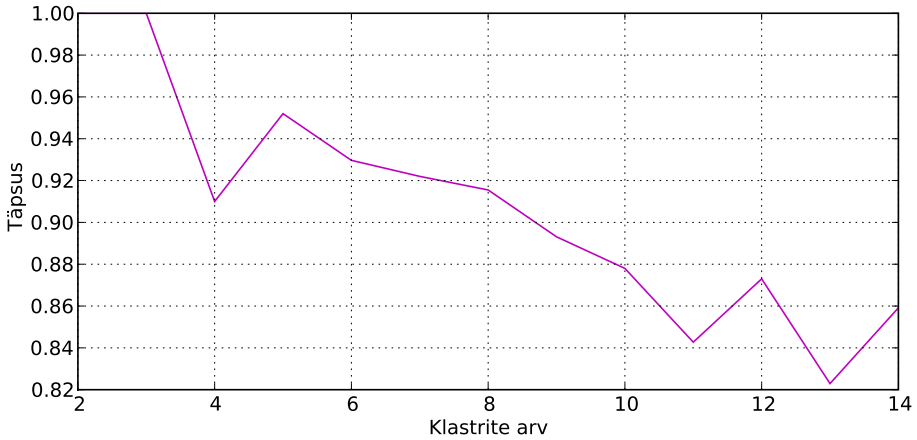
Kuna algoritm kasutab kauguse leidmiseks maatrikseid, mille dimensionaalsus sõltub erinevate lehekülgede arvust, siis uurisime kuidas käitub algoritm, kui seada lehekülgede arvu vähemaks, $n_p = 20$ (Joonis 3.3).



Joonis 3.3: Klasterdamise täpsus klastrite arvu kasvades,$n_p = 20$.

Võrreldes tulemusi $n_p = 50$ ja $n_p = 20$ korral, ei ole märgata suuri erinevusi. Pigem töötab algoritm paremini suuremate lehekülgede arvu korral, kuna siis avaldub rohkem

iseäralikke üleminekuid sessioonides. Praktikas võib aga veebilehekülgede arv olla veel suurem. Selleks uurisime meetodi tööd lehekülgede arvu väärtusega n_p=100 (Joois 3.4).



Joonis 3.4: Klasterdamise täpsus klastrite arvu kasvades,$n_p = 100$.

Antud meetod on võimeline töötama ka suurte lehekülgede arvuga ehk olekuruumiga. Väärtuste $k = 2, 3$ korral saavutas meetod 100% täpsuse, mida väiksema lehekülgede arvuga saavutati vähem. Kasutatud tehisandmestiku korral on tõenäolisem, et suurte lehekülgede arvu korral kasutab iga mudel ainult mingit osa lehekülgedest ning kahe erineva mudeli poolt kasutatavate üleminekute ühisosa on väiksem.

3.2.3 Alternatiivid

Nüüd kui on selge, et Markovi mudelil baseeruv meetod on võimeline leidma andmestikus olevaid klastreid, pakub huvi, kuidas käitub Markovi mudelitel baseeruv meetod võrreldes teiste olemasolevate lahendustega. Uurisime lahendusi, milles ei kasutata sessiooni klasterdamiseks parameetrina leheküljel viibitud aega, vaid ainult külastatuid lehekülgi, ning rakendasime algoritme meie tehisandmestikul.

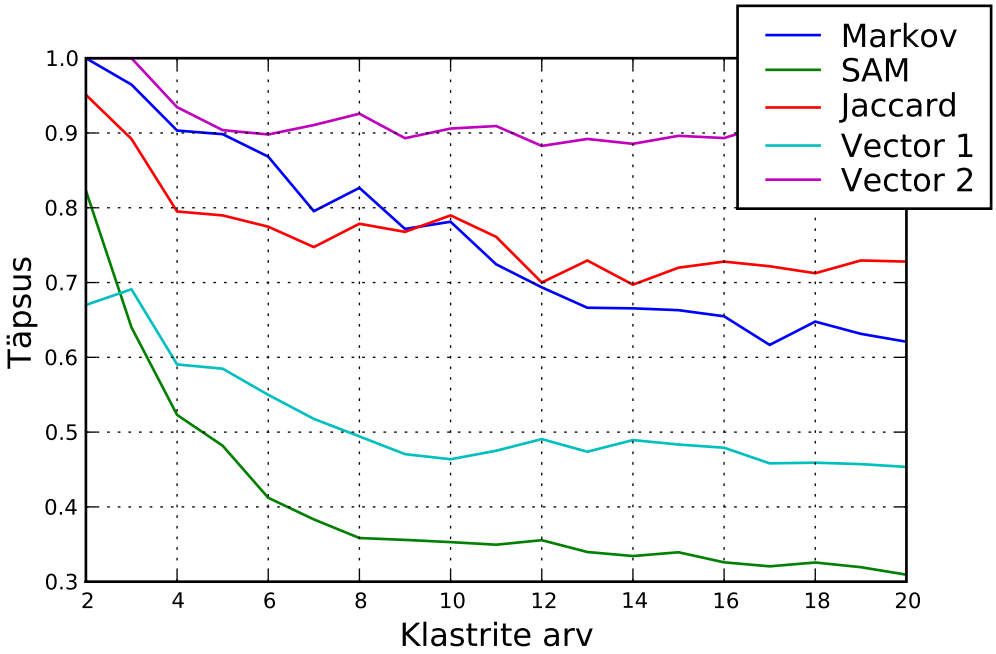
Sõnepõhise kaugusmõõduna kasutasime sekventside joondamise meetodit (*SAM*, *sequence alignment method*), mida on rakendatud töös [WZ02]. Võrdlusesse lisasime veel hulkade sarnasusel põhineva lahenduse(*Jaccard*), mis mõõdab sessioonide vahelist sarnasust kui sarnasust hulkade vahel kasutades *Jaccard* mõõtu(*Jaccard distance*)[SLdCT06]. Mõlemate meetodi puhul kasutasime klasterdamiseks *k-medoids* algoritmi, mis on sarnane k-keskmiste algoritmile, kuid mõeldud töötama kategooriliste atribuutidega ja objektidega, kus ei saa defineerida keskmist.

Tööst [WME04] võtsime vektoripõhised meetodid sessioonide vaheliste kauguste mõõtmiseks. Esimene nendest(*Vector 1*), esitab sessiooni n_p elemendilise vektorina

$$(x_1, x_2, \cdots, x_{n_p}),$$

kus element x_i tähistab mitu korda külastati i-ndat lehekülge. Teine lahendus (*Vector 2*), kasutab sama esitust, kuid $x_i = 1$, kui lehekülge i külastati vähemalt üks kord, ning $x_i = 0$, kui lehekülge ei külastatud mitte kordagi. Nende meetodite puhul kasutasime k-keskmiste algoritmi, mida rakendati ka algses töös [WME04].

Eksperiment sooritati andmestikul, kus oli 30 lehekülge, keskmine sessiooni pikkus oli 25 ning sessioone klastri kohta oli 50.



Joonis 3.5: Kaugusmõõtude võrdlus.

Tulemustest (Joonis 3.5) ilmneb, et Markovi meetod ei anna kõige täpsemaid tulemusi, vaid kaotab täpsust klastrite arvu kasvades. Seevastu meetodid *Jaccard* ning *Vector 2* on palju stabiilsemad. Meetod *SAM* on suhteliselt täpne $k = 2$ korral, kuid kaotab suuremate k väärtuste korral oma väärtused.

Vaadeldud Markovi meetod sessioonide grupeerimiseks ei ole alati kõige täpsem, kuid kombineerides endas lehekülgede järjekorra ning võime esitada mitut sessiooni, on sellel meetodil palju eeliseid teiste vaadeldud meetodite ees. Samuti peab arvestama, et võrdluse tulemused kehtivad pigem meie tehisandmestikul, ning meetod *Vector 2* ei pruugi alati anda absoluutselt paremaid tulemusi. Pigem aitab võrdlus põhjendada, et vaadeldud meetodi on võimelised leidma klastreid ning seega on nende kasutamine argumenteeritud reaalsuses.

Peatükk 4

Praktiline näide

4.1 Tööjõujälgimise tarkvara sessioonigruppide analüüs

Markovi meetodil sessioonide klasterdamise reaalseks rakendamiseks uurisime sessioonklastrite eksisteerimist AS Regio poolt toodetud tööjõujälgimise[1](toodetud *Reach-U* ettevõtte nime all) tarkvara kasutuslogidest. Antud veebirakenduses saab kasutaja jälgida reaalajas tööjõu ehk inimeste asukohta. Iga rakenduse kasutaja omab ühte eelnevalt defineeritud rolli, kokku on kasutajarolle neli. Iga rolliga on ligipääs võimaldatud erinevatele lehekülgedele, ning see väljendub ka kasutaja sessioonides.

Rakenduse haldajale on info sessioonigruppide kohta oluline kahest aspektist. Esiteks puudub haldajal selge ülevaade, kuidas kasutajad rakendust kasutavad. Teiseks on haldaja huvitatud, et rakendus oleks lihtsasti kasutatav, mille jaoks otsime enamlevinud järjestikuseid teekondi igast leitud sessioonigrupist.

Logifailidest sessioonide eraldamiseks kasutasime veebiserveri poolt määratud unikaalset sessiooni identifikaatorit, mis oli määratud automaatselt kasutaja tuvastamisel ning grupeerisime kõik kirjed sama identifikaatoriga kokku. Logifailis oleva info puhul kasutasime ära külastatud lehekülge ja kasutajatunnust, kellele sessioon kuulus. Lisaks logifailis olevale kastuajatunnusele oli teada ka info iga kasutaja rolli kohta.

4.2 Protsess

Sessioonide grupeerimise sooritasime kahes osas. Esitaks uurisime gruppide olemasolu terves sessioonikogus, kus olid esindatud kõikide rollidega kasutajad. Kuna avastasime tugeva seose kasutaja rolli ja klastrisse kuulumise vahel, uurisime lisaks klastreid kõige rohkemate sessioonide arvuga rolli sees, milleks on *company.operator*.

Klasterdamisel üritasime tuvastada ka optimaalset klastrite arvu, ning koostasime igale klastrile Markovi mudeli, et paremini mõista kasutajate käitumist. Kui klasterdasime üldistatud lehekülgedega sessioonidel ning tekkinud tsentroidid esindasid üldistatud lehekülgedega käitumist, siis klastreid iseloomustavad mudelid koostasime

[1]Tarkvara info: `http://www.reach-u.com/?op=body&id=23`, viimati kehtiv 16.05.2009

üldistamata sessioonidest. Avastasime, et üldistatud klastrimudelites ei ole piisavalt infot tegevuste täpse järjekorra kohta, kuna sageli on sama üldistatud lehekülg mitukorda järjest. Üldistamata mudelid koostasime suurima tõepära hinnanguga kõikidest sessioonidest, mis kuulusid klastrisse.

4.3 Eeltöötlus ja statistika

Sessioonimudelite koostamisel andmetest kasutasime suurima tõepära hinnangut. Eeltöötluse käigus seati sessiooni pikkuse alampiiriks 6 lehekülje, sest leidsime, et sellise pikkusega sessioonidel on juba sisulisi väärtusi. Kuna rakenduses oli ka proovikasutajaid, siis oli palju sessioone, mis piirdusid kasutajatuvastusega ning menüüst paari lehekülje valimisega.

Logifailid pärinevad ajavahemikust 2008.10.30 kuni 2009.01.21. Need sisaldavad 73979 kirjet, kokku 2360 sessiooni (ilma pikkuse järgi filtreerimata oli kõikide sessioonide arv 3079). Keskmine sessiooni pikkus on 31.3, minimaalne 6, maksimaalne 1047, pikkuste mediaan 16.0 ja standardhälve 56.0.

Nende sessioonide jooksul sooritati 51 üldistatud tegevust. Täielik tegevuste arv on 242, seega saavutatud olekuruumi vähenemine Markovi ahela mudeli korral on ligikaudu 22 kordne.

Kokku kasutas rakendust selle aja jooksul 144 erinevat kasutajat. Kasutaja kohta keskmine sessioonide arv on 16.4, minimaalne arv sessioone kasutaja kohta 1 ning maksimaalne 140.

4.3.1 Tegevuste üldsitamine

Tegevuste arvu vähendamiseks kasutasime üldistamist. Rakenduses on kasutaja tegevused antud järgmisel kujul

$$PersonAddEditWidget\#save.$$

Enne sümbolit # olev sõne märgib ära millisel leheküljel tegevus sooritati ning pärast sümbolit on tegevus ise. Ühel leheküljega on seotud tavaliselt mitu tegevust, näiteks andmete uuendamine ja kustutamine. Eeltöötluse käigus üldistasime kõik tegevused lehekülje tasemele, et vähendada tegevuste arvu, ehk eelnevast kirjest jäi alles

$$PersonAddEditWidget.$$

4.4 Meetod klastrite iseloomustamiseks

Klasterdamise tulemusel saadud klastrite iseloomustamiseks kasutame meetodit, kus kõigi klastris olevate sessioonide üldistamata kujudest koostasime suurima tõepära hinnangu alusel klastrit iseloomustava mudeli. Mudelist genereerisime kõige tõenäolisemaid tegevuste järjendeid ning sorteerisime need tõenäosuse järgi. Genereerimiseks

kasutasime laiuti otsingut mudeli üleminekute ruumi peal. Iga genereeritud jada tõenäosus avaldub kujul

$$\prod_{i=1}^{n-1} p_i,$$

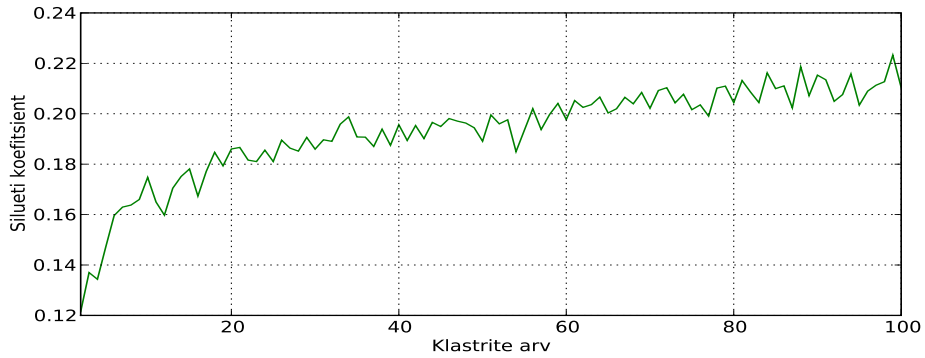
kus p_i sümboliseerib i-nda järjestikusse ülemineku tõenäosust. Kõige tõenäolisemad üleminekud annavad edasi klastrile iseloomulikke omadusi, kuna kahe erineva klastri vaheline kaugus on seda suurem, mida enam erinevad nende üleminekute tõenäosused paari kaupa. Tõenäolisemad üleminekud on kujul

$$OperatorPersonListWidget\#view- > WebMenuEvent\#map- > LastLocationTab\#refreshMapWithCenter,$$

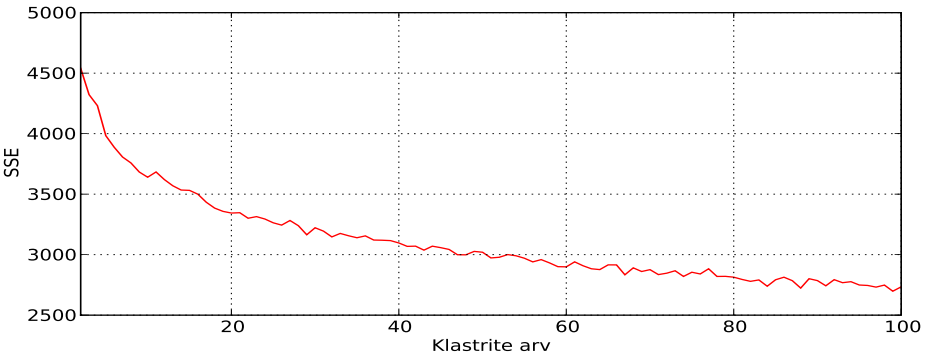
kus noolega suunas toimub kasutaja liikumine. Antud jada iseloomustab sageli esinevat mustrit, kus leheküljelt $OperatorPersonListWidget$ liiguti menüü vahendusel leheküljele $LastLocationTab$ ning sooritatid kaardi tsentreerimise tegevus $\#refreshMapWithCenter$.

4.5 Klasteradamine üle kõikide rollide

Esimese sammuna uurisime me võimalikke klastrite arvu. Selleks klasteradasime sessioonid $2 - 100$ klastriks, ning iga klasterduse korral uurisime k-keskmiste kriteeriumfunktsiooni ja silueti koefitsiendi muutust. Joonistel 4.5 ja 4.1 on näha tulemused. Väärtuse kõige suuremad muutused toimuvad k<10 korral, millest võib oletada, et kõige tõenäolisem klastri arv jääb sinna vahemikku.



Joonis 4.1: Silueti koefitsiendi muutus.



Joonis 4.2: Ruutvea väärtuse muutus.

Teades, et meil on neli erinevat rolli andmestikus, mis võivad mõjutada klasterdamise tulemust, uurisime kuidas jaotuvad erinevad kasutaja rollid klastrite vahel. Selleks valisimegi $k = 4$ ning klasterdamise tulemusel saadud puhtus kasutaja rollide suhtes on 0.89.

	company.admin	*company.operator*	*superadmin*	*wfm.admin*	Kokku
Klaster 1	0	535	0	0	535
Klaster 2	89	0	7	423	519
Klaster 3	108	700	20	30	858
Klaster 4	0	448	0	0	448

Joonis 4.3: Kasutaja rollide jaotus erinevate klastrite vahel.

	Keskmine sessiooni pikkus	Maksimaalne sessiooni pikkus	Minimaalne sessiooni pikkus	Erinevate lehekülgede arv
Klaster 1	39.51	1047	6	33
Klaster 2	20.96	179	6	26
Klaster 3	30.52	480	6	45
Klaster 4	35.2	758	6	29

Joonis 4.4: Keskmine, maksimaalne ja minimaalne sessiooni pikkus klastris, erinevate lehekülgede arv klastris.

Klastrite hulgas eristuvad kõige selgemini esimene ja viimane klaster(Joonis 4.3), mis sisaldavad ainult ühe rolli, *company.operator* sessioone. Esimese klastris paiknevate sessioonide põhitegevuseks on tööjõu asukoha määramine ning selle kaardi pealt uuendamine.

```
OperatorPersonListWidget#view-> WebMenuEvent#map-> LastLocationTab#refreshMapWithCenter
WebMenuEvent#map-> LastLocationTab#refreshMapWithCenter-> LastLocationTab#refreshMapWithCenter
OperatorPersonListWidget#requestPosition-> OperatorPersonListWidget#view-> WebMenuEvent#map
```

Neljandas klastris olevate põhitegevuste hulgas domineerivad samasugused eesmärgid, ainult selle erinevusega, et kasutaja poolt sooritatakse tegevus kasutades teisi navigeerimisteekondi ja komponente rakenduses. Inimese asukoha määramiseks kasutatakse komponenti, kus on kirjas tema viimati teadaolev asukoht. Samuti on neljandas klastris rohkem järjestikust kaardi värskendamist.

```
OperatorPersonListWidget#showOnMap-> LastLocationTab#requestPositioning-> LastLocationTab#refreshMap
LastLocationTab#requestPositioning-> LastLocationTab#refreshMap-> LastLocationTab#requestPositioning
LastLocationTab#refreshMap-> LastLocationTab#requestPositioning-> LastLocationTab#refreshMap
LastLocationTab#requestPositioning-> LastLocationTab#refreshMap-> LastLocationTab#refreshMap
```

Selle info põhjal võib rakenduses viia sisse kaks muudatust. Esiteks tuleks kasutajale võimaldada asukohamääramine ja kaardi värskendamine ühe tegevusena, sest sageli esinevad need järjestikku. Teiseks võib sisse viia lisavõimaluse automaatselt värskendada kaardinfot teatud ajaintervalli tagant.

Teise klastri, mis sisaldab peamiselt administraatori rolliga kasutajad, tegevused iseloomustavad otseselt kasutaja rolli. Domineerivad kasutajate administreerimine ning

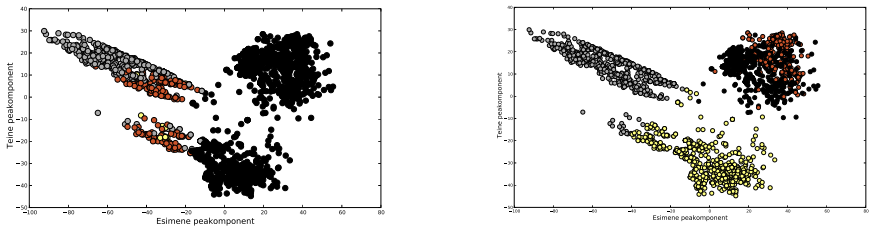
vastavate andmete uuendamine. Samuti on teise klastri puhul näha (Joonis 4.5), et kasutatava lehekülgede arv on väiksem kui ülejäänutes ning sessioonid on lühemad.

Kolmanda klastri sisu on sarnane esimesele ja neljandale, kuid selle erinevusega, et esineb rohkem menüüs liikumist ning lisaks kaardifunktsionaalsuse kasutamisele kasutatakse ka seadistuste muutmist ning rohkem pidevat kaardiandmete uuendamist.

Lisaks uurisime ka $k = 10$ korral klastrite sisu. Sellisel juhul on võrreldes eelmisega eraldunud klaster, mille sessioonid seavad automaatseid uuendusi kasutajate asukoha muutumisel, samuti on erinevad administreerimistegevused jagunenud eraldi klastrite vahel, vastavalt sellele kas uuendati kasutajate infot, kasujate ettevõtete profiili või lisati uusi kasutajaid. Kaardiinfoga seonduvad klastrid olid jagunended mitmeks, seejuures erinevates klastrites tuli esile, milliseid kasutajad kasutasid sündmuste ajalugu ja millised mitte. Lisaks eraldusid selgemalt need sessioonid, kus pidevalt värskendati kaardiandmeid. Lisas 1 on võimalus uurida klastreid $k = 10$ korral visuaalselt.

4.5.1 Peakomponentide analüüs

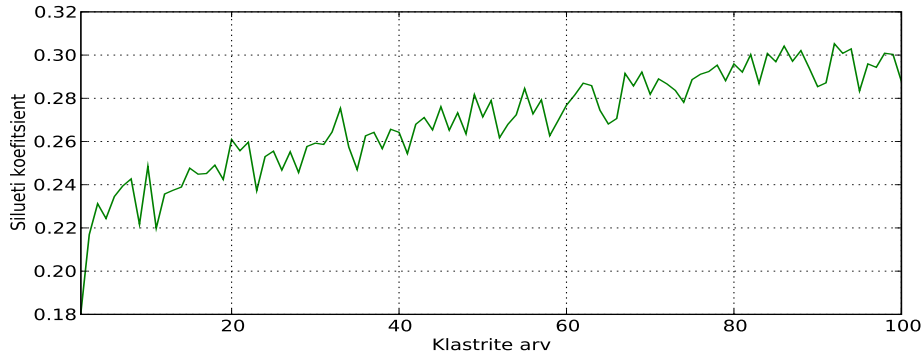
Praktikas huvitab, kas klastrite arvu on võimalik tuvastada ka lihtsamalt, kui klasterdada paljude erinevate k väärtuste korral. Üks võimalus on uurida sessioonide jagunemist gruppideks, kasutades peakomponentide analüüsi üleminekumaatriksitel. Joonisel 4.5.1 on näha kuidas erinevad kasutaja rollid eralduvad üksteisest. Diagonaalselt keskel asuvad on *wfm.admin* ja *superadmin* privileegi omavad sessioonid. See iseloomustab erinevate rollide kasutamisharjumusi, kuna nad saavad külastades erinevaid lehekülgi.



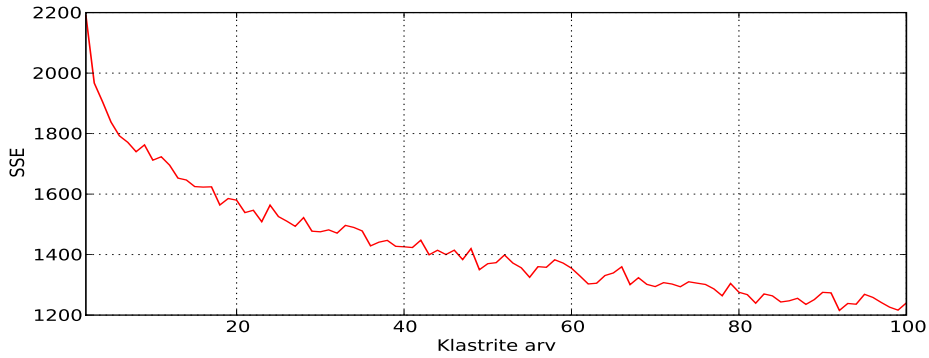
Joonis 4.5: Sessioonide kasutajarollide jaotus(a), sessioonid jaotus nelja klastrisse(b)

4.6 Klastrid ühe rolli sees

Nüüd kui oleme näinud klastrite jagunemist vastavalt tema rollide järgi, uurime millised sessioonigrupid tekivad ühe konkreetse rolli, *company.operator*, sessioonide hulgas. Vastava rolli kasutajaid on kõige enam ning rakenduse sisuline osa, asukoha määramine, ongi eelkõige mõeldud selle rolli kasutajatele. Selliseid sessioone on 1683.



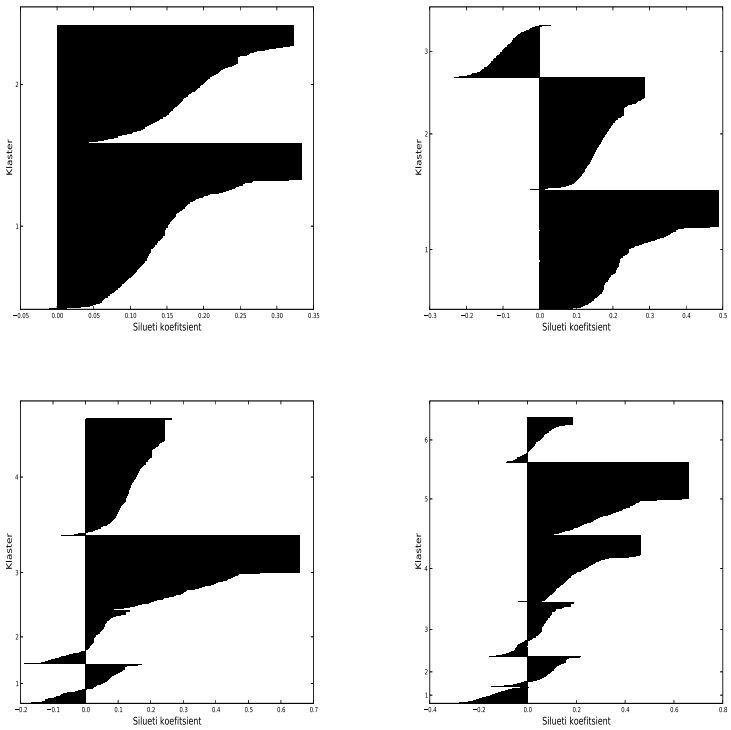
Joonis 4.6: Silueti koefitsiendi muutus.



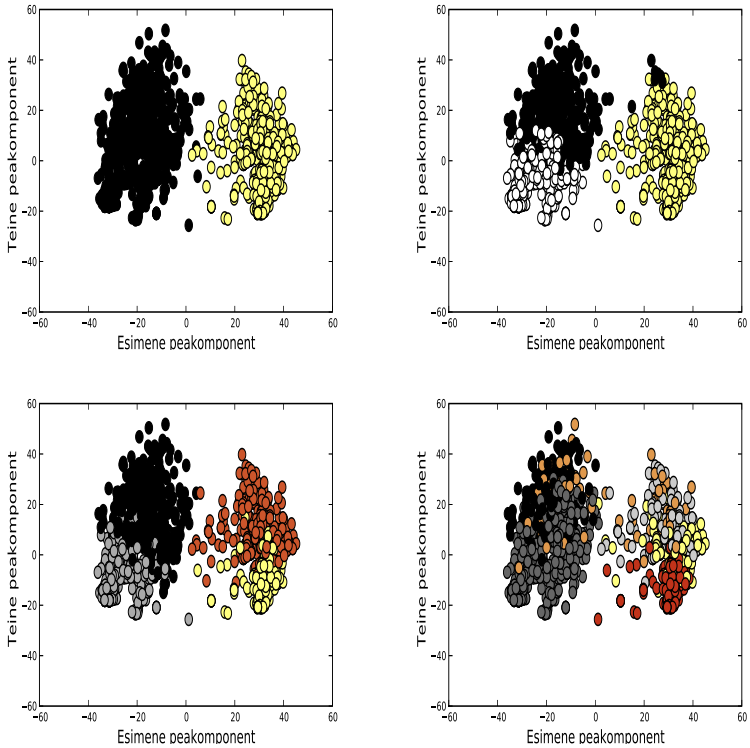
Joonis 4.7: Ruutvea väärtuse muutus.

Sarnaselt klasterdamisele kõikide rollide hulgas, on ka siin näha(Joonis 4.6, 4.7), et *company.operator* rolli hulgas toimuvad kõige suuremad muutused k väärtuste vahemikus $2 \cdots 6$. Selleks uurisime lähemalt klastreid $k = 2, 3, 4, 6$ korral.

Klastrite elementide silueti väärtused annavad paremini edasi, kuidas klastrid on jaotunud. Joonisel 4.6 on näha, et $k = 2$ korral on kõikide elementide silueti indeks suurem nullist, ning klastreid võib lugeda selgelt eristunuks, kuid kuna koefitsiendi maksimaalne väärtus on suhteliselt väike mõlema klastri korral (maksimaalselt 0.3), siis ei ole klastrid väga tugevalt eristunud. $k = 3$ korral on üks selgelt hajutatud klaster ning on ilmne, et $k = 3$ ei ole parem kui $k = 2$. Klastrite arvu suurendes suureneb ka elementide maksimaalne silueti koefitsient klastri sees(Joonis 4.8(c)), kus kolmas klaster on juba märgtavalt tugevamalt eraldunud kui ükskõik milline teine klaster $k = 2$ või $k = 3$ korral. Silueti koefitsientide info peegeldub ka peakomponentide analüüsil, kus on näha, osade k väärtuste korral pole klastrid selgelt eristunud(Joonis 4.6). Uurisime lähemalt klastreid $k = 6$ korral (Joonis 4.10).



Joonis 4.8: Silueti koefitsient kahe(a), kolme(b), nelja (c) ja kuue (d) klastri korral.



Joonis 4.9: Peakomponentide analüüs *company.operator* rollide sessioonidel jagatuna kahte klastrisse(a), sessioonid jagatuna kolme klastrisse(b), sessioonid jagatud nelja klastrisse (c), sessioonid jagatud kuueks klastriks (d). Tulemused on analoogselt silueti koefitsiendi väärtustele, kus on näha, et negatiivse koefitsientidega klastrid on peitunud teiste klastrite sisse.

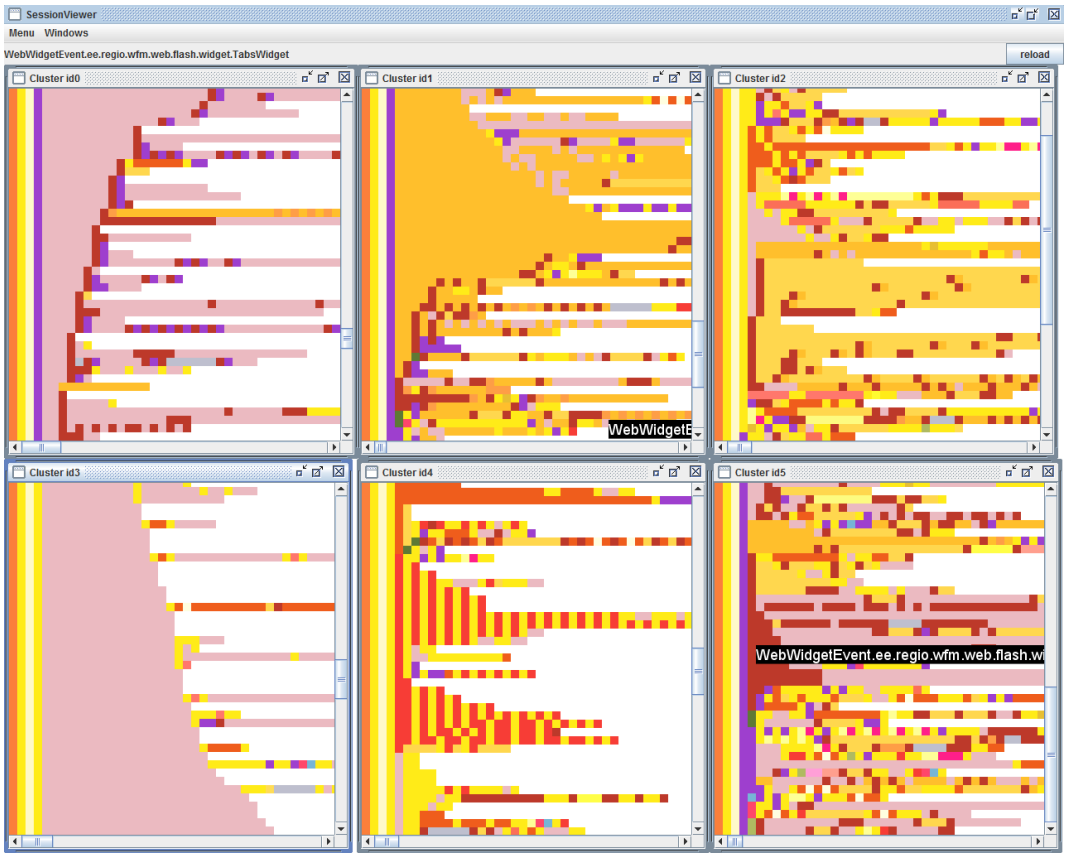
4.6.1 Klastrite iseloomulikud jooned

	Suurus	Iseloomulikud jooned
Klaster 1	667	Pidev kaardiandmete uuendamine ja tööjõu positsioneerimine kasutades $\#refreshMap$ ja $\#requestPositioning$
Klaster 2	205	Kaardi andmete värskendamisel kasutatakse $\#jsDataNotification$, mis uuendab andmeid ilma tervelt lehekülge uuendamata. See on puhtalt kasutaja eelistus, ning pigem tehniline eriärasus.
Klaster 3	73	Andmete ajaloo uurimine kasutades $\#eventHistory$ ning vastavate andmete kuvamine kaardil.
Klaster 4	343	Enamlevinud on teekond $\#requestPositioning$ -> $\#refreshMap$. Erinevus esimese klastriga on, et ei kasutada tegevuste jadas funktsiooni $\#showOnMap$, vaid eelpool loetud kaht, kus juures $\#requestPositioning$ ja $\#refreshMap$ on tugevalt järjestatud, ning väga sageli järgneb positsioneerimisele kaardi värskendamine.
Klaster 5	293	Klastrit iseloomustab kõigeparemini $\#notifications$ ja $\#showMap$ järjestikune kasutamine
Klaster 6	102	Sessioonid koosnevad Esimese klastri ja kolmanda klastri tegevustest, sisaldades lisaks sündmuste ajaloo vaateid ning uute andmete pärimist.

Joonis 4.10: Klastrid $k = 6$ korral kasutajarolli *company.operator* jaoks

4.6.2 Sessioonide visuaalne inspekteerimine

Analüüsisime ka veebisessioonide klastrites asetsemist visuaalselt kasutades [CHM+00] tutvustatud meetodit. Visuaalsel inspekteerimisel avastasime, et eelpool koostatud klastrid sisaldavad ka palju selliseid sessioone, mis on pea ühe tegevuse kordus terve sessiooni vältel.



Joonis 4.11: Veebisessioonide visuaalne esitus klastrites. Esimesel klastris on näha roosa värviga *LastLocationTab* sagedased tegevused. Teises klastris on domineeriva värviga kaardiandmete uuendamine. Kolmandal klastril kõige enamlevinud värviga sündmuste ajaloo vaatamine. Neljas klaster on analoogne esimesele, kuid erinevad viimased vaadatud leheküljed sesioonides. Viienda klastriga tuleb välja antud meetodi eelis - visuaalselt on võimalik tuvastada korduvaid mustreid. Kollase ja punasega vahelduvad tegevused iseloomustavad tegevusi $\#notifications- > \#showMap$. Viimase klastri puhul on värvide jaotuse järgi näha, et sessioonid omavad esimese ja kolmanda klastri sessioonide omadusi.

Lisaks oli visuaalsel uurimisel näha, kuidas sessioonide pikkused kasvasid iga külastuskorraga. Joonisel 4.11 on sessioonid klastrites sorteeritud kasutaja järgi ajaliselt, ehk eespool asuvad varem sooritatud sessioonid ning ühe kasutaja sessioonid on järjest.

4.7 Järeldused

Klasterduse analüüsi tulemusel leidsime paar ilmset tulemust: seose kasutaja külastatud lehekülgede (ehk kasutaja rolli) ja klastrite vahel. Sisuliste tulemuste korral avastasime, et sageli on kasutaja sunnitud kaardi andmete uuendamiseks sooritama järjestikku mitu tegevust, mida saaks tegelikult viia ühe tegevuse alla. Lisaks eksisteerib tööjõu asukoha uuendamiseks mitu võimalust, mis võib tekitada segadust eelkõige uute kasutajate seas.

Hea ülevaate klastritest andis sessioonide visuaalne esitus, mille korral on võima-

lik leida lihtsalt huvitavaid mustreid. Visuaalne inspekteerimine on mõeldav praeguse andmemahu juures, kuid andmemahtude ja klastrite arvu kasvades muutub see keerulisemaks. Lisaks nägime, et kahe tähtsama peakomponendi kujutamisel tasandil võib leida info klastrite arvu hindamiseks.

Kokkuvõte

Töö eemärgiks oli uurida veebisessioonide grupeerimise võimalusi kasutades selleks veebirakenduse logifaile ning leida universaalne, lihtsasti kasutatav algoritm.

Olemasolevatest lahendustest uurisime lähemalt meetodit, mis kasutab kasutajakäitumise modelleerimiseks Markovi ahelate mudeleid, ning klasterdamiseks k-keskmiste algoritmi. Sellise lähenemise eeliseks teiste lahenduste ees on mudeli võime salvestada kasutaja tegevuste järjekorda, samuti loob Markovi ahelate kasutamine veebisessiooni modelleerimisel hea vahendi tüüpprofiilide interpreteerimiseks. Meetodi hindamiseks kasutasime selleks spetsiaalselt koostatud tehisandmestikku, uurides kui täpselt leiab lahendus andmestikus olevad klastrid. Tehisandmestikel kaotab meetod täpsust klastrite arvu kasvadas. Meetodi plussiks on võime töötada ka suurema olekuruumi ehk keeruliste veebilehekülgedega, kus on palju erinevaid võimalikke tegevusi. Võrreldes muude lahendustega ei anna meetodi töö alati täpsuselt kõige paremaid tulemusi, jäädes alla hulkade sarnasusel baseeruvale meetodile, mis kasutab sessioonide vahelise kauguse mõõtmiseks *Jaccard* mõõtu.

Analüüsitud meetodi abil uuritakse sessioonigruppide olemasolu tööjõujälgimise tarkvara kasutajate hulgas. Näidatakse, et on olemas erinevad sessioonigrupid, kelle huvid on erinevad. Tulemuste visualiseermisel rakendatakse peakomponentide analüüsi Markovi ahelate üleminekutemaatriksitel ning koostatakse mudelist kõige tõenäolisemaid kasutajate tegevuste järjestusi. Jõutakse järeldusele, et sessioonide visuaalne interpreeterimine aitab paremini mõista sessiooni gruppide tulemusi. Sisuliste tulemusena leiti, et tarkvaras on tegevusi, mida saaks lahendada ühe kombineeritud tegevusena, mis hetkel nõuavad kasutajalt mitut eraldi tegevust.

Clustering web sessions based on user navigation

Bachelor thesis

Riivo Kikas

Summary

To understand behaviour and aims of a user in a web application, it is possible to group web sessions by similarity. Clusters of similar sessions can describe user interests and combining this with most frequently used navigation paths for each cluster, we can gain insight into actions and preferences of users. This provides information for a website owner how to redesign software, so that users cant find easily what they need.

The aim of this thesis is to study web session clustering problem using log files and to find a simple and reasonably universal algorithm for it. An overview of existing solutions is presented in the thesis.

Author investigates previously used method for web session clustering that uses Markov chains for modelling user navigation and k-means algorithm for clustering. This method provides a way to capture order of events into Markov chain model, also using Markov chains presents a possibility to describe a cluster of sessions with a single model, that captures properties of all sessions. An experiment is carried out to see if the method can find existing clusters in an artificial dataset. It is observed that accuracy of found clusters decreases with growing number of clusters. A benefit of the method is the ability to perform with web page count nearing to 100. A comparison is presented with existing methods and results show that given method does not always give best results. Jaccard distance based similarity measure performs better in terms of purity of found clusters in data set.

The method is applied to real world web log data that originates from workforce management software. It is shown that different groups of users are present with different behaviours. Results are visualised using principal component analysis, also most frequent user navigation paths are found based on the Markov model that captures properties of all items in a cluster. It is concluded that visualizing web sessions helps to understand its content. It is discovered that changes can be made into software's user interface to make it more simpler and substitute several sequential actions with a single one.

Kirjandus

[BG01] Arindam Banerjee and Joydeep Ghosh. Clickstream clustering using weighted longest common subsequences. In *in: Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pages 33–40, 2001.

[BKM+00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, pages 309–320, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[BL04] Jose; Borges and Mark Levene. A dynamic clustering-based markov model for web usage mining. *http://arxiv.org/abs/cs/0406032v1*, Jun 2004.

[BMMM05] Andrea Bianco, Gianluca Mardente, Marco Mellia, and Maurizio Munafo. Web user session characterization via clustering techniques. *Globecom 2005 Autonomic Internet*, nov 2005.

[Bre01] Pierre Bremaud. *Markov Chains*. Springer, January 2001.

[CHM+00] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigation patterns on a web site using model-based clustering. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–284, New York, NY, USA, 2000. ACM.

[DEKM99] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1999.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA, 1999. ACM.

[FL05] Federico Michele Facca and Pier Luca Lanzi. Mining interesting knowledge from weblogs: a survey. volume 53, pages 225–241, Amsterdam, The Netherlands, The Netherlands, 2005. Elsevier Science Publishers B. V.

[FSyS99] Yongjian Fu, Kanwalpreet Sandhu, and Ming yi Shih. Clustering of web users based on access patterns. In *In Proceedings of the 1999 KDD Workshop on Web Mining*. Springer-Verlag, 1999.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, January 1997.

[HK00] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, September 2000.

[HNC+01] Zhexue Huang, Joe Ng, David W. Cheung, Michael K. Ng, and Wai ki Ching. A cube model for web access sessions and cluster analysis. In *In Proceedings of the 3rd international workshop on mining web data (WEBKDD 2001*, pages 48–67. Springer-Verlag, 2001.

[JJK00] Anupam Joshi, Karuna Joshi, and Raghu Krishnapuram. On mining web access logs. In *In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 63–69, 2000.

[Juh03] Mihhail Juhkam. Klasterdamine andmekaevanduses. *http://www.egeen.ee/u/vilo/edu/2003-04/DM_seminar_2003_II/ver1/P05/Clustering.pdf*, *Viimati vaadatud 16. mai 2009*, 2003.

[Kav09] Lydia E. Kavraki. Dimensionality reduction methods for molecular motion. *http://cnx.org/content/m11461/latest/*, Viimati vaadatud: 16. mai 2009.

[LAV05] George Pallis Lefteris, Lefteris Angelis, and Athena Vakali. Model-based cluster analysis for web users sessions. In *In Proceedings of the 15th international symposium on methodologies for intelligent systems (ISMIS 2005*, pages 219–227. Springer-Verlag, 2005.

[MAFM99] Daniel A. Menascé, Virgilio A. F. Almeida, Rodrigo Fonseca, and Marco A. Mendes. A methodology for workload characterization of e-commerce sites. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 119–128, New York, NY, USA, 1999. ACM.

[mCC01] Hui min Chen and Michael D. Cooper. Using clustering techniques to detect usage patterns in a web-based information system. *Journal of the American Society for Information Science and Technology*, 52:888–904, 2001.

[MCS00] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[QYH05] Aijun An Qingsong Yao and Xiangji Huang. *Database Systems for Advanced Applications*, volume Volume 3453/2005 of *Lecture Notes in Computer Science*, chapter Finding and Analyzing Database User Sessions, pages 851–865. Springer Berlin / Heidelberg, 2005.

[Shl05] Jonathon Shlens. A tutorial on principal component analysis. In *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.

[SLdCT06] Alzennyr Silva, Yves Lechevallier, Francisco de Carvalho, and Brigite Trousse. Mining web usage data for discovering navigation clusters. In *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications*, pages 910–915, Washington, DC, USA, 2006. IEEE Computer Society.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 1 edition, May 2005.

[WME04] Qing Wang, Dwight J. Makaroff, and H. Keith Edwards. Characterizing customer groups for an e-commerce website. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 218–227, New York, NY, USA, 2004. ACM.

[WZ02] Weinan Wang and Osmar R. Zaiane. Clustering web sessions by sequence alignment. In *In Proceedings of the 13th international workshop on database and expert systems applications (DEXA 2002). Aix-en-Provence*, pages 394–398. Springer-Verlag, 2002.

[YJGMD96] Tak Woon Yan, Matthew Jacobsen, Hector Garcia-Molina, and Umeshwar Dayal. From user access patterns to dynamic hypertext linking. In *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, pages 1007–1014, Amsterdam, The Netherlands, The Netherlands, 1996. Elsevier Science Publishers B. V.

Lisad

Lisa 1. DVD algoritmide implementatsioonidega ja sessioonide visualiseerimise tarkvaraga.