Kohosen itseorganisoituvasta kartasta tekstidokumenttien tiedonhaussa

Jyri Saarikoski

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Helmikuu 2007

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Jyri Saarikoski: Kohosen itseorganisoituvasta kartasta tekstidokumenttien tiedonhaussa
Pro gradu -tutkielma, 63 sivua
Helmikuu 2007

Tässä tutkielmassa tarkastellaan tekstidokumenttien tiedonhakua itseorganisoituvan kartan (self-organizing map, SOM) avulla. Itseorganisoituva kartta on Teuvo Kohosen kehittämä neuroverkkomalli, jossa verkolle annetut syötteet organisoituvat kartaksi ohjaamattoman oppimisen seurauksena. Mallia on tähän mennessä sovellettu tiedonhaussa enimmäkseen erilaisten dokumenttijoukkojen järjestämiseen ja luokitteluun, mutta hakusovelluksissa sen käyttäminen on ollut melko vähäistä. Tutkielman tavoitteena oli kehittää hakujärjestelmä, jossa tekstidokumentit järjestetään itseorganisaation avulla kartaksi, jolta dokumenttien hakeminen olisi tehokasta.

Itseorganisoituvan kartan syötteisiin, alustamiseen ja opettamiseen liittyy useita muuttujia, jotka saattavat vaikuttaa lopputuloksena syntyvän kartan laatuun. Tutkielmassa on pyritty selvittämään näiden kartan luomiseen liittyvien muuttujien vaikutusta tiedonhakutilanteessa. Kehitettyä järjestelmää testattiin luomalla sen avulla 1160 saksankielistä uutisdokumenttia sisältävä dokumenttikartta ja hakemalla kartalta dokumentteja 20 eri aiheeseen liittyen. Aluksi luotiin kartta perusasetuksilla, jota testattiin sitten asetuksia muuttamalla. Tulosten perusteella luotiin muuttuja-asetuksiltaan paranneltu kartta, jota verrattiin sitten perusasetuksilla luotuun karttaan.

Kehitetyn itseorganisoituvaan karttaan perustuvan hakukoneen suoriutuminen tekstidokumenttien tiedonhaussa osoittautui vähintäänkin kelvolliseksi. Testaamisen tuloksena sen suorituskykyä pystyttiin myös tehostamaan, mikä osoittaa, että kartan luomiseen liittyvillä muuttujilla on merkitystä tiedonhaun kannalta. Lisäksi järjestelmän tuottama kaksiulotteinen dokumenttikartta antaa tiedonhakijalle mahdollisuuden selata järjestettyä dokumenttijoukkoa intuitiivisella tavalla, mikä tuo merkittävästi lisäarvoa itseorganisoituvan kartan käyttämiselle tiedonhaussa.

CR-luokat: I.2.6 [**Artificial Intelligence**]: Learning – *Connectionism and neural nets*; H.3.3. [**Information Storage and Retrieval**]: Information search and retrieval - *Retrieval models.*

Avainsanat ja -sanonnat: neuroverkko, itseorganisoituva kartta, SOM, Kohosen kartta, tiedonhaku, tekstitiedonhaku.

Sisällys

1. Johdanto

Nykyaikainen tietoyhteiskunta tuottaa jatkuvasti dokumentoitua informaatiota mitä erilaisimpiin aihepiireihin liittyen, mistä johtuen lähes jokaiseen tiedonhakijan eteen tulevaan tiedontarpeeseen on olemassa useita hyviä vastauksia dokumentoidussa muodossa. Samalla informaatiotulva kuitenkin tekee entistä vaikeammaksi tarvittavan informaation löytämisen. Tähän perusongelmaan etsii vastausta tiedonhaun tutkimus [Salton and McGill, 1983], jossa pyritään kehittämään erilaisia menetelmiä relevanttien tietoalkioiden löytämiseksi suuresta joukosta tietoalkioita.

Tiedon etsimisestä on Internetin nopean leviämisen myötä tullut arkipäiväinen osa elämää sekä työssä että vapaa-ajalla. Ennen tietoa hakivat enimmäkseen opiskelijat kirjastosta, mutta nykyään tiedonhaun osaamista tarvitsevat lähes kaikki ja tietoa etsitään kirjaston lisäksi myös työpaikalla ja kotona tietokoneen välityksellä verkosta. Voitaneenkin siis todeta, että relevantin tiedon löytäminen on yhä tärkeämmässä osassa ihmisten jokapäiväistä elämää.

On selvää, että kasvava informaation määrä ja tiedon hakemisen muuttuminen arkipäiväiseksi osaksi elämää aiheuttavat suuren tarpeen kehittää entistä tehokkaampia menetelmiä tiedonhakuongelmien ratkaisemiseen. Yksi perinteisestä poikkeava lähestymistapa on soveltaa tekoälyn ja koneoppimisen menetelmiä tiedonhaussa. Neuroverkot [Haykin, 1994] ovat neurolaskentaan perustuvia tekoälymenetelmiä, joissa tietokone suorittaa laskentaa eräässä mielessä ihmisen aivojen toimintaa mallintavalla tavalla pyrkien rinnakkaisuuden ja laskennan hajauttamisen avulla tehokkuuteen, mikä tekee niistä potentiaalisen vaihtoehdon perinteisempien menetelmien mahdollisina korvaajina.

Yksi mielenkiintoisimmista neuroverkkomenetelmistä on suomalaisen Teuvo Kohosen kehittämä itseorganisoituva kartta [Kohonen, 1995], joka pystyy luomaan moniulotteisista syötearvoista kaksiulotteisen kartan pyrkien säilyttämään syötejoukon sisäiset suhteet etäisyyksinä kartalla. Kartan luomista varten tekstidokumentit koodataan numeeriseen muotoon ja annetaan syötteenä kartalle, joka itseorganisaation kautta järjestää dokumentit pinnalleen luoden dokumenttikartan koko dokumenttijoukosta. Kohosen menetelmää on sovellettu paljon erilaisten tekstidokumenttijoukkojen automaattiseen järjestämiseen ja luokitteluun, mihin sen luoma kaksiulotteinen kartta sopiikin mainiosti.

Tässä tutkielmassa tarkastellaan sitä, miten hyvin Kohosen itseorganisoituva dokumenttikartta pystyy säilyttämään dokumenttien suhteet niiden käsittelemiin aiheisiin liittyen. Tutkimusta varten kehitettiin hakukoneen prototyyppi, joka luo annetuista tekstidokumenteista dokumenttikartan, jolta sitten voidaan hakea eri aiheisiin liittyviä dokumentteja sanahaulla. Koska Kohosen menetelmä pyrkii säilyttämään syötteiden välillä esiintyvät suhteet, voidaan olettaa, että samaa aihetta käsittelevät dokumentit

sijoittuvat dokumenttikartalla lähelle toisiaan, jolloin dokumenttikarttaa voitaisiin hyödyntää etsimällä annettuun hakuun parhaiten täsmäävä sijainti kartalta ja palauttamalla dokumentit, jotka sijaitsevat kyseisessä sijainnissa tai sen välittömässä lähiympäristössä. Hieman yllättäen itseorganisoituvaa karttaa ei ole aiemmin kovinkaan paljon sovellettu tähän tarkoitukseen, mikä olikin yksi tämän tutkimuksen suurimmista motivaatiotekijöistä. Tutkielmassa on pyritty tarkastelemaan tiedonhakutilannetta mahdollisimman todenmukaisessa tapauksessa, joten dokumenttikartta luotiin 1160 uutisdokumentista ja kyselyinä käytettiin suhteellisen lyhyitä kyselyitä, joissa oli keskimäärin alle 20 sanaa.

Tutkielman varsinainen sisältö koostuu aiheeseen liittyvän teoreettisen taustan läpikäymisestä, hakukoneprototyypin kehittämisen kuvauksesta, hakukoneen testaamisen esittelystä, tulosten tarkastelusta sekä loppuyhteenvedosta. Samalla on tavoitteena antaa lukijalle yleiskuva siitä, miten itseorganisoituvaa karttaa on tähän mennessä sovellettu tekstitiedonhaussa, sekä pohtia, miten sitä voitaisiin kenties jatkossa hyödyntää.

Tutkielman toinen luku esittelee neuroverkkoihin liittyvät peruskäsitteet ja tarkastelee neuroverkkojen luokittelua niiden ominaisuuksien perusteella. Tavoitteena on kuvata neuroverkkojen taustalla olevat biologiset ja tekoälyyn liittyvät lähtökohdat lyhyesti sekä esitellä kaikille neuroverkoille yhteiset ominaisuudet.

Kolmannessa luvussa tarkastellaan lähemmin Kohosen itseorganisoituvaa karttaa neuroverkkomenetelmänä sekä esitellään sen erityispiirteet. Yleisten ominaisuuksien esittelyn jälkeen syvennytään syötteiden esiprosessointiin sekä kartan alustamiseen ja oppimisvaiheeseen. Lopuksi pohditaan valmiin kartan ominaisuuksia.

Neljäs luku keskittyy tiedonhaun teoriaan. Esitellään tiedonhaku yleisesti sekä siihen liittyen perinteiset täsmäytysmenetelmät, vektoriavaruusmalli sekä tiedonhakumenetelmien evaluoinnin perusteet.

Viidennessä luvussa pohditaan puolestaan itseorganisoituvan kartan soveltamista tiedonhakuun. Esitellään menetelmän soveltamisen edut ja toisaalta suurimmat ongelmakohdat, jotka ovat kriittisessä asemassa lopputuloksen kannalta. Lisäksi tarkastellaan, miten itseorganisoituvaa karttaa on aiemmin sovellettu tiedonhaun ongelmiin. Tarkempana esimerkkinä tutkitaan WEBSOM-järjestelmää [Lagus *et al.*, 2004]. WEBSOM on itseorganisoituvaan karttaan perustuva ohjelmisto, joka toimii suurten dokumenttijoukkojen selaamiseen tarkoitettuna työkaluna. Sen toiminta perustuu dokumenttikartan kykyyn sijoittaa samanlaiset dokumentit lähelle toisiaan. Joissain sovelluksissa WEBSOMiin on liitetty myös sanahakuominaisuus, mutta varsinaisesti sen toiminta perustuu siihen, että käyttäjä selaa karttaa sen tarjoamien vihjesanojen avulla. WEBSOM on kenties merkittävin itseorganisoituvien karttojen sovellus tiedonhaussa ja lisäksi se on erittäin tärkeä vertailukohta tämän tutkimuksen kannalta.

Kuudennessa luvussa esitellään luodun hakukoneprototyypin kehitysprosessi päävaiheineen sekä prototyypin ominaisuudet. Lisäksi tarkastellaan hakukoneen

testaamisessa käytettyä dokumenttiaineistoa sekä esitellään testausmenetelmä. Aineisto koostui kaikkiaan 1160 saksankielisestä uutisdokumentista. Testiaineistona käytettiin yhteensä 20 kyselyä, joihin liittyen dokumenttiaineistossa tiedettiin olevan yhteensä 580 relevanttia tekstidokumenttia.

Seitsemännessä luvussa esitellään hakukoneen testaamisen tärkeimmät tulokset sekä analysoidaan niitä tarkemmin. Luvussa pyritään myös vertailemaan tuloksia aiempaan tutkimukseen alalla.

Viimeisessä luvussa kootaan tutkimuksen tulokset yhteen ja pohditaan mahdollisuuksia aiheen jatkotutkimukseen.

2. Neuroverkot

Neurolaskenta (neural computing) on ihmisen aivojen oletettua toimintatapaa jäljittelemään pyrkivä tietokonelaskentamenetelmä. Ihmisaivot kykenevät luonnostaan erittäin tehokkaaseen rinnakkaisprosessointiin ja laskennan hajauttamiseen. Toisaalta ihmisellä on myös kyky yleistää oppimaansa hyvin. Nämä molemmat ominaisuudet olisivat luonnollisesti toivottavia piirteitä myös tietokoneiden ominaisuuksina, mutta perinteisesti tietokoneet eivät ole pystyneet kilpailemaan aivojen kanssa näissä ominaisuuksissa lainkaan. Neurolaskennassa ihmisaivojen toimintaa jäljitellään *neuroverkkojen* (neural networks) avulla. Haykin [1994] kuvailee neuroverkkoa yleisesti koneeksi, joka on suunniteltu mallintamaan aivojen toimintatapaa tietyssä tehtävässä.

2.1. Biologinen perusta

Ihmisaivojen biologinen rakenne on toiminut neuroverkkomenetelmien kehittämisen innoittajana. Aivojen perusyksikkö on *hermosolu* eli *neuroni* (neuron). Hermosolut ovat yhteydessä toisiinsa ja viestivät keskenään näiden yhteyksien kautta. Hermosolun perusosat ovat solukeskus, dendriitit, aksoni ja synapsit.

Dendriitit toimivat tiedonkuljetuskanavina ja ne välittävät hermosoluun saapuvan sähköimpulssin muodossa olevan tiedon solukeskukseen. *Aksoni* puolestaan kuljettaa solukeskuksesta eteenpäin lähtevän tiedon muille hermosoluille. *Synapsit* ovat hermosolujen välisiä liitospisteitä, jotka liittävät edellisen hermosolun aksonin seuraavien hermosolujen dendriitteihin. Ne kontrolloivat tiedon liikkumista eteenpäin hermosolujen välillä päästäen impulssin läpi vasta, kun hermosolun saama syöte ylittää tietyn raja-arvon eli kun syöte on tarpeeksi voimakas, jolloin hermosolu laukeaa ja impulssi siirtyy eteenpäin. Biologisessa neurosysteemissä oppiminen tarkoittaa muutosten syntymistä synapseissa ja sitä kautta tiedon liikkumisessa.

Vaikka biologinen perusta onkin innoittanut suuresti neurolaskentamenetelmien kehittämistä, ovat ne kuitenkin käytännössä niin etäällä alkuperäisistä lähtökohdistaan, ettei niiden käyttäjän tarvitse tietää mitään aivojen rakenteesta. Aivojen biologian tarkempi tarkastelu ei ole tämänkään tutkielman kannalta oleellista.

2.2. Neuroverkon rakenne

Erotuksena biologiseen neuroverkkoon neurolaskennan yhteydessä puhutaan usein keinotekoisista neuroverkoista (artificial neural networks). Käytännössä kuitenkin keinotekoisia verkkoja nimitetään yksinkertaisesti neuroverkoiksi, koska sekaannuksen vaaraa ei yleensä ole. Keinotekoisen neuroverkon rakenne jäljittelee edellä kuvattua aivojen perusrakennetta. Biologinen neuroverkko on malli siitä, miten aivojen uskotaan toimivan. Neuroverkon rakenne on puolestaan yksinkertaistus tästä mallista.

Ennen neuroverkkojen tarkempaa käsittelyä lienee paikallaan esitellä niiden niiden luonnetta yleisesti ja vapaamuotoisesti. Neuroverkot ovat tietokonelaskentamenetelmiä,

joiden avulla voidaan toteuttaa monenlaisia prosessointitehtäviä. Yksinkertaisimmillaan verkolle annetaan jokin syöte, ja verkko antaa syötteen seurauksena tuloksen. Usein verkolle annetaan myös useita syötteitä, joiden avulla neuroverkko muuttaa sisäistä tilaansa opetellen niiden ominaisuuksia, minkä jälkeen verkko pystyy yleistämään oppimaansa ja antamaan myös aiemmin tuntemattomille syötteille oikeita tuloksia. Yksi tärkeimmistä ehdoista on se, että syötteiden on oltava numeerisia, jotta ne ovat neuroverkon ymmärrettävissä ja laskettavissa. Luonnollisesti myös tulosteet ovat numeerisia. Neuroverkolle annettavat syötteet on siis ensin muutettava numeeriseen muotoon, mikä saattaa usein olla jo oma ongelmansa.

Seuraavassa perehdytään keinotekoisen neuroverkon perusrakenteeseen ja toimintaan pääosin Haykinin [1994] ja Rojaksen [1996] esitysten pohjalta. Ensin tarkastellaan yksittäisen neuronin yleistä mallia ja sitten neuroneista koostuvien verkkojen perusarkkitehtuureja.

2.2.1. Neuronin yleinen malli

Neuroverkkojen perusyksikkö on neuroni, joka on saanut nimensä biologiselta hermosolulta. Sen rakenteen pääosan muodostavat syötekanavat, tuloskanava, summain sekä aktivaatiofunktio. Syötekanavat ja tuloskanava mallintavat biologisten neuroverkkojen dendriittejä ja aksonia toimien siis yksittäisten neuronien välissä tiedonkuljetusväylinä. Aktivaatiofunktio laskee neuroniin tulevan kokonaissyötteen perusteella neuronin tuloksen eli aktivaation. Jokaiseen yksittäiseen syötteeseen liittyy painoarvo, joka kuvaa saapuvan syötteen voimakkuutta. Kokonaissyötteen yksittäisistä syötteistä tuottaa summain, joka yksinkertaisesti laskee yhteen neuronin saamat painotetut syötteet. Neuroniin liittyy myös vakiosyöte, jota kutsutaan myös siirtotermiksi, koska sen avulla voidaan säädellä, tavallaan siis siirtää, neuronin aktivoitumisherkkyyttä.

Edellä kuvattua neuronimallia on havainnollistettu kuvassa 2.1. Kuvassa on esitelty yleiseen neuroniin k liittyvät rakenteet. Neuroni saa syötteet $x_0,x_1,\ldots,x_p$, jotka kerrotaan kukin vastaavalla painoarvoilla $w_{k0},w_{k1},\ldots,w_{kp}$ ja lasketaan sen jälkeen yhteen. Matemaattisesti neuronin k kokonaissyöte u_k voidaan esittää yksikertaisella kaavalla

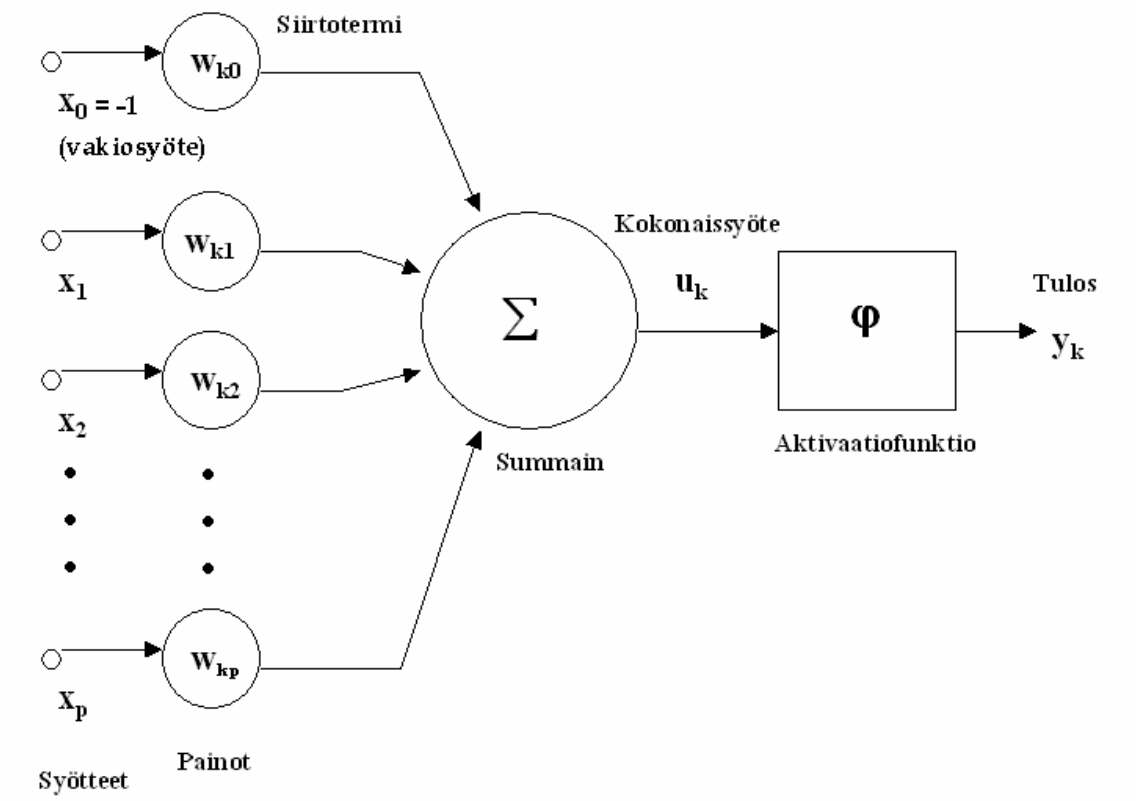
$$u_k = \sum_{j=0}^{p} w_{kj} x_j \ , \tag{2.1}$$

missä p on neuronin saamien syötteiden lukumäärä, w_{kj} syötekanavan j painoarvo ja x_j syötekanavaan j liittyvä syöte. Kaavan 2.1 tuottama tulos u_k on siis neuronin kokonaissyöte, jonka summain laskee saapuvista syötteistä. Aktivaationfunktio laskee neuronin lopullisen tuloksen summaimen tuottaman kokonaissyötteen perusteella, mikä voidaan esittää kaavalla

$$y_k = \varphi(u_k), \tag{2.2}$$

missä φ on aktivaatiofunktio ja u_k on neuroniin k liittyvä kokonaissyöte.

Kaavojen 2.1 ja 2.2 kohdalla on syytä huomata, että siirtotermiksi kutsuttu tekijä eli vakiosyöte x_0 ja siihen liittyvä painoarvo w_{k0} sisältyvät tässä esitystavassa mukaan kokonaissyötteeseen u_k. Kuvassa 2.1 on vakiosyötteelle x_0 annettu kiinnitetty arvo -1, jolloin painon w_{k0} arvon asettamisen avulla voidaan pienentää neuronin kokonaissyötettä, mikä johtaa aktivaation siirtymiseen, kun muilta syötekanavilta tarvittava kokonaissyöte neuronin laukaisemiseksi kasvaa, mikä onkin siirtotermin varsinainen tarkoitus.



Kuva 2.1. Neuronin *k* perusrakenne.

Aktivaatiofunktio siis tuottaa summaimen laskemasta kokonaissyötteestä neuronin tuloksen, jonka tuloskanava kuljettaa eteenpäin verkossa mahdollisesti seuraavan neuronin syötteeksi. Käytännössä aktivaatiofunktio määrääkin siis koko neuronin aktivoitumisen, sillä jos funktio antaa tulokseksi 0, ei neuroni anna ollenkaan tulosta, eli neuroni ei aktivoidu. Haykinin [1994] mukaan aktivaatiofunktiot rajaavat yleensä neuronin antaman tulosimpulssin reaalilukuvälille [0,1] tai [-1,1]. Haykin jakaa erilaiset aktivaatiofunktiot kolmeen eri ryhmään: askelfunktiot, paloittain lineaariset funktiot ja sigmoidifunktiot.

Askelfunktio on hyvin yksinkertainen ja klassinen ratkaisu aktivaatiofunktioksi. Se määritellään kaavalla

$$\varphi(v) = \begin{cases} 1, & jos \quad v \geq 0 \\ 0, & jos \quad v < 0 \end{cases}, \tag{2.3}$$

missä v on summaimen laskema kokonaissyöte. Askelfunktio muuntaa siis saamansa kokonaissyötteen neuronin tulosteeksi yksinkertaisesti siten, että positiivinen kokonaissyöte muuntuu tulokseksi 1 ja muuten tulos on 0.

Paloittain lineaarinen funktio on hieman monipuolisempi tapa määritellä neuronin aktivaatio. Määrittely tapahtuu kaavalla

$$\varphi(v) = \begin{cases} 1, & jos \quad v \geq \frac{1}{2} \\ v + \frac{1}{2}, & jos \quad -\frac{1}{2} < v < \frac{1}{2} \\ 0, & jos \quad v \leq -\frac{1}{2} \end{cases}, \tag{2.4}$$

missä v on neuronin saama kokonaissyöte. Paloittain lineaarinen vaihtoehto antaa siis joustavamman tavan määritellä neuronin aktivaatio, sillä tulosten 0 ja 1 lisäksi se antaa myös tuloksia niiden väliltä. Jos funktion lineaarista keskimmäistä osaa pienennetään siirtämällä sen rajoja lähemmäs toisiaan, lähestytään askelfunktiota. Suurentamalla lineaarista osaa lähestytään puolestaan täysin lineaarista aktivaatiofunktiota.

Kolmas vaihtoehto on sigmoidifunktio, joka on yleisin aktivaatiofunktio neuroverkkosovelluksissa [Haykin 1994]. Se määritellään kaavalla

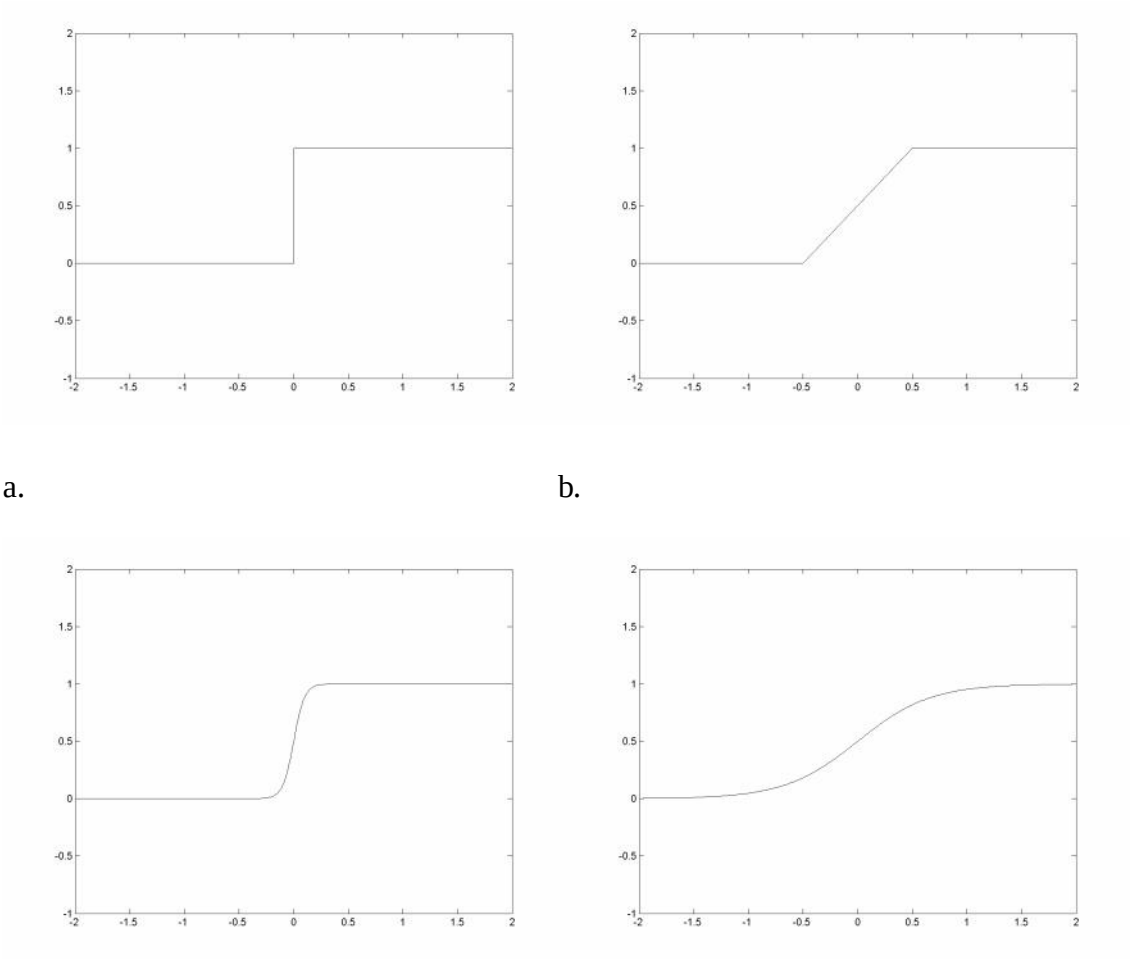
$$\varphi(v) = \frac{1}{1 + e^{-av}}, \tag{2.5}$$

missä v on neuronin saama kokonaissyöte ja a on funktion muotoon vaikuttava parametri. Tätä aktivaatiofunktiota voidaan parametri a:n arvoa muuttamalla muokata haluttaessa lähelle askelfunktiota. Kasvatettaessa a:ta lähestytään askelfunktiota ja a:n lähestyessä nollaa lähestytään suoraa

$$y = \frac{1}{2}.$$

Edellä esitellyissä tapauksissa aktivaatiofunktiot saivat arvoja väliltä [0,1], mutta Haykin [1994] toteaa, että usein käytetään myös funktioita, joiden arvot vaihtelevat välillä [-1,1], koska tämä voi olla joskus toivottavaa verkon toiminnallisuuden kannalta. Arvovälin muuttaminen onnistuu edellä esitettyjen funktioiden kohdalla melko

yksinkertaisesti, joten aiheen syvempi tarkasteleminen sivuutetaan tarpeettomana. Kuvassa 2.2 on esitetty edellä käsiteltyjen aktivaatiofunktioiden kuvaajat.



Kuva 2.2. Perusaktivaatiofunktiot: askelfunktio (a), paloittain lineaarinen funktio (b) sekä sigmoidifunktio parametrin arvoilla a=20 (c) ja a=3 (d).

2.2.2. Neuroverkkojen perusarkkitehtuurit

Edellä esiteltiin yksittäisen neuronin perusrakenne. Seuraavaksi tarkastellaan neuronien muodostaman verkon perusmalleja, joita kutsutaan yleisesti myös arkkitehtuureiksi. Haykin [1994] jakaa neuroverkkoarkkitehtuurit neljään eri luokkaan: yksikerroksiset eteenpäin syöttävät verkot, monikerroksiset eteenpäin syöttävät verkot, rekursiiviset verkot ja hilarakenteiset verkot.

Yleisesti neuroverkkojen arkkitehtuuri jaetaan *kerroksiin* (layers), jotka ovat neuroneista muodostuvia tasoja, jotka ottavat perustilanteessa vastaan syötteinä edellisen

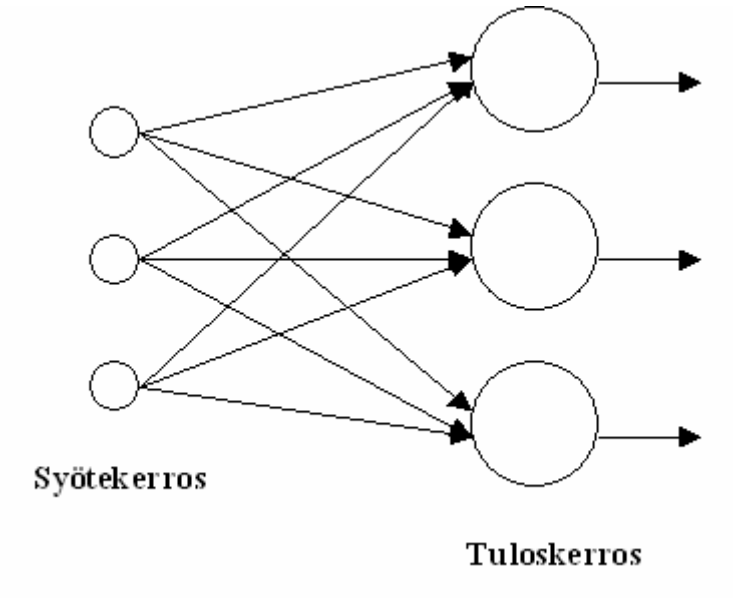
tason tuloksia painotettuina ja laskevat niistä omat tuloksensa, jotka ne sitten ohjaavat eteenpäin seuraavalle neuronitasolle, eli seuraavalle kerrokselle.
Ensimmäinen kerros verkossa on syötekerros, jonka neuronit välittävät poikkeuksellisesti saamansa syötteen eteenpäin muuttumattomana. Muissa kerroksissa neuronit toimivat kuten edellä neuronin perusmallin tarkastelussa esiteltiin, eli laskevat syötteistään oman aktivaationsa ja antavat tuloksen eteenpäin. Viimeinen kerros on puolestaan tuloskerros, jonka tulokset toimivat koko verkon prosessoinnin tuloksena.

Yksikerroksiset verkot (single-layer networks) ovat yksinkertaisia neuroverkkoja, joissa on vain syötekerros ja neuroneista koostuva tuloskerros. *Monikerroksisissa verkoissa* (multilayer nerworks) syötekerroksen ja tuloskerroksen välissä on lisäksi ainakin yksi neuroneista koostuva lisäkerros, jota kutsutaan myös piilokerrokseksi, koska se ei ole suorassa yhteydessä syötteisiin tai tuloksiin, mikä tavallaan tekee siitä verkon käyttäjälle näkymättömän verkon sisäisen kerroksen.

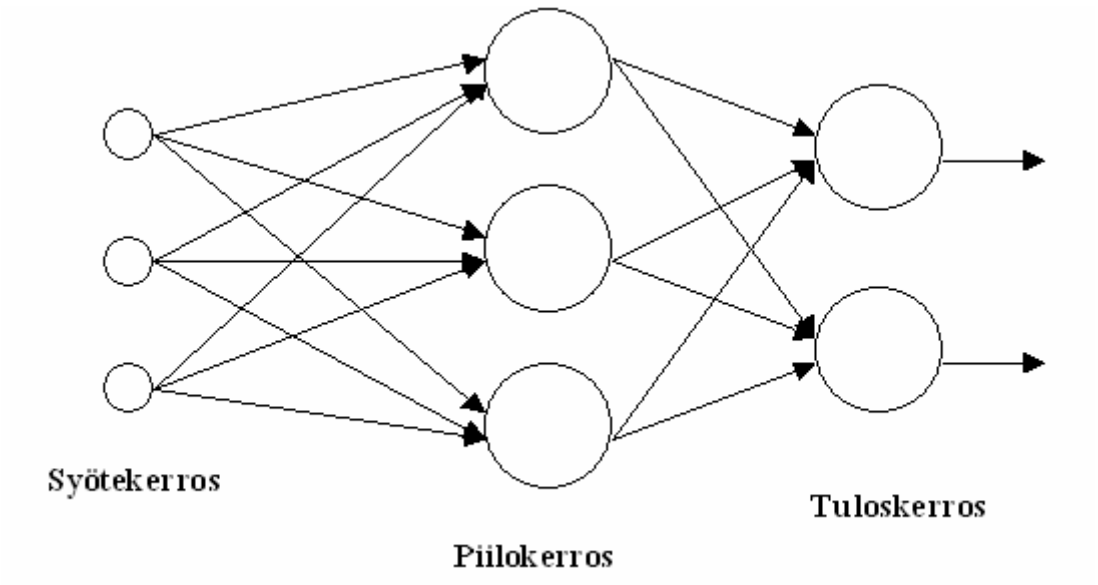
Eteenpäin syöttävät verkot (feedforward networks) ovat neuroverkkoja, joissa tieto, eli neuronien tulosimpulssi, kulkee vain yhteen suuntaan. Käytännössä eteenpäin syöttävissä verkoissa siis tieto kulkee vain alusta loppuun päin, mutta ei lainkaan takaisin alkua kohti. Verkkoja, joissa impulssit kulkevat myös takaisinpäin, kutsutaan vastaavasti *rekursiivisiksi verkoiksi* (recurrent networks). Rekursiivisen ominaisuuden saavuttaakseen verkko tarvitsee ainakin yhden takaisinpäin syöttävän yhteyden neuronien välillä.

Hilarakenteiset verkot (lattice structures) ovat eteenpäin syöttäviä verkkoja, joissa tuloskerros muodostaa yksi-, kaksi- tai useampiulotteisen hilan. Esimerkiksi kaksiulotteisen hilan tapauksessa on kyseessä verkko, jossa tuloskerroksen neuronit on järjestetty riveihin ja sarakkeisiin.

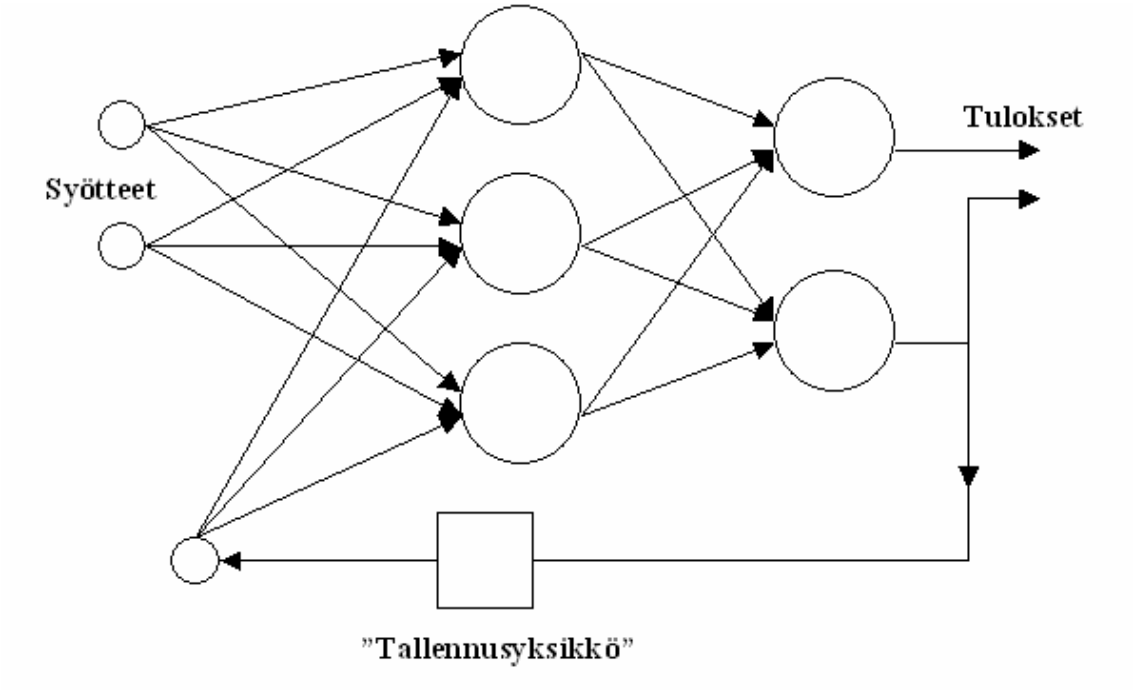
Kuvissa 2.3 - 2.6 on esitetty yksinkertaiset esimerkit kaikista edellä mainituista neljästä neuroverkkoarkkitehtuuriluokasta. Käytännössä neuronien väliset yhteydet voivat olla paljon monimutkaisempia ja arkkitehtuurit vaikeasti havaittavissa.



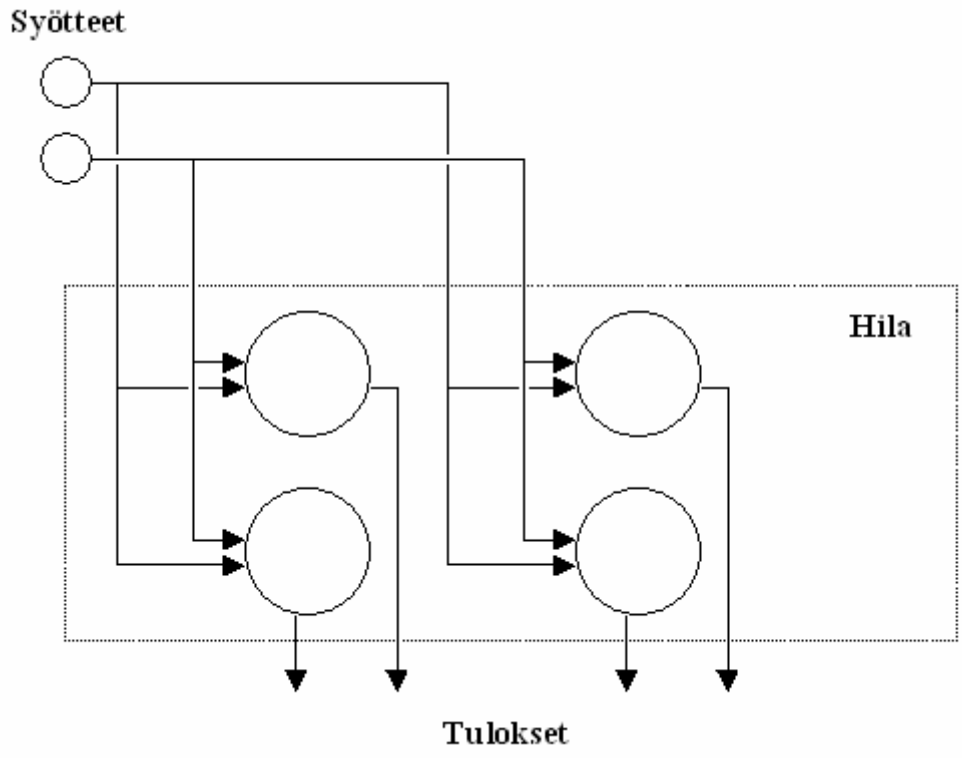
Kuva 2.3. Yksikerroksinen eteenpäin syöttävä verkko



Kuva 2.4. Kaksikerroksinen eteenpäin syöttävä verkko



Kuva 2.5. Rekursiivinen verkko



Kuva 2.6. Hilarakenteinen verkko

2.3. Neuroverkkojen oppimisalgoritmeista

Tähän mennessä on esitelty neuroverkon yksittäinen neuroni sekä neuronien muodostamat yleisimmät arkkitehtuurit. Seuraavaksi tarkastellaan hieman neuroverkon toiminnan oleellisinta osaa eli oppimista. Oppiminen neuroverkoissa tapahtuu erilaisten *oppimisalgoritmien* (learning algorithms) avulla.

2.3.1. Neuroverkkojen oppimisesta yleisesti

Haykin [1994] kuvailee neuroverkon oppimista iteratiiviseksi prosessiksi, jossa verkko oppii ympäristöstään säätämällä neuronien välisten yhteyksien painoarvoja ja siirtotermejä. Neuroverkko voi käyttää oppimisessa hyväkseen annettua syötettä, odotettua tulosta ja prosessoinnissa saatua tulosta. Näiden avulla verkko muuttaa painoarvojaan, ja mahdollisesti myös siirtotermiensä arvoja, päästäkseen lähemmäs haluttua tulosta. Arvojen muuttaminen tapahtuu verkkoon liittyvän oppimisalgoritmin mukaisesti. Neuroverkon oppima tieto tallentuu siis käytännössä verkon painoarvoihin useiden oppimisiteraatioiden tuloksena.

2.3.2. Oppimisalgoritmien luokittelu

Rojaksen [1996] mukaan neuroverkkojen oppimisalgoritmit jaetaan kahteen pääluokkaan, jotka ovat *ohjattu oppiminen* (supervised learning) sekä *ohjaamaton oppiminen* (unsupervised learning). Ohjatussa oppimisessa verkolle annetaan syötteitä ja tutkitaan verkon antamaa tulosta vertaamalla sitä ennalta odotettuun, oikeaan tulokseen. Saadun tuloksen ja odotetun tuloksen ero sekä verkon oppimisalgoritmi määräävät, miten verkon painoarvoja muutetaan. Ohjatussa oppimisessa tiedetään, mitä tuloksia halutaan, jolloin voidaan opettaa verkkoa, siis säätää sen painoarvoja, niin että se lopulta antaa halutun tuloksen.

Rojas jakaa edelleen ohjatun oppimisen kahteen luokkaan: korjaavaan ja vahvistavaan oppimiseen. Korjaavassa oppimisessa tarkoitetaan tilannetta, jossa tehty tulosvirheen suuruus tiedetään ja se pyritään yleensä poistamaan tämän tiedon ja annetun syötteen perusteella yhdellä korjaavalla opetuskierroksella. Vahvistavan oppimisen tapauksessa taas tiedetään vain, onko tulos oikea vai väärä, jolloin oppimisen täytyy perustua vain tähän tietoon ja annettuun syötteeseen.

Ohjaamaton oppiminen on täysin erilainen lähestymistapa. Siinä verkkoa ei opeteta tavoitetulosten avulla, vaan verkon toiminta on itsenäistä. Se saa syötteitä ja muodostaa niiden perusteella tulokset ilman ulkopuolista opetusta, jota ohjatussa oppimisessa hyödynnetään. Ohjaamatonta oppimista voidaan hyödyntää esimerkiksi syötteiden luokittelussa niiden ominaisuuksien perusteella erilaisiin ryhmiin. Välttämättä ei edes tiedetä ennakolta moneenko luokkaan syötteet jakautuvat, vaan verkko muodostaa luokat itsenäisesti syötteiden avulla.

2.4. Neuroverkkojen sovelluksista

Neuroverkkomenetelmiä on sovellettu monilla aloilla, niin tekniikassa kuin tieteessäkin. Erityisesti neuroverkkoja on käytetty luokittelussa, hahmontunnistuksessa, erilaisten ennusteiden tekemisessä sekä yhä enemmän myös tiedonhaun eri tehtävissä. Tämän tutkielman kannalta mielenkiintoisin sovellus on suurten tekstikokoelmien tutkimiseen ja tiedonhakuun tarkoitettu Teuvo Kohosen tutkimusten pohjalta pääosin 1990-luvun loppupuolella kehitetty WEBSOM-järjestelmä [Lagus *et al.*, 2004], jota käsitellään tarkemmin vielä luvussa 5. Myös Kwok [Kwok, 1989; Kwok *et al.*, 2005] on soveltanut neuroverkkomenetelmiä tiedonhaussa menestyksekkäästi jo 1980-luvulta lähtien.

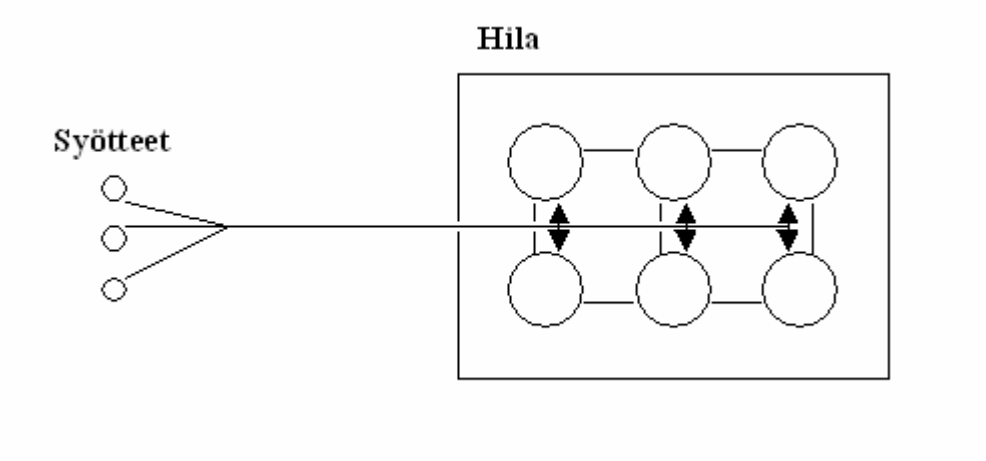
3. Itseorganisoituva kartta

Itseorganisoituva kartta (self-organizing map) [Kohonen, 1995] on Teuvo Kohosen kehittämä ohjaamattomaan oppimiseen perustuva neuroverkkomenetelmä, jonka esikuvana toimivat ihmisaivot ja biologiset neuroverkot. Menetelmää kutsutaan usein myös Kohosen (piirre)kartaksi, itseorganisoituvaksi verkoksi tai lyhyesti SOMiksi. Tässä tutkielmassa pyritään käyttämään järjestelmällisesti nimeä itseorganisoituva kartta, koska se vastaa parhaiten yleisesti käytettyä englanninkielistä termiä *self-organizing map* ja kartaksi kutsuminen on hieman verkkoa osuvampi nimitys kyseiselle neuroverkkomenetelmälle.

Seuraavaksi syvennytään tarkemmin menetelmän rakenteeseen, syötteisiin, varsinaiseen oppimisalgoritmiin ja tuloksena syntyvään karttaan.

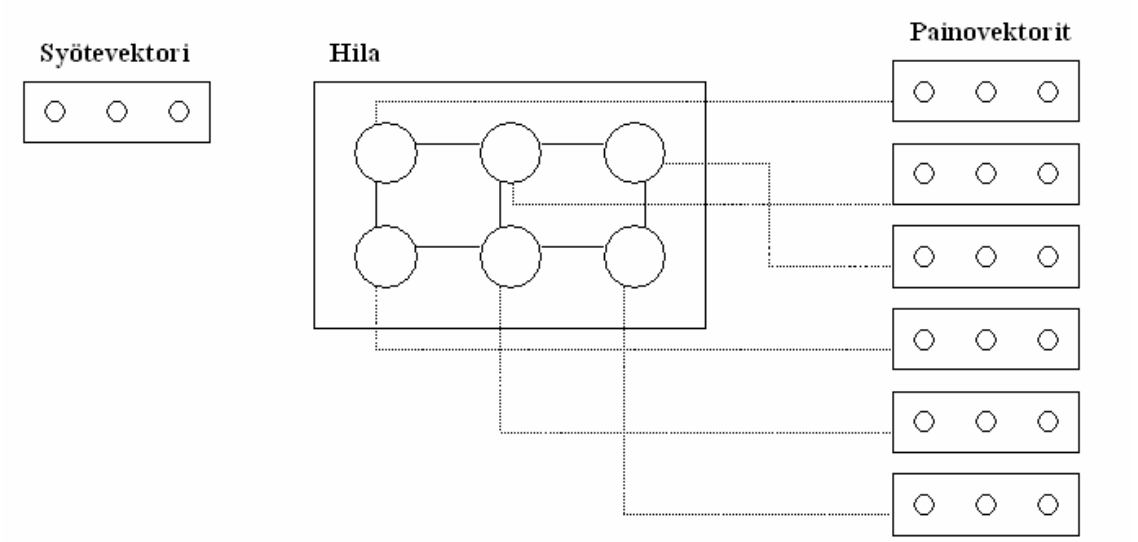
3.1. Verkon rakenne

Perinteisesti neuroverkot jakautuvat neuronien muodostamiksi syöte-, piilo- ja tuloskerroksiksi. Itseorganisoituva kartta poikkeaa tästä perusmallista siten, että verkko koostuukin yhdestä ainoasta neuronien hilasta, jonka jokainen neuroni on suorassa yhteydessä kaikkiin syötekerroksen neuroneihin, ja perinteinen tuloskerros puuttuu kokonaan. Lisäksi jokainen neuroni on yhteydessä viereisiin neuroneihin. Itseorganisoituva kartta on siis hilarakenteinen neuroverkko. Tuloskerrokseksi voidaankin tulkita koko neuronien hila, jolloin hilan neuronien tila muodostaa prosessoinnin lopputuloksen. Esimerkki tästä on esitetty kuvassa 3.1, jossa kartan syötteitä on kolme ja hilassa kuusi neuronia järjestettynä kahteen kolmen neuronin riviin. Hilaneuroneita kutsutaan usein myös solmuiksi.



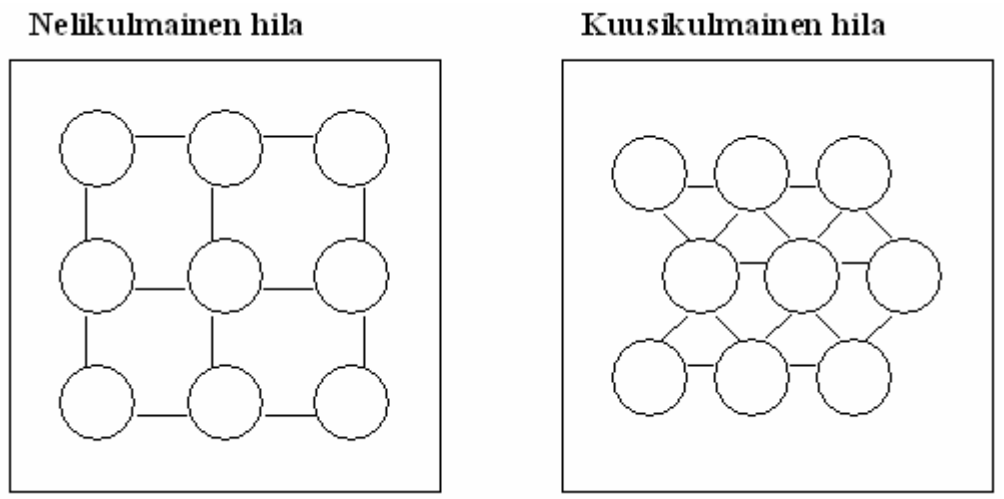
Kuva 3.1. Itseorganisoituvan kartan perusrakenne.

Koska jokainen yksittäinen hilaneuroni on suorassa yhteydessä jokaiseen syöteneuroniin liittyy jokaiseen hilan neuroniin myös syöteneuronien määrän verran painoarvoja. Itseorganisoituvan kartan yhteydessä ajatellaankin usein, että nämä painoarvot muodostavat painovektorin, joka liittyy kyseiseen hilaneuroniin. Vastaavasti kartan syöteneuronit muodostavat syötevektorin. Edellä esitetyn perusteella on selvää, että sekä kartan syötevektori että jokaisen yksittäisen hilaneuronin painovektori sisältävät yhtä monta alkiota kuin kartalla on syötteitä. Vektorit ovat myös samanrakenteisia siinä mielessä, että kartan *i*. syötteen arvo löytyy syötevektorin *i*. alkiosta ja *i*. syötteeseen liittyvät painot löytyvät kunkin painovektorin *i*. alkiosta. Vektorimallia on selvennetty kuvassa 3.2, jossa kuvan 3.1 kartta on esitetty vektoriesityksenä.



Kuva 3.2. Itseorganisoituvan kartan rakenne vektorinäkökulmasta.

Kartassa käytetään yleensä kaksiulotteista hilaa, joka on joko nelikulmainen tai kuusikulmainen. Nelikulmaisessa hilassa jokaisesta neuronista, joka ei sijaitse hilan reunalla, on suora yhteys neljään naapurineuroniin ja kuusikulmaisessa hilassa kuuteen. Hilan reunoilla naapurineuroneita on luonnollisesti vähemmän. Monimutkaisempien hilarakenteiden muodostaminen on mahdollista, mutta harvoin sovelluksen kannalta tarpeellista. Hila voi periaatteessa olla useampiulotteinen, rakenteeltaan hierarkkinen tai vaikkapa sylinterin muotoinen. Lisäksi syötteiden ja hilaneuronien yhteydet sekä hilaneuronien keskinäiset suhteet voidaan määritellä monimutkaisemmin, mutta useimmiten tyydytään käyttämään edellä esiteltyä perusmallia. Kuvassa 3.3 on esitetty nelikulmainen ja kuusikulmainen hila.



Kuva 3.3. Neuronien yhteydet ja sijainnit nelikulmaisessa ja kuusikulmaisessa hilassa.

3.2. Verkon alustaminen

Ennen itseorganisoituvan kartan opettamista on suoritettava hilassa olevien neuronien painoarvojen, eli käytännössä painovektorien, alustaminen. Yleisesti suositeltu tapa on alustaa jokainen painovektori pienehköillä satunnaisluvuilla. Myös erilaisia kehittyneempiä menetelmiä on olemassa ja ne perustuvat yleensä syötejoukon ominaisuuksien tarkastelemiseen. Painovektorit voidaan esimerkiksi alustaa niin, että ne ovat jo lähtötilanteessa lähellä syötevektoreiden arvoja, jolloin oppimisvaiheen voidaan ajatella nopeutuvan.

3.3. Verkon oppimisvaihe

Verkon alustamisen jälkeen on vuorossa oppimisvaihe. Itseorganisoituva kartta toteuttaa oppimisessa ohjaamatonta oppimista, mikä tarkoittaa sitä, että se organisoituu alkutilastaan ilman ulkoista ohjausta lopputilaansa. Verkolle annettavat syötevektorit vastaavat kukin jotain syöteoliota ja vektorin jokainen alkio vastaa tiettyä syöteolion ominaisuutta. Ominaisuudet on koodattava numeeriseen muotoon, jotta verkko voi käsitellä vektoria.

3.3.1. Oppimisalgoritmi

Lagus [2000] tarkastelee itseorganisoituvaa karttaa väitöskirjassaan ja kuvailee sitä ohjaamattoman oppimisen neuroverkkoalgoritmiksi, joka pystyy järjestämään korkeadimensioisen datan niin, että samanlaiset datasyötteet sijoittuvat lähelle toisiaan kartalla. Seuraavaksi itseorganisoituvan kartan oppimisalgoritmia tarkastellaan Laguksen väitökseen perustuen.

Itseorganisoituvan kartan oppimisalgoritmissa syöteoliot eli käytännössä niitä edustavat syötevektorit annetaan kartalle satunnaisessa järjestyksessä. Syötevektorien

joukko annetaan lisäksi kartalle yleensä useita kertoja peräkkäin. Jokaisen kartalle annetun syötevektorin, eli yksittäisen oppimisaskeleen, kohdalla etsitään kartalta sitä parhaiten vastaava painovektori, jota kutsutaan myös *voittajasolmuksi* (BMU, best matching unit). Kun voittajasolmu on löydetty, päivitetään voittajan ja sen naapurisolmujen painovektoreita. Päivitysoperaatio suoritetaan kaavalla

$$m_i(t+1) = m_{i(t)} + h_{c(x),i}(t)\left[x(t) - m_i(t)\right], \tag{3.1}$$

missä $x(t)$ on opetusaskeleen t syötevektori, $m_i(t)$ on yksittäinen kartan painovektori opetusaskeleella t, i=0,…, n-1 kertoo painovektoriin liittyvän neuronin indeksin ja n on neuronien määrä kartalla. Kaavassa esiintyvä $c(x)$ on löydetyn voittajasolmun indeksi ja se saadaan kaavasta

$$c(x) = \arg\min_i \left\{ \left\| x - m_i \right\| \right\}. \tag{3.2}$$

Kaavassa 3.1 esiintyvä $h_{c(x),i(t)}$ on puolestaan *naapurustofunktio* (neighborhood function), joka määrittelee voittajasolmun ympärillä olevan naapuruston opetusaskeleella t. Laguksen [2000] mukaan naapuruston määrittämänä funktiona käytetään usein Gaussin funktiota (Gaussian), jolloin naapurusto voidaan määritellä muodossa

$$h_{c(x),i}(t) = \alpha(t) \exp\left(- \frac{\left\| r_i - r_{c(x)} \right\|^2}{2\sigma^2(t)} \right), \tag{3.3}$$

missä $0 < \alpha(t) < 1$ on *oppimiskerroin* (learning rate factor), r_i kuvaa neuronin i sijainnin kartalla ja $\sigma>0$ on naapuruston koko. Sekä oppimiskerroin että naapuruston koko pienenevät monotonisesti oppimisen edetessä.

Käytännössä siis edellä esitellyssä oppimisalgoritmissa kartta saa kaikki syötteet satunnaisessa järjestyksessä useita kertoja peräkkäin ja joka kerta annetulle syötteelle etsitään osumakohta kartalta, jota lähiympäristöineen sitten muutetaan lähemmäs syötteen hahmoa. Usein naapurustofunktiona käytetään myös niin sanottua kuplanaapurustoa, jossa naapurusto määrittyy yksinkertaisesti voittajasolmun ympärille piirretyn halutun kokoisen ympyrän sisälle ja kaikkia naapuruston painoarvovektoreita muutetaan samalla tavalla kuin voittajasolmua.

3.3.2. Oppimisen muuttujat

Verkon oppimiseen vaikuttavat useat muuttujat. Esimerkiksi kartan koko ja muoto, hilaneuronien painovektorien alustustapa, opetusaskelten määrä, naapurustofunktion muoto, oppimiskerroin ja naapuruston koko vaikuttavat kaikki siihen, miten kartta oppii

syötejoukon ominaisuudet. On olemassa vain melko vähän tietoa siitä, miten eri muuttujat tulisi asettaa parhaan mahdollisen oppimistehon saavuttamiseksi.

Kohonen [1995] käsittelee asiaa laajasti antaen joitain suosituksia hyväksi havaituista muuttuja-asetuksista. Näiden tutkimusten pohjalta on yleiseen käyttöön alalla noussut suositus, jonka mukaan opetuskierrokset tulisi jakaa kahteen osaan: alun karkeaan opetusvaiheeseen ja loppuvaiheen hieno-opetukseen. Karkeassa opetusvaiheessa naapuruston koko pidetään suurena ja opetuskerroin korkeana, jotta kartta järjestyisi nopeasti vastaamaan karkeasti syötejoukon ominaisuuksia. Hieno-opetusvaiheessa puolestaan naapuruston koko on pienempi ja opetuskerroin matalampi, jotta kartta järjestyisi vastaamaan syötejoukon hienovaraisempia suhteita. Opetuskierrosten määrää säädellään tässä opetustavassa myös siten, että karkeassa vaiheessa käydään vain muutamia kierroksia läpi koko opetusjoukon ja hienovaiheessa opetuskierroksia läpi joukon tehdään merkittävästi enemmän. Nämäkin yleisesti käytössä olevat ohjeet ovat hyvin suhteellisia ja mitään tarkkoja yleisiä asetuksia ei ole olemassa. Ohjeistus on siis suhteutettava kulloiseenkin datajoukkoon yksilöllisesti ja eri asetuksia on syytä testata parhaan yhdistelmän löytämiseksi. Varsinaisia tutkimustuloksia eri asetusten eroista ei juurikaan ole saatavilla.

3.3.3. Oppimisen arviointi

On erittäin vaikeaa mitata sitä, miten hyvin kartta on oppinut syötejoukkonsa ominaisuudet. Yksi menetelmä tähän on kuitenkin yleisessä käytössä. Menetelmää kutsutaan *keskimääräiseksi kvantisaatiovirheeksi* (average quantization error). Keskimääräinen kvantisaatiovirhe lasketaan syöttämällä valmiille kartalle kaikki syötevektorit kertaalleen ja laskemalla kunkin etäisyys voittajasolmun painovektorista ja ottamalla sitten tästä keskiarvo. Tämä keskiarvo kuvaa sitä, miten hyvin kartan neuronit ovat järjestyneet kuvaamaan syötejoukon ominaisuuksia. Keskimääräistä kvantisaatiovirhettä ei voida käyttää eri asetuksilla luotujen karttojen vertailussa, mutta tietyillä asetuksilla luotujen karttojen vertailemiseen se soveltuu.

Koska itseorganisoituvan kartan luominen perusalgoritmilla on satunnainen prosessi, saattaa kartan luominen joskus epäonnistua, jos esimerkiksi annetut painovektorien alkuarvot eivät mahdollista hyvään lopputulokseen pääsemistä. Keskimääräisellä kvantisaatiovirheellä voidaankin arvioida itseorganisaation onnistumista esimerkiksi luomalla useita karttoja samoilla asetuksilla ja vertailemalla kvantisaatiovirhettä.

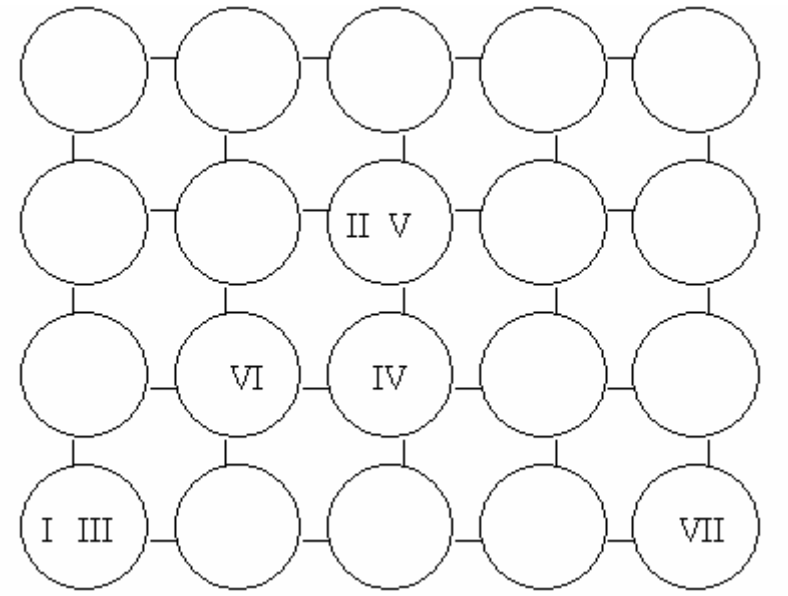
3.3.4. Eräalgoritmi

Edellä esiteltiin perinteinen itseorganisoituvan kartan oppimisalgoritmi, jota on käytetty tähän tutkielmaan liittyvien karttojen luomisessa. WEBSOM-sovelluksen [Lagus *et al*., 2004] yhteydessä on kehitetty myös uusi algoritmi, joka on kehittäjiensä mukaan lähes yhtä suorituskykyinen kuin perinteinen algoritmi, mutta toimii huomattavasti nopeammin. Uutta algoritmia kutsutaan *eräalgoritmiksi* (batch mode algorithm) ja sen

yhteydessä on kehitetty myös useita algoritmin tietokonetoteutusta nopeuttavia menetelmiä. Koska näitä uusia menetelmiä ei sovellettu tässä tutkielmassa, niiden tarkempi tarkastelu sivuutetaan tarpeettomana. Lisää tietoa uusien menetelmien toteutuksesta ja vertailusta perinteiseen algoritmiin löytyy useista WEBSOM-ryhmän julkaisuista (esimerkiksi [Kohonen *et al*., 2000] ja [Lagus *et al*., 2004]).

3.4. Valmis kartta

Opetusvaiheiden päätyttyä verkon tila on organisoitunut vastaamaan syötejoukon ominaisuuksia ja suhteita. Kartan eri alueet ovat muuttaneet painojaan siten, että ne vastaavat erityyppisiä syöteluokkia, jotka menetelmä on oppinut ohjaamattomasti syötejoukosta. Syötehahmot voidaan nyt sijoittaa kartalle etsimällä valmiilta kartalta kullekin syötehahmolle vielä kerran voittajasolmu ja asettamalla syötehahmo kartalla voittajasolmun sijaintiin. Samaan kartan solmuun eli neuroniin sijoittuvat syötteet ovat oletettavasti samanlaisia jossain suhteessa. Samalla tavoin esimerkiksi vierekkäisissä solmuissa sijaitsevat syötteet ovat lähempänä toisiaan kuin syötteet, joilla on suuri etäisyys kartalla. Tästä on annettu esimerkki kuvassa 3.4, jossa kartta on järjestänyt seitsemän (I - VII) syötehahmoa pinnalleen. Kuvasta voidaan päätellä, että esimerkiksi syötteet II ja V vastaavat ominaisuuksiltaan toisiaan hyvin läheisesti, kun taas syöte VII on hyvin erilainen kaikkiin muihin syötehahmoihin nähden. Samoin syöte VI on etäämmällä syötteestä II kuin syöte IV, jolloin oletettavasti II ja IV vastaavat toisiaan paremmin kuin II ja VI.



Kuva 3.4. Seitsemän (I-VII) syötehahmoa organisoituneena kartalle.

3.5. Visualisointi, yleistäminen ja luokittelu

Edellä toteutettiin opetusvaihe siten, että kartta opetettiin käymällä läpi koko syötejoukko opettamisessa. Tällöin voidaan ajatella, että kyseessä on syötejoukon visualisointi karttana. Jos tavoitteena on muodostaa annetusta syötejoukosta automaattisesti kartta, joka kuvaa joukon suhteita on paras ratkaisu käyttää opettamisessa koko syötteiden joukkoa, jolloin koko joukon ominaisuudet vaikuttavat kartan järjestymiseen ja kartta edustaa paremmin joukon sisäisiä suhteita. Toinen sovellustapa on kartan yleistämiskyvyn hyödyntäminen. Esimerkiksi voidaan luoda kartta, joka osaa luokitella syötteitä tiettyihin luokkiin. Tällöin osaa syötteistä voidaan käyttää opetusjoukkona, jonka perusteella kartta muodostetaan. Opetusjoukkoon kuulumattomien syötteiden joukkoa kutsutaan usein testijoukoksi. Jos opetusjoukon alkioiden luokat tunnetaan, voidaan kartalle muodostaa luokkia ja karttaa voidaan käyttää aiemmin esittelemättömien syötteiden luokitteluun. Kartta siis sijoittaa syötteet hahmojensa mukaan luokkia vastaaville alueille. Tällä tavalla kartta yleistää opetusjoukosta oppimaansa ja toimii syötteiden luokittelijana.

Sovelluksen luonteesta riippuu, kannattaako koko syötejoukkoa käyttää opettamisessa vai käyttää erillistä opetusjoukkoa ja testijoukkoa.

3.6. Itseorganisoituvan kartan sovelluksista

Itseorganisoituvan kartan sovelluskenttä on laaja. Kohosen [1995] mukaan sovellusalueeseen kuuluvat ainakin: konenäkö ja kuva-analyysi, optiset lukulaitteet, puheen analyysi ja tunnistus, akustiikan ja musiikin tutkimus, signaalin prosessointi ja tutkalaitteet, telekommunikaatio, teollinen mittaaminen, prosessinhallinta, robotiikka, kemia ja fysiikka, lääketieteen kuvien prosessoimiseen liittyvät sovellukset, tiedon prosessointi, kielitieteet ja tiedonhaku, tekoäly, matemaattiset ongelmat ja neurofysiologinen tutkimus. Hyvänä esimerkkinä pienimuotoisesta perussovelluksesta voidaan mainita esimerkiksi Faba-Pérez *et al*. [2003], jossa tutkittiin verkkosivujen luokittelua. Tutkimuksessa valittiin erilaisia numeerisia ominaisuuksia kuvaamaan verkkosivua ja muodostettiin näin jokaisesta verkkosivusta vektori. Nämä vektorit annettiin syötteenä kartalle, joka järjesti sivut itseorganisaation avulla. Karttaa käytettiin tässä siis kohteiden välisten monimutkaisten suhteiden pelkistämiseen kaksiulotteiseen tasoon. Tällaisia sovelluksia voidaankin kuvitella olevan lukuisia lähes kaikilla aloilla, sillä periaatteessa ainoa edellytys on, että ongelmasta voidaan muodostaa numeerinen syötedatan joukko, joka sitten organisoidaan automaattisesti kartalle analysointia varten.

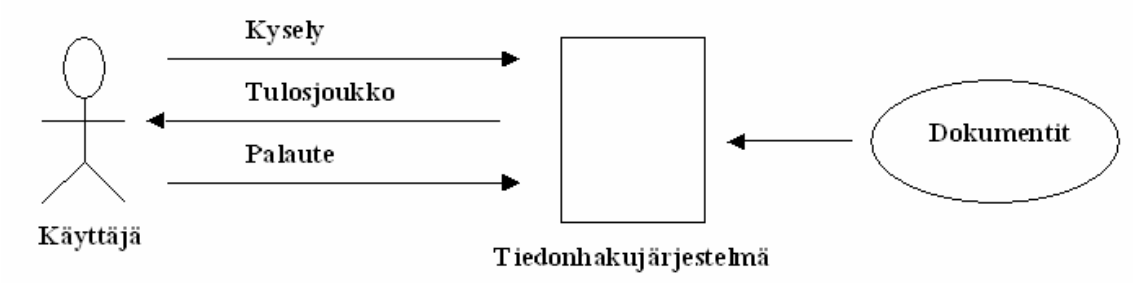
Edellä esitettyihin sovellusalueisiin ei ole tässä yhteydessä syytä perehtyä tarkemmin, mutta luvussa 5 palataan tarkemmin itseorganisoituvan kartan tiedonhakusovelluksiin.

4. Tiedonhaku

Tiedonhaku (IR, information retrieval) on tieteenala, jossa käsitellään tiedon esitystä, tallentamista, organisoimista ja saatavuutta [Salton ja McGill, 1983]. Sen merkitys alana on jatkuvasti kasvava nyky-yhteiskunnassa, kun hyödyllisen tiedon löytäminen suuresta määrästä tietoa on yhä arkipäiväisempi ja tärkeämpi tehtävä yhä useammalle ihmiselle niin työssä kuin vapaa-ajallakin. Erityisesti Internetin kasvaminen merkittäväksi tiedonlähteeksi on lisännyt tiedonhaun alaan liittyvien sovellusten tarvetta. Dokumentoitua tietoa on nykyään saatavilla enemmän kuin koskaan, mutta samalla on yhä vaikeampi löytää tarvitsemaansa tietoa tästä informaatiotulvasta. Juuri tähän ongelmaan haetaan ratkaisuja tiedonhaun tutkimuksessa.

Tiedonhaussa käsiteltävä tieto voi olla periaatteessa missä muodossa tahansa, kunhan sitä pystytään jollain tavoin käsittelemään. Esimerkiksi tietoalkiot voivat olla tekstiä, kuvaa, ääntä tai videota. Tässä tutkielmassa keskitytään pelkästään tekstidokumenttien tiedonhakuun, jota kutsutaan yleensä *tekstitiedonhauksi*. Jatkossa käytetään kuitenkin yksinkertaisuuden vuoksi tiedonhaku-sanaa, vaikka tarkoitetaankin yleensä nimenomaan tekstitiedonhakua. Pääosin kaikki tässä luvussa esitetty pätee yleisesti myös muuhun kuin tekstitiedonhakuun.

Perinteisesti tiedonhaussa lähdetään liikkeelle tiedontarpeesta, joka syntyy käyttäjälle. Käyttäjä pyrkii ilmaisemaan tiedontarpeensa kyselynä, jonka hän muotoilee jollain kyselykielellä tiedonhakujärjestelmälle. Tiedonhakujärjestelmän tehtävänä on palauttaa käyttäjälle tietoalkioita, jotka olisivat relevantteja annetun kyselyn suhteen. Määritellään nyt tähän tutkielmaan liittyen dokumentin relevanssi annetun kyselyn suhteen yksinkertaisesti siten, että relevantti dokumentti vastaa kyselyn muotoilemaan tiedontarpeeseen. *Relevanssin* käsite ei välttämättä ole lainkaan näin yksinkertainen, mistä lisää luvussa 4.3. Joidenkin tiedonhakujärjestelmien kohdalla käyttäjällä on vielä tulosjoukon saatuaan mahdollisuus antaa järjestelmälle palautetta tulosjoukon dokumenttien relevanssista tiedontarpeen suhteen. Kuvassa 4.1 on esitetty yksinkertainen malli tiedonhakuprosessista. Tiedonhaun tutkimuksessa tutkitaan kaikkia tähän perinteiseen tiedonhakutapahtumasarjaan liittyviä ilmiöitä. Yksi tärkeimmistä tehtävistä tiedonhaun tutkimuksessa on pyrkimys kehittää tiedonhakujärjestelmä, joka pystyisi mahdollisimman hyvin vastaamaan käyttäjän esittämään tiedontarpeeseen.



Kuva 4.1. Tiedonhakuprosessin malli.

Oma ongelmansa on se, että käyttäjä ei välttämättä edes osaa ilmaista tiedontarvettaan kunnolla annetulla kyselykielellä tai hakumenetelmällä, mikä saattaa johtaa siihen, että varsinaisen annetun kyselyn suhteen hyväkin vastaus saattaa osoittautua täysin hyödyttömäksi käyttäjän varsinaisen tiedontarpeen suhteen. On siis selvää, että tiedonhaku on paljon muutakin kuin pelkästään tehokkaiden tiedonhakujärjestelmien teknistä suunnittelua. Tiedonhaun ongelmissa joudutaankin usein yhdistämään tehokas tekninen toteutus hyvään ymmärrykseen ihmisen sisäisestä toiminnasta ja psykologiasta.

Tämän tutkielman kannalta ei ole oleellista pohtia itse tiedonhakuprosessia kovin syvällisesti, joten tarkastellaan tiedonhakua jatkossa lähinnä kuvan 4.1 mukaisena yksinkertaisena prosessina, jossa käyttäjä esittää tiedontarpeensa pohjalta kyselyn ja saa vastauksena tiedonhakujärjestelmän tuottaman tulosjoukon, joka koostuu järjestelmän arvion mukaan parhaiten käyttäjän kyselyyn vastaavista dokumenteista. Seuraavassa esitellään klassiset tiedonhakumenetelmät, tiedonhaunmenetelmien evaluointi sekä tarkastellaan lyhyesti tiedonhakumenetelmät liittyvistä ongelmista.

4.1. Klassiset tiedonhakumenetelmät

Tiedonhaun kolme klassista menetelmää ovat Boolen malli, vektoriavaruusmalli ja todennäköisyysmalli [Baeza-Yates and Ribeiro-Neto, 1999]. Seuraavassa tutustutaan kuhunkin menetelmään lyhyesti.

4.1.1. Boolen malli

Boolen malli (Boolean model) on joukko-oppiin ja Boolen algebraan perustuva tiedonhakumenetelmä. Mallissa jokaista dokumenttia edustaa yksinkertaisesti indeksisanojen joukko. Vastaavasti kyselyt muodostuvat indeksisanoista, joita voidaan yhdistellä Boolen loogisilla operaattoreilla (*not*, *and* ja *or*). Jokainen kysely on hyvin määritelty siinä mielessä, että tietyn muotoisen kyselyn tuottama tulosjoukko on aina yksikäsitteinen. Jokainen dokumentti on siis aina joko täysin relevantti tai täysin epärelevantti annetun kyselyn suhteen. Boolen mallia kutsutaankin tästä syystä

täystäsmäystä (exact matching) toteuttavaksi tiedonhakumenetelmäksi. Yksinkertaisen ja joukko-oppiin perustuvan tyylikkään teoreettisen perustansa ansiosta menetelmä on ollut hyvin suosittu ratkaisu tiedonhakujärjestelmien toteuttamisessa. Itse asiassa Boolen malli on edelleen suosituin tiedonhakumenetelmä kaupallisissa tietokantajärjestelmissä [Baeza-Yates and Ribeiro-Neto, 1999].

Vastaavasti Boolen menetelmällä on myös omat ongelmansa, jotka Lagus [2000] tiivistää kolmeen pääkohtaan:

1. Sopivien sanojen valitseminen kyselyyn voi olla vaikeaa, varsinkin jos tiedonhakuaihe on ennestään tuntematon.
2. Tulosjoukon kokoa ei voida rajata. Tulosjoukko voi olla tyhjä tai sisältää kaikki dokumenttijoukon dokumentit.
3. Tulosjoukkoa ei ole järjestetty relevanssin suhteen.

Ensimmäinen ongelma on merkittävä, sillä tiukasti Boolen logiikalla tehdyissä kyselyissä yksikin ”ylimääräinen” hakusana saattaa aiheuttaa täysin vääränlaisia tuloksia tulosjoukkoon tai rajata merkittävän osan hyödyllisistä dokumenteista ulos tuloksesta. Toinen ja kolmas ongelma ovat osittain yhteydessä toisiinsa, sillä Boolen mallissa dokumentteja ei pisteytetä mitenkään kyselyn suhteen, vaan jokainen dokumentti joko kuuluu tulosjoukkoon tai ei kuulu, mikä on suoraan ja yksikäsitteisesti pääteltävissä kyselystä. Tällöin dokumenttien järjestäminen on mahdotonta, koska ei ole olemassa suhteellista relevanssia, jonka mukaan järjestäminen voitaisiin tehdä. Kun järjestäminen on mahdotonta, ei tulosjoukkoakaan voida supistaa, koska jokaisella joukon dokumentilla on yhtä suuri arvo kyselyn suhteen. Voisi ajatella, että kyseessä ei ole suurikaan ongelma, mutta pohdittaessa tilannetta, jossa tiedonhakujärjestelmä A tarjoaa tulosjoukkoon 1000 dokumenttia satunnaisessa järjestyksessä ja järjestelmä B tuottaa 100 relevanteinta relevanssin mukaan järjestettynä, huomataan, että järjestelmän B tuottama lisäarvo dokumenttien järjestämisessä ja tulosjoukon rajaamisessa on merkittävä. Baeza-Yates ja Ribeiro-Neto [1999] pitävätkin Boolen mallia edellä esitettyjen ongelmien takia klassisista menetelmistä heikoimpana.

4.1.2. Vektoriavaruusmalli

Vektoriavaruusmalli [Salton and McGill, 1983] pyrkii vastaamaan Boolen mallin ongelmiin toteuttamalla täystäsmäyksen sijaan *osittaistäsmäystä* (partial matching). Osittaistäsmäyksessä sanalistoina annetut kyselyt täsmäytetään dokumentteja kuvaaviin sanalistoihin ja lasketaan siten kullekin dokumentille kyselyyn suhteutettua samanlaisuutta kuvaava pisteytys. Tämän pisteytyksen avulla dokumenttijoukko voidaan järjestää paremmuusjärjestykseen kyselyn suhteen. Sanalistojen sanoille voidaan osittaistäsmäyksessä antaa myös painoarvoja niiden merkittävyyden mukaan.

Vektoriavaruusmallin peruslähtökohtana on esittää dokumenttikokoelman jokainen dokumentti vektorina, jonka alkiot vastaavat kokoelman sanoja (tai valittuja hakuavaimia). Dokumentit esitetään siis muodossa

$$D_i = (a_{i1}, a_{i2}, a_{i3}, ..., a_{it}), \tag{4.1}$$

missä a_{ik} on sanan k painoarvo dokumentissa i, $1 \leq i \leq$ n, $1 \leq k \leq$, t , n on dokumenttien lukumäärä kokoelmassa ja t on sanojen lukumäärä kokoelmassa. Kun kaikki dokumenttivektorit kootaan yhteen, saadaan kokoelmasta muodostettua seuraavanlainen matriisi:

$$\begin{array}{c} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_n \end{array} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1t} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2t} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n2} & \cdots & a_{nt} \end{bmatrix}, \tag{4.2}$$

Matriisin rivit muodostuvat dokumenttivektoreista ja sarakkeet sanavektoreista. Myös kyselyt voidaan vektoriavaruusmallissa esittää vektoreina, jonka alkiot vastaavat samalla tavalla kokoelman sanoja kuin edellä dokumenttivektorien tapauksessa. Kyselyvektori voidaan esittää muodossa

$$Q_j = (q_{j1}, q_{j2}, q_{j3}, ..., q_{jt}), \tag{4.3}$$

missä q_{jk} on sanan k painoarvo kyselyssä j, ja t on sanojen kokonaismäärä dokumenttikokoelmassa. Koko vektoriavaruusmallin perustan luo se tosiasia, että vektoreina esitettyihin dokumentteihin ja kyselyihin voidaan soveltaa suoraan lineaarialgebran peruslaskutoimituksia. Täsmäys perustuukin vektorimallissa kyselyä ja dokumenttia edustavien vektorien välisen samanlaisuuden laskemiseen ja oletukseen, että tietyn kyselyn ja sen suhteen relevantin dokumentin vektorit ovat samanlaiset.

Edellä kerrottiin, että dokumentit ja kyselyt muodostavat vektoriavaruusmallissa vektoreita, joissa alkioina käytetään dokumenttijoukon sanojen painoarvoja, joten esitellään seuraavaksi sanojen painotukseen liittyviä näkökulmia. Yksinkertaisimmat tavat sanan painoarvon laskemiselle tietyssä dokumentissa ovat binääri- ja frekvenssipainotus. Binääripainotuksessa kutakin sanaa vektorissa edustaa joko 1 tai 0 sen mukaan, esiintyykö sana dokumentissa (1) vai ei (0). Frekvenssipainotus ottaa huomioon myös sanan esiintymien määrän, jolloin esimerkiksi viisi kertaa dokumentissa esiintyvää sanaa edustaa vektorissa arvo 5 ja sanaa, joka ei esiinny kertaakaan arvo 0. Vastaavasti kyselyvektorien sanat painotetaan saman periaatteen mukaisesti kuin dokumenttivektorien sanatkin.

Sanojen painottamiseksi on olemassa myös kehittyneempiä menetelmiä, jotka pyrkivät mittaamaan sanojen paremmuutta dokumenttinsa edustajana. Baeza-Yatesin ja Ribeiro-Neton [1999] mukaan kehittyneintä painotusmenetelmää edustaa *tf·idf*-

menetelmä ja sen erilaiset variaatiot, jotka kaikki jakautuvat kahteen eri tekijään: termifrekvenssiin (tf) ja käänteiseen dokumenttifrekvenssiin (idf). *Termifrekvenssi* (term frequency, tf) kuvaa yleisesti sitä, miten hyvin kyseinen sana edustaa dokumentin sisältöä. Termifrekvenssi ilmoitetaan yleensä sanojen lukumääränä tai yksinkertaisena binäärisenä arvona, jossa sana joko löytyy (1) tai ei löydy (0) dokumentista. Lukumääräpainotusta sovellettaessa tf_{ik} kertoo sanan k esiintymien määrän dokumentissa D_i. *Käänteinen dokumenttifrekvenssi* (inverse document frequency, idf) on tekijä, joka pyrkii suosimaan sanoja, jotka esiintyvät kyseisen dokumentin lisäksi vain harvoissa muissa dokumenttijoukon dokumentissa. Lähestymistavan motivaationa on se huomio, että sana, joka esiintyy lähes kaikissa dokumenteissa, on käytännössä huono erottelemaan dokumentteja toisistaan.

Tarkastellaan seuraavaksi erästä hyvin yleistä *tf·idf*-painotuksen toteutusta, jossa termifrekvenssi on toteutettu niin, että lukumääräpainotus jaetaan vielä kyseiseen dokumenttiin liittyvien sanojen lukumääräpainojen maksimilla. Tämän toteutuksen tarkoituksena on suhteuttaa esiintymien määrä dokumentin kokoon ja siten vähentää pitkien dokumenttien saamaa etua, joka esiintyy vertailussa pelkällä lukumääräpainolla. Näin saatava suhteellinen termifrekvenssi sanalle k dokumentissa D_i lasketaan kaavalla

$$tf_{ik} = \frac{freq_{ik}}{\max_l freq_{il}}, \tag{4.4}$$

missä $freq_{ik}$ on sanan k esiintymien lukumäärä dokumentissa D_i ja l käy läpi kaikki dokumenttikokoelman sanat eli l=1,2,3,..., t-1, t. Käänteinen dokumenttifrekvenssi sanalle k dokumenttijoukossa puolestaan saadaan kaavalla

$$idf_k = \log \frac{N}{n_k}, \tag{4.5}$$

missä N on dokumenttien määrä koko dokumenttijoukossa ja n_k on niiden dokumenttien lukumäärä, jotka sisältävät sanan k. Yhdistämällä kaavojen 4.4 ja 4.5 tekijät saadaan lopulta painotus sanalle k dokumentissa D_i:

$$a_{ik} = tf_{ik} \cdot idf_k \ . \tag{4.6}$$

Näin saadaan siis laskettua dokumenttivektorien alkiot. Kyselyvektorien sanapainot saadaan puolestaan kaavasta

$$q_{jk} = \left(0{,}5 + \frac{0{,}5 \cdot freq_{jk}}{\max_l freq_{jl}} \right) \cdot \log \frac{N}{n_k}, \tag{4.7}$$

missä $freq_{jk}$ on sanan k esiintymien lukumäärä kyselyssä j ja l käy läpi kaikki dokumenttikokoelman sanat eli l=1,2,3,...,t-1,t. Kyselyvektoreille on kirjallisuudessa annettu oma painotustapansa, koska niiden luonne poikkeaa dokumenttivektoreista merkittävästi esimerkiksi siksi, että kyselyissä sanat esiintyvät yleensä vain kertaalleen ja kyselyt ovat selvästi lyhyempiä sanamäärältään.

Myös kaavaan 4.6 voidaan lisätä normalisointitekijä, jolla voidaan pyrkiä poistamaan pitkien dokumenttien etua, joka johtuu niiden suuresta sanamäärästä. Belew [2000] jakaakin painotusmenetelmän kolmeen tekijään. Termifrekvenssin ja käänteisen dokumenttifrekvenssin lisäksi kolmantena tekijänä toimii normalisointitekijä, jolloin painotus saa yleisen muodon:

$$a_{ik} = \frac{freq_{ik} \cdot discrim_k}{norm}, \tag{4.8}$$

missä *freq* vastaa termifrekvenssiä, *discrim* käänteistä dokumenttifrekvenssiä ja *norm* normalisointitekijää.

Kun dokumentti- ja kyselyvektorit on muodostettu, tarvitaan vielä tapa, jolla näiden samankaltaisuutta voidaan mitata, jotta dokumentit voidaan järjestää relevanssin mukaiseen järjestykseen kyselyn suhteen. Kuten edellä todettiin, on vektoriesityksen suuri etu se, että voidaan käyttää lineaarialgebran laskutoimituksia. Yleisesti samanlaisuuden laskemiseen käytetään (esimerkiksi [Belew, 2000] ja [Baeza-Yates and Ribeiro-Neto, 1999]) kosinituloa. Kosinitulo lasketaan kaavalla

$$\cos(D_i, Q_j) = \frac{\sum_{k=1}^{t} a_{ik} \cdot q_{jk}}{\sqrt{\sum_{k=1}^{t} a_{ik}^2} \cdot \sqrt{\sum_{k=1}^{t} q_{jk}^2}}, \tag{4.9}$$

missä merkinnät vastaavat edellä esitettyjä kaavoja.

Näillä edellä esitetyillä ”työkaluilla” voidaan toteuttaa vektoriavaruusmalli, joka toteuttaa osittaistäsmäyksen dokumenttijoukon dokumenttien ja annetun kyselyn kesken ja järjestää tuloksena dokumentit relevanssijärjestykseen kyselyn suhteen. Vektoriavaruusmallia pidetään yleisesti parhaana menetelmänä dokumenttien osittaistäsmäykseen perustuvassa tiedonhaussa. Baeza-Yatesin ja Ribeiro-Neton [1999] mukaan menetelmä on osoittautunut osassa sovelluksista ylivertaiseksi ja lähes kaikkialla vähintäänkin muiden vaihtoehtojen tasoiseksi menetelmäksi.

4.1.3. Todennäköisyysmalli

Todennäköisyysmallin perusajatuksena on lähtötilanne, jossa käyttäjän antamaan kyselyyn voidaan aina ajatella liittyvän tulosjoukko, joka sisältää kaikki relevantit dokumentit eikä mitään muita dokumentteja [Baeza-Yates and Ribeiro-Neto, 1999]. Hakutehtäväksi muodostuukin tällöin käytännössä etsiä ideaalin tulosjoukon ominaisuudet. Lähtökohdaksi otetaan yleensä arvaus tulosjoukosta, josta pyritään sitten erilaisin menetelmin kohti ideaalia tulosta.

Tavoitteeseen pääsemiseksi voidaan toimia vuorovaikutuksessa käyttäjän kanssa siten, että käyttäjä luokittelee tulosjoukon kärkidokumentit relevantteihin ja ei-relevantteihin. Tämän tiedon avulla järjestelmä tekee uuden arvauksen ja antaa sen edelleen käyttäjällä. Näin iteroimalla lähestytään todennäköisesti koko ajan etsittyä ideaalia tulosjoukkoa. Tämä oli todennäköisyysmalliin perustuvien menetelmien alkuperäinen idea. Sittemmin on kehitetty myös menetelmiä, joissa käyttäjän apua ei tarvita, vaan järjestelmä suorittaa tulosjoukon parantamista omien kriteeriensä perusteella.

Todennäköisyysmalli voi olla jopa vektoriavaruusmallia tehokkaampi, mutta tutkimusten ja tutkijoiden enemmistö on edelleen sitä mieltä, että vektorimalli on yleisessä tilanteessa parempi [Baeza-Yates and Ribeiro-Neto, 1999]. Joka tapauksessa menetelmä on yksi nykyaikaisen tiedonhaun merkittävistä malleista.

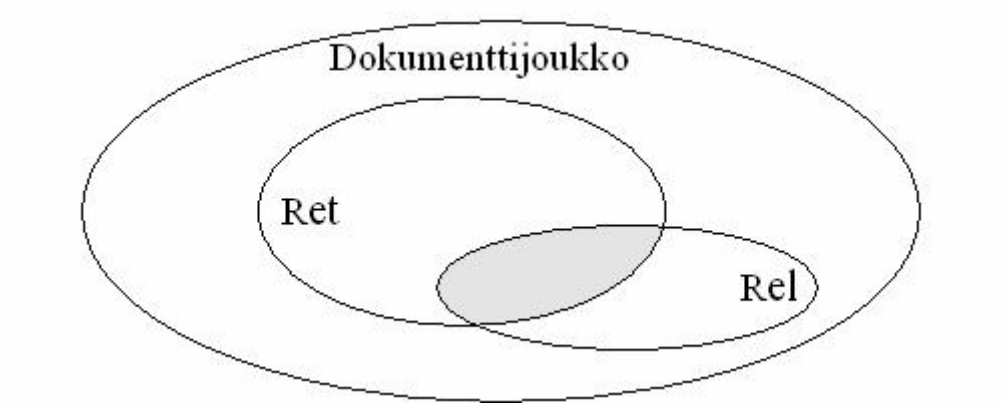
4.2. Tiedonhakumenetelmien evaluointi

Koska tiedonhakumenetelmien kehittämisen tavoitteena on aina pyrkiä vastaamaan mahdollisimman tehokkaasti käyttäjän tiedontarpeeseen, tarvitaan myös mittareita, joiden avulla tehokkuutta voidaan arvioida ja menetelmien suoritusta verrata toisiinsa. Belew [2000] esittää kolmeksi perusmittariksi saannin, tarkkuuden ja unohdetut. Kuvan 4.2 merkinnöillä nämä voidaan esittää seuraavasti:

1. $$saanti \equiv \frac{|Ret \cap Rel|}{|Rel|} \tag{4.10}$$

2. $$tarkkuus \equiv \frac{|Ret \cap Rel|}{|Ret|} \tag{4.11}$$

3. $$unohdetut \equiv \frac{|\overline{Ret} \cap Rel|}{|Rel|} \tag{4.12}$$



Kuva 4.2. Haku dokumenttijoukosta Venn-diagrammiesityksenä. *Ret* on haun palauttama tulosjoukko ja *Rel* on kaikkien relevanttien dokumenttien joukko.

Saanti (recall) kertoo siis, miten suuri osuus koko dokumenttijoukon relevanteista dokumenteista löydettiin. *Tarkkuus* (precision) puolestaan kertoo, miten suuri osa muodostetusta tulosjoukosta oli relevantti. Nämä kaksi mittaria ovat muodostaneet tiedonhakumenetelmien evaluoinnin kivijalan jo 1950-luvulta lähtien [Belew, 2000]. Belew mainitsee kolmantena vähemmän käytettynä perusmittarina *unohdetut* (fallout), joka kertoo miten suuri osa relevanteista dokumenteista jäi kokonaan löytämättä.

Kuvassa 4.2 on esitetty dokumenttijoukko sekä annetun kyselyn (eli haun) tulosjoukko *Ret* ja dokumenttijoukossa olevien kyselyn suhteen relevanttien dokumenttien joukko *Rel*. Harmaalla merkitty alue edustaa niitä relevantteja dokumentteja, jotka kyselyn perusteella löydettiin. Harmaan alueen suurentaminen on tietysti koko tiedonhakumenetelmien kehittämisen tavoitteena. Ihanteellisessa tilanteessa joukkoon *Ret* kuuluvat kaikki joukon *Rel* alkiot, mutta ei mitään muita alkioita. Silloin kuva muuttuisi niin, että *Ret* ja *Rel* joukot olisivat tarkalleen ”päällekkäin” kuvassa. Tällaisessa tilanteessa saanti = 1 (tai 100 %, riippuen esitystavasta) ja tarkkuus = 1 (tai 100 %). Käytännössä tällaiseen tilanteeseen on kuitenkin miltei mahdotonta päästä. Usein onkin tehtävä valinta sen suhteen pyritäänkö saavuttamaan korkea saanti vai tarkkuus [Belew, 2000].

Edellä esitellyt saanti ja tarkkuus muodostavat siis tiedonhakumenetelmien evaluoinnin peruskivet. On kuitenkin olemassa myös muita tapoja tutkia tiedonhakumenetelmän toimivuutta. Salton ja McGill [1983] järjestävät kuusi tärkeintä evaluointikriteeriä seuraavasti:

1. Saanti eli kyky löytää käyttäjälle hyödyllisiä dokumentteja.
2. Tarkkuus eli kyky jättää käyttäjälle hyödyttömät dokumentit ulos tulosjoukosta.

3. Henkinen ja fyysinen työ kyselyn muodostamisessa, hakemisessa ja tulosten tutkimisessa.
4. Aika, joka kuluu kyselyn antamisesta tulosten esittämiseen. Siis käytännössä tulosten prosessointiin kuluva aika.
5. Tuloksen esitystapa suhteessa helppoon hyödynnettävyyteen.
6. Kokoelman kattavuus relevanttien dokumenttien suhteen. Kuinka kattavasti relevantti tieto löytyy kokoelmasta?

Salton ja McGill toteavat, että kahta ensimmäistä lukuun ottamatta kriteerien mittaaminen voi olla melko vaikeaa. Näin onkin varmasti kaikkien paitsi ehkä suoritusajan mittaamisen suhteen. Tutkimalla alan kirjallisuutta on helppo huomata, että saanti, tarkkuus ja suoritusaika ovat tästä joukosta käytetyimmät mittarit järjestelmien evaluoinnissa.

On siis selvää, että saanti ja tarkkuus hallitsevat tiedonhaun evaluoinnin kenttää edelleen ja ovat vakiinnuttaneet asemansa standardimittareiksi. Käytännössä menetelmien vertailussa käytetään yleensä erilaisia saanti-tarkkuuskäyriä, joiden avulla voidaan tutkia menetelmien ja saman menetelmä erilaisten asetusten paremmuutta esimerkiksi useita kyselyitä sisältävien testisarjojen avulla. Erilaisista saantiin ja tarkkuuteen liittyvistä mittareista ja käyristä kattavan esityksen ovat antaneet Baeza-Yates ja Ribeiro-Neto [1999].

4.3. Tiedonhaun evaluoinnin ongelmista

Tiedonhaun evaluointi ei ole käytännössä ihan niin suoraviivaisen ongelmallista kuin edellä annetaan ymmärtää. Ensinnäkin jo relevanssin käsite on osin kiistanalainen. On hyvin epätodennäköistä, että relevanssi yksittäisen dokumentin ja kyselyn välillä on käyttäjän kokemuksesta, ammatista, hakutilanteesta ja muusta dokumenttijoukosta riippumaton käsite.

Lisäksi tiedonhaun evaluoimiseksi on yleensä luotava testijoukko, jonka dokumenttien relevanssi testikyselyiden suhteen on etukäteen päätettävä. Belew [2000] tuo tähän liittyen esille, että asiantuntijan näkemys relevanteista dokumenteista voi poiketa merkittävästi tavallisten käyttäjien näkemyksestä. Tällöin saattaa tulla ongelma, jos asiantuntija päättää, mitkä dokumentit ovat relevantteja, mutta käyttäjät kokevat eri dokumentit hyödyllisiksi. Onhan kuitenkin, niin että yleensä tietoa hakevat ei-asiantuntijat (riippuen toki sovelluksen luonteesta), joten relevanssi olisi ehkä syytä määritellä heidän mielipiteensä suhteen. Jos jostain dokumentista on heille hyötyä, on se melkoisen relevantti vastaus, vaikka asiantuntijan mielestä se ei kuuluisikaan relevanttien joukkoon. Belew [2000] esittää ratkaisuksi eräänlaista konsensukseen perustuvaa relevanssia, jossa relevantti tulosjoukko muodostuu usean käyttäjän esittämien relevanssimäärittelyiden perusteella.

Saantia ja tarkkuuttakin on kritisoitu, sillä ne ovat osittain toisistaan riippuvia mittareita ja esimerkiksi Baeza-Yates ja Ribeiro-Neto [1999] esittävät, että niiden sijasta

voisi olla usein parempi käyttää jotain yhteen arvoon perustuvaa mittaria, kuten erilaisia saannin ja tarkkuuden yhdistäviä tunnuslukuja.

5. Itseorganisoituva kartta tekstitiedonhaussa

Edellä luvussa 3 esiteltiin itseorganisoituvien karttojen perusteet. Luvussa 4 puolestaan keskityttiin tiedonhaun perusteisiin. Tässä luvussa siirrytään soveltamaan itseorganisoituvaa karttaa tekstitiedonhaussa. Käydään kohta kohdalta itseorganisoituvan kartan luomisen vaiheet ja käsitellään niiden erityispiirteet tiedonhaussa pohtien mahdollisia etuja ja ongelmia, jotka kartan soveltamiseen liittyvät. Lopuksi tutustutaan itseorganisoituvan kartan tiedonhakusovelluksiin.

5.1. Dokumenttivektorien luominen

Itseorganisoituvaa karttaa varten data on muunnettava numeeriseen muotoon. Tekstidokumenttidatan tapauksessa luonnollinen ratkaisu on muodostaa jokaisesta dokumentista oma dokumenttivektorinsa vektoriavaruusmallin mukaisesti, niin että dokumenttivektori muodostuu dokumenttijoukon sanojen painoarvoista kyseisessä dokumentissa. Tässä törmätäänkin heti yhteen itseorganisoituvan kartan tekstitiedonhakusovelluksen suurimmista ongelmakohdista. Olisi pystyttävä luomaan dokumenttivektoreita, joissa olisi tarpeeksi tietoa kuvaamaan kutakin dokumenttia niin, että dokumentin ominaisuudet säilyisivät, mutta toisaalta olisi vältettävä liian monen sanan mukaan ottaminen. Ongelma on vaikea, sillä usein ollaan tilanteessa, jossa kokoelmassa on esimerkiksi 1000 dokumenttia, joista löytyy yhteensä 20 000 eri sanaa. Tällöin kaikkien sanojen ottaminen dokumenttivektoreihin johtaa tilanteeseen, jossa dokumenttivektorit muodostavat matriisin, jossa on kaikkiaan 20 miljoonaa alkiota. Tämä saattaa olla laskennallisesti liian raskas tilanne, varsinkin myöhemmin kartan opetusvaiheessa, kun karttaa opetetaan useiden iteraatioiden avulla läpi syötevektorien joukon.

Tarvitaan siis syötevektorien tiivistämistä. Tähän tarkoitukseen on olemassa useita menetelmiä. Kirjallisuudessa on yleisesti käytetty seuraavia:

- sulkusanalista
- sanavartaloiksi muuttaminen
- satunnaismatriisiprojektio
- sanojen klusterointi
- sanakategoriakartta
- idf-valinta
- frekvenssivalinta
- erilaiset vektori- ja matriisimenetelmät.

Sulkusanalistan käyttäminen on tiedonhaussa standardimenetelmä, jossa kokoelman sanastosta poistetaan erillisen kielikohtaisen listan mukaan merkityksettömiä tai merkitykseltään olemattomia täytesanoja. Esimerkiksi saksankielessä poistetaan sanojen sukuja kuvaavat *der*, *das* ja *die*, jotka eivät sinällään kerro dokumentin merkityksestä

mitään. Toinen tiedonhaun perusmenetelmä on sanojen *palauttaminen sanavartaloiksi* (stemming). Tämä tapahtuu poistamalla sanojen taivutuspäätteet. Esimerkiksi suomen kielessä taivutuspäätteiden poistaminen saattaa supistaa sana-avaruutta merkittävästi.

Satunnaismatriisiprojektio (random matrix projection) [Kaski, 1998] on yksinkertainen menetelmä vektoreiden tiivistämiseen. Menetelmässä korkeadimensioinen data muutetaan matalampaan dimensioon matriisiprojektiolla. Tutkimusten mukaan [Bingham and Mannila, 2001; Kaski,1998] matriisiprojektio säilyttää näennäisestä yksinkertaisuudestaan huolimatta vektorien informaation verrattain hyvin, mistä syystä menetelmää käytetään esimerkiksi WEBSOM-järjestelmän [Lagus *et al.*, 2004] vektorien tiivistämisessä.

Sanojen klusteroinnissa dokumenttivektorin kokoa voidaan tiivistää klusteroimalla sanat erilaisiin ryhmiin ja korvaamalla vektorin sanat näillä ryhmillä vektorin alkioina. Siis esimerkiksi, jos sanojen klusterointi liittää sanat auto, rekka ja traktori yhteen ryhmään, voidaan kolmen erillisen sanan (auto, rekka, traktori) sijasta käyttää dokumenttivektorissa alkiona vain yhtä ryhmää. Klusterointi voidaan suorittaa monella tavalla. Esimerkiksi manuaalisesti tai vaikkapa muodostamalla ensin sanoista itseorganisoituva kartta, josta sanojen klusterit saadaan esimerkiksi asettamalla samaan solmuun osuneet sanat samaan ryhmään. Sanakarttaa kutsutaan *sanakategoriakartaksi* (word category map) [Honkela, 1997].

Eräs tapa lähestyä asiaa on tutkia sanojen esiintyvyyttä kokoelmassa. Fernández *et al.* [2004] järjestävät sanat *idf-painon* mukaan järjestykseen ja valitsevat sopivan määrän korkeimmalle rankattuja dokumenttivektoriin. Toinen vaihtoehto on valita sanat niiden frekvenssin suhteen. Belewin [2000] mukaan parhaita indeksisanoja ovat *keskifrekvenssien sanat*, sillä liian usein, tai liian harvoin, esiintyvät sanat eivät erottele dokumentteja samalla tavalla, koska niiden kohdalla suurin osa dokumenteista on samanlaisia keskenään.

Viimeisen kohdan muodostavat erilaiset *matriisilaskennan menetelmät*, joita voidaan soveltaa suoraan dokumenttidataan, koska data on vektoroitu vektoriavaruusmallin mukaan. Esimerkkinä voidaan mainita pääkomponenttianalyysin (principal component analysis, PCA) menetelmät. Suurin ongelma näissä on, että suurilla dokumenttimatriiseilla niiden laskennan vaatimat resurssit saattavat olla liian suuret.

Yksi näkökohta sanaston pienentämiseksi on käyttää itseorganisoituvaa karttaa jonkin tietyn *rajatun aihepiirin* dokumenttien järjestämiseen ja tiedonhakuun, sillä silloin myös dokumenttikokoelmaan liittyvä sanasto on huomattavasti suppeampi kuin esimerkiksi vaikkapa aiheelta yleisten uutisartikkelien kokoelmassa.

Vektorien tiivistämisen lisäksi on valittava, miten vektorit painotetaan. Perinteiset ja käytetyimmät painotusvaihtoehdot ovat samat kuin vektoriavaruusmallissa yleensäkin: binääri-, frekvenssi- tai *tf·idf*-painotus.

5.2. Alustaminen ja oppimisvaihe

Kartan alustamisessa tekstitiedonhakusovellus ei aiheuta juurikaan erityistoimia. Hilaneuronien painovektorien alustaminen voidaan tehdä normaaliin tapaan joko satunnaisesti tai lineaarisesti dokumenttivektorien ominaisuuksien pohjalta. On huomattava, että jos dokumentteja on kokoelmassa paljon ja vektorien dimensio on korkea, saattaa lineaarinen alustus viedä huomattavan paljon aikaa. Kartan koko sen sijaan on merkittävämpi asia tekstitiedonhaun yhteydessä. Kirjallisuudessa (esimerkiksi [Lagus *et al.*, 2004]) suositellaan valitsemaan kartan koko siten, että jokaista kartan solmua kohti on keskimäärin 10 dokumenttia. Tämä on lähinnä käytännöllinen ohje, koska silloin kartan mahdollinen selaaminen on mielekkäämpää, kun kussakin solmussa on todennäköisesti hallittavissa oleva määrä dokumentteja. On kuitenkin selvää, että kartan koolla on paljon merkitystä, koska isommalla kartalla on enemmän mallivektoreita, joihin syötteitä voidaan samaistaa, mikä vaikuttaa varmasti lopputulokseen.

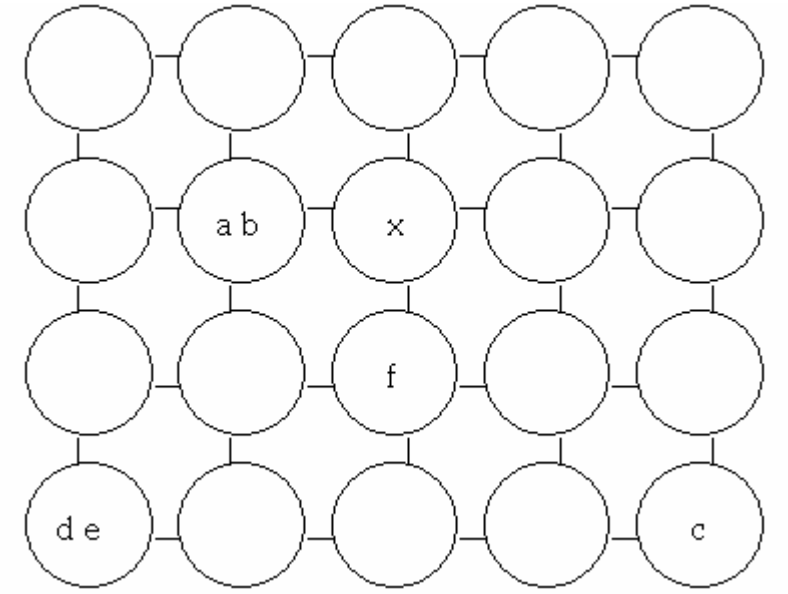
Oppimisvaiheen suhteen ei tiedonhakusovellus luo mitään erityisiä vaatimuksia. On kuitenkin hyvä muistaa, että mitä enemmän dokumentteja, karttasolmuja ja valittuja indeksisanoja on, sitä enemmän opetuskierroksia tarvitaan. Yleinen nyrkkisääntö on, että ainakin 10 kierrosta läpi koko syötejoukon olisi syytä suorittaa aina, jos se laskenta-ajan puitteissa on vain mahdollista. Oppiminen kannattaa jakaa karkea- ja hieno-opetukseen, kuten itseorganisoituvan kartan oppimisessa on tapana.

5.3. Valmis dokumenttikartta

Kun oppimisvaihe on valmis, voidaan syötejoukon dokumenttivektoreille etsiä sijainnit kartalta etsimällä jokaiselle voittajasolmu (BMU) ja sijoittamalla dokumentti sitten voittajan sijaintiin kartalla. Valmiille kartalle voidaan myös sijoittaa sanoja otsikoiksi, jotka kuvaavat eri alueiden sisältöä [Lagus and Kaski, 1999] [Azcarraga and Yap Jr., 2001]. Kartan visualisoimiseen on olemassa erilaisia työkaluja [Yang *et al.*, 1999] [Vesanto *et al.*, 2000].

Nyt kun dokumentit ovat kartalla, voidaan kartalle tehdä kyselyitä. Kyselyiden muodostamisessa voidaan käyttää samaa periaatetta kuin vektoriavaruusmallissa, eli luoda myös kyselyistä vektoreita. Itseorganisoituvan kartan tapauksessa luotuja kyselyvektoreita verrataan valmiin kartan solmujen painovektoreihin ja lähimmän painovektorin solmu vastaa kyselyn osumakohtaa kartalla. Käytännössä yksinkertaisessa toteutuksessa lähimmän solmun sisältämät dokumentit muodostavat tulosjoukon kyselylle. Myös monimutkaisempia toteutuksia tulosjoukon muodostamiseksi voidaan kehittää. Jos ajatellaan yksinkertaista toteutusta, jossa kyselyn osumakohdan dokumentit muodostavat tulosjoukon, on mielenkiintoinen ja ratkaiseva kysymys se, miten hyvin samanaiheiset dokumentit järjestyvät samalle alueelle kartalla ja miten hyvin kysely osuu tuolle alueelle. Kartalla dokumentit ovat Kohosen algoritmin seurauksena järjestyneet ominaisuuksiensa mukaan siten, että samanlaiset dokumentit löytyvät samasta

karttasolmusta tai naapurisolmuista ja täysin erilaiset dokumentit sijoittuvat kauas toisistaan. Järjestyminen perustuu tekstitiedonhakutilanteessa dokumenttien sanavektorien vertailun perusteella, jolloin vektorin hahmo tavallaan kuvaa dokumentin aihetta, minkä pitäisi näkyä myös kartalla aiheiden säilymisenä. Kuvassa 5.1 on kuvattu esimerkki valmiista dokumenttikartasta, johon on merkitty dokumentit *a*, *b*, *c* ,*d* ,*e* ,*f* ja *x*. Dokumentit *a* ja *b* ovat keskenään samankaltaisia, samoin *d* ja *e*. Dokumentti *x* on puolestaan ominaisuuksiltaan lähellä dokumentteja *a*, *b* ja *f*, kun *c* on puolestaan hyvin erilainen kaikkiin muihin verrattuna. Itseorganisoituva kartta siis kuvaa dokumenttien samanlaisuutta etäisyyksinä kartalla. Tekstitiedonhaun kannalta olisi tietenkin toivottavaa, että tuo samanlaisuus vastaisi dokumenttien aiheiden samanlaisuutta. Tämä on tietenkin riippuvainen siitä, miten dokumenttivektorit on painotettu.



Kuva 5.1. Valmis dokumenttikartta.

Entä mitkä voisivat olla itseorganisoituvan kartan edut tekstitiedonhaussa? Ensimmäisenä huomataan helposti, että kartan käyttäminen vähentää laskenta-aikaa, koska kyselyvektoreita verrataan kartan painovektoreihin eikä dokumenttivektoreihin. Jos kartalla on esimerkiksi keskimäärin 10 dokumenttia jokaista solmua kohti, kuten standarditilanteessa yleensä on, on vertailuja kyselyiden ja painovektorien välillä tehtävä vain kymmenesosa siitä, mitä suoraan dokumenttivektoreihin verrattaessa olisi tehtävä. Tietenkin vastaavasti kartan luominen vie huomattavan kauan aikaa, mutta kartan järjestäminen voidaan tehdä etukäteen kokoelmalle, jolloin sitä ei tietenkään suoriteta jokaisen kyselyn kohdalla uudestaan. Dokumentteja lisättäessä kartta voidaan aina luoda

uudestaan. On kuitenkin huomattava, että kartan luominen uudestaan ei muutaman uuden dokumentin lisäämisen jälkeen ole tarpeellista, koska kartta osaa yleistämällä sijoittaa uuden dokumentin vanhalle kartalle, jonka opettamisessa kyseistä dokumenttia ei ole käytetty. Kun uusia dokumentteja kertyy merkittävästi, on kuitenkin syytä luoda kartta uudestaan, koska suuri määrä uusia syötteitä voi vaikuttaa kartalle muodostuviin alueisiin.

Toinen etu on se, että kartalla osumasolmuun saattaa järjestyä myös relevantteja dokumentteja, jotka eivät vastaa täysin annettua kyselyä sanojen suhteen, mutta vastaavat muita solmun dokumentteja sanastoltaan siinä määrin, että ovat järjestyneet kyseiseen solmuun. Tämä saattaisi parantaa hakutulosta tilanteessa, jossa esiintyy relevantteja dokumentteja, joiden sanasto poikkeaa kyselystä ja jotka jäisivät perinteisessä hakumenetelmässä ehkä tästä syystä löytämättä. Tämä ilmiö kuitenkin vaatisi lisää tutkimusta, sillä toisaalta tämä ominaisuus voi myös heikentää tulosta.

Kolmas, ja ehkä ainutlaatuisin, etu on siinä, että itseorganisoituva kartta muodostaa kokoelmasta intuitiivisesti selattavan kartan kaksiulotteiselle tasolle, joka on helppo esittää näytöllä. Yhdistettynä selaintyökaluun kartta tuo merkittävää lisäarvoa hakujärjestelmälle, koska käyttäjä voi halutessaan selata järjestettyä dokumenttikokoelmaa helposti omaksuttavassa karttamuodossa. Toiselta kannalta katsottuna itseorganisoituva kartta visualisoi dokumenttikokoelman kaksiulotteiselle kartalle, mikä on jo itsessään hyödyllinen ominaisuus, sillä tällöin mistä tahansa dokumenttijoukosta voidaan muodostaa kartta, josta voidaan tutkia joukon suhteita ja rakennetta.

5.4. Itseorganisoituvan kartan tiedonhakusovelluksista

Itseorganisoituvaa karttaa on sovellettu melko paljon tiedonhaun alalla. Erityisesti sen ominaisuus organisoida kokoelma ”itsestään” kartalle luokitellen on tehnyt siitä merkittävän välineen luokittelun ja klusteroinnin sovelluksissa. Merkittävästi vähemmälle on jäänyt kartan soveltaminen varsinaisesti hakutilanteessa. Tärkein sovellus on kenties WEBSOM-järjestelmä [Lagus *et al.*, 2004], jonka kehittämisessä on ollut mukana itseorganisoituvan kartan kehittäjä Teuvo Kohonen. WEBSOM on erityisesti laajojen tekstikokoelmien automaattiseen järjestämiseen ja selaavaan tutkimiseen kehitetty järjestelmä, jonka avulla käyttäjä voi etsiä tietoa kokoelmasta pääosin selaamalla kokoelmaa, joka on järjestetty kaksiulotteiselle kartalle. Järjestelmään tutustutaan tarkemmin luvussa 5.5, mutta tarkastellaan sitä ennen joitain esimerkkejä siitä, mitä muuta itseorganisoituvalla kartalla on tehty tiedonhaun alalla.

Jo 90-luvun alussa Lin *et al.* [1991] esittivät itseorganisoituvaan karttaan perustuvan tiedonhakujärjestelmän, jossa dokumentit on organisoitu kartalle, jossa kokoelman selaaminen on käyttäjälle helppo ja intuitiivista. Myöhemmin Lin [1997] on pohtinut esimerkiksi itseorganisoituvaa karttaa yleisenä tiedonkuvausmenetelmänä soveltaen sitä esimerkiksi tiedonhaussa suuren tulosjoukon merkitykselliseen järjestämiseen. Fernández

et al. [2004] ovat kehittäneet kahdesta kartasta, erisnimien ja muiden sanojen kartoista, muodostuvan järjestelmän, jonka avulla haku espanjankielisestä kokoelmasta on tehokasta. Myös Lee ja Yang [1999] käyttävät kahta karttaa, joista toinen sisältää sanojen luokittelun ja toinen dokumenttien luokittelun, suunnittelemassaan verkkotiedonhakusovelluksessa. Chowdhury ja Saha [2005] puolestaan ovat luokitelleet urheiluartikkelien kokoelmia menestyksellisesti todella mielenkiintoisella itseorganisoituvaan karttaan perustuvalla sovelluksella. Guerrero Bote *et al*. [2002] sekä Moya-Anegón *et al*. [2006] soveltavat menetelmää tieteellisten dokumenttien klusterointiin, joka helpottaa aihepiireihin liittyvän tiedon ja asiayhteyksien omaksumista. Merkl [1998] on kehittänyt hierarkkisen itseorganisoituvan kartan, jossa on useita karttoja päällekkäin hierarkkisessa rakenteessa. Tätä sovellusta ovat tutkineet myös Dittenbach *et al*. [2002]. Tangsripairoj ja Samadzadeh [2005] ovat soveltaneet samaa perusideaa käyttämällä monikerroksisia karttoja dokumenttien luokitteluun. Edellä mainitut esimerkit luovat hyvän kuvan siitä, millaisissa tehtävissä itseorganisoituvaa karttaa yleensä tiedonhaussa sovelletaan. Kuten huomataan, ovat pääosassa luokittelun ja klusteroinnin sovellukset.

Perinteisempien sovelluskohteiden lisäksi itseorganisoituva kartta soveltuu monenlaisiin muihinkin tehtäviin. Tästä esimerkkinä voidaan mainita Lindénin väitös [2005], jossa sovelletaan karttaa sanojen monimerkityksellisyyden ongelman ratkaisemisessa. Toinen tavallisimmista sovelluksista poikkeava esimerkki on Marinai *et al*. [2006], jossa itseorganisoituvaa karttaa käytettään kuvaperustaisen sanahaun sovelluksessa.

5.5. WEBSOM

WEBSOM-järjestelmä [Lagus *et al*., 2004] on ohjelmisto, joka on tarkoitettu laajojen tekstidokumenttikokoelmien järjestämiseen kaksiulotteiselle kartalle merkitykselliseen järjestykseen. Sen tavoitteena on helpottaa kokoelman selaamista ja mahdollistaa sanahakujen tekeminen kartalle. Järjestelmä perustuu Kohosen [1995] itseorganisoituvan kartan menetelmään, joten luodulla kartalla samankaltaiset dokumentit sijaitsevat samassa karttasolmussa tai naapurisolmuissa. WEBSOM on kiistatta yksi tärkeimmistä itseorganisoituvan kartan tiedonhakusovelluksista. Sen tärkein ominaisuus on sen kyky järjestää erittäin suuria kokoelmia helposti selattavalle kartalle. Kartalle on selaamisen helpottamiseksi lisätty sanoja, jotka kuvaavat kutakin aluetta parhaiten. Lisäksi kartan väritys kuvaa vierekkäisten alueiden samankaltaisuutta. Joissain sovelluksissa WEBSOM tarjoaa myös mahdollisuuden sanahakuun, joka etsii käyttäjälle sopivimman tai sopivimpia sijainteja selaamisen aloittamiseksi.

Tarkastellaan seuraavaksi kolmea WEBSOM-järjestelmän sovellusta keskittyen lähinnä niiden kartan asetuksiin sekä käytettyjen kokoelmien ominaisuuksiin.

5.5.1. Patenttikokoelma

Suurin kokoelma, joka WEBSOM-menetelmällä on järjestetty, on peräti 6 840 568 englanninkielistä patenttiabstraktia käsittänyt kokoelma. Järjestämisessä käytettiin 1 002 240 solmua sisältänyttä karttaa [Kohonen *et al.*, 2000].

Datan esiprosessoinnissa sanat muutettiin perusmuotoihinsa, jolloin jäljelle jäi 733 179 sanaa. Tästä määrästä poistettiin kaikki alle 50 kertaa kokoelmassa esiintyvät sanat sekä merkityksettömät sanat sulkusanalistan perusteella, minkä jälkeen sanastoon jäi 43 222 sanaa. Tämän jälkeen kokoelmasta poistettiin 122 524 abstraktia, joihin jäi esiprosessoinnin jälkeen alle viisi sanaa jäljelle. Vektorien tiivistäminen suoritettiin satunnaismatriisiprojektiolla [Kaski, 1998], jolloin 43 222 sanasta jäi jäljelle 500 sanaa. Dokumenttivektorit luotiin vektoriavaruusmallin mukaan käyttäen entropiaperusteista (Shannonin entropia, katso [Kohonen *et al.*, 2000]) sanapainotusta.

Kartan luominen toteutettiin käyttämällä yhdistetysti alkuperäistä itseorganistoituvan kartan algoritmia sekä uudempaa eräalgoritmia ja muita WEBSOM-ryhmän kehittämiä optimoituja menetelmiä, kuten kartan suurentamista opetuksen edetessä. Laskenta kesti noin kuusi viikkoa. Lopuksi luotiin vielä automaattisesti käyttöliittymä, mikä kesti noin viikon. Käyttöliittymä mahdollistaa kokoelman selaamisen annettujen vihjesanojen avulla sekä sanahaun selaamiselle sopivien aloituspisteiden löytämiseksi.

Kartan suoritusta mitattiin laskemalla jokaiselle solmulle luokka ja tutkimalla, miten suuri osa kyseisen solmun patenteista kuului tuohon luokkaan. Tulokseksi saatiin 64 %.

5.5.2. Encyclopaedia Britannica -kokoelma

Tässä sovelluksessa kokoelmana toimi Encyclopaedia Britannicasta valitut 68 000 artikkelia sekä lisäksi muuta materiaalia 43 000 artikkelin verran. Pisimmät artikkelit jaettiin kahtia, joten lopputulokseksi saatiin noin 115 000 tekstidokumenttia.

Esiprosessointivaiheessa poistettiin HTML-merkkaus, numerot ja kuvat ja loput sanat muutettiin perusmuotoihinsa, minkä jälkeen sanastossa oli 325 275 sanaa ja keskimääräinen dokumentin pituus oli 490 sanaa. Sulkusanalista hyödyntämisen ja alle 30 kertaa esiintyvien sanojen poistamisen jälkeen sanastoon jäi jäljelle enää 39 058 sanaa. Vektorit tiivistettiin vielä käyttämällä satunnaismatriisiprojektiota, jolla sanasto supistettiin 1000 sanan kokoon. Vektorit painotettiin idf-painoilla. Kartan kooksi valittiin 72 × 168. ja kartta luotiin eräalgoritmin ja optimoitujen menetelmien avulla.

Kartalle luotiin myös käyttöliittymä, jolla käyttäjän apuna toimivat solmujen läheisyyttä kuvaava väritys sekä apusanat, jotka muodostettiin kartalle automaattisesti dokumenttien järjestymisen pohjalta. Myös sanahakutoiminto tarjottiin, jolla käyttäjä löytää sopivia aloituspaikkoja selaamiselle.

5.5.3. CISI-kokoelma

WEBSOM-ryhmään kuuluva Krista Lagus tutki itseorganisoituvaan karttaan perustuvaa tiedonhakusovellusta CISI-kokoelmalla [Lagus, 2002]. Sovellus ei varsinaisesti perustu

WEBSOM-järjestelmän käyttämiseen, sillä Lagus soveltaa tässä itseorganisoituvaa karttaa hieman eri tavalla, mutta kuitenkin WEBSOM-järjestelmästä hankittujen kokemusten pohjalta. Sovellus on mielenkiintoinen siinä mielessä, että tutkimuksessa on evaluoitu kehitetyn sovelluksen hakujen tehokkuutta.

Toteutetussa systeemissä itseorganisoituvan kartan avulla dokumentit jaetaan luokkiin, joita edustaa yksi keskimääräinen vektori. Kyselyä verrataan edustajavektoreihin ja vertailun perusteella valitaan parhaat luokat dokumentteineen mukaan tulosjoukkoon, niin että haluttu määrä dokumentteja saadaan kokoon. Tästä tulosjoukosta etsitään sitten haluttu määrä parhaita dokumentteja lopulliseen tulosjoukkoon vertaamalla niitä suoraan kyselyyn.

Käytetyssä CISI-kokoelmassa on 1460 dokumenttia ja 76 kyselyä. Keskimäärin jokaiseen kyselyyn liittyy 41 tunnettua relevanttia dokumenttia. Dokumenttien keskimääräinen pituus on 115 sanaa ja kyselyiden 51 sanaa.

Kartta luotiin kaikista 1460 dokumentista luomalla 150 solmun kartta, jolloin solmua kohti dokumentteja on keskimäärin noin kymmenen. Kokoa perusteltiin sillä, että kyseinen määrä dokumentteja solmua kohti on sopiva kartan selaamista ajatellen.

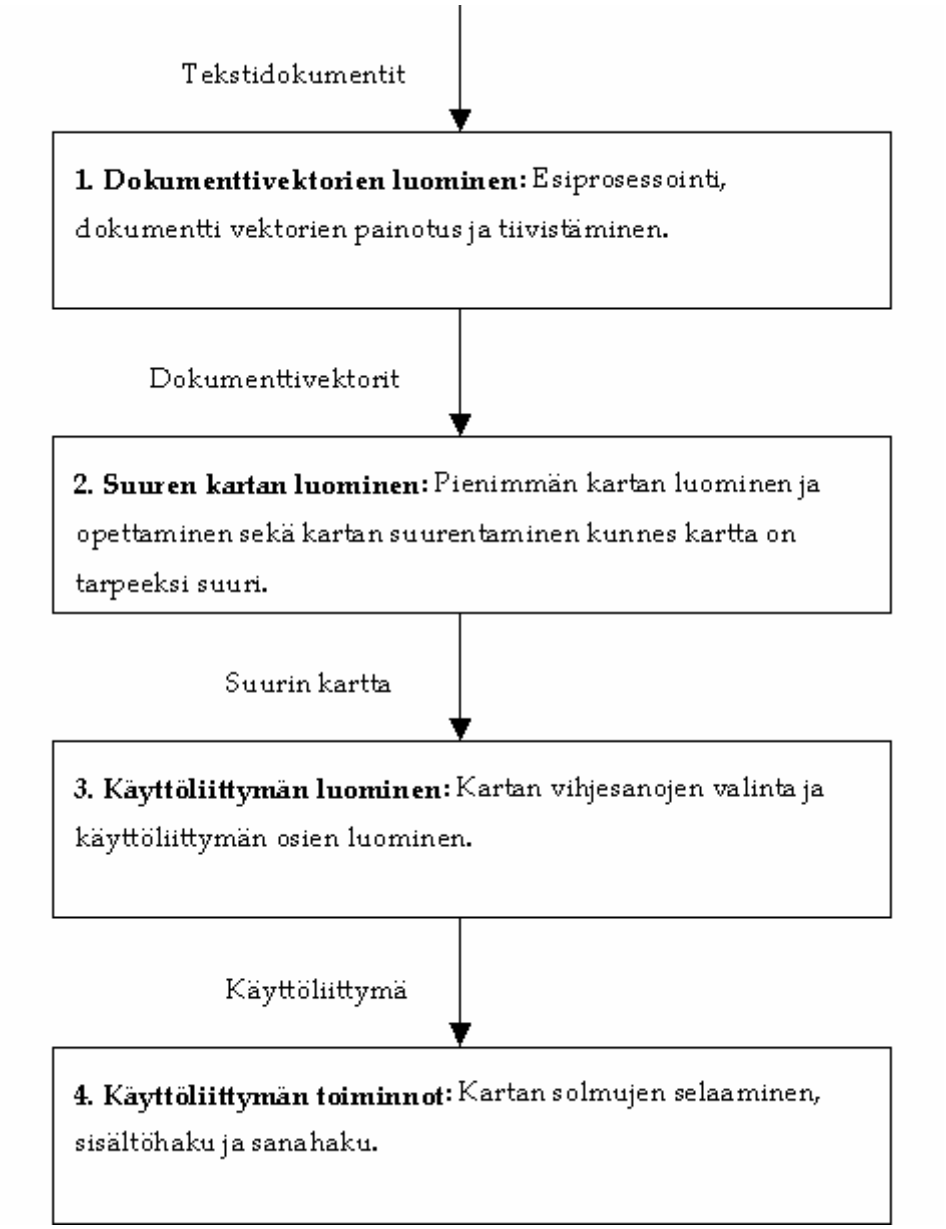
Tutkimuksen tuloksena itseorganisoituvan kartan suoriutuminen tiedonhausta verrattuna perinteisiin menetelmiin osoittautui lupaavaksi.

5.5.4. WEBSOM-järjestelmän perustoteutus

Edellä käsitellyt esimerkit antavat kohtuullisen hyvän kuvan WEBSOMin soveltamisesta tiedonhaussa. Annetaan vielä lopuksi yleinen malli siitä, miten WEBSOM-järjestelmän perustoteutus toimii. Kuvassa 5.2. on esitetty WEBSOMin malli Kohonen *et al.* [2000] pohjalta.

Järjestelmä saa siis syötteenään tekstidokumentteja. Ensimmäisessä vaiheessa luodaan dokumenttivektorit. Aivan aluksi dokumenttidata esiprosessoidaan samalla vähentäen sanojen määrää kokoelmassa. Seuraavaksi luodaan varsinaiset dokumenttivektorit ja päätetään sanojen painotusvektoreissa. Tämän jälkeen on vuorossa yksi tärkeimmistä vaiheista, joka on syötevektorien tiivistäminen.

Valmiit dokumenttivektorit annetaan eteenpäin vaiheeseen, jossa varsinainen kartta luodaan. Ensin luodaan pieni kartta perinteisellä itseorganisoituvan kartan algoritmilla ja sitten suurennetaan kartta suuremmaksi erityisellä laajennusmenetelmällä ja hienosäädetään eräalgoritmilla. Tätä toistetaan kunnes kartta on saavuttanut halutun kokonsa ja kartta on valmis.



Kuva 5.2. WEBSOM-järjestelmän perusmalli

Valmiin kartan pohjalta luodaan käyttöliittymä. Ensimmäinen vaihe on kuvaavien vihjesanojen valitseminen kullekin kartan osalle. Nämä lasketaan kartta-alueiden sisältämien dokumenttien sanastojen pohjalta. Kun vihjesanat ja kartta ovat valmiit, luodaan muut käyttöliittymäkomponentit.

Lopulta saadaan tuloksena valmis kartta, jonka pääominaisuudet ovat:

1. Käyttäjä voi selata kokoelmaa solmu kerrallaan navigoiden.
2. Käyttäjä voi hakea parhaita karttasijainteja sisältöhaulla.
3. Käyttäjä voi hakea parhaita sijainteja sanahaun avulla.

WEBSOMin perusmalli on erittäin hyvä lähtö- ja vertailukohta tässä tutkimuksessa toteutetulle hakukoneelle. Se on kuitenkin selvästi kehittyneempi kuin tutkielmassa kehitetty yksinkertainen prototyyppi. Lisäksi WEBSOM-ryhmä on käyttänyt toteutuksissaan omia optimointimenetelmiään ja uutta eräalgoritmia. Näitä menetelmiä ei tässä tutkielmassa käytetä, vaan testataan itseorganisoituvan kartan perusalgoritmia.

6. Hakukoneen toteutus

Tutkielman tavoitteena oli tutkia, miten hyvin itseorganisoituva kartta säilyttää syötteenään saamiensa dokumenttien väliset suhteet niiden käsittelevien aihepiirien suhteen. Käytännössä tavoitteena oli siis toteuttaa eräänlainen dokumenttien karttaesitykseen perustuva hakukoneprototyyppi, jolla tekstidokumenttien sanahaku olisi mahdollisimman tehokasta. Hakukoneen sovelluskohteeksi valittiin 1160 saksankielisestä uutisartikkelista koostuva kokoelma. Seuraavaksi esitellään ensin dokumenttikokoelma ja sen jälkeen hakukoneen suunnittelu ja toteutus.

6.1. Aineisto

Dokumenttikokoelmaksi valittiin 1160 saksankielistä uutisartikkelia. Artikkelit valittiin suuresta saksankielisten uutisartikkelien kokoelmasta, johon kuuluu yhteensä 294 810 uutista vuosilta 1994 ja 1995. Uutiset ovat ilmestyneet muun muassa Frankfurter Allgemeinessa ja Der Spiegelissä. Kokoelmaan liittyy 60 kyselyä ja tieto kuhunkin liittyvistä relevanteista dokumenteista.

Lopullinen 1160 artikkelin kokoelma valittiin seuraavasti. Ensin valittiin satunnaisesti 20 kyselyä ja otettiin dokumenttikokoelmaan niihin liittyvät 580 relevanttia uutista. Jokaista kyselyä kohti kokoelmaan saatiin siis keskimäärin 29 relevanttia dokumenttia. Tämän jälkeen valittiin satunnaisesti toiset 580 uutista alkuperäisestä lähes 300 000 uutisen kokoelmasta. Näin saatiin yhteensä 1160 dokumenttia sisältävä kokoelma ja siihen liittyen 20 kyselyä.

Sekä uutisaineisto, että kyselyt olivat SGML-muotoista tekstidataa. Uutisesta löytyy seuraavat ominaisuuskentät:

1. dokumentin id-numero
2. kieli
3. päivämäärä
4. otsikko
5. teksti.

Joistakin uutisista puuttuu esimerkiksi otsikko- tai kielikenttä, mutta kaikista löytyvät id-numero ja teksti. Esimerkki uutinen löytyy kuvasta 6.1, jossa on tekstikentän osalta lyhennetty esimerkki satunnaisesta SGML-muotoisesta uutisesta kokoelmassa. Kyselyistä puolestaan löytyvät:

1. kyselyn numero
2. kyselyn otsikko
3. varsinainen kysely
4. kyselyn selite.

Näistä kentistä kyselyn muodostamiseen käytetään yleisen tavan mukaan kyselyn otsikkoa ja varsinaista sisältöä. Kyselyn pidempää selitettä ei käytetä yleensä kyselyn muodostamiseen. Esimerkki kyselystä löytyy kuvasta 6.2.

```
<DOC>
<DOCID>FR940515-000383</DOCID>
<DOCNO>FR940515-000383</DOCNO><LANGUAGE>german</LANGUAGE>
<DATE>19940515</DATE>
<TITLE>Ahornblatt nach 33 Jahren vergoldet</TITLE>


<TITLE>Zum 20. Mal Eishockey-Weltmeister</TITLE>


<TITLE>Sieg im Penaltyschießen</TITLE>


<TITLE>Finnland </TITLE>


<TEXT>Das Eishockey-Mutterland Kanada ist nach 33 Jahren wieder die
Nummer eins in der Welt. Durch einen 3:2-Erfolg im Penaltyschießen
gegen Finnland lösten die Ahornblätter im WM-Finale in Mailand den einst
übermächtigen Rivalen und Titelverteidiger Rußland ab, der bereits im
Viertelfinale gegen die USA (1:3) ausgeschieden war. Nach regulärer
Spielzeit und Verlängerung hatte es 1:1 (0:0, 0:0, 1:1, 0:0) gestanden.
Zuvor hatte Brind'Amour (56.) die Führung der Finnen durch Keskinen
(47.) ausgeglichen. Im Penaltyschießen zeigten die Kanadier die
besseren Nerven, die Finnen verschossen viermal in sechs Versuchen.
Robitaille verwandelte den sechsten Penalty für Kanada.
.
.
.
</TEXT>
</DOC>
```

Kuva 6.1. Esimerkki eräästä kokoelmaan kuuluvasta uutisesta SGML-muodossa (uutisen sisältöä on lyhennetty tilan säästämiseksi).

```
<top>
<num> C171 </num>
<DE-title> Eishockeyfinale in Lillehammer </DE-title>
<DE-desc> Welche Mannschaften spielten im Eishockeyfinale der Olympischen Spiele
von Lillehammer 1994?
</DE-desc>
<DE-narr> Relevante Dokumente berichten darüber, welche Teams im Eishockeyfinale
der Olympischen Spiele von Lillehammer 1994 spielten. Dokumente, die darüber
informieren, welches Team den ersten und zweiten Platz der Veranstaltung
belegte, ohne speziell das Finale zu erwähnen, sind ebenfalls relevant.
</DE-narr></top>
```

Kuva 6.2. Esimerkki eräästä kokoelmaan kuuluvasta kyselystä SGML-muodossa.

6.2. Työkalut

Hakukoneen prototyypin toteuttamiseksi oli käytettävissä seuraavat valmiit työkalut:

1. sanavartaloon palauttamisen toteuttava ohjelmisto
2. sulkusanalistaohjelmisto
3. sanojen dokumenttifrekvenssien laskemiseen tarkoitettu työkalu
4. SOM Toolbox
5. SOM_PAK.

Käytettävissä ollut sanavartalo-ohjelmisto pystyy muuntamaan annetun saksankielisen sanan sen sanavartaloksi. Seuraavaksi ohjelmiston toiminnan valaisemiseksi muutama esimerkki saksankielisten sanojen muuntamisesta sanavartaloikseen:

Reisimporte ► reisimport

Olympische ► olymp

Antike ► antik .

Sulkusanapoistojen tekeminen onnistuu myös automaattisesti sanavartalo-ohjelmistolla. Sulkusanalistalle (kuva 6.3) kuuluvat sanat ovat yleisesti tunnettuja merkityksettömiä saksankielisiä sanoja. Yhteensä listalle kuuluu 1320 sanaa.

```
Ab
ab
Aber
aber
Abseits
abseits
Absolut
absolut
Abwärts
abwärts
All
all
Alle
alle
Allein
allein
Allemal
allemal
Allenfalls
allenfalls
Aller
aller
Allerdings
allerdings
Allerhand
allerhand
Allerlei
allerlei
Allezeit
allezeit
Allgemein
allgemein
Allmählich
```

Kuva 6.3. Saksankielisen sulkusanalistan alkupään sanoja.

Sanojen dokumenttifrekvenssien laskemiseen tarkoitettu työkalu puolestaan pystyy laskemaan SGML-muotoisista tekstidokumenteista niissä esiintyvien sanojen frekvenssit. Kuvassa 6.4 on esimerkki erään uutisdokumentin sanojen frekvensseistä laskettuna edellä mainitulla työkalulla. On huomattava, että ennen frekvenssien laskemista on kuvassa esitetty uutinen käsitelty sekä sanavartaloiden että sulkusanojen poistamisen suhteen, kuten edellä esiteltiin.

```
###
ruckwirk  2
bundessozialgericht  1
anerkannt  1
beginnt  1
februar  1
asylbewerb  1
nordrhein  1
anspruch  1
urteil  1
erziehungsgeld  1
reg  1
aufenthaltserlaubnis  1
aufgehob  1
bekomm  1
gilt  1
gegenteil  1
erteil  1
entschied  1
kassel  2
unbefristet  1
betreu  1
ap  1
az  1
kind  1
fr  2
german  1
demnach  1
westfal  1
```

Kuva 6.4. Esimerkki erään uutisen sanojen frekvensseistä.

SOM Toolbox [Vesanto *et al.*, 2000] on Matlab-ohjelmiston lisäpaketti, joka on verkossa vapaasti saatavilla tutkimustyötä varten. Pakettiin kuuluu useita hyödyllisiä itseorganisoituvien karttojen luomiseen liittyviä työkaluja. Paketista löytyy muun muassa kartan alustamiseen, oppimiseen, visualisoimiseen ja laadun tarkasteluun liittyviä työkaluja. SOM Toolbox on helppokäyttöinen Matlabin graafisella käyttöliittymällä käytettävä ohjelmistopaketti, joka soveltuu lähinnä suhteellisen pienten itseorganisoituvien karttojen toteuttamiseen. Paketista löytyy sekä perinteisen itseorganisoituvan kartan että uuden eräalgoritmin toteutukset.

SOM_PAK [Kohonen *et al.*, 1996] on verkossa vapaasti tutkimustarkoituksiin saatavilla oleva C-kielellä toteutettu itseorganisoituvien karttojen rakentamiseen suunniteltu ohjelmistopaketti. SOM_PAK on komentorivityökalu, jolla ei ole graafista

käyttöliittymää. Se on kuitenkin selvästi tehokkaampi kuin SOM Toolbox, joten se soveltuu myös erittäin suurten karttasovellusten luomiseen. Paketista löytyy työkaluja kartan alustamiseen, oppimiseen, arviointiin ja kevyeen visualisointiin. Paketti ei tue uutta eräalgoritmia, vaan kaikki sen toiminta perustuu alkuperäiseen itseorganisoituvan kartan algoritmiin.

6.3. Hakukoneen suunnittelu

Tavoitteena oli suunnitella prototyyppi suhteellisen yksinkertaisesta hakukoneesta, jolla voitaisiin tekstidokumenteista muodostaa itseorganisoituvan kartan algoritmilla kaksiulotteisia karttoja, joilta sanahaku olisi mahdollisimman tehokasta. Suunnittelun perusteina olivat seuraavat ominaisuudet:

- Järjestelmä saa syötteenään tekstidokumentteja SGML-muodossa.
- Tavoitteena on muodostaa itseorganisoituva dokumenttikartta.
- Aihepiirit merkitään valmiille kartalle sanoina.
- Kartalle levitetyn kokoelman selaaminen oltava mahdollista käyttäjälle.
- Käyttäjän sanahakujen mahdollistaminen kartalta.
- Haun tuloksena on annettava parhaiten täsmäävä sijainti kartalta ja sen sisältämät dokumentit.

Suunnittelun lähtökohtana ja tavoitteena oli siis luoda järjestelmä, joka syötteenään saamiensa SGML-muotoisten tekstidokumenttien pohjalta kykenee luomaan itseorganisoituvan kartan algoritmia hyväksikäyttäen dokumenttikartan, jolla käyttäjä voi liikkua ja selata kokoelmaa vapaasti. Kartalle sijoitellaan sanoja kuvaamaan eri aihealueiden sijainteja. Käyttäjällä on mahdollisuus antaa sanahakuja, joiden tuloksena järjestelmä osoittaa käyttäjälle sijainnin kartalta, joka parhaiten vastaa annettua kyselyä. Löydettyään parhaan sijainnin kartalla käyttäjä voi jatkaa tiedonhakua siirtymällä viereisiin solmuihin tai antamalla uuden mahdollisesti paremman kyselyn ja siirtymällä sitä vastaavaan sijaintiin.

Tavoitteena oleva järjestelmä vastaa siis melko pitkälle WEBSOM-järjestelmän ominaisuuksia, mutta tässä tutkielmassa sen toteuttamiseen käytetyt välineet poikkeat hivenen WEBSOMin toteutuksesta. Toiseksi tässä tutkimuksessa toteutetaan vain yksinkertainen prototyyppi, jolla voidaan testata menetelmän toimivuutta tiedonhaussa. Alkuperäinen motivaatio tutkimukselle olikin toteuttaa WEBSOMia vastaava järjestelmä ja tutkia sen toimivuutta hakutilanteessa ja kenties parannella sen ominaisuuksia niin, että se toimisi tehokkaammin tiedonhaussa.

6.4. Hakukoneen prototyypin toteutus

Tämän tutkielman tavoitteena on toteuttaa vain prototyyppi hakukoneesta. Käytännössä tavoite tarkoittaa, että toteutetaan kaikki välttämättömät osat hakukoneen suorituksen testaamiseksi, mutta esimerkiksi näiden osien yhdistäminen yhdeksi toimivaksi ohjelmistoksi ja käyttöliittymän toteuttaminen jätetään pääosin tekemättä. Tavoitteena

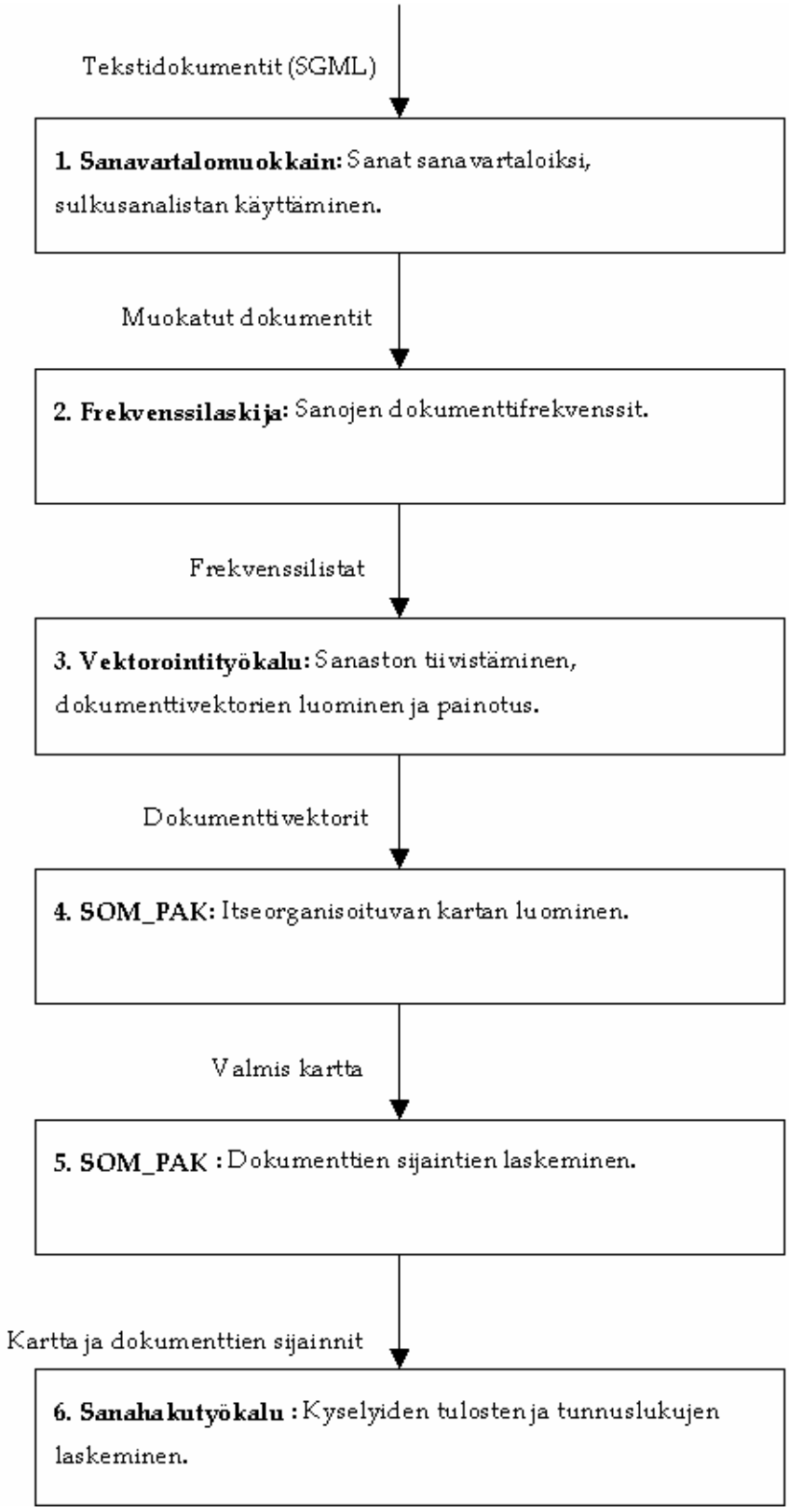
on siis saada toteutettua testattava prototyyppi, joka edustaa suunnitellun hakukonetta tehokkuudeltaan. Tällöin edellä luvussa 6.3 esitetyistä ominaisuuksista jäävät pois kohdat, jotka liittyvät käyttöliittymään, siis kuvaavien sanojen levittäminen kartalle ja käyttäjän selausmahdollisuus. Selaaminen on kuitenkin itseorganisoituvan kartan ominaisuuksista johtuen helposti toteutettavissa ja vihjesanojen valitsemiseen on olemassa valmiita menetelmiä [Lagus and Kaski, 1999]. Voidaan siis sanoa, että näiden prototyypin ulkopuolisten osien toteuttaminen on helppoa, jos suunniteltu hakukone halutaan toteuttaa kokonaisuudessaan. Edellä esiteltyjä periaatteita noudattaen toteutettiin hakukoneen prototyyppi, joka koostuu kuvan 6.5 mukaisesti seuraavista osista:

1. sanavartalomuokkain ja sulkusanojen poistaja
2. frekvenssilaskija
3. vektorointityökalu: dokumenttivektorien ja kyselyiden luominen
4. SOM_PAK: kartan luominen
5. SOM_PAK: dokumenttien osumakohdat
6. sanahakutyökalu: kyselyiden tulokset.

Ensimmäinen osa muodostuu valmiista sanavartalo-ohjelmistosta, joka saa syötteenään tekstidokumentteja SGML-muodossa. Ohjelmisto muokkaa sanat sanavartaloiksi ja poistaa samalla sulkusanalistalle kuuluvat sanat. Tämän jälkeen frekvenssejä laskeva työkalu laskee kullekin dokumentille siinä esiintyvien sanojen frekvenssit.

Frekvenssit annetaan syötteenä nimenomaan tätä tutkielmaa varten toteutetulle Java-työkalulle, joka toimii systeemissä vektoroijana. Työkalu poistaa ensin lyhyet alle kaksimerkkiset sanat sanastosta, minkä jälkeen on vuorossa vektorien tiivistäminen. Työkalu toteuttaa tiivistämisen valitsemalla vektoriin ennalta valitun määrän (normaalisti 1000 kpl) sanoja, jotka esiintyvät mahdollisimman lähelle puolessa dokumenteista. Ideana on se, että sanat, jotka esiintyvät kaikissa tai ei missään dokumentissa ovat merkityksettömiä erottelukyvyltään, kun taas puolessa dokumenteista esiintyvät sanat jakavat dokumenttijoukon kahtia. Tiivistämisen lisäksi tämä osa ohjelmistoa vastaa myös dokumenttivektorien luomisesta ja painottamisesta. Vaihtoehtoina ovat binääri-, frekvenssi- ja *tf·idf*-painotus. Tuloksena vektoroinnista saadaan kokoelman dokumenttivektorit.

Neljännessä vaiheessa dokumenttivektorit annetaan SOM_PAK-ohjelmistopaketille, joka luo niistä itseorganisoituvan kartan. Kartan luomisessa voidaan käyttää erilaisia asetuksia, esimerkiksi alustustapa, kartan koko, kartan malli, naapuruston malli ja oppimiskierrosten määrät asetetaan ennen kartan luomista. Tuloksena neljännestä vaiheesta saadaan karttatiedosto. Viidennestä vaiheesta vastaa edelleen SOM_PAK, joka laskee dokumenttien sijainnin kartalla.



Kuva 6.5. Hakukoneprototyypin toimintamalli.

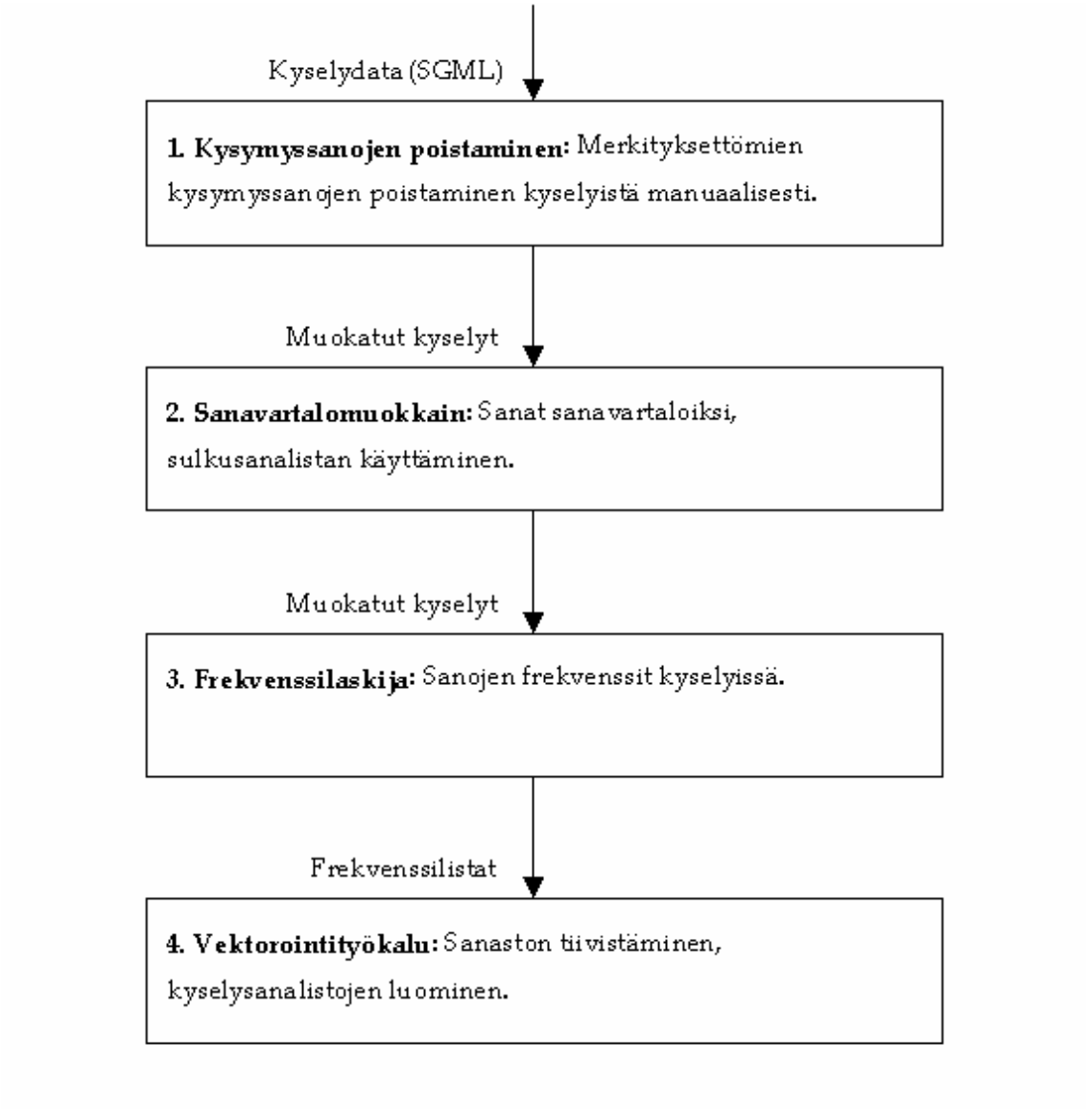
Viimeisessä vaiheessa valmis kartta, dokumenttien sijaintitiedot ja kyselyt sekä tieto relevansseista dokumenteista annetaan juuri tätä tutkielmaa varten valmistetulle sanahakutyökalulle, joka on toteutettu Java-kielellä. Sanahakutyökalu laskee kyselyiden sijainnit kartalla sekä laskee evaluointiin liittyviä tunnuslukuja sen perusteella, miten paljon relevantteja dokumentteja löydetään (katso luku 7).

Edellä esitetyn hakukoneen päätoiminnan lisäksi kuuluu prototyyppiin yhtenä osatehtävänä vielä kyselyiden muodostaminen SGML-muotoisesta kyselydatasta. Luonnollisesti valmiissa hakukoneessa tätä ei tarvita, koska kyselyiden muodostaminen liittyy vain prototyypin testaamisen mahdollistamiseen. Kyselyiden muodostaminen jakautuu seuraaviin osa-alueisiin:

1. kysymyssanojen poisto
2. sanavartalomuokkain ja sulkusanojen poistaja
3. frekvenssilaskija
4. vektorointityökalu.

Ensimmäisessä vaiheessa jokaisesta kyselystä poistetaan merkityksettömät kysymyssanat manuaalisesti, jotka SGML-muotoisen datan sisällöstä jäävät muuten kyselyihin mukaan häiritsemään kyselyitä. Tämän jälkeen kyselydatan sanat ajetaan sanavartaloiksi ja poistetaan sulkusanalistan sanat, kuten edellä toimittiin dokumenttivektorienkin osalta. Muokattu kyselydata annetaan seuraavaksi frekvenssityökalulle, jonka tuloksena syntyvät frekvenssilistat ohjataan edelleen vektorointityökalulle. Vektorointityökalu muodostaa tiivistykseen jälkeen kustakin kyselystä jäljelle jäävistä sanoista sanalistan, joka edustaa nyt kyselyä. Kyselyitä ei painoteta, vaan listataan vain sanat, jotka kyselyssä esiintyvät. Kyselyiden muodostamista on havainnollistettu kuvassa 6.6.

Kun kartta on valmis ja kyselyt on muodostettu, on sanahakutyökalun tehtävänä laskea kyselyiden osumakohdat kartalta. Tässä hakukoneessa parhaan osumakohdan määrittäminen on toteutettu niin, että jokaisen kyselyn kohdalla käydään läpi jokainen kartan solmu ja lasketaan, missä solmussa kyselyn sanoja vastaavien painoarvojen yhteissumma on suurin. Kun kaikki solmut on tutkittu, tiedetään, missä summa oli suurin ja tuo solmu määrätään sitten kyselyn osumakohdaksi kartalla. Esimerkiksi jos kysely muodostuisi sanoista ’auto’ ja ’rengas’, etsittäisiin kartalta se solmu, jossa noiden kahden sanan yhteenlaskettu painoarvo on suurin ja kyselyn tulosjoukon muodostaisivat tuolloin tuon solmun dokumentit.



Kuva 6.6. Kyselyiden muodostamisen toimintamalli.

6.5. Epäonnistuneet toteutuskokeilut

Ennen edellä esitetyn onnistuneen prototyypin toteuttamista kokeiltiin useita erilaisia tapoja onnistuneen prototyypin toteuttamiseksi. Esitellään seuraavaksi lyhyesti merkittävimmät yritykset sekä pohditaan, mikä kussakin tapauksessa epäonnistui.

6.5.1. Matlab-kokeilu 1

Ensimmäisessä Matlabiin perustuneessa kokeilussa tavoitteena oli toteuttaa prototyyppi siten, että itseorganisoituvan kartan luomisesta olisi vastannut Matlabiin asennettu SOM Toolbox, jonka käyttämiseen päädyttiin, koska Matlab tarjosi helpon graafisen kehitysympäristön.

Tavoitteena oli luoda karttoja 30 000 dokumentilla. Nopeasti osoittautui kuitenkin mahdottomaksi toteuttaa niin isoja karttoja Matlabissa, sillä muisti loppui jatkuvasti kesken ennen kuin karttoja saatiin edes alustettua. Tämän jälkeen käytettävän kokoelman kokoa pudotettiin 20 000 dokumenttiin ja melko pian samojen ongelmien uusiuduttua 10 000 dokumenttiin. Vielä 10 000 dokumentin kokoelmallakin muistin riittäminen oli ongelma, joten kokoa pudotettiin 5000 dokumentin tasolla, ja kun tässäkin kokoluokassa esiintyi muistiongelmia päätettiin SOM Toolboxin käytöstä luopua kokonaan.

6.5.2. SOM_PAK-kokeilu 1

Epäonnistuneen ensimmäisen Matlab-kokeilun jälkeen otettiin käyttöön SOM_PAK-ohjelmistopaketti, jonka käyttöä ei tähän asti ollut testattu. Vaikka paketista puuttui kokonaan graafinen kehitysympäristö, osoittautui paketti muistinkäytöltään merkittävästi paremmaksi.

SOM_PAK osoittautui teholtaan niin paljon paremmaksi, että nopeasti kokoelman kooksi päätettiin ottaa peräti 100 000 dokumenttia. Näinkään suuressa kokoluokassa muistimäärä koneessa ei rajoittanut mahdollisuuksia lainkaan. Laskenta-ajatkin pysyivät kohtuullisella tasolla, vaikka suurimpien karttojen laskemiseen saattoikin nyt kulua jopa viisi tuntia. Karttojen luominen onnistui siis mainiosti, mutta ongelmaksi muodostui karttojen luominen, sillä suurin osa dokumenteista kasautui pienelle alueelle kartalla. Esimerkiksi tilanteessa, jossa kartan kussakin solmussa olisi pitänyt olla keskimäärin noin 10 dokumenttia, saattoivat kartan tietyt solmut kerätä jopa tuhansia dokumentteja. Asiaa yritettiin korjata muuttamalla opetusasetuksia, mutta merkittävää parannusta ei saatu aikaan. Ongelmaa lisäsi vielä se, että erilaisten asetusten kokeileminen kartoilla, joiden laskeminen voi kestää parikin tuntia, oli työlästä. Niinpä päätettiin toteuttaa sovellus pienemmällä kokoelmalla, jossa eri asetusten testaaminen olisi helpompaa. Päätettiin supistaa kokoelma noin tuhanteen dokumenttiin.

6.5.3. Matlab-kokeilu 2

Kokoelman radikaalin supistumisen vuoksi Matlabin SOM Toolbox valittiin helppoutensa vuoksi jälleen kartan luomisen toteutustavaksi. Vaikka kokoelmassa oli jäljellä vain 1160 dokumenttia, esiintyi muistiongelmia satunnaisesti, kun kartan koko ja dokumenttivektorien dimensio kasvoivat liian suuriksi. Alkoi käydä selväksi, että Matlab-toteutuksen rajat kulkevat nykytietokoneiden osalta noin 1000 dokumentin paikkeilla. Koska tutkimuksen tavoitteena on, varsinkin jatkossa, keskittyä kuitenkin suurempiin kokoluokkiin, päätettiin Matlab-toteutuksesta luopua lopullisesti.

Tämän viimeisen Matlab-kokeilun jälkeen siirryttiin käyttämään SOM_PAK-ohjelmistopakettia kokoelman koolla 1160, mikä olikin lopulta onnistunut yhdistelmä.

7. Tulokset

Tässä luvussa esitellään hakukoneen testaamiseen käytetty menetelmä sekä tärkeimmät testitulokset. Testaamisen tavoitteena oli mitata sitä, miten hyvin itseorganisoituvan kartan algoritmi säilyttää dokumenttien suhteet niiden käsittelemien aihepiirien suhteen sekä selvittää, miten kartan luomiseen liittyvät asetukset vaikuttavat kartan suorituskykyyn tiedonhaussa.

7.1. Testimenetelmä

Koska tavoitteena oli toteuttaa hakukone, joka löytää kutakin kyselyä vastaavan parhaan sijainnin kartalta, on sen tehokkaan toimimisen kannalta oleellista, että kyselyä parhaiten vastaavassa solmussa olevista dokumenteista mahdollisimman suuri osa on relevantteja kyseisen kyselyn kannalta (tarkkuus) ja toisaalta, että mahdollisimman suuri osa kaikista kyseisen kyselyn suhteen relevanteista dokumenteista löytyy tuosta solmusta (saanti). Toisaalta on myös toivottavaa, että parhaan solmun viereisissä solmuista löytyy myös relevantteja dokumentteja, koska ne ovat solmuja, joihin käyttäjä voi kyselyn antamisen jälkeen siirtyä etsiessään lisää sopivia vastuksia kyselyynsä. Samoin olisi toivottavaa, että tulosjoukot olisivat sopivan kokoisia eli kyselyyn parhaiten täsmäävässä solmussa olisi hallittavissa oleva määrä dokumentteja. Kenties ihanteellinen koko olisi noin 10 dokumenttia solmua kohti.

Edellä esitellyn pohjalta valitaan hakukoneen suorituskyvyn mittareiksi seuraavat:

1. Tarkkuus eli relevanttien dokumenttien prosenttiosuus tulosjoukosta.
2. Saanti, joka kertoo, miten suuri prosenttiosuus kaikista kyseiseen kyselyyn liittyvistä relevanteista dokumenteista löydettiin.
3. Tulosjoukon koko dokumenttien määrällä mitattuna.

Edellä esitetyt kolme tunnuslukua lasketaan kaikkien 20 kyselyn keskiarvona sekä kyselyn parhaassa osumasolmussa että parhaan osumasolmun ja sen välittömien lähinaapurien muodostamalla alueella, joka sisältää viisi solmua, kun osumasolmu ei sijaitse kartan reunalla. Reunalla sijaitessaan osumasolmulla on, riippuen sijaitseeko solmu nurkassa, kahdesta kolmeen naapuria, jolloin alue on kooltaan vastaavasti kolmen tai neljän solmun kokoinen.

Koska kartan organisoituminen on osittain satunnainen prosessi, tarvitaan jonkinlaista menetelmää, jonka avulla pystytään mittaamaan kartan keskimääräistä suoriutumista. Kohonen *et al.* [2000] käyttää tämän ongelman ratkaisemiseen menetelmää, jossa kartan suorituskyky lasketaan luomalla viisi eri karttaa ja laskemalla niiden suorituksen keskiarvoa. Tässä tutkielmassa keskimääräisen suorituksen saavuttamiseksi on toimittu siten, että jokaista testiä varten kustakin kartasta on luotu viisi eri versiota, joita on verrattu toisiinsa keskimääräisen kvantisaatiovirheen perusteella, siis sen keskimääräisen virheen perusteella, joka tehdään, kun kartan

painovektorit edustavat syötevektoreita. Testiin on valittu viidestä kartasta se, jonka kvantisaatiovirhe on ollut noiden viiden arvon joukossa mediaani. Tämä vastaa sitä, että valitaan kartta, joka on organisaationsa onnistuvuuden kannalta keskinkertainen, eli edustaa tällöin tavallaan kyseisen kartan keskimääräistä suoriutumista.

Erilaisten kartan luomiseen liittyvien asetusten testaamiseksi muuttujiksi valittiin seuraavat kartan ominaisuudet:

1. dokumenttivektorien painotus: binääri, frekvenssi tai tf·idf
2. dokumenttivektorien dimensio: 500, 1000, 2000, 3000, 4000, 5000
3. kartan koko: 9×9, 11×11, 13×13, 15×15, 17×17
4. alustustapa: satunnainen, lineaarinen
5. naapuruston tyyppi: kupla, Gaussin funktio
6. oppimiskierrokset läpi koko joukon (karkea/hieno): 1/5, 2/10, 3/15, 4/20.

Muuttujien testattavat arvot on valittu siten, että kartan luominen ei vie liikaa laskenta-aikaa tavallisellakaan tietokoneella. Kartan koossa kaikki eri vaihtoehdot valittiin samanmuotoisiksi, jotta kartan muoto ei vaikuttaisi tulokseen. Samoin oppimiskierrosten määrä valittiin suhteessa 1:5, jotta karkean ja hieno-oppimisen suhde ei vaikuttaisi testaamiseen, vaan voitaisiin testata nimenomaan tarvittavaa opetuskierrosten määrää. Vektorien samankaltaisuuden laskemisessa käytettiin euklidista etäisyyttä.

Testaamista varten on luotava myös perusjärjestelmä, jotta kunkin edellä esitetyn muuttujan vaikutusta voidaan testata muuttamalla sen arvoa perusjärjestelmässä ja tarkastelemalla tulosta. Perusjärjestelmään valitaan seuraavat asetukset:

1. dokumenttivektorin painotus: binääri
2. dokumenttivektorin dimensio: 1000
3. kartan koko: 11×11
4. alustustapa: satunnainen
5. naapuruston tyyppi: kupla
6. oppimiskierrokset: 2/10.

Perusjärjestelmän valinnassa on pyritty valitsemaan yksinkertaiset perusvaihtoehdot. Testaamisessa muutetaan kutakin muuttujaa kerrallaan perusjärjestelmässä ja tutkitaan muutoksen vaikutusta tulokseen. Testaamisen perusteella pyritään valitsemaan paras muuttujayhdistelmä, jonka perusteella luodaan lopuksi paranneltu järjestelmä, jonka suoritusta voidaan sitten verrata perusjärjestelmään.

7.2. Testitulokset

Seuraavaksi tarkastellaan testiajojen tuloksia. Tulostaulukoissa käytetään seuraavia merkintöjä:

- Tarkkuus 1 = tarkkuus kyselyn osumasolmussa 20 kyselyn keskiarvona
- Saanti 1 = saanti kyselyn osumasolmussa 20 kyselyn keskiarvona
- Tulosjoukon koko 1 = tulosjoukon koko osumasolmussa 20 kyselyn keskiarvona

- Tarkkuus 2 = tarkkuus kyselyn osumasolmussa ja sen naapurustossa 20 kyselyn keskiarvona
- Saanti 2 = saanti kyselyn osumasolmussa ja sen naapurustossa 20 kyselyn keskiarvona
- Tulosjoukon koko 2 = tulosjoukon koko osumasolmussa ja sen naapurustossa 20 kyselyn keskiarvona.

On huomattava, että laskettaessa tuloksia lähinaapurustossa tarkastellaan osumasolmua sekä sen välittömiä naapureita. Solmuilla, jotka eivät sijaitse kartan reunalla, lähinaapureita on neljä. Reunasolmuilla lähinaapureita on vain kolme, paitsi kartan kulmassa sijaitsevilla vain kaksi. Esitellään ensin perusjärjestelmän testitulos ja sen jälkeen testitulokset muuttuja kerrallaan muutellussa perusjärjestelmässä. Lopuksi vertaillaan parannellun järjestelmän tulosta perusjärjestelmään.

7.2.1. Perusjärjestelmä

Perusjärjestelmä luotiin luomalla itseorganisoituva kartta edellä esitellyillä asetuksilla ja testaamalla sitten sen toimivuutta tiedonhaussa 20 kyselyllä ja tutkimalla relevanttien dokumenttien esiintymistä kyselyiden osumasolmuissa ja niiden lähinaapurustossa. Testin tulokset löytyvät taulukosta 7.1. Kaikki taulukon tunnusluvut on laskettu 20 kyselyn keskiarvona. Tarkkuus 1 kertoo esimerkiksi, että keskimäärin 17,2 % kyselyn osumasolmun dokumenteista on ollut relevantteja kyselyn suhteen. Vastaavasti saanti 1 kertoo, että osumasolmusta on löytynyt keskimäärin 4,1 % kaikista kyselyyn liittyvistä relevanteista dokumenteista ja tulosjoukon koko 1 kertoo keskimääräisen osumasolmun sisältäneen 6,7 dokumenttia. Vastaavasti tarkkuus 2, saanti 2 ja tulosjoukon koko 2 kertovat samat tunnusluvut, kun tarkastellaan osumasolmun lähinaapurustoa, johon kuuluu yhteensä viisi solmua, jos osumasolmu ei ole kartan reunalla.

Järjestelmä	Perus
Tarkkuus 1 (%)	17,2
Saanti 1 (%)	4,1
Tulosjoukon koko 1	6,7
Tarkkuus 2 (%)	15,6
Saanti 2 (%)	22,1
Tulosjoukon koko 2	39,1

Taulukko 7.1. Testitulokset perusjärjestelmälle.

7.2.2. Dokumenttivektorin painotus

Toisessa testiajossa testattiin dokumenttivektorin painotuksen vaikutusta tiedonhaun tulokseen. Käytännössä siis muutettiin perusjärjestelmää yhden muuttujan osalta ja tutkittiin tuloksia.

Taulukon 7.2. tulosten perusteella nähdään, että *tf·idf*-painotus antaa selvästi parempia tuloksia kuin binääri- ja frekvenssipainotus. Tulos on melko kiistaton, sillä *tf·idf* antaa sekä paremman saannin, että tarkkuuden niin osumasolmussa kuin sen lähiympäristössäkin, vaikka sen tulosjoukko on suunnilleen samaa kokoa kuin muiden painotusmenetelmien antamat tulosjoukot.

Painotus	Binääri	Frekvenssi	tf•idf
Tarkkuus 1 (%)	17,2	12,8	28,8
Saanti 1 (%)	4,1	2,3	9,3
Tulosjoukon koko 1	6,65	5,9	7,0
Tarkkuus 2 (%)	15,6	17,1	32,0
Saanti 2 (%)	22,1	15,3	30,6
Tulosjoukon koko 2	39,1	25,4	27,1

Taulukko 7.2. Testitulokset dokumenttivektorin eri painotustavoille.

7.2.3. Dokumenttivektorin dimensio

Kolmannessa testitilanteessa vertailtiin dimensioltaan erikokoisten dokumenttivektorien vaikutusta tulokseen. Dimensio vaihteli 500:sta 5000:een. Taulukon 7.3 pohjalta on vaikea sanoa juuri mitään, sillä mikään dimensio ei erotu joukosta selvästi.

Dimensio	500	1000	2000	3000	4000	5000
Tarkkuus 1 (%)	14,1	17,2	11,3	13,1	14,3	12,9
Saanti 1 (%)	2,9	4,1	8,0	2,1	1,3	7,8
Tulosjoukon koko 1	5,6	6,7	13,2	15,4	7,6	14,1
Tarkkuus 2 (%)	15,0	15,6	14,9	12,3	11,7	14,8
Saanti 2 (%)	20,0	22,1	19,0	19,0	21,8	20,8
Tulosjoukon koko 2	39,4	39,1	42,5	42,5	56,9	53,9

Taulukko 7.3. Testitulokset dokumenttivektorin eri dimensioille.

7.2.4. Kartan koko

Neljännessä ajossa (taulukko 7.4) testattiin kartan koon vaikutusta tiedonhakutulokseen. Näyttäisi siltä, että suurempi kartta antaa paremman tuloksen, mutta toisaalta pienenevä tulosjoukko haittaa vertailu jonkin verran. Selvästi muita paremmaksi osoittautuu

17×17-kokoinen kartta, joka löytää osumasolmun lähinaapurustosta 28,4 % relevantteja dokumentteja, kun tulosjoukon koko on 19,7.

Kartan koko	9x9	11x11	13x13	15x15	17x17
Tarkkuus 1 (%)	6,1	17,2	16,0	20,9	22,9
Saanti 1 (%)	1,0	4,1	5,0	2,6	2,6
Tulosjoukon koko 1	19,2	6,6	9,1	4,4	4,6
Tarkkuus 2 (%)	7,6	15,6	16,9	18,5	28,4
Saanti 2 (%)	12,3	22,1	19,3	14,1	15,6
Tulosjoukon koko 2	63,5	39,1	37,4	20,0	19,7

Taulukko 7.4. Testitulokset kartan eri kokovaihtoehdoille.

7.2.5. Alustustapa

Taulukossa 7.5 on vertailtu satunnaisen ja lineaarisen alustuksen vaikutusta tulokseen. Tämän testin pohjalta satunnainen alustus on jopa hivenen parempi, varsinkin osumasolmussa. Lähinaapurustossa molemmat alustustavat ovat tulokseltaan tasaisia.

Alustus	Satunnainen	Lineaarinen
Tarkkuus 1 (%)	17,2	10,2
Saanti 1 (%)	4,1	3,1
Tulosjoukon koko 1	6,6	13,6
Tarkkuus 2 (%)	15,6	15,9
Saanti 2 (%)	22,1	17,8
Tulosjoukon koko 2	39,1	40,9

Taulukko 7.5. Testitulokset satunnaiselle ja lineaariselle alustukselle.

7.2.6. Naapuruston tyyppi

Taulukossa 7.6. ovat vertailussa kupla- ja Gauss-naapurustot. Kupla näyttäisi pärjäävän hivenen paremmin osumasolmussa, mutta koko lähinaapuruston alueella Gaussin funktioon perustuva naapurusto on lähes yhtä tehokas.

Naapurusto	Kupla	Gauss
Tarkkuus 1 (%)	17,2	12,6
Saanti 1 (%)	4,1	3,3
Tulosjoukon koko 1	6,6	21,7
Tarkkuus 2 (%)	15,6	13,7
Saanti 2 (%)	22,1	21,3
Tulosjoukon koko 2	39,1	52,2

Taulukko 7.6. Testitulokset kuplanaapurustolle ja Gaussin naapurustolle.

7.2.7. Oppimiskierrokset

Viimeisenä muuttujavertailuna tutkittiin oppimiskierrosten määrän vaikutusta (taulukko 7.7). Selvästi muita paremmaksi osoittautui oppiminen kolmen karkean ja 15 hieno-oppimiskierroksen kautta. Tässä kierros tarkoittaa koko dokumenttijoukon läpikäymistä.

Oppiminen	1/5	2/10	3/15	4/20
Tarkkuus 1 (%)	11,3	17,2	19,1	13,0
Saanti 1 (%)	3,1	4,1	7,4	1,2
Tulosjoukon koko 1	9,5	6,6	12,0	9,4
Tarkkuus 2 (%)	11,1	15,6	18,4	15,5
Saanti 2 (%)	13,9	22,1	23,0	16,7
Tulosjoukon koko 2	36,0	39,1	53,3	51,9

Taulukko 7.7. Testitulokset oppimiskierrosten eri vaihtoehdoille.

7.2.8. Paranneltu järjestelmä

Edellä esitettyjen testien pohjalta muodostettiin asetuksiltaan paranneltu järjestelmä valitsemalla jokaisen muuttujan kohdalla parhaiten testeissä menestynyt vaihtoehto. Paranneltuun järjestelmään valittiin seuraavat asetukset:

1. Dokumenttivektorin painotus: tf·idf
2. Dokumenttivektorin dimensio: 1000
3. Kartan koko: 17×17
4. Alustustapa: satunnainen
5. Naapuruston tyyppi: kupla
6. Oppimiskierrokset: 3/15.

Paranneltua järjestelmää testattiin ja verrattiin perusjärjestelmään. Tulokset löytyvät taulukosta 7.8. Paranneltu järjestelmä osoittautui paremmaksi sekä osumasolmussa, että lähinaapurustossa. Selkeästi paremman tarkkuuden paranneltu versio antoi

lähinaapurustossa, mutta pienestä tulosjoukosta johtuen saanti jäi alle perusjärjestelmän saannin. On kuitenkin melko selvää, että paranneltu versio oli tehokkaampi.

Järjestelmä	Perus	Paranneltu
Tarkkuus 1 (%)	17,2	19,6
Saanti 1 (%)	4,1	4,3
Tulosjoukon koko 1	6,6	5,2
Tarkkuus 2 (%)	15,6	28,1
Saanti 2 (%)	22,1	15,7
Tulosjoukon koko 2	39,1	16,0

Taulukko 7.8. Testitulokset perusjärjestelmälle ja parannellulle järjestelmälle.

7.3. Tulosten analysointi

Edellä esitetyt tulokset antavat paljon aihetta pohdintaan. Tulokset eivät olleet kaikilta osin kovinkaan johdonmukaiset. Herää kysymys johtuvatko vaihtelevat tulokset siitä, että testatuilla parametreilla ei ole käytännön merkitystä kartan luomisessa vai siitä, että itseorganisoituvan kartan algoritmin satunnaisuus vaikuttaa tulokseen enemmän kuin parametrien valinta.

Testien perusteella saadut käytännön suositukset ovat vähissä. Näyttäisi kuitenkin siltä, että ainakin *tf·idf*-painotus parantaa tiedonhakutehokkuutta jopa merkittävästi. Samalla tavoin suuri kartan koko näyttäisi auttavan paremman tuloksen saavuttamisessa. Tulosten mukaan myös 9×9-kokoinen kartta tuntuisi olevan liian pieni 1160 dokumentin järjestämiseen ja kolmen karkean ja 15 hieno-oppimisvaiheen oppimismalli saattaisi olla lähellä optimaalista tässä sovelluksessa.

Vaikka käytännön neuvot hyvien parametrien valitsemiseksi jäivätkin melko vähiin, voidaan testien perusteella ainakin todeta, että itseorganisoituvaa karttaa voidaan soveltaa tekstidokumenttien tiedonhaussa. Parhaimmillaan tällä toteutuksella tässä kokoelmassa näytettäisiin pääsevän noin 30 %:n tarkkuuteen 20 - 30 tulosdokumentin joukossa, mikä tarkoittaa käytännössä sitä, että haettaessa satunnaisella kyselyllä saataisiin vastauksena 30 dokumenttia, joista 10 olisi relevantteja kyselyn suhteen. Tämä ei ehkä ole mikään mullistavan hyvä tulos, mutta vähintäänkin kelvollista tasoa ja ennen kaikkea se osoittaa, että itseorganisoituva kartta säilyttää dokumenttien väliset suhteet niiden aihepiirin suhteen. Pohditaan vertailun vuoksi montako relevanttia dokumenttia voidaan odottaa osuvan valittuun joukkoon, jos valitaan 1160 dokumentin joukosta satunnaisesti 30 dokumenttia ja verrataan niitä johonkin annettuun kyselyyn. Kokoelmassa on jokaista kyselyä kohti keskimäärin 29 relevanttia dokumenttia. Kokoelmassa siis noin joka 40. dokumentti on relevantti edellä mainitun kyselyn suhteen.

Tällöin valitsemalla satunnaisesti 30 on odote noin 0,75 relevanttia dokumenttia. Voidaan siis sanoa, että tutkielmassa toteutettu hakukone pystyy tässä mielessä tehostamaan satunnaista tilannetta yli kymmenen kertaa paremmaksi. Tuloksen pohjalta on selvää, että kartta todellakin säilyttää dokumenttien suhteet niiden aihepiirien osalta, ainakin kelvollisessa määrin.

8. Yhteenveto

Tutkielman päätavoite oli tutkia säilyttääkö Kohosen itseorganisoituva kartta tekstidokumenttien aihepiiriin liittyvät ominaisuudet ja suhteet siinä määrin, että karttaa voisi hyödyntää tehokkaasti tekstidokumenttien tiedonhaussa. Toisena tavoitteena oli suunnitella ja toteuttaa hakukoneen prototyyppi, jonka avulla edellä mainitun ongelman tutkiminen olisi mahdollista. Prototyypin tuli olla WEBSOM-järjestelmän kaltainen yksinkertainen hakukone, jolla kartalta voisi hakea dokumentteja tunnettujen hakukoneiden tapaan muutamalla sanalla. Yksi suurimmista motivaatioista työlle oli aiemman tutkimuksen puute nimenomaan Kohosen kartan sovelluksissa hakukoneiden kehittämiseksi tekstidokumenttidatalle.

Hakukoneen prototyyppi suunniteltiin ja toteutettiin onnistuneesti ja sen avulla pyrittiin testaamaan itseorganisoituvan kartan tiedonhakutehoa. Aineistona kartalle käytettiin 1160 saksankielistä uutisartikkelia sekä testaamiseen 20 kyselyä, joista jokaiseen liittyi keskimäärin 29 relevanttia dokumenttia uutisartikkelien joukossa. Hakutestien perusteella kehitetyn järjestelmän suoriutuminen oli kelvollista, mutta melko vaihtelevaa ja satunnaista, ilmeisesti Kohosen algoritmin satunnaisesta luonteesta johtuen. Joka tapauksessa tulokset osoittavat kiistattomasti, että Kohosen itseorganisoituva kartta pystyy säilyttämään ja kuvaamaan dokumenttikokoelmien aihepiirisuhteita. Tulosten vertaileminen aiempaan tutkimukseen on vaikeaa, sillä vastaavaa tutkimusta on vain vähän julkaistu ja toisaalta aineiston ja sovelluksen koon erilaisuus tekevät yleensäkin tämän kaltaisten sovellusten vertailemisen vaikeaksi. Joka tapauksessa itseorganisoituva kartta tuo lisäarvoa tiedonhakuun järjestämällä dokumentit kartaksi, jolla käyttäjä voi liikkua etsien tietoa tai tutkia karttaa dokumenttikokoelman rakenteen ja suhteiden selvittämiseksi.

Jatkotutkimukseen tämä työ on antanut runsaasti aiheita. Ensiksikin olisi erittäin mielenkiintoista pyrkiä kehittämään itseorganisoituvaan karttaan perustuva hakumenetelmä, jossa kartta järjestäisi dokumentit relevanssin mukaiseen järjestykseen ja palauttaisi tuloksen sitten listana käyttäjälle. Tällaista menetelmää olisi helppo verrata esimerkiksi klassiseen vektoriavaruusmalliin, jolloin saataisiin selville tuoko suuritöinen kartta lisäarvoa tiedonhakuun. Tässä työssähän järjestelmä ei palauta listaa tai relevanssijärjestettyä tulosjoukkoa, vaan osoittaa käyttäjälle kyselyyn parhaiten vastaavan sijainnin kartalla. Toisaalta olisi mielenkiintoista tutkia kartan tehoa suuremmissa sovelluksissa, joissa kokoelmassa olisi mukana esimerkiksi 10 000, 25 000 tai peräti 100 000 dokumenttia. Tämän työn opettamana suurempien sovellusten kehittäminen voisi olla mahdollista. Toisessa ääripäässä olisi kiinnostavaa tehdä jatkotutkimusta erittäin pienillä kokoelmilla, esimerkiksi 100 dokumentin joukoilla. Näiden tutkimisesta saattaisi olla suurta hyötyä erilaisten kartan luomiseen liittyvien lainalaisuuksien ymmärtämisessä. Tutkittavaa riittäisi varsinaisen kartan luomisen

optimoinnin lisäksi myös erilaisten kartan visualisointitapojen ja selailukäyttöliittymien kehittelyssä, sillä niiden osalta ei tutkimusta ole tehty vielä lähellekään riittävästi. Tässä työssä suunnitellun hakukoneen toteuttaminen käytännössä mahdollistaisi visualisoinnin ja selailukäyttöliittymän testaamisen ja kehittämisen hakukoneen rinnalla.

Tutkielman tavoitteet onnistuivat siis tärkeimmiltä osin, mutta uutta tietoa syntyi lopulta, kenties huomattavastikin, vähemmän kuin uusia kysymyksiä ja mielenkiinnon kohteita ilmestyi horisonttiin. Itseorganisoituvassa kartassa ja sen tekstitiedonhakusovelluksissa riittäisi tutkittavaa vielä vuosiksi eteenpäin.

Viiteluettelo

[Azcarraga and Yap Jr., 2001] Arnulfo P. Azcarraga and Teddy N. Yap Jr., Extracting Meaningful Labels for WEBSOM Text Archives. In: *Proc. of 10th International Conference on Information and Knowledge Management* CIKM'01 (2001), ACM Press, 41-48.

[Baeza-Yates and Ribeiro-Neto, 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[Belew, 2000] Richard K. Belew, *Finding Out About. A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, 2000.

[Bingham and Mannila, 2001] Ella Bingham and Heikki Mannila, Random projection in dimensionality reduction: applications to image and text data. In: *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001), ACM Press, 245-250.

[Chowdhury and Saha, 2005] Nirmalya Chowdhury and Diganta Saha, Unsupervised Text Classification Using Kohonen's Self Organizing Network. *Lecture Notes in Computer Science, Computational Linguistics and Intelligent Text Processing* **3406** (2005), Springer, 715-718.

[Dittenbach *et al.*, 2002] Michael Dittenbach, Andreas Rauber and Dieter Merkl, Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing* **48/1-4** (2002), 199-216.

[Faba-Pérez *et al.*, 2003] Cristina Faba-Pérez, Vicente P. Guerrero-Bote and Félix de Moya-Anegón, Self-organizing maps of Web spaces based on formal characteristics. *Information Processing and Management* **41** (2005), 331-346.

[Fernández *et al.*, 2004] Javier Fernández, Ricardo Mones, Irene Díaz, José Ranilla and Elías F. Combarro, Experiments with Self Organizing Maps in CLEF 2003. *Lecture Notes in Computer Science, Comparative Evaluation of Multilingual Information Access Systems* **3237** (2004), Springer, 358-366.

[Guerrero Bote *et al.*, 2002] Vicente P. Guerrero Bote, Félix de Moya Anegón and Victor Herrero Solana, Document organization using Kohonen's algorithm. *Information Processing and Management* **38** (2002), 79-89.

[Haykin, 1994] Simon Haykin, *Neural Networks. A Comperenhensive Foundation*. Prentice Hall, 1994.

[Honkela, 1997] Timo Honkela, *Self-organizing maps in natural language processing*. PhD Dissertation. Helsinki University of Technology, 1997.

[Kaski, 1998] Samuel Kaski, Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering. In: *Proc. of IEEE International Joint Conference on Neural Networks IJCNN'98* **1** (1998), 413-418.

[Kohonen, 1995] Teuvo Kohonen, *Self-Organizing Maps*. Springer-Verlag, 1995.

[Kohonen *et al.*, 1996] Teuvo Kohonen, Jussi Hynninen, Jari Kangas and Jorma Laaksonen, *SOM_PAK: The Self-Organizing Map Program Package*. Helsinki University of Technology, 1996. (Available at: http://www.cis.hut.fi/research/papers/som_tr96.ps.Z, 5.2.2007)

[Kohonen *et al.*, 2000] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero and Antti Saarela, Self organization of a massive document collection. *IEEE Transactions on Neural Networks* **11**, 3 (May 2000), 574-585.

[Kwok, 1989] Kui-Lam Kwok, A Neural Network for Propabilistic Information Retrieval. In: *Proc. of the 12th annual international ACM SIGIR Conference on Research and Develpment in Information Rretrieval SIGIR '89* **23/SI** (1989), ACM Press, 21-30.

[Kwok *et al.*, 2005] Kui-Lam Kwok, Sora Choi, Norbert Dinstl and Peter Deng, NTCIR-5 Chinese, English, Korean Cross Language Retrieval Experiments using PIRCS. In: *Proc. of NTCIR-5 Workshop Meeting* (2005).

[Lagus, 2000] Krista Lagus, *Text Mining with the WEBSOM*. ScD Dissertation. Helsinki University of Technology, 2000.

[Lagus, 2002] Krista Lagus, Text Retrieval Using Self-Organized Document Maps. *Neural Processing Letters* **15** (2002), Kluwer Academic Publishers , 21-29.

[Lagus and Kaski, 1999] Krista Lagus and Samuel Kaski, Keyword selection method for characterizing text document maps. In: *Proc. of 9th International Conference on Artificial Neural Networks ICANN'99* **1** (1998), 371-376.

[Lagus *et al.*, 2004] Krista Lagus, Samuel Kaski and Teuvo Kohonen, Mining massive document collections by the WEBSOM method. *Information Sciences* **163/1-3** (2004), 135-156.

[Lee and Yang, 1999] Chung-Hong Lee and Hsin-Chang Yang, A Web Text Mining Approach Based on Self-Organizing Map. In: *Proc. of 2nd international workshop on Web Information and Data Management* (1999), 59-62.

[Lin, 1997] Xia Lin, Map Displays for Information Retrieval. *Journal of the American Society for Information Science*. **48/1** (1997), John Wiley & Sons Inc, 40-54.

[Lin *et al.*, 1991] Xia Lin, Dagobert Soergel and Gary Marchionini, A Self-organizing Semantic Map for Information Retrieval. In: *Proc. of 14th annual international ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR'91* (1991), ACM Press, 262-269.

[Lindén, 2005] Krister Lindén, *Word sense discovery and disambiguation*. PhD Dissertation. University of Helsinki, 2005.

[Marinai *et al.*, 2006] Simone Marinai, Stefano Faini, Emanuele Marino and Giovanni Soda, Efficient Word Retrieval by Means of SOM Clustering and PCA. *Lecture*

Notes in Computer Science, Document Analysis Systems VII **3872** (2006), Springer, 336-347.

[Merkl, 1998] Dieter Merkl, Text classification with self-organizing maps: Some lessons learned. *Neurocomputing* **21/1-3** (1998), 61-77.

[Moya-Anegón *et al.*, 2006] Félix Moya-Anegón, Víctor Herrero-Solana and Evaristo Jiménez-Contreras, A connectionist and multivariate approach to science maps: the SOM, clustering and MDS applied to library and information science research. *Journal of Information Science* **32/1** (2006), 63-77.

[Rojas, 1996] Raúl Rojas, *Neural Networks. A Systematic Introduction*. Springer-Verlag, 1996.

[Salton and McGill, 1983] Gerard Salton and Michael J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[Tangsripairoj and Samadzadeh, 2005] Songsri Tangsripairoj and M. H. Samadzadeh, Organizing and visualizing software repositories using the growing hierarchical self-organizing map. In: *Proc. of the 2005 ACM symposium on Applied Computing* (2005), ACM Press, 1539-1545.

[Vesanto *et al.*, 2000] Juha Vesanto, Johan Himberg, Esa Alhoniemi and Juha Parhankangas, *SOM Toolbox for Matlab 5*. Libella Oy, 2000. (Available at: http://www.cis.hut.fi/projects/somtoolbox/package/papers/techrep.pdf, 5.2.2007)

[Yang *et al.*, 1999] Christopher C. Yang, Hsinchun Chen and K. K. Hong, Visualization Tools for Self-organizing Maps. In: *Proc. of the 4th ACM conference on Digital Libraries DL'99* (1999), ACM Press, 258-259.