Miika Nurminen

Tiedonlouhinta rakenteisista dokumenteista

Tietotekniikan (ohjelmistotekniikka)
pro gradu -tutkielma
27.1.2005

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

URN:NBN:fi:jyu-200594

Tekijä: Miika Nurminen

Yhteystiedot: minurmin@cc.jyu.fi

Työn nimi: Tiedonlouhinta rakenteisista dokumenteista

Title in English: Data mining from structured documents

Työ: Tietotekniikan (ohjelmistotekniikka) pro gradu -tutkielma

Sivumäärä: 141

Tiivistelmä: Tutkielman kokonaistavoite on vastata tietotulvan tuomiin haasteisiin tiedonlouhinnan tekniikoita käyttäen. Yleisenä tutkimuskohteena on tiedonlouhinta rakenteisista dokumenteista. Täsmällisemmin määriteltynä tutkimusongelma käsittää samaa skeemaa noudattavien XML-dokumenttien klusteroinnin ja tiedonhaun. Lisäksi käsitellään erilaisten haku- ja klusterointitekniikoiden yhdistämisen tuomia mahdollisuuksia dokumenttikokoelmien hahmottamisessa. Teoreettisessa osuudessa käydään läpi erilaisia indeksirakenteita, samanlaisuusmittoja, klusterointialgoritmeja ja hakumenetelmiä. Empiirisessä osuudessa on kehitetty ExtMiner-sovellus, joka tukee hakua, klusterointia ja visualisointia erilaisille XML-dokumenttikokoelmille.

English abstract: The overall objective of this thesis is to consider the challenges posed by information overflow using data mining techniques. The research concentrates on data mining from structured documents. More precisely, the research problem involves information retrieval and clustering from XML documents conforming to the same schema. The potential of combining various search and clustering techniques in order to comprehend document collections is considered. Various index structures, similarity measures, clustering algorithms and ranking techniques are reviewed in the theoretical part of this thesis. In the empirical part the ExtMiner-application is developed, supporting searching, clustering and visualization for various XML document collections.

Avainsanat: Bibliometriikka, dokumenttien klusterointi, laajennettu vektorimalli, läheisyysmitat, rakenteinen indeksointi, rakenteiset dokumentit, tiedonhaku, tekstitiedonlouhinta, tiedonlouhinta, tietämyksen muodostaminen, web-louhinta, XML.

Keywords: Bibliometrics, data mining, document clustering, extended vector model, information retrieval, knowledge discovery, similarity measures, structured documents, structured indexing, text mining, web mining, XML.

Esipuhe

Tämän tutkielman tekeminen on ollut pitkä ja vaativa prosessi, joka on kestänyt yli vuoden. Lähteisiin tutustuminen alkoi jo lokakuussa 2003. Aihe tarkentui ja ensimmäiset luvut syntyivät saman syksyn aikana pidetyssä graduseminaarissa. Päälukujen sisällöt hahmottuivat myös seuraavilla kursseilla, joiden harjoitustyöt ja muut tehtävät pyrin valitsemaan gradun aihetta silmälläpitäen: *Semantic web and web services*, *Johdatus paikkatiedon analyysiin*, *Tekstitiedonhaku* sekä viimeisenä ja merkittävimpänä *Rakenteiset dokumentit*. Kurssien ja muiden töideni ohella tutkielman ja ExtMiner-sovelluksen tekeminen kuului osaksi Agora Centerin *Data Mining*- ja sittemmin *Knowledge Mining* -projekteja. Aikataulujen sovittamisen ohella tutkielman suurin haaste oli aiheen laajuus ja olennaisimpien lähdemateriaalien löytäminen. Lukemattomien klusterointimenetelmien ja toinen toistaan erikoisempien indeksirakenteiden läpikäynti oli aikaavievää, mutta välttämätöntä kokonaisnäkemyksen muodostamiseksi. Uskon, että prosessi on ollut oppimiskokemuksena vaivan arvoinen.

Kiitokset professori Tommi Kärkkäiselle merkittävästä ohjauksesta, tuesta ja kannustuksesta koko prosessin ajan. Tutkija Sami Äyrämön ohjaus ja palaute on ollut myös hyödyllistä ja tervetullutta tutkielman edetessä. Kiitokset myös lehtori Anne Honkarannalle XML-osuuteen liittyvästä palautteesta. Kiitän työtovereitani Turo Kilpeläistä ja Raija Suvista yhteistyöstä *Data Mining*- ja *Knowledge Mining* -projekteissa. Kiitokset *Johdatus ohjelmistotekniikkaan* -kurssin opettajille ja opiskelijoille mahdollisuudesta testata ExtMiner-sovellusta laajalla dokumenttikokoelmalla. Luennoitsijat Hilkka Heikkilä, Sami Kollanus ja Jonne Itkonen testasivat ohjelmaa kurssin aikana ja sain heiltä arvokasta palautetta. Lisäksi kiitokset projektipäällikkö Antti Saariselle UPM:ltä mahdollisuudesta testata ExtMiner-sovellusta koulutusmateriaaleilla sekä jatkokehitysideoista. Kiitokset myös ystävilleni. Juhani Honkalan kanssa opintojeni alkupuolella tehty harjoitustyö Vesa Lappalaisen *Ohjelmointi++* -kurssilla oli ensikosketukseni XML:n kiinnostavaan maailmaan ”hierarkkisine relaatiomalleineen”. Jukka Penttinen auttoi oikoluvussa työn viimeistelyvaiheessa.

Lopuksi haluan aivan erityisesti kiittää perhettäni oleellisesta tuesta ja kannustuksesta koko opiskelujeni ajalta. Suuret kiitokset Paulalle palautteesta ja kärsivällisyydestä, jota tämän tutkielman kirjoitus ja sovelluksen toteutus on pitkien työpäivien muodossa vaatinut.

Jyväskylässä, tammikuu 2005

Miika Nurminen

Sisältö

1 Johdanto

Tietotulva on viime vuosina kasvanut Internetin, tietokantojen ja digitaalisten kirjastojen myötä mittoihin, jotka eivät ole enää yhden ihmisen käsitettävissä. Eri alojen tieteellisen tutkimuksen kannalta tietotulvasta seuraa sisällöllisesti tai rakenteellisesti samanlaisten ongelmien käsittely täysin toisistaan poikkeavilla termeillä ja käsitteillä. Tällöin vuoropuhelu tieteiden välillä vaikeutuu. Tietotulva aiheuttaa haasteita myös organisaatioiden dokumenttien ja tietämyksen hallintaan. WWW:stä on tullut valtava linkitetty dokumenttikokoelma, johon eri hakukoneet tarjoavat rajallisia näkymiä järjestettyjen tuloslistojen muodossa. Tuloslistat eivät kuitenkaan aina vastaa käyttäjän tarpeisiin, tai käyttäjä ei yksinkertaisesti keksi ”oikeita” hakusanoja. Parhaimmassakin tapauksessa relevantit dokumentit voivat hajautua ympäri tuloslistaa, joka saattaa sisältää tuhansia hakutuloksia.

Tietotulvan ohella toinen viime vuosien ilmiö on ollut XML:n nousu merkittäväksi tiedon esitysformaatiksi. XML on laajalti korvannut SGML:n dokumenttikeskeisen tiedon osalta (merkittävänä poikkeuksena HTML – XHTML-sivujen osuus kasvaa kuitenkin jatkuvasti), mutta tullut myös uusille sovellusalueille mm. vaihtoehtona tietokannoissa olevan tiedon esittämiseen ja sovellusintegraatioon. Tästä kattavuudesta johtuen XML:n käytöstä on kuitenkin tullut lähes itseisarvo: vaikka formaatti itsessään on hyvä ja monipuolinen, XML:ää hyödyntävät sovellukset eivät ole vielä kypsää teknologiaa. Uusi formaatti ei sellaisenaan ole perusteltu syy korvata aiemmin toimivaa järjestelmää uudella. Kaksi erityisongelmaa ovat standardin editointiympäristön puuttuminen (XML:n kirjoittaminen käsin on työlästä eikä sitä voida vaatia loppukäyttäjiltä – kaupallisia editointiympäristöjä on saatavilla, mutta niitä on käytössä lähinnä raskaissa yritysjärjestelmissä) ja relaatioalgebran kaltaisen selkeän ja laajasti hyväksytyn kyselymallin puuttuminen, mikä on aiemmin mahdollistanut relaatiotietokantojen menestyksen.

Tutkimuksen tavoitteena on yrittää vastata tietotulvan tuomiin haasteisiin rakenteisten dokumenttikokoelmien osalta ja samalla luoda yleishyödyllinen sovellus XML-muotoisen tiedon käsittelyyn. Menetelmäksi on valittu tiedonlouhinta, joka on eräs mahdollisuus dokumenttien (tai minkä tahansa laajan tietomassan) hahmottamiseen. Tiedonlouhinnan avulla kokoelman tietoja voidaan tiivistää ja löytää sieltä uusia suhteita automaattisesti. Tarkoituksena on laajentaa perinteisten hakukoneiden toimintaa niin, että käyttäjä näkee tuloslistan lisäksi yleiskuvauksen dokumenteista klusterien (ryppäiden, ryhmien) muodossa. Pitkän ai-

kavälin tavoitteena empiirisen osuuden työllä pyritään luomaan pohjaa uusille tietämyksen hallintatavoille memex-koneen [21] hengessä.

Valittua aihetta leimaa vahvasti monitieteisyys. Yleistä viitekehystä edustavat tiedonlouhinta ja tietämyksen muodostaminen ovat lähes kaikenkattavia yleiskäsitteitä suurten tietomassojen käsittelyyn. Klusteroinnilla, joka on tutkimukseen valittu perusmenetelmä tiedonlouhinnan alueelta, on pitkät perinteet myös tilastotieteen, hahmontunnistuksen ja koneoppimisen osana. Tutkielmassa käsitellyistä algoritmeista suurin osa on kuitenkin luonteeltaan heuristisia, vailla vahvaa teoreettista pohjaa. Tutkimuskohde on vaatinut yleistä tiedon rakenteisuuden ja XML:n ominaisuuksien käsittelyä. Lisäksi menetelmät dokumenttien esittämiseen ja osaltaan myös käyttöliittymän toteutukseen ovat tiedonhaun sovelluksia. Tiedonhaun lähestymistapa on tiedonlouhinnasta poikkeava: jälkimmäinen pyrkii löytämään automaattisesti uusia hahmoja, kun taas edellisen päätavoite on tyydyttää käyttäjän tietotarpeet. Tiedonhakua käsitellään tässä lähinnä teknisestä näkökulmasta käytettävyysasioiden jäädessä vähemmälle huomiolle. Myös hypertekstikirjallisuutta, bibliometriikkaa ja linkkianalyysia sivutaan, koska viimeistään HTML-dokumenttien myötä linkit ovat tulleet erottamattomaksi osaksi rakenteisia dokumentteja. Empiirinen osuus on vaatinut myös visualisointimenetelmiin tutustumista, joten lähdemateriaali on kaikenkaikkiaan ollut varsin laaja. Tieteidenvälisyydestä johtuen lähteissä on ollut jonkin verran rinnakkaista termistöä ja ristiriitaisiakin näkökulmia, joita on pyritty sovittamaan. Aiheen laajuudesta johtuen työn kuluessa on pitänyt myös tehdä rajauksia.

Tarkemmin määriteltynä tutkimusongelma on XML-dokumenttien klusterointi ja tiedonhaku sekä niiden yhdistämisen tarjoamat mahdollisuudet samaa skeemaa noudattavien dokumenttikokoelmien hahmottamisessa. Kummatkin menetelmät hyödyntävät indeksiä, joka sisältää dokumentin tekstisisällön, linkit, rakenteen ja metatiedot tiivistetyssä muodossa. Klusteroinnissa dokumentit ryhmitellään niiden keskinäisen samanlaisuuden perusteella, tiedonhaun hakutulokset saadaan vertaamalla käyttäjän kyselyä dokumentteihin. Kirjallisuuskatsauksessa käydään yleisellä tasolla läpi erilaisia indeksirakenteita, klusterointialgoritmeja, samanlaisuusmittoja ja hakumenetelmiä. Empiirisessä osuudessa on kehitetty haku- ja klusterointisovellus ExtMiner, joka hyödyntää osaa teoreettisessa osuudessa tutkituista malleista.

Tutkielma jakautuu seuraaviin osiin: luvussa 2 määritellään tiedonlouhinta ja käydään läpi sen lähialueita, menetelmiä ja sovelluksia. Luku 3 keskittyy rakenteisten dokumenttien ominaisuuksiin. Luvussa 4 tarkastellaan klusterointia tiedonlouhinnan osa-alueena ja käydään läpi eri klusterointimenetelmiä. Luvussa 5 perehdytään tekstitiedon analysointitekniikoihin sekä tiedonhaun että klusteroinnin näkökulmasta. Luku 6 kuvaa empiirisessä osuudessa toteutetun sovelluksen sekä siinä käytetyn haku- ja klusterointimallin. Luku 7 on yhteenveto.

2 Tiedonlouhinta

Tietämyksen muodostaminen tarjoaa yleisen kehyksen ja näkökulman myöhemmissä luvuissa esitettäville haku- ja klusterointimenetelmille. Luvussa käsitellään tiedonlouhintaa ja tietämyksen muodostamista yleisellä tasolla. Tiedonlouhinnan lähitieteiden ja siitä edelleen kehittyneiden louhintatieteiden suhteita pohditaan.

2.1 Tiedonlouhinnan taustaa

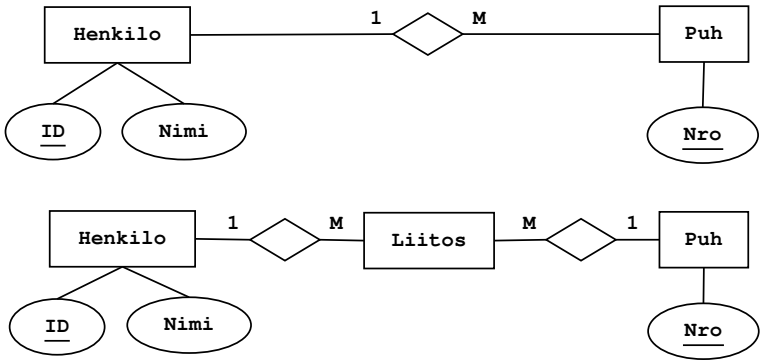
Luvussa käsitellään tiedonlouhinnan taustaa tietokantojen näkökulmasta. Lisäksi arvioidaan tietotulvan aiheuttamia ongelmia ja tiedonlouhinnan merkitystä sen mahdollisena ratkaisuna. Data, informaatio ja tietämys määritellään tämän tutkielman kannalta.

2.1.1 Tietokannat, tietovarastot ja tietojoukot

Elmasri & Navathe [50, sivut 4-5] määrittelevät tietokannan (*database*) olevan *loogisesti yhtenäinen kokoelma tietoa, jolla on jokin merkitys. Tietokanta esittää reaalimaailman asioita kuvattuna omassa minimaailmassaan. Tietokanta on suunniteltu tiettyä tarkoitusta varten*. Tietokannat ovat mahdollistaneet suurten tietomäärien järjestelmällisen käsittelyn yhtenäisellä tavalla. Lähes kaikki tietojärjestelmät hyödyntävät tietokantoja tavalla tai toisella.

Tietokanta sijaitsee yleensä omalla palvelimellaan. Tietojen fyysisestä käsittelystä vastaa tietokannan hallintajärjestelmä, joka tarjoaa standardin rajapinnan sovellusohjelmille. Ylivoimaisesti suosituimpia tietokantoja ovat relaatiotietokannat, joissa standardi tapa hakea ja päivittää tietoja on SQL (*Structured Query Language*) -kieli [50, sivut 243-244]. Tietojen haku ja raporttien teko vaatii erillisten kyselyjen tekoa, ja tietokantasovellukset ovatkin usein olennaisesti SQL-kyselymuotoilijoita, jotka käyttäjän hakukriteerien perusteella generoivat SQL-kyselyjä ja näyttävät tulokset käyttäjälle muotoiltuina raportteina. Kyselyn tulos on tietokannan tietojen tapaan taulukko, jossa voi olla myös alkeellisia koosteita, kuten summia, keskiarvoja ja ryhmittelyjä tietyn kentän mukaan. Edellytyksenä hauille ja päivityksille on kuvaus tietokannan rakenteesta, jota kutsutaan skeemaksi (*schema*). Muita tietokantatyyppejä ovat mm. olio- ja paikkatietokannat, mutta niitä ei käsitellä tässä tarkemmin.

Teoreettinen malli relaatiotietokantojen tiedon mallintamiseen on E. Coddin 1970 kehittämä relaatiomalli, jossa tietokohteet esitetään määrämuotoisina tauluina eli relaatioina. Kohteen ominaisuudet ovat kenttiä ja ilmentymät tietueita. Relaatiomallin erityisvaatimuksena on, että kenttien arvojen pitäisi olla jakamattomia [50, sivut 195-199]. Esimerkiksi tilanteessa, jossa henkilöllä voi olla monta puhelinnumeroa, niitä ei voida määrittää henkilön kentäksi, vaan tarvitaan erillinen *puhelinnumerot*-aputaulu. Siinä jokainen puhelinnumero on erillinen tietue ja viittaa edelleen johonkin henkilöön. Jos monella henkilöllä voi olla vielä sama puhelinnumero, tilanne monimutkaistuu entisestään. Tämän *monesta-moneen*-suhteiden hallinnan hankaluuden takia relaatiomallia pidetään joustamattomana tietomallina. Kuvassa 2.1 on ER-kaavio puhelinnumerotietokannasta kummassakin tilanteessa.



Kuva 2.1: Puhelinnumeroesimerkki.

Tästä huolimatta relaatiotietokannat ovat standardi ja hyväksi havaittu tapa tiedonhallintaan erityisesti kaupallis-hallinnollisissa sovelluksissa. Relaatiomalli soveltuu parhaiten tiedolle, jolle voidaan määritellä selkeä kenttäkohtainen rakenne eikä suhteisiin liittyvien lukumäärien suhteen ole poikkeuksia. Tiedot ovat muodoltaan yleensä numeerista tietoa, lyhyitä merkkijonoja, päivämääriä tai aikoja.

Suurissa organisaatioissa voi olla lukuisia tietokantoja eri tietojärjestelmien osana. Niiden tietoja voidaan kerätä yhteen suureen tietokantaan, jota kutsutaan tietovarastoksi (*data warehouse*). Tavallisiin tietokantoihin verrattuna tietovarastoissa on monipuoliset haku- ja analysointitoiminnot. Hakutoimintoja kutsutaan suora-analyysiksi (*OLAP, On-Line Analytical Processing*). Sille on ominaista moniulotteinen tietomalli, jossa esimerkiksi aika- tai paikkatiedot esitetään omina akseleinaan. Tietovarastoja käytetään johdon päätöksenteon tukena ja keskitetyssä raportoinnissa. Tietovarastoa muodostettaessa lähdetietokantojen mahdollisesti eri esitystavoilla kuvatut tiedot yhtenäistetään ja data pyritään puhdistamaan syöttövirheistä. Kerätty data sovitetaan tietovaraston moniulotteiseen tietomalliin. [50, sivut 842-855]

Relaatiotietokantojen kaupallisesta merkityksestä huolimatta kaikkea digitaalista tietoa ei ole järkevää pakottaa relaatiomallin rakenteiseen kehykseen (rakenteisen ja rakenteettoman

tiedon eroista enemmän luvussa 3.2). Kaikkea tietoa ei yleensäkään pidetä tietokannoissa. Tämän tutkielman kannalta kiinnostavinta tietoa ovat dokumenttikokoelmat, jotka voivat olla yhtä lailla tiedostojärjestelmässä tai dokumenttitietokannassa. WWW on itsessään valtava dokumenttikokoelma, jonka erityispiirteenä ovat mm. hyperlinkit dokumenttien välillä. Teollisuudessa merkittäviä tietomääriä saadaan esim. prosessi- ja mittausdatasta, joka on tyypillisesti numeerista tietoa, mutta jonka varastointitavat ja -aika voivat vaihdella merkittävästi. Myös sähköposti- ja uutisryhmäarkistot muodostavat laajan relaatiotietokannoista erillisen joukon. Tietojen analysoinnin kohde ei siis välttämättä ole (relaatio)tietokanta, joskin kirjallisuudessa usein painotetaan juuri tietämyksen muodostamista *tietokannoista* (katso luku 2.2.1). Jatkossa käytetään yleisempää käsitettä tietojoukko (*dataset*) korostamaan analysoitavan tiedon monimuotoisuutta – tietokannat yhtenä mahdollisena tutkimuskohteena.

2.1.2 Tarve tietämykselle

Vannevar Bush totesi jo vuonna 1945 *As We May Think* -artikkelissaan, kuinka tieteenalojen erikoistumisen ja tutkimustiedon määrän kasvaessa tutkijat eivät pysty seuraamaan kaikkia oman alansa tuloksia. Erikoistuminen on välttämätöntä edistykselle, mutta tieteidenvälinen toiminta jää erikoistumisen myötä pinnalliseksi [21]. Sisällöllisesti tai rakenteellisesti samanlaisia ongelmia tai ratkaisuja saatetaan käsitellä eri aloilla täysin toisistaan poikkeavilla termeillä ja käsitteillä, jolloin vuoropuhelu tieteiden välillä vaikeutuu (puhumattakaan yleistajuisesta materiaalista). Tutkijat kehittävät samoja asioita toisistaan tietämättä. Lopullisena riskinä on, että aidosti uusia teorioita ei synny, vaan joku on keksinyt sen aiemmin toisessa asiayhteydessä. Tietotulva on myös yhteiskunnallinen ongelma, koska se edistää asiantuntijavaltaa demokratian kustannuksella. Kuvaava esimerkki tieteenalojen eriytymisestä on, että suosittu K-means-klusterointialgoritmi on eri aloilla keksitty useaan otteeseen. Esimerkiksi informaatioteorian puolella K-means tunnetaan myös yleistettynä Lloydin algoritmina tai Linde-Buzo-Gray-algoritmina [91].

Nykyään tietotulva on kasvanut Internetin, tietokantojen ja digitaalisten kirjastojen myötä käsittämättömiin mittoihin. Luotettavasta ja tiivistetystä tiedosta on tullut ”kriittinen menestystekijä” ja arvo sinänsä. WWW:n monimuotoinen, linkitetty ja jatkuvasti kasvava tietojoukko aiheuttaa suuria haasteita tiedon etsijöille ja hakukoneille, joiden mahdollisesti tuhansia dokumenttiehdokkaita sisältävät tuloslistat eivät ole kokemattoman käyttäjän käsiteltävissä. Tietoa tuotetaan nopeammassa tahdissa kuin ihmiset pystyvät sitä käsittelemään, jolloin vaarana on, että relevantti tieto sekoittuu hälyyn. Bush esitti tietotulvan ratkaisuksi memex-konetta: hypertekstijärjestelmää, joka muistuttaa ideoiltaan (mutta ei tekniikaltaan) etäisesti nykyistä WWW:tä linkkeineen:

> A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.
>
> ...associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the memex. The process of tying two items together is the important thing. [21]

Hakukoneisiin tottuneelle käyttäjälle WWW:stä on tullut memexin kaltainen muistin jatke. Linkit liittävät tietoa sivulta toisille kuten memexin assosiaatiot – tosin vain yksisuuntaisesti. Pelkät linkit eivät ratkaise tietojen organisointiongelmaa, koska ne ovat tyypillisesti käsin päivitettäviä eikä kukaan WWW:n sisällöntuottaja pysty linkittämään sivuilleen kaikkea aiheeseensa olennaisesti liittyvää tietoa. Lisäksi käyttäjien tarpeet vaihtelevat. Tietoa pitäisi organisoida automaattisesti, laajoina mielekkäinä kokonaisuuksina, jotka esitetään käyttäjälle tiivistetyssä muodossa ja tarvittaessa eri näkökulmista. Tarkastelutaso on siirrettävä informaatiosta tietämykseen (*knowledge*).

Data, informaatio ja tietämys ovat toisiinsa liittyviä käsitteitä, joilla on tiedonlouhinnan lisäksi tärkeä merkitys mm. tietämyksen hallinnassa, informaatiotieteissä ja tekoälyssä. Johtuen käsitteiden yleisyydestä ja käytöstä eri tieteenaloilla niillä ei ole yleisesti hyväksyttyä eikä yhtenäistä määrittelyä. Useimmissa määritelmissä todetaan kuitenkin, että tietämys on käsitteellisesti informaatiota rikkaampaa, joka on edelleen datan yläpuolella. Suomen kielessä lisäongelmia tuo myös suomennos *tieto*, jota käytetään sekä datasta että informaatiosta (myös tässä tutkielmassa, ellei sekaannuksen vaaraa ole). Eräissä määrittelyissä hierarkiaan on lisätty myös korkeampia tasoja, kuten viisaus (*wisdom*), mutta ne sivuutetaan tässä. Varhaisin tunnettu viittaus ”tiedon hierarkiaan” on vuodelta 1934 T. S. Eliotin näytelmästä *The Rock* [122]:

> Where is the Life we have lost in living?
>
> Where is the wisdom we have lost in knowledge?
>
> Where is the knowledge we have lost in information?

Kirjallisuudessa vallitsee melko laaja yksimielisyys siitä, että *data* on joukko erillisiä faktoja tai havaintoja. Datalla itsessään ei ole merkitystä, ellei tiedetä kontekstia, jossa se on kerätty [87, sivu 8]. Tietokantojen kannalta on myös merkittävää, että data on tallennettavissa [50, sivu 4]. *Informaatio* rikastaa dataa relevanssilla ja tarkoituksella. Informaatio voidaan hahmottaa viesteinä, jotka voivat olla esim. dokumentteja [39, sivut 3-4]. Informaatiota

voidaan tutkia joko prosessipainotteisesti (painopisteenä siirtomedia) tai käyttäytymispainotteisesti (painopisteenä viestit) [87, sivu 9]. *Tietämys* on yhdistelmä kokemusta, arvoja, kontekstuaalista informaatiota ja asiantuntijatietoa, jonka avulla informaatiota voidaan arvioida ja ottaa käyttöön. Organisaatioissa päätöksenteko perustuu tietämykseen [39, sivut 5-6]. Muiden näkemysten mukaan informaatiosta tulee tietämystä, kun se tulkitaan, sijoitetaan oikeaan kontekstiin ja sille lisätään merkitys. Toisaalta myös *perusteltu, tosi uskomus* voi kuulua tietämyksen piiriin [87, sivut 10-13]. Tiedonlouhinnan näkökulmasta (katso luku 2.2.1) tietämys on kokoelma datasta koostettuja malleja ja hahmoja, jotka ovat ymmärrettäviä, käyttökelpoisia [58], kiinnostavia ja riittävän varmoja [61] (kriteerit kiinnostavuudelle ja käyttökelpoisuudelle riippuvat käyttäjästä eikä niitä voi välttämättä määritellä formaalisti). Lisäksi tietämys liittyy kiinteästi todelliseen maailmaan [90]. Määritelmä on rajoittunut verrattuna edellä esitettyihin yleisempiin näkemyksiin, mutta tämän tutkielman kannalta se on riittävä ja vastannee tietämyksen muodostamisen alalla vallitsevaa nykyistä käytäntöä.

Perinteinen tapa tietämyksen muodostamiseen datasta on ollut manuaalinen data-analyysi ja tulkinta. Analyysin tekee asiantuntija tilastollisia menetelmiä hyödyntäen. Tietojoukkojen koon kasvaessa tämä ei kuitenkaan ole enää käytännöllistä: on arvioitu, että maailmassa olevan informaation määrä tuplaantuu 20 kuukauden välein [61] eivätkä miljardeja tietueita sisältävät tietokannat ole enää harvinaisia[1] [58]. Tietueiden määrän kasvun myötä myös datan piirteiden (ulottuvuuksien, kenttien) määrä kasvaa. Esimerkiksi tekstikokoelmaa analysoitaessa yleinen menetelmä on tulkita yksittäiset (perusmuotoon palautetut) sanat piirteinä. Tällöin pienenkin, muutamia satoja dokumentteja sisältävän kokoelman piirteiden määrä kasvaa helposti tuhansiin, mikä voi heikentää analyysin luotettavuutta (katso luku 4.4.1). Suuret data- ja ulottuvuusmäärät vaativat data-analyysin automatisointia. Erityisesti tiede ja yritykset hyötyvät jalostetusta tiedosta. Tietämys muodostaa perustan uusien teorioiden ja mallien muodostamiselle, yritykset voivat käyttää tietämystä parempien palvelujen tuottamiseen ja tuotannon tehostamiseen [58].

Tiedon hyödyntäminen ei ole suoraviivaista. Frawley *et al.* [61] luettelevat ongelmia, joita tietokannoissa olevan tiedon analysointiin liittyy. Nämä pätevät yhtä lailla muillekin tietojoukoille ja liittyvät myös datan puhdistukseen tietovarastoa muodostettaessa.

- **Dynaaminen data**: tietokantojen tyypillinen piirre on niissä olevien tietojen dynaaminen luonne. Tietoja lisätään ja päivitetään, usein samanaikaisesti eri käyttäjien toimesta.
- **Epäolennaiset kentät**: vastakohtana tieteellisille koeasetteluille tietämyksen muodostamiselle on tyypillistä, ettei kohteena olevan tietokannan tietoja ole kerätty data-

[1] Esim. Google-hakukone mainosti 20.8.2004 hakunsa kohdistuvan 4 285 199 774 sivun joukkoon. Ks. http://www.google.com/

analyysia ajatellen. ”Oikeiden” kenttien valinta voi vaikuttaa ratkaisevasti analyysin järkevyyteen.

- **Puuttuvat arvot**: tietokantoja päivitettäessä kaikkia kenttiä ei yleensä pidetä pakollisina. Joskus oletusarvoja voidaan käyttää, mutta yleisesti ottaen puuttuvat arvot vaikuttavat analyysin laatuun.
- **Kohina ja epävarmuus**: erityisesti mittauksia tehdessä mahdolliset mittausvirheet on aina huomioitava. Virheitä voi tapahtua myös syötettäessä tietoja tietokantaan. Esimerkiksi henkilön nimellä voi olla useita mahdollisia kirjoitusasuja, joita tietokanta ei osaa samaistaa.
- **Puuttuvat kentät**: sama perusongelma kuin epäolennaisilla kentillä. Tietokantaa suunniteltaessa ei pystytä huomioimaan kaikkia tutkimusongelmia, joihin tietoja mahdollisesti yritetään soveltaa.

Tietämyksen muodostamisen sovellusalueita ovat esimerkiksi asiakasprofiilien muodostaminen, hakukoneiden toiminnan tehostaminen, roskapostin suodatus, vakuutuspetosten jäljitys, televerkon virheiden analysointi ja geenitutkimus [145]. Käytännön esimerkkinä Bohnacker *et al.* [15] esittävät yrityksen ja asiakkaan välisen kommunikoinnin tehostamisen. Kohdeyrityksessä käytössä olleella palautejärjestelmällä asiakkaat pystyivät antamaan palautetta ja parannusehdotuksia yrityksen tuotteista. Palautteet jaettiin manuaalisesti yli 10000 eri kategoriaan, jotka käsittävät joukon eri tuotteita ja tunnistettuja vikatyyppejä. Palautetietojen arviointi ja etsiminen suuresta kategoriajoukosta vei työaikaa. Lisäksi järjestelmä sopeutui huonosti odottamattomaan, mutta silti relevanttiin asiakaspalautteeseen. Palautejoukko klusteroitiin, jolloin työntekijät saavat nopeasti yleiskäsityksen palautemassasta ja voivat arvioida manuaalisen luokituksen onnistumista. Sovelluksen klusterointi perustuu yksinkertaiseen ideaan verrata palautteita niissä esiintyvien sanojen jakauman perusteella. Sovelluksen kehittäjien mukaan kiinnostavia (uusia ongelma-alueita ilmaisevia) klustereita ovat sellaiset, jotka on jaoteltu manuaalisesti eri kategorioihin, mutta silti muistuttavat sanatasolla toisiaan. Myös tämän tutkielman yhteydessä kehitetty ExtMiner-sovellus (katso luku 6) pyrkii vastaamaan tarpeeseen hahmottaa laajoja tekstikokoelmia tekstitiedonhaun ja klusteroinnin avulla.

2.2 Mitä tiedonlouhinta on?

Luvussa määritellään tiedonlouhinta, tarkastellaan sen suhteita lähitieteisiin ja käydään läpi sen yleisimmät menetelmät. Tiedonlouhinta nähdään osana tietämyksen muodostamispro-

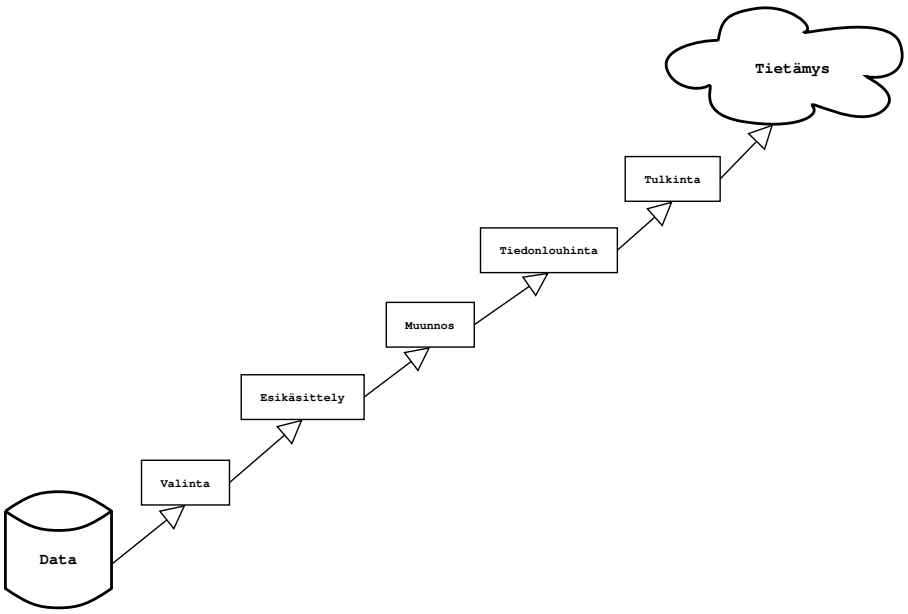
sessia, jonka kohteena voivat olla mitkä tahansa tietojoukot, eivät pelkästään tietokannat. Lopuksi käsitellään mahdollisuuksia tiedonlouhinnan ”yhtenäisteorialle”.

2.2.1 Määritelmä ja lähitieteet

Hand *et al.* [77, sivut 1-4] määrittelevät tiedonlouhinnan (*data mining, tiedonrikastus*) olevan *suurten, tiettyä tarkoitusta varten kerättyjen tietojoukkojen analyysia, jonka tarkoituksena on löytää odottamattomia suhteita ja tiivistää dataa uusilla tavoilla, jotka ovat sekä ymmärrettäviä että käyttökelpoisia*. Tuloksena saatavat mallit ja hahmot voivat olla esim. tilastollisia tai loogisia [58]. Tiedonlouhinta on tieteidenvälistä toimintaa, jonka piiriin kuuluu joukko erilaisia menetelmiä ja algoritmeja. Tiedonlouhinnan lähitieteitä ovat tilastotiede, tietokannat, koneoppiminen, hahmontunnistus, tekoäly ja visualisointi. Tiedonlouhintaa kutsutaan joskus myös sekundääriseksi data-analyysiksi, koska tiedonlouhinnan kohteena olevaa dataa ei ole yleensä kerätty data-analyysia silmälläpitäen. Tämä korostaa luvussa 2.1.2 mainittuja tietokantojen analysointiin liittyviä ongelmia.

Tiedonlouhinta voidaan nähdä osana tietämyksen muodostamista tietokannoista (*Knowledge Discovery in Databases, KDD*). Fayyad *et al.* [58] määrittelevät tietämyksen muodostamisen olevan *epätriviaali prosessi, jossa pyritään muodostamaan päteviä, uusia, potentiaalisesti käyttökelpoisia ja lopulta ymmärrettäviä malleja datasta*. Prosessi on interaktiivinen ja iteratiivinen. *Valinnassa* (vaihe 1) kerätään prosessin kohteena oleva tietojoukko tai valitaan alkiot suuremmasta joukosta. *Esikäsittelyssä* (vaihe 2) eli datan puhdistuksessa pyritään poistamaan ”kohinaa” ja täyttämään tyhjät havainnot – kuten tietovarastoa muodostettaessa (Fayyad *et al.* pitävätkin tietämyksen muodostamista automatisoituna suora-analyysina). *Muunnos* (vaihe 3) vastaa tilastollisen hahmontunnistuksen *piirteiden valintaa ja erottelua*, jossa datasta valitaan käsiteltävät ominaisuudet ja mahdollisesti muunnetaan dataa toiseen muotoon jatkokäsittelyä varten [83] (vaihe on tärkeä erityisesti tekstidatan käsittelyssä, koska tiedonlouhinta-algoritmit eivät käsittele suoraan tekstiä, vaan siitä johdettua numeerista esitystä). *Tiedonlouhinnan* (vaihe 4) tulokset esitetään usein visualisoinnin avulla. *Tulkinta* (vaihe 5) tarkoittaa tulosten arviointia sekä mahdollista jatkokäsittelyä. *Tietämys* merkitsee tässä yhteydessä hahmoa tai mallia, joka on käyttäjän kannalta kiinnostava ja riittävän varma [61]. Kodratoff [90] korostaa, että tietämyksen täytyy liittyä todelliseen maailmaan, jolloin se vaikuttaa sovelluksen käyttäjän toimintaan. Prosessi on esitetty kuvassa 2.2.

Edellisten (sinänsä lähellä toisiaan olevien) määritelmien ohella termejä *tiedonlouhinta* ja *tietämyksen muodostaminen* käytetään kirjallisuudessa synonyymeina. Tietämyksen muodostamisprosessi painottaa erityisesti tietokannoissa olevan tiedon analysointia, mutta tiedonlouhintaa voidaan tarkastella myös yleisemmin ottamatta kantaa syötetiedon formaattiin.



Kuva 2.2: Tiedonlouhinta osana tietämyksen muodostamisprosessia.

Sitä voidaan soveltaa kaikkeen digitaalisessa muodossa olevaan dataan [145]. Oikeampi nimitys KDD-termille olisi siis *Knowledge Discovery from Datasets*, tietämyksen muodostaminen tietojoukoista. Hand *et al.* [77, sivut 4-9] käyttävät *dataset*-termiä merkitsemään vain mittausjoukkoa, mikä vastaa matriisia tai yhtä relaatiotietokannan taulua. Tässä tutkielmassa tietojoukon käsite on yleisempi, eli datan ei välttämättä tarvitse olla matriisimuodossa, vaan esim. XML-dokumenttikokoelma käy tietojoukoksi. Jos tietojoukko on relaatiotietoa, tauluja voi olla useampia ja niiden rakenne voi vaihdella. Tässä tutkielmassa tiedonlouhintaa (teksti- web- ja relaatiotieto mukaanlukien) pidetään KDD-prosessin osana siten, että analysoitava data voi olla peräisin mistä tahansa tietojoukosta.

2.2.2 Louhintatieteiden perhe

Tiedonlouhinnassa käytetty tietojoukko on perinteisesti ollut matriisi, joka sisältää numeeristen tai lueteltujen muuttujien arvoja [77, sivut 4-9]. Yksittäinen relaatiotietokannan taulu tai tietovarasto soveltuvat tietojoukoksi. Numeerisen tiedon louhinnan rinnalle on kehittynyt lukuisia uusia monimuotoisen syötetiedon louhintaan keskittyviä tutkimusalueita. Näistä keskeisimpiä ovat tekstitiedonlouhinta (*text mining*), web-louhinta (*web mining*) ja relaatiotiedonlouhinta (*multirelational data mining*).

Tekstitiedonlouhinta (dokumenttien louhinta, tekstianalyysi) on tiedonlouhinnan tekniikoiden soveltamista (yleensä rakenteettomaan tai puolirakenteiseen) tekstidataan tai dokument-

tijoukkoon. Tyypillisiä sovelluksia ovat esimerkiksi lyhennelmien generointi, dokumenttien luokittelu, klusterointi ja tekstitiedonhaun tehostaminen [49]. Tekstitiedonlouhintaa ja sen suhdetta tiedonhakuun sekä luonnollisen kielen käsittelyyn käsitellään perusteellisemmin luvussa 3.1.2.

Web-louhinta on tiedonlouhintaa WWW-ympäristössä [93]. Web-louhinnan tunnusomaisia piirteitä ovat HTML-dokumenttien, linkkirakenteiden, web-sivustojen ja aihehakemistojen analyysi. Rakenteisen tekstin lisäksi WWW sisältää monimuotoista tietoa: kuvia, ääntä, animaatioita, ohjelmakoodia (esim. Java-sovelmat ja ActiveX-kontrollit) ja valmistajakohtaisia sisältölaajennuksia (esim. Flash). Dokumentit ja sivustot ovat jatkuvasti muuttuvia ja kohinaisia: suurin osa web-dokumenteista ei noudata tarkasti HTML-kielen skeemaa ja relevantti tieto on erotettava esim. mainoksista tai harhaanjohtavasta metatiedosta [22, sivut 11-12]. Etzioni [55] jakaa web-louhintaprosessin *resurssien löytämiseen*, *tiedon eristämiseen* (vastaavat web-hakukoneiden indeksointia tai KDD-prosessin kahta ensimmäistä vaihetta) ja *yleistämiseen* (vastaa tiedonlouhinnan mallien löytämistä). Kosala [93] lisää tähän vielä *analyysin*, joka vastaa KDD-prosessin *tulkintaa* (vaihe 5). Resurssien löytäminen on tarpeen WWW:n valtavan dokumenttimäärän takia – muusta tiedonlouhinnasta poiketen tutkittava datajoukko ei välttämättä ole etukäteen tiedossa. Web-louhinta voidaan edelleen jakaa sisällön, rakenteen ja käytön analysointiin. Rakenneanalyysin granulariteetti voi vaihdella dokumenttien osien hausta useita sivustoja kattavien WWW-yhteisöjen kartoittamiseen. Käytön analysointi voi kohdistua esim. web-palvelimien käyttäjälokeihin.

Relaatiotiedonlouhinta etsii malleja ympäristössä, joka käsittää useita erilaisilla suhteilla toisiinsa liittyviä tietojoukkoja. Tietojoukot voivat olla esim. relaatiotietokannan tauluja. Perinteiseen tiedonlouhintaan liittyy yleensä oletus, että tutkittavat näytteet ovat riippumattomia ja samasta jakaumasta (*Independent and Identically Distributed, IID*). Relaatiodata rikkoo tämän oletuksen. Relaatiodata voidaan ”pakottaa” yhteen tauluun korvaamalla viitetiedot toisessa taulussa olevilla ilmentymillä. Tämä ”latteistaminen” on käänteinen operaatio tietokannan muodostamisen aikana tehtävälle normalisoinnille ja se mahdollistaa relaatiotiedon analysoinnin tavallisilla tiedonlouhintamenetelmillä. IID-oletus ei kuitenkaan ole voimassa, koska datassa on jo lähtökohtaisesti riippuvuuksia. Lisäksi latteistamisen yhteydessä eksplisiittinen riippuvuustieto häviää. Relaatiotiedon luokittelusäännöt ovat yhden taulun luokittelusääntöjä ilmaisuvoimaisempia: jälkimmäiset perustuvat propositiologiikkaan ja edelliset 1. asteen predikaattilogiikkaan. Perusmenetelmä relaatiotiedonlouhintaan on induktiivinen logiikkaohjelmointi (*Inductive Logic Programming, ILP*), joka on päättelysääntöjen etsimistä ja yleistämistä annettujen esimerkkien ja taustatiedon avulla. Deduktiivisessa päättelyssä hyödynnetään vain annettuja aksioomia ja päättelysääntöjä, mutta induktiivinen päättely pyrkii löytämään uusia. Tässä mielessä ILP on deduktiivisen päättelyn vastakohta. [48]

2.2.3 Tiedonlouhinnan menetelmät

Tiedonlouhinnan menetelmillä pyritään muodostamaan malleja syötedatasta löydettyjen hahmojen ja säännönmukaisuuksien pohjalta. Malli (*model*) on globaali tiivistelmä tietojoukosta. Esimerkki yksinkertaisesta mallista on regressiomalli $Y = aX + b$, missä Y ja X ovat satunnaismuuttujia ja a ja b ovat mallin parametreja. Hahmo (*pattern*) on väite tai sääntö, joka kuvaa rajoitettua osaa datasta [77, sivut 9-11]. Hahmojen ja mallien ero ei ole aina selkeä (eikä parametrittomassa klusteroinnissa malleista puhuminen ole välttämättä edes mielekästä), mutta käsitteet ovat laajassa käytössä tiedonlouhinnan kirjallisuudessa.

Tiedonlouhinnan menetelmät voidaan jakaa kuvaileviin ja ennustaviin. Kuvailevat menetelmät pyrkivät esittämään datan yleisiä ominaisuuksia, ennustavien menetelmien avulla voidaan tehdä päättelyä. Tiedonlouhinnan menetelmät voidaan jakaa karkeasti klusterointiin, luokitteluun ja assosiaatiosääntöjen etsimiseen [76, sivut 21-28]. Hand *et al.* [77, sivut 11-15] käyttävät yleisempiä termejä *kuvaileva mallinnus*, *ennustava mallinnus* sekä *hahmojen ja sääntöjen etsiminen*.

- **Klusterointi** on kuvaileva menetelmä, jossa tietoalkiot ryhmitellään äärelliseen määrään klustereita niiden keskinäisen samanlaisuuden perusteella. ”Samanlaisuus” määritellään yleensä etäisyysfunktiolla. Tähän ryhmään voidaan liittää myös muut kuvailevat mallinnusmenetelmät, kuten muuttujien välinen riippuvuusanalyysi. Klusterointia käsitellään tarkemmin luvussa 4.

- **Luokittelu** ja **regressio** ovat ennustavia menetelmiä. Luokittelussa tietoalkiolle pyritään määrittämään jokin ennalta määrätyistä luokista. Luokittelumallia rakennettaessa pyritään aina minimoimaan luokitteluvirhe. Regressiota voidaan pitää luokittelun yleistyksenä, jossa tietoalkioille määritellään numeerinen arvo luokan sijaan. Toisaalta myös mallipohjaisen klusteroinnin tulosta voidaan käyttää luokittelumallina. Luokittelun ja klusteroinnin suhdetta käsitellään luvussa 4.1, regression käsittely sivuutetaan.

- **Assosiaatioiden** ja **peräkkäisten toimintojen** etsintä. Assosiaatiot ovat kuvaileva malli toistuvasti yhdessä esiintyville tietueille (*frequent itemsets*). Peräkkäiset toimintosäännöt ovat ennustavia assosiaatioita, joiden kohdalla on tiedossa tietueiden suhteellinen järjestys. Sääntöjen yleinen muoto on implikaatio $X \Rightarrow Y$. Assosiaatiosääntöihin liittyviä tärkeitä validointimittoja ovat tuki (*support*) ja luottamus (*confidence*), jotka voidaan määritellä tietuepareille seuraavasti [76, sivut 27-28]:

$$tuki(X \Rightarrow Y) = P(X \cup Y)$$

$$luottamus(X \Rightarrow Y) = P(Y|X),$$

missä $P(Y|X)$ on ehdollinen todennäköisyys Y ehdolla X. Tuki mittaa, kuinka suuressa osassa tietokantaa X ja Y (voidaan yleistää myös useampiin arvoihin) esiintyvät yhdessä. Luottamus mittaa assosiaation todennäköisyyttä saatavilla olevan datan perusteella. Perusmenetelmä yhdessä esiintyvien tietueiden etsintään on Apriori-algoritmi. Se etenee rekursiivisesti alkaen yksittäisistä tietueista, joiden esiintymistodennäköisyys (tuki) on kynnysarvon yläpuolella. Tämän jälkeen kokoelmaa laajennetaan asteittain suurempiin yksiköihin yhdistämällä kokoelman joukot ristitulolla itseensä. Tulosjoukoista poistetaan ne, joiden tuki on kynnysarvon alapuolella. Lisäksi laajennuksen aikana karsitaan pois ne joukot, joiden osajoukot on aiemmin poistettu [76, sivut 230-235].

Han & Kamber [76, sivut 21-28] lisäävät omaksi menetelmäkseen kuvausten ja koosteiden luomisen (Hand *et al.* kutsuvat tätä eksploratiiviseksi data-analyysiksi). Kuvausten ja koosteiden luominen liittyy kuitenkin lähes aina tietämyksen muodostamisprosessiin, joten tämän kirjoittajan mielestä sitä ei ole perusteltua erotella omaksi menetelmätyypiksi. Sama pätee Hanin *et al.* ehdottamaan poikkeamien (*outliers*) analyysiin. Hand *et al.* ehdottavat omaksi tiedonlouhinnan lajikseen myös sisältöhakua (*retrieval by content*), samanlaisten näytteiden hakua tietojoukosta käyttäjän esimerkin perusteella. Tämä voidaan kuitenkin tulkita erikoistapauksena klusteroinnista tai (teksti)tiedonhausta, joiden suhdetta pohditaan tarkemmin luvussa 3.1

Fayyad *et al.* [58] osittavat yksittäiset tiedonlouhintamenetelmät edelleen mallin esittämiseen, mallin arviointiin ja hakuun. Mallin esittäminen tarkoittaa kieltä, jolla löydetyt hahmot kuvataan. Esitystapoja voivat olla esim. luonnollinen kieli, loogiset säännöt tai kaaviot [61]. Mallin arviointikriteerit kuvaavat, kuinka hyvin löydetyt hahmot täyttävät tietämyksen muodostamisprosessin tavoitteet. Esimerkiksi ennustavia malleja voidaan arvioida ennustustarkkuudella. Kuvaavia malleja voidaan arvioida myös uutuudella, käyttökelpoisuudella ja ymmärrettävyydellä. Haku voidaan jakaa edelleen mallin ja parametrien etsintään. Hand *et al.* [77, sivut 15-18] lisäävät tähän vielä tiedonhallintastrategian. Tiedonhallinnasta tulee ongelma, kun tietojoukko ei mahdu kerralla muistiin. Tällöin tiedonlouhinta-algoritmin on otettava kantaa tietojen tallennukseen, indeksointiin ja saantiin. Kun mallin esitystapa ja arviointikriteerit on valittu, tiedonlouhintaongelma voidaan tulkita optimointiongelmana, jossa on löydettävä parhaat mallit ja parametrit arviointikriteerien kannalta. Ositus toimii käytännössä, jos tiedonlouhintatehtävä voidaan tulkita tilastollisessa ympäristössä ja arviointikriteerit ovat selkeät. Valitettavasti monilla sovellusalueilla (erityisesti klusteroinnissa) näin ei kuitenkaan välttämättä ole.

2.2.4 Kohti tiedonlouhinnan teoriaa?

Luvussa 2.2.1 todettiin tiedonlouhinnan lähitieteitä olevan tilastotiede, tietokannat, koneoppiminen, hahmontunnistus, tekoäly ja visualisointi. Voidaan kysyä, onko tiedonlouhinta todella oma tieteenalansa vai voisiko sen ”upottaa” esimerkiksi hahmontunnistukseen, jolla on jo kymmenien vuosien tutkimusperinne. Tietämyksen muodostamisen kannalta nämä tieteenalat tarjoavat osan menetelmistä, joita KDD-prosessin tiedonlouhintavaiheessa käytetään. Tietämyksen muodostaminen keskittyy prosessiin kokonaisuutena ottaen data-analyysialgoritmien lisäksi huomioon tiedonhallinnan, algoritmien skaalauksen suuriin tietomääriin, tulosten tulkinnan, visualisoinnin ja käyttöliittymäkysymykset. Tietämyksen muodostaminen pyrkii automatisoimaan manuaalisen data-analyysin hypoteesien valintoineen niin pitkälle kuin on mahdollista ja järkevää. Erityisesti tilastotieteellisessä kirjallisuudessa tiedonlouhintaa on kritisoitu siitä, ettei kone voi korvata data-analyytikon arvostelukykyä: etsimällä tarpeeksi kauan mistä tahansa (mahdollisesti täysin satunnaisesta) tietojoukosta on mahdollista löytää hahmoja, jotka vaikuttavat tilastollisesti merkitseviltä, mutta eivät todellisuudessa sitä ole. Tämän takia tiedonlouhinnassa on aina otettava huomioon tutkittavan ongelman tilastolliset ominaisuudet [58]. Klusteroinnin kannalta validointiongelmaa käsitellään luvussa 4.4.

Tiedonlouhinnan lähitieteistä tietokannat ja tiedonhallinta tarjoavat perustekniikan suurten tietojoukkojen käsittelyyn, päivitykseen ja kyselyihin. Tietokantajärjestelmät eivät kuitenkaan pysty itsessään päättelemään, mitkä kyselyt ovat toivottavia tai arvioimaan tulosten kiinnostavuutta. Tekoälyn alueella kehitetyt asiantuntijajärjestelmät ovat erikoistuneet tietämyksen esittämiseen ja sääntöjen päättelyyn, mutta lähtödata on tyypillisesti ollut paljon tietokantojen tietoa laadukkaampaa (vähemmän kohinaa ja epäolennaisia näytteitä). Lisäksi järjestelmän koulutusvaiheessa on hyödynnettävä asiantuntijaa. Tilastotiede on luonnollinen teoreettinen perusta tiedonlouhinnalle, mutta sen menetelmät ovat riittämättömät luokka- ja rakenteisen tiedon käsittelyyn. Sovellusaluekohtaisen taustatiedon käyttö ei yleensä ole mahdollista tai sen pukeminen informatiiviseksi prioriksi bayesilaisessa päättelyssä on erittäin vaativaa. Analyysin tulkinta vaatii asiantuntemusta. Koneoppimisen algoritmeja on perinteisesti sovellettu yhden taulun staattiseen tietojoukkoon eivätkä ne skaalaudu erittäin suuriin tietojoukkoihin [61]. Hahmontunnistus tarjoaa menetelmiä tiedonlouhinta- ja arviointivaiheeseen – luokittelua ja klusterointia on hahmontunnistuksen piirissä tutkittu jo pitkään. Visualisointi on apuna KDD-prosessin loppuvaiheessa arvioitaessa sitä, miten tiedonlouhinnan tulokset ovat parhaiten esitettävissä käyttäjälle.

Mannila [103] on pohtinut mahdollisuuksia sovittaa tiedonlouhinta kokonaisuutena johonkin teoreettiseen kehykseen. Motivaationa ”tiedonlouhinnan teorian” kehittämiseen hän esit-

tää tietokantojen teorian kehityksen: 1960-luvulla alaa pidettiin epäyhteensopivien sovellusten sekamelskana vailla teoreettista mielenkiintoa, kunnes Coddin relaatiomalli mahdollisti tiedon ja sille tehtävien operaatioiden kuvaamisen yhtenäisesti. Tämän seurauksena voitiin kehittää kyselyjen optimointia, transaktioita ja yleiskäyttöisiä tietokannan hallintajärjestelmiä. Teorian täytyisi pystyä mallintamaan tyypilliset tiedonlouhinnan tehtävät (klusterointi, luokittelu, sääntojen etsintä), käsitellä löydettyjä sääntöjä tilastollisesti, yleistää dataa, sallia eri formaatteja tiedolle (relaatiotieto, sekvenssit, teksti, WWW). Teorian täytyisi myös ottaa huomioon KDD-prosessin interaktiivisuus, iteratiivisuus ja löydettyjen mallien ymmärrettävyys. Mannila esittää mahdollisina lähtökohtina tilastotieteen, koneoppimisen, todennäköisyysteorian (tavoitteena datan muuttujien todennäköisyysjakauman löytäminen), tietojen pakkauksen (tietämys tulkitaan esityksenä, joka tiivistää alkuperäistä tietoa mahdollisimman paljon), mikrotaloustieteen (tavoitteena sellaisen mallin löytäminen, joka tuottaa maksimaalisen hyödyn) ja induktiiviset tietokannat (tietokanta sisältää datan lisäksi teorian yleistyksistä – uusia sääntöjä luodaan käyttäjän kyselyjen mukaan). Vaihtoehtojen moninaisuudesta voidaan havaita, että mahdollinen tiedonlouhinnan teoria on vielä lapsenkengissään, joten sen tarkempi käsittely menee tämän tutkielman ulkopuolelle.

3 Tutkimuskohde: rakenteiset dokumentit

Tutkielman kohteena on yleisesti tiedonlouhinta rakenteisista dokumenteista. Erityisesti keskitytään XML-dokumenttien indeksointiin, klusterointiin ja hakuun. Handin *et al.* luokittelua käyttäen myös sisältöhaku olisi oma kohteensa, mutta tässä sen katsotaan kuuluvan näkökulmasta riippuen osaksi klusterointia tai tiedonhakua.

3.1 Tiedonhakua vai tiedonlouhintaa?

Luvussa pohditaan tiedonhaun ja tekstitiedonlouhinnan suhdetta ja käydään läpi kummankin tutkimusalueen peruskäsitteitä. Empiirisessä osuudessa kehitettävää sovellusta tarkastellaan kummankin tutkimusalueen näkökulmasta.

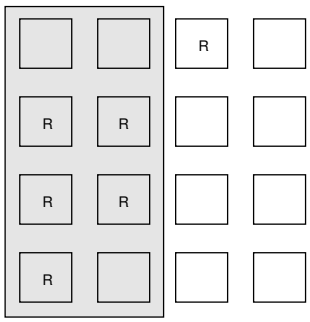
3.1.1 Tiedonhaun peruskäsitteet

Tiedonhaku (*Information Retrieval, IR*) on tietojenkäsittelytieteen osa-alue, joka tutkii relevanttien dokumenttien hakua dokumenttikokoelmasta. Dokumenttien haun tarkoitus on tyydyttää käyttäjän tietotarpeet [6, sivu 444]. Nämä ilmaistaan ideaalisesti luonnollisella kielellä. Hakusanat voivat kuulua kontrolloituun sanastoon. Korostettaessa haettavien dokumenttien tyyppiä voidaan puhua myös tekstitiedonhausta, multi- tai hypermediatiedonhausta tai rakenteisesta hausta. Jatkossa käytetään kuitenkin yleensä lyhyttä nimitystä. Tiedonhakuprosessi on interaktiivinen ja iteratiivinen. Käyttäjä arvioi hakutuloksena saatua dokumenttikokoelmaa ja tarvittaessa tarkentaa kyselyään, kunnes hakutulos täyttää tietotarpeet (tai käyttäjä luovuttaa, jos soveltuvia dokumentteja ei löydy) [26]. Tiedonhaku voidaan tulkita myös loogisena päättelynä. Tällöin tarkoituksena on löytää ne dokumentit, joista käyttäjän kysely on pääteltävissä [63].

Tiedonhaun määritelmässä oleva tieto viittaa *informaatioon*, ei *dataan*. Data on käsitteellisesti informaatiota matalammalla tasolla (katso luku 2.1.2) viitaten esim. säännöllisillä lausekkeilla tai relaatioalgebralla haettavaan tietoon. Käyttäjän täytyy tietää täsmällisesti haettavan tiedon muoto (esim. tietokannan skeema) ja kyselykielen syntaksi (esim. SQL). Haettavan datan relevanssia ei arvioida, vaan hakutulokset ovat syntaktisella tasolla käyt-

täjän kyselyyn sopivat tietueet. Vastakohtana ”datahaulle” tiedonhaun tarkoituksena on palauttaa käyttäjälle tietoa käyttäjän haluamista *aiheista*. Tiedonhakujärjestelmän on tulkittava käyttäjän tekemä kysely, verrattava sitä dokumenteista erotettuun tietoon (indeksitermit) ja esitettävä hakutulokset käyttäjälle järjestettynä arvioidun relevanssin mukaan. Tiedonhakujärjestelmän päätavoite on palauttaa mahdollisimman paljon käyttäjän kyselyyn sopivia relevantteja dokumentteja jättäen pois epärelevantit [6, sivut 1-3]. Käytännössä kysely koostuu useimmiten hakusanoista, jotka voivat olla osa jotain kontrolloitua sanastoa.

Relevanssin käsitteen tarkka määrittely on hankalaa, koska se riippuu käyttäjän yksilöllisistä tietotarpeista ja tiedonhakujärjestelmän kyselykieli asettaa rajoituksia näiden tarpeiden muotoilulle. Käytännössä tiedonhakujärjestelmiä arvioitaessa on tapauskohtaisesti arvioitava, mitkä dokumentit ovat tietyn haun suhteen relevantteja ja mitkä eivät. Tämän jälkeen tutkitaan, kuinka hyvin järjestelmä pystyy löytämään ko. dokumentit. Formaalisti tiedonhakujärjestelmän suorituskykyä arvioidaan tarkkuuden (*precision*) ja saannin (*recall*) avulla, joita käytettäessä oletetaan, että relevantit dokumentit ovat tiedossa (ottamatta kantaa siihen, miksi ne ovat relevantteja). Tarkkuus on palautettujen relevanttien dokumenttien osuus kaikista palautetuista, saanti on palautettujen relevanttien dokumenttien osuus kaikista relevanteista dokumenteista. Formaalisti tämä voidaan määritellä seuraavasti: olk. A relevanttien dokumenttien joukko ja B palautettujen dokumenttien joukko. Tällöin tarkkuudelle saadaan määritelmä $P = \frac{|A \cap B|}{|B|}$, ja saannille $R = \frac{|A \cap B|}{|A|}$ [129, sivut 113-115]. Tiedonhakujärjestelmän suorituskyky on tyypillisesti kompromissi haun ja saannin suhteen. Jos järjestelmällä on korkea tarkkuus, saanti on matalampi ja päinvastoin. Tarkkuutta ja saantia on havainnollistettu kuvassa 3.1.



Kuva 3.1: Esimerkki tarkkuudesta ja saannista. R-merkintä kuvaa relevantteja dokumentteja ($|A| = 6$) ja harmaalla pohjalla oleva alue hakutuloksia ($|B| = 8$). Nyt tarkkuus on $\frac{5}{8} \approx 62.5\%$ ja saanti $\frac{5}{6} \approx 83.3\%$.

Tiedonhakujärjestelmän toiminnallisuuden perustana on tiedonhakumalli (*information retrieval model*), joka määrittää perusoletukset ja ennusteet dokumenttien relevanssille. Tärkeimpiä tiedonhakumalleja (esim. vektorimalli, probabilistiset mallit) käsitellään tekstia-

nalyysia käsittelevässä luvussa 5 yhdessä tekstin samanlaisuusmittojen kanssa. Formaalisti määriteltynä tiedonhakumalli on nelikko $(\mathbf{D}, \mathbf{Q}, \mathcal{F}, R(q,d))$, missä [6, sivut 19-34]

- $\mathbf{D}$ on loogisten näkymien joukko dokumenttikokoelmaan. Loogisia näkymiä voivat olla dokumentin sisältötiedot, looginen rakenne, ulkoasutiedot tai metatiedot [63].
- $\mathbf{Q}$ on loogisten näkymien joukko käyttäjän tietotarpeille. Käytännössä nämä esitetään hakukyselyinä.
- $\mathcal{F}$ on viitekehys dokumenttiesitysten, kyselyjen ja niiden suhteiden mallinnukseen. Viitekehyksen rakenne vaihtelee suuresti tiedonhakumallista riippuen.
- $R(q,d) : \mathbf{Q} \times \mathbf{D} \rightarrow \mathbb{R}$ on täsmäytysfunktio (*matching function*), joka liittää jokaiseen kysely- ja dokumenttiesitysparin reaaliluvun, jota sanotaan relevanssiksi. Täsmäytys määrittää dokumenteille järjestyksen yksittäisten kyselyjen suhteen.

3.1.2 Tekstitiedonlouhinta - uusi tieteenala?

Tekstitiedonlouhinta määriteltiin lyhyesti luvussa 2.2.2 tiedonlouhinnan osa-alueeksi. Määritys vaatii selvennystä, koska tekstidokumentteja on analysoitu lukuisilla eri tieteenaloilla ennen termin yleistymistä 1990-luvun lopulla. Kodratoff [90] huomauttaa, kuinka useita eri tuotteita on markkinoitu *text mining* -leiman alla, vaikka toiminnaltaan tuotteet ovat käsittäneet mm. syntaktista tai semanttista analyysia, tekstitiedonhakua tai tiedon eristämistä – kaikki omia selkeästi määriteltyjä tutkimusalueitaan. Kroeze *et al.* [94] luettelevat peräti 8 erilaista määritelmää tekstinlouhinnalle. Niiden painotukset vaihtelevat tutkijoiden erilaisten taustojen mukaisesti (esim. tiedonhaun ja laskennallisen kielitieteen näkökulmat). Mitä *uutta* tekstitiedonlouhinnassa on?

Tekstitiedonlouhintaa sivuavia, ajallisesti varhaisempia lähialueita ovat laskennallinen kielitiede (*computational linguistics*), luonnollisen kielen käsittely (*Natural Language Processing, NLP*), (teksti)tiedonhaku ja tiedon eristäminen (*Information Extraction, IE*). Laskennallisessa kielitieteessä ja luonnollisen kielen käsittelyssä etsitään tiedonlouhinnan tapaan tilastollisia malleja tekstikokoelmasta, mutta tavoitteena on mallintaa kieltä yleisesti sovellusaluekohtaisten mallien etsinnän sijaan [80]. Tyypillisiä luonnollisen kielen käsittelyn tehtäviä ovat sanojen merkkaus sanaluokkiin ja syntaktinen analyysi [32]. NLP:n katsotaan kuuluvan tietojenkäsittelytieteeseen, kun taas laskennallinen kielitiede on kielitieteen osa-alue. Tiedon eristäminen on prosessi, jonka syötteenä on tiedonhaulla saatu dokumenttikokoelma ja tuloksena analysoitua ja tiivistettyä tietoa dokumenteista yhtenäisessä, määrämuotoisessa kehyksessä (IE:tä onkin luonnehdittu myös automaattiseksi ”lomakkeiden täytöksi”) [32].

Tekstitiedonlouhinta on sovitettavissa KDD-prosessiin (katso luku 2.2.1). Ahosen *et al.* [3] malli käsittää esikäsittelyn (mm. morfologinen analyysi), tiedonlouhintavaiheen (esim. assosiaatiosääntojen muodostaminen) ja jälkikäsittelyn (sääntöjen ryhmittely ja karsinta). Dixon [45] tarkentaa mallia enemmän KDD-prosessia muistuttavaan suuntaan jakamalla esikäsittelyn tiedonhakuun ja tiedon eristämiseen. Tällä tulkinnalla tiedonhaku vastaisi siis KDD-prosessin *valintavaihetta* (vaihe 1) ja tiedon eristäminen vastaisi *muunnosta* (vaihe 3). Tiedonlouhintavaiheen jälkeen on lisätty vielä tulkintavaihe. Malli määrittää helposti ymmärrettävällä tavalla tiedonhaun ja -eristämisen suhteen tekstitiedonlouhintaan, mutta sitä voi myös kritisoida. Näkökulma tiedonhakuun on kapea, koska kyseessä on itsessään interaktiivinen ja iteratiivinen prosessi [26], jota voi ainakin klusterointisovelluksissa hyödyntää myös tulosten tulkinnassa (esim. ”Hae jokaisen klusterin relevantit dokumentit”). Lisäksi tiedonhaun edellytyksenä on indeksointi, joka vaatii yhtä lailla piirteiden valintaa ja dokumenttien esikäsittelyä kuin tiedon eristäminenkin. Tiedonhaun piirteet voivat tosin olla tiedon eristämisessä käytettyjä piirteitä yksinkertaisempia. Tyypillinen tiedonhaun malli on dokumentin esitys sanavektorina, kun taas tiedon eristämisessä piirteet ovat rakenteisia ”lomakkeita”. Olisi siis yleisempää puhua tiedonhaun tai -eristämisen sijaan tilastollisen hahmontunnistuksen terminologiaa käyttäen *piirteiden* valinnasta ja tarvittaessa erottelusta. Piirteiden muodostuksessa voisi käyttää tiedonhaun, -eristämisen tai jopa luonnollisen kielen käsittelyn tekniikoita sovelluksesta ja tekstijoukon muodosta riippuen.

Hearst [80] pitää artikkelissaan tekstitiedonlouhintaa selkeästi erillisenä alueena tiedonhausta, laskennallisesta kielitieteestä ja tekstien luokittelusta. Tiedonhaussa käyttäjän täytyy tietää vähintään avainsana etsimästään tiedosta ja tiedon eristämisessä käyttäjän on määriteltävä tarkasti, millä säännöillä tietoa erotellaan dokumenteista. Tekstitiedonlouhinnan tavoitteena on tiedonlouhinnan tapaan löytää uutta, mahdollisesti odottamatonta ja sovellusalueeseen liittyvää tietoa tekstijoukosta automaattisesti. Laskennallisen kielitieteen Hearst erottaa tekstitiedonlouhinnasta, koska sen löytämä tietämys ei liity sovellusalueeseen, vaan itse kieleen. Tekstien luokittelussa taas luokat ovat ennalta määrättyjä, jolloin uudesta tiedosta ei välttämättä voi puhua. Hearstin kriteerit ”uudelle” tiedolle ovat varsin tiukat ja esim. Kroeze *et al.* [94] ovat kritisoineet mallia ja erottaneet toisistaan *standardin* ja *älykkään* tekstinlouhinnan. Heidän mallinsa mukaan standardi tekstinlouhinta löytää tekstidatasta siihen jo sisältyviä hahmoja ja trendejä, jotka ovat silti käyttäjälle tuntemattomia. Älykäs tekstinlouhinta luo uutta tietämystä ja hyödyntää luonnollisen kielen käsittelyn ohella myös tekoälyä. Lyhennelmien generointi dokumenteista ja löydettyjen hahmojen automaattinen tulkinta perusteluineen ovat esimerkkejä älykkäästä tekstinlouhinnasta.

Tyypillisempi näkökulma tekstitiedonlouhintaan on mm. Dörrellä *et al.* [49]. He pitävät myös luokittelua osana tekstitiedonlouhintaa, mikä tuntuu luontevalta, koska se lasketaan

myös tiedonlouhinnan perusmenetelmiin. Verrattuna perinteiseen data-analyysiin esikäsittelyn, piirteiden valinnan ja erottelun merkitys kasvaa tekstitiedonlouhinnassa. Ihmisen on mahdotonta edes nähdä kerralla, mitä kaikkia piirteitä dokumenttimassa sisältää. Yksinkertainen lähestymistapa on pitää jokaista perusmuotoon palautettua sanaa yhtenä piirteenä, jolloin jo pienelläkin (kymmeniä dokumentteja) datajoukolla potentiaalisten piirteiden määrä nousee helposti yli tuhannen. Tekstitiedonlouhinnalle tyypillistä on siis suuri dimensiomäärä ja voimakkaasti kohinainen data. Kun huomioon otetaan vielä erilaiset datan muunnosmahdollisuudet (esim. jakaumiin perustuvat kielimallit, n-grammit, ulottuvuuksien määrän vähentäminen erilaisilla data-analyysitekniikoilla) tilanne mutkistuu entisestään. Kroezen *et al.* luokittelun perusteella Dörren määritys edustaa ”standardia” tekstinlouhintaa.

Kodratoff [90] kutsuu tekstidokumenttien hakuun ja käsittelyyn liittyvää prosessia nimellä tietämyksen muodostaminen teksteistä (*Knowledge Discovery from Texts, KDT*). Prosessin keskeisin ominaisuus on sen tietämystä luova vaikutus. Tämä erottaa tietämyksen muodostamisen muista tekstianalyysitekniikoista (sen lisäksi, että tekstitiedonlouhinta kokoaa yhteen lukuisia tekniikoita eri tieteenaloilta ja sitä sovelletaan suuriin dokumenttimassoihin). Tietämys on käyttäjälle ymmärrettävässä muodossa olevaa tietoa, joka liittyy suoraan sovellusalueeseen ja todelliseen maailmaan. Tietämyksen muodostamisen taustalla oleva perusperiaate on induktio: mallin yleistäminen esimerkeistä; jopa niin, että uusi tieto voi olla ristiriitaista vanhan tiedon kanssa. KDT-prosessin tietämys on lähellä Kroezen *et al.* ”älykkään” tekstinlouhinnan käsitettä. Termejä *tekstitiedonlouhinta* ja *tietämyksen muodostaminen teksteistä* on käytetty myös synonyymeina [45], mutta tässä tutkielmassa niiden katsotaan olevan samassa suhteessa toisiinsa kuin tiedonlouhinnan ja KDD-prosessin.

Tämän kirjoittajan näkökulmasta tekstitiedonlouhinta on tiedonlouhinnan osa-alue, joka puolestaan on KDD-prosessin vaihe. Kun prosessin kohteena oleva tietojoukko on tekstidataa, nimitys tietämyksen muodostaminen teksteistä on käyttökelpoinen. Empiirisessä osuudessa tietämyksen muodostamisen menetelmänä on käytetty klusterointia, joka on tiedonlouhinnan perusmenetelmä. Toisaalta kehitetyn sovelluksen käyttöliittymässä on hyödynnetty tekstitiedonhakua. Lisäksi klusterointimallin piirteet (tai indeksi) perustuvat laajennettuun vektorimalliin, joka on tiedonhaun hakumalli. Empiiristä osuutta voidaan siis pitää yhtä lailla tiedonlouhinnan kuin tekstitiedonhaun sovelluksena riippuen siitä, mistä näkökulmasta asiaa katsoo. Tästä kaksijakoisuudesta johtuen luvussa 5 kuvattuja menetelmiä kutsutaan yleisemmällä nimellä *tekstitiedon analysointitekniikat*. Terminologiaa tärkeämpää on kuitenkin se, mitä sovellus todella tekee, millainen menetelmän teoreettinen tausta on ja kuinka *käyttökelpoista* dokumenteista saatu tietämys on – olkoon se sitten ennalta tunnettua dataa, ennalta tuntemattomia malleja tai jotain oikeasta maailmasta kertovaa, ns. uutta tietämystä. Tämän määrittelee viime kädessä käyttäjä.

3.2 Rakenteisuuden merkitys

Klusteroinnin ja sisältöhaun kannalta rakenteisen tiedon erityisongelmana on oikean esitystavan valinta [77, sivu 470]. Rakenteisten dokumenttien osalta esitystavat vaihtelevat esimerkiksi sanalistasta monimutkaisiin puuindekseihin. Esitystapa vaikuttaa suoraan klusteroinnissa käytettävään etäisyysmittaan tai tiedonhaussa sovellettavaan täsmäytykseen. Luvussa pohditaan rakenteisuuden asteita ja ilmenemistä eri tietojoukoissa. Rakenteisuuden seurauksia käsitellään lähinnä tiedonhaun kannalta.

3.2.1 Puolirakenteinen tieto ja rakenteiset dokumentit

Eräs näkökulma digitaalisen tiedon luokitteluun on tarkastella sen rakenteisuutta. Voidaan puhua rakenteisesta, puolirakenteisesta tai rakenteettomasta tiedosta (tässä tapauksessa tiedolla tarkoitetaan ennemmin dataa kuin informaatiota). Rakenteeton data on esim. puhtaasta tekstistä koostuvia dokumentteja, bittikarttakuvia tai digitoitu äänisignaali. Rakenteettomalla datallakin voi olla rakennetta (esim. lauserakenteet, kuvassa olevat kohteet), mutta niiden esitysformaatti ei tue rakenteiden koneellista käsittelyä. Tällöin tarvitaan erityistekniikoita, kuten luonnollisen kielen käsittelyä tai kuva-analyysia. Rakenteisella datalla on skeema, joka määrittää tiedon muodon, tietoyksiköiden suhteet toisiinsa ja rajoitteet. Tyypillisiä esimerkkejä rakenteisesta datasta ovat relaatiotietokannat ja esim. CAD-tiedostot. Perinteinen tiedonhaku on yleensä kohdistunut rakenteettomaan tietoon, rakenteista tietoa on haettu skeemaa hyödyntävillä kyselykielillä (esim. SQL).

Rakenteisen ja rakenteettoman tiedon rinnalle on edellisen vuosikymmenen aikana noussut puolirakenteinen tieto (*semistructured data*). Buneman [20] määrittelee puolirakenteisen datan olevan tietoa, joka hyödyntää jotain yleistä rakennemallia (tyypillisesti puu tai graafi), mutta tiedolla ei ole skeemaa – ainakaan tietokantojen vaatimalla tarkkuudella. Lisäksi osa tiedoista (esim. ”kenttien” nimet), jotka yleensä on kuvattu skeemassa, kuvataan puolirakenteisessa datassa itsessään. Tästä johtuen puolirakenteista dataa kutsutaan joskus itsekuvaavaksi, vaikka se ei sitä semanttisessa mielessä ole. Buneman samaistaa puolirakenteisen ja rakenteettoman datan, mutta tämän kirjoittajan mielestä niillä on selvä ero: puolirakenteisella datalla on yleinen rakennemalli, joka puuttuu rakenteettomalta datalta. Wang & Liu [132] lisäävät myös heterogeenisista lähteistä yhdistetyn tiedon puolirakenteisen datan joukkoon.

Tärkein esimerkki puolirakenteisesta tiedosta ovat rakenteiset dokumentit. Nimitys on yleistynyt erityisesti SGML- ja XML-kielten myötä, mutta yhtä lailla sähköpostit, uutisryhmäarkistot, ohjelmakoodit ja LaTeX-tiedostot [132] ovat rakenteisia dokumentteja. Myös WWW:tä voidaan kokonaisuutena pitää valtavana puolirakenteisena tietokantana. Bunemanin mielestä

kaikki rakenteiset dokumentit ovat puolirakenteista tietoa. Perusteluna tähän on, että rakenteisiin dokumentteihin käytetyt skeemat (SGML-dokumenteissa DTD, XML-dokumenteissa myös XML Schema) ovat olennaisesti relaatiomallia joustavampia. Relaatiomallilla sisäkkäisten tietoalkioiden ja erityisesti *monesta moneen* -suhteiden esittäminen on työlästä, puolirakenteisella datalla – skeemalla tai ilman – tämä on helpompaa. Luvun 2.1.1 puhelinnumeroesimerkki voitaisiin esittää yhtenä XML-dokumenttina helposti:

```
<henkilot>
  <henkilo id="h1">
    <nimi>A. A.</nimi>
    <puh type="tyo">435-345345</puh>
  </henkilo>
  <henkilo id="h2">
    <nimi>N. N.</nimi>
    <puh type="koti">11-343255</puh><puh type="tyo">435-365552</puh>
  </henkilo>
</henkilot>
```

Monesta moneen -suhteen esittämiseen tarvittaisiin kaksi dokumenttia (tai erillistä dokumentin alipuuta), mutta rakenne on edelleen relaatiomallia yksinkertaisempi. Jos on odotettavissa, että henkilöitä pitäisi hakea puhelinnumeron mukaan, voitaisiin myös `<puh>`-elementin alle lisätä viitteet henkilöihin. Tämä ei ole kuitenkaan välttämätöntä, kuten relaatiomallissa.

```
<henkilot>                      <puhelinnumerot>
  <henkilo id="h1">               <nro type="tyo" id="p1">435-345345</nro>
    <nimi>A. A.</nimi>            <nro type="koti" id="p2">11-343255</nro>
    <puh xref="p1" />             <nro type="tyo" id="p3">435-365552</nro>
  </henkilo>                    </puhelinnumerot>
  <henkilo id="h2">
    <nimi>N. N.</nimi>
    <puh xref="p2" /><puh xref="p3" />
  </henkilo>
  <henkilo id="h3">
    <nimi>M. N.</nimi>
    <puh xref="p2" />
  </henkilo>
</henkilot>
```

Esityskyvyltään rakenteisissa dokumenteissa ei sinänsä ole mitään uutta tietokantoihin verrattuna. Samaa tyyppiä olevat dokumentit voitaisiin tulkita relaatiotietona siten, että sisäkkäiset elementit ja dokumentit ovat tauluja, muut elementit ja attribuutit ovat kenttiä, linkit suhteita ja tietueet muodostuisivat dokumenttien sisällöstä. Tämä ei kuitenkaan ole luonnollinen tapa rakenteisten dokumenttien kuvaamiseen, koska dokumentin teksti hajaantuu moneen

eri tauluun ja esim. järjestyksen esittäminen vaatii ylimääräisiä apukenttiä. Hajautuksen takia tiettyyn dokumenttiin liittyvien termien hakeminen monimutkaistuu. Relaatiotiedonlouhintaa käytettäessä dokumenttien termit pitäisi lisäksi muuttaa numeerisiksi tai kategorisiksi piirteiksi, koska menetelmät vaativat relaatiotiedon esittämistä loogisina lausekkeina [48]. Relaatiotiedonlouhintaan perustuvia analyysimenetelmiä ei käsitellä tässä tarkemmin. Huomattavaa on, että dokumenttikokoelmasta koostetun indeksin esittäminen tietokannassa on eri asia itse dokumenttien säilyttämiseen nähden. Monet tiedonhakujärjestelmät käyttävätkin tietokantaa indeksin tallentamiseen. Tutkielmassa esitellyt indeksointimenetelmät eivät pääsääntöisesti ota kantaa siihen, miten indeksi on tallennettu.

Koska XML-dokumenttien rakennemalli on puu, sanoja *rakenne* ja *hierarkia* käytetään usein synonyymeina. *Hyperteksti*-termin keksijä Ted Nelson on kritisoinut voimakkaasti tätä käsitystä vedoten siihen, ettei hierarkialla voida esittää luontevasti mielivaltaisia informaatiorakenteita kuten rinnakkaisuutta, kaksisuuntaisia linkkejä tai olion sijaintia eri paikoissa samanaikaisesti [107]. Hierarkkinen dokumenttimalli on relaatiotietokantaa joustavampi, mutta ei tarpeeksi yleinen. Tutkielmassa rakenteisuuden ymmärretään olevan hierarkiaa laajempi käsite, mutta aiheen käsittely rajataan yksinkertaisuuden vuoksi XML-muotoiseen tietoon. Dokumentit voivat sisältää yksisuuntaisia linkkejä ja niillä voi olla ulkopuolista metadataa.

3.2.2 Rakenteiset dokumentit tiedonhaussa

Tiedonhaussa dokumentteja on pidetty tyypillisesti jakamattomina ja rakenteettomina, mistä johtuen perinteisiä tiedonhakumalleja täytyy laajentaa rakenteisia dokumentteja silmälläpitäen. Hypermediamallit ovat käyttökelpoisia rakenteisten dokumenttien käsittelyyn. Tällöin rakenteisen dokumentin osaa pidetään hypertekstijärjestelmän solmuna, joka on linkittynyt muihin saman dokumentin osiin ja mahdollisesti muiden dokumenttien solmuihin. Hypertekstijärjestelmät ovat aiemmin keskittyneet lähinnä dokumenttikokoelman selaukseen linkkien avulla, mutta rakenteesta saadaan myös arvokasta lisätietoa, jota voidaan hyödyntää tiedonhaussa [66]. Yhtä lailla rakenteesta saadaan lisätietoa dokumenttien klusterointiin ja tietämyksen muodostamiseen yleisesti. Chiaramella [26] on analysoinut rakenteisuuden vaikutusta perinteisissä tiedonhakujärjestelmissä (esim. pelkkiin avainsanoihin perustuva WWW-hakukone). Huomioitavia tekijöitä ovat ainakin seuraavat:

- **Kyselyt**. Hakukoneet eivät osaa erotella web-sivuilla olevaa oleellista tietoa ”kohinasta” (mainokset ym.). Lisäksi web-sivu saattaa jakautua moneen loogiseen osioon, joiden aiheet vaihtelevat. Laajemmassa mittakaavassa hakukoneet eivät osaa erottaa, mitkä sivut kuuluvat asiayhteydeltään samaan sivustoon – kaikki linkit ovat samanarvoisia.

- **Selaus**. Jos hypertekstijärjestelmässä on olemassa loogista rakennetta, tämän rakenteen pitäisi tulla käyttäjälle selväksi. WWW-linkit eivät kuvaa tätä rakennetta riittävällä tarkkuudella. Linkeistä ei yleisesti voida päätellä, millaista tietoa sen ”takana” on, viekö linkki toiseen paikkaan nykyisellä sivustolla vai siirrytäänkö kokonaan toiseen asiakokonaisuuteen. Tämä ja linkkien yksisuuntaisuus johtavat siihen, että kokemattomat käyttäjät voivat tuntea ”eksyvänsä hyperavaruuteen”.

- **Hakutulosten järjestys**. Puhtaaseen tekstiin keskittyvät tiedonhaun tekniikat ovat tehottomia rakenteisia dokumentteja haettaessa. Web-hakukoneiden tuhansia dokumentteja sisältävät hakutuloslistat ovat tästä tunnettu esimerkki. Laajassa dokumentissa oleva relevantin osion painoarvo on vähäinen, jos dokumentti indeksoidaan jakamattomana ja sanoille annetaan sama paino rakenneosista riippumatta (esim. otsikot). Hakutulosten suuren määrän lisäksi myös niiden järjestys saattaa siis olla vääristynyt.

- **Hakutulosten esittäminen**. Hakukoneiden tulisi viitata tulosdokumenttien relevantiksi arvioituun osaan eikä koko dokumenttiin. Tämä vähentäisi käyttäjän työmäärää hakutulosten arvioinnissa.

Rakenteisten dokumenttien ja puhtaaseen tekstiin erikoistuneiden hakukoneiden epäyhtenäisyys yhdistettynä WWW:n valtavaan kokoon ja korkeintaan muutaman sanan mittaisiin hakukyselyihin takaavat hakutuloslistojen epäkäytännöllisyyden vielä pitkälle tulevaisuuteen. Myönnettäköön kuitenkin, että uudemmat hakukoneet ovat korjanneet tilannetta jonkin verran. Esim. linkkitiedon (PageRank-algoritmi, katso luku 5.2.2) käyttö Googlessa tai hakutulosten esittäminen klustereina Vivísimo-yhtiön Clusty[1]-metahakukoneessa mahdollistavat täysin käyttökelpoisen haun, vaikka niiden kyselykielet eivät dokumenttien rakennetta tuekaan. Hakusanojen näyttäminen kontekstissaan on tehokas tapa hakutulosten esittämiseen, vaikka dokumentin osan relevanssia ei muuten arvioitaisi. Tästä huolimatta käyttäjän täytyy tietää melko tarkasti aiheeseen liittyvät oikeat avainsanat.

Rakenteisia dokumentteja varten on kehitetty runsaasti erilaisia kyselykieliä ja indeksirakenteita. Tyypilliset lähestymistavat ovat SQL:n tyyppisen kielen laajentaminen tukemaan puolirakenteista tietomallia ja kokotekstihakua, tai määrittelemällä kieli suoraan puolirakenteisen tietomallin pohjalta [20]. Kyselyt voivat sisältää rajoitteita dokumentin sisällön, rakenteen tai linkkitiedon suhteen [66]. Jos käytettävissä on metatietoa, sitä pitäisi pystyä myös hyödyntämään. Haun tuloksena saatavat tietoyksiköt (*information units, dokumenttikomponentit*) voivat olla kokonaisia dokumentteja, dokumenttifragmentteja tai eri dokumenteista saatu dokumenttifragmenttien yhdistelmä [101] (viimeistä vaihtoehtoa voidaan verrata tilanteeseen, jossa haetaan kenttiä usean taulun liitoksesta relaatiotietokannassa). Dokumenttifrag-

[1] http://www.clusty.com

menttien haku on mielekästä, jos käyttäjä tietää etukäteen, millaista tietoa halutaan hakea (esim. ”Hae kaikkien dokumenttien otsikot ja ensimmäiset tekstikappaleet”). Laajat dokumentit voivat sisältää tietoa useista aiheista (esim. monesta osasta koostuva kirja voi olla yksi looginen dokumentti), jolloin hyvän hakualgoritmin pitäisi pystyä myös päättelemään dokumentin relevantit osat ja näyttämään ne käyttäjälle Chiaramellan huomioiden mukaisesti. Erityisen hankalaksi tilanne menee, jos haettava dokumenttikokoelma on heterogeeninen sisältäen laajuudeltaan ja tyypiltään erilaisia dokumentteja – kuten WWW.

Tutkielmassa toteutetussa hakumallissa (katso luku 6.2) oletetaan yksinkertaisuuden vuoksi, että dokumenttikokoelma on homogeeninen: dokumentit ovat rakenteeltaan samaan skeemaan kuuluvia ja laajuudeltaan samaa kertaluokkaa. Dokumentin indeksointivaiheessa pyritään etsimään sovellusaluekohtaisen konfiguroinnin avulla dokumenttien relevantit ja kuvaavimmat osat: esim. eri elementtityypit painotetaan eri tavalla. Hakuvaiheessa käyttäjälle palautetaan kuitenkin vain kokonaisia dokumentteja eikä hakua voida osoittaa dokumentin mielivaltaiseen elementtiin. Tämän yksinkertaistuksen vuoksi dokumenttifragmentteja hakevien kyselykielten tarkempi käsittely sivuutetaan (kiinnostunut lukija voi tutustua esim. Luk’n *et al.* kirjallisuuskatsaukseen [101]). Muilta osin rakenteisia indeksi- ja hakumenetelmiä käsitellään luvussa 5.3.

3.3 XML:n erityispiirteet

XML (Extensible Markup Language) on W3C:n kehittämä metakieli rakenteisten dokumenttien esittämiseen. XML on kehitetty Charles Goldfarbin kehittämän ja ISO:n standardoiman[2] SGML-kielen (Standard Generalized Markup Language) pohjalta siten, että XML on oleellisesti SGML:n yksinkertaistettu osajoukko. XML:n alkuperäiset suunnittelutavoitteet olivat seuraavat[3] [18]:

1. XML:n pitää olla suoraviivaisesti käytettävissä Internetissä.
2. XML:n täytyy tukea laajaa ohjelmistosovellusten joukkoa.
3. XML:n täytyy olla yhteensopiva SGML:n kanssa.
4. Täytyy olla helppo kirjoittaa ohjelmia, jotka prosessoivat XML-dokumentteja.
5. Vaihtoehtoisten ominaisuuksien määrä täytyy pitää XML:ssä minimissään.

[2] ISO 8879:1986

[3] Suomennos Airi Salmisen XML-kurssin luentomateriaalista (Jyväskylän yliopisto, Tietojenkäsittelytieteiden laitos, 2004)

6. XML-dokumenttien tulisi olla ihmisen luettavissa ja kohtuullisen selviä.
7. XML:n suunnittelun täytyy tapahtua nopeasti.
8. XML:n suunnittelun täytyy olla formaalia ja tiivistä.
9. XML-dokumentteja täytyy olla helppo luoda.
10. XML-esitystavassa esitystavan tiiviys ei ole tärkeää.

XML-kielen voidaan sanoa ylittäneen tavoitteensa. Kieli on epäilemättä kaikista rakenteisista dokumenttiformaateista laajimmalle levinnyt. Syy tähän lienee se, ettei XML-määritys ole sitoutunut mihinkään tiettyyn sovellusalueeseen, vaan tarjoaa puolirakenteisen puumaisen rakennemallin, jota joustavuudessaan voidaan soveltaa lähes minkä tahansa tiedon kuvaamiseen. Varsinaisen XML-määrityksen lisäksi kieliperheeseen kuuluu joukko ns. liitännäiskieliä, joista tärkeimmät ovat XML-pohjaiset, tiettyä sovellusaluetta varten suunnitellut kielet eli XML-sovellukset (*XML Applications*). XML-pohjaisten kielten välillä voidaan tehdä karkea jako data- tai dokumenttikeskeisiin kieliin [17, sivut 3-13]. Salminen [114] jakaa muut liitännäiskielet XML-laajennuskieliin ja XML-manipulointikieliin. Laajennuskielillä voidaan määritellä lisäsääntöjä muita XML-pohjaisia kieliä varten. Tällaisia ovat esimerkiksi XML-linkit [43], nimiavaruudet[4] ja skeemakieli XML Schema[5]. Manipulointikieliä käytetään muiden XML-dokumenttien muuntamiseen. Tärkeimpiä manipulointikieliä ovat XSLT-muunnoskieli[6] ja XQuery-kyselykieli [14]. Uusien liitännäiskielten ja erityisesti laajennuskielien myötä XML-kieliperheestä on tullut monimutkaisuudessa täysin SGML-standardiin verrattava kokonaisuus.

Datakeskeiset kielet on tarkoitettu sovellusten väliseen kommunikointiin, tietorakenteiden ja esim. relaatiotietokannoissa olevan tiedon esittämiseen. Esimerkkejä datakeskeisistä XML-kielistä ovat W3C:n kehittämä, web-sovelluspalvelujen perustana oleva SOAP[7] (*Simple Object Access Protocol*), graafien kuvauskieli GXL[8] (*Graph eXchange Language*) ja yritysten sähköiseen tiedonvaihtoon kehitetty ebXML[9]. Datakeskeisten formaattien tarkempi käsittely sivuutetaan, koska ne ovat muodoltaan lähempänä tietokantoja tai viestiprotokollakehyksiä kuin tekstidokumentteja. Siten tavanomaisen tekstitiedonhaun soveltaminen niihin ei ole mielekästä, ellei kohdedokumenteissa ole merkittävää määrää tekstiä. Parempi ratkaisu on käyttää XQuery-tyyppistä rakenteista kyselykieltä, joka tukee hakua yksittäisten elementtien tarkkuudella.

[4] http://www.w3.org/TR/REC-xml-names/
[5] http://www.w3.org/XML/Schema
[6] http://www.w3.org/TR/xslt
[7] http://www.w3.org/TR/soap12-part0/
[8] http://www.gupro.de/GXL/
[9] http://www.ebxml.org/

Dokumenttikeskeiset kielet on tarkoitettu perinteiselle SGML:n sovellusalueelle, elektroniseen julkaisemiseen ja yleensäkin rakenteisten *dokumenttien* käsittelyyn. SGML:n ja sittemmin XML:n suuri lupaus on ollut monikanavajulkaisun mahdollistaminen: sisältö kirjoitetaan kertaalleen rakenteiseen dokumenttiin, josta tyylitiedostojen avulla muunnetaan automaattisesti esitykset eri formaatteihin, kuten paperijulkaisuun (PDF), WWW-ympäristöön tai esim. CD-ROM:ille (käytännössä prosessi tuskin on näin suoraviivainen). Dokumenttikeskeisiä kieliä ovat XHTML[10], romaanitekstien merkkaamiseen suunniteltu TEI[11] (*Text Encoding Initiative*) ja teknisessä dokumentoinnissa käytetty DocBook[12].

XML on valittu empiirisessä osuudessa käytettäväksi formaatiksi. Perustelut tähän ovat sekä käytännölliset että periaatteelliset. XML on levinnyt laajalle; työkaluja, ohjelmakirjastoja ja tietojoukkoja (vähintäänkin WWW-sivustoja) on helposti saatavilla. Formaatin avulla saadaan edustava otos rakenteisista dokumenteista, koska mikä tahansa rakenteinen tai puolirakenteinen tieto on muunnettavissa XML-muotoon. Periaatteellinen syy on löytää lisäarvoa formaatin käytöstä (ja tietovarastoinnista yleensäkin): 1990-luvun lopulla XML-kieltä markkinoitiin ihmelääkkeeksi lähes mihin tahansa organisaation tietohallinnolliseen ongelmaan. Uusi dokumenttiformaatti ei kuitenkaan sellaisenaan ole perusteltu syy korvata aiemmin toimivaa järjestelmää uudella. Seuraava lause puhuu puolestaan:

> All my data is in XML. Now what?[13]

Eräs mahdollisuus XML-muotoisen tiedon hyödyntämiseen ja jalostamiseen on tiedonlouhinta. WWW-dokumenttien osalta tämä näkyy web-louhinnan yleistymisenä ja esim. Han & Kamber [76, sivut 435-441] pitävät web-louhintaa ratkaisuna WWW:n nykyiseen kaoottisuuteen. Tutkielmassa on erityisesti tutkittu klusteroinnin ja tiedonhaun yhdistämisen tarjoamia mahdollisuuksia samaa skeemaa noudattavien dokumenttikokoelmien hahmottamisessa. On kuitenkin korostettava, että lähes kaikki tutkielmassa esitettävät algoritmit ja menetelmät ovat sellaisenaan sovellettavissa myös muuhun (puoli)rakenteiseen tietoon ja esimerkiksi tekstidokumentteihin. Mitään pakottavaa tarvetta rajoittua XML-kieleen ei siis ole.

3.3.1 Dokumentin rakenne ja sisältö

XML-muotoisen tiedon abstrakti tietomalli kuvataan XML InfoSet -määrityksessä [31]. Dokumentit koostuvat sisäkkäisistä ja peräkkäisistä elementeistä muodostaen puolirakenteisel-

[10] `http://www.w3.org/MarkUp/`

[11] `http://www.tei-c.org/`

[12] `http://docbook.org/`

[13] Muokattu J. Fauseyn ja K. Shaferin artikkelin otsikosta All my data is in SGML. Now what? (*JASIS* 48(7):638–643, 1997. Sisällöltään artikkeli käsitteli lähinnä SGML-tyylitiedostoja, mutta otsikko kuvaa laajemminkin tilannetta, missä XML-merkattu data odottaa hyötykäyttöä.)

le tiedolle tyypillisen puumaisen rakenteen, jonka elementit ovat solmuja. Solmu voi sisältää alisolmujen lisäksi tekstiä ja/tai attribuutteja. Elementit muodostavat dokumentin loogisen rakenteen, attribuuteilla voidaan merkitä yksittäisiin elementteihin liittyvää metatietoa. Lisäksi dokumentilla on entiteeteistä (*entity*) muodostuva fyysinen rakenne. Entiteetit voivat olla mitä tahansa tietoyksiköitä, kuten yksittäisiä merkkejä, dokumenttifragmentteja tai viittauksia tiedostoihin. Esimerkiksi elementtitunnisteita (*tageja*) varten varattuihin <- ja >-merkkeihin voidaan viitata `<` ja `>` -entiteeteillä. Tiedosto voidaan määritellä entiteetiksi DTD:ssä komennolla `<!ENTITY included SYSTEM "includefile.xml">`, jolloin sen sisältöön viitataan entiteetillä `&included`. Jos tiedosto on binäärinen, XML-prosessori on vastuussa sen oikeasta käsittelystä. Muussa tapauksessa tiedoston sisältö tai entiteetiksi määritelty merkkijono sijoitetaan osaksi dokumenttia ja jäsennetään. Entiteetit eivät siis vaikuta dokumentin loogiseen rakenteeseen. Entiteetit mahdollistavat alkeellisen dokumentin tai DTD:n osien uudelleenkäytön ohjelmointikielten makrojen tapaan (joskaan entiteettejä ei voi parametrisoida). [17, sivut 3-13]

XML-dokumentteja voidaan käsitellä ilman erillistä skeemaa käyttämällä jäsentimen tarjoamaa rajapintaa tai jäsennyspuuta. Skeemattoman XML-dokumentin täytyy olla syntaktisesti oikea (esim. vain yksi juurielementti, jokaisen elementin alkutunnistetta vastaa lopputunniste, attribuutit on erotettu lainausmerkeillä, elementtitunnisteet eivät mene ”ristiin”) ja samojen sääntöjen täytyy olla voimassa dokumentissa viitatuilla entiteeteillä. Tällöin dokumentin sanotaan olevan hyvinmuodostettu (*well-formed*) [18]. Hyvinmuodostetut dokumentit täyttävät täysin Bunemanin puolirakenteisen datan määritelmän. Täsmällisen käsittelyn kannalta on kuitenkin toivottavampaa, että dokumenteilla on jokin skeema ja jatkossa näin oletetaan, ellei toisin ole mainittu (myös skeemaa noudattavat XML-dokumentit lasketaan tässä puolirakenteiseksi dataksi). Tärkeimmät kielet skeeman kuvaamiseen ovat DTD ja XML Schema, mutta muitakin on.

DTD (*Document Type Definition*) on mekanismi formaaliin rakennesääntöjen määrittelyyn. DTD määrittelee käytettävät elementit ja niiden suhteet. Lisäksi DTD:ssä määritellään elementteihin liittyvät attribuutit. Attribuuteille voidaan antaa yksinkertaisia rajoitteita: keskeisimpiä ovat omat luetellut tyypit, vapaa teksti tai viite toisaalla dokumentissa olevaan tunnisteeseen [17, sivut 47-69]. DTD-määritykset ovat aiemmin olleet käytössä SGML-kielessä, mutta kieltä on jonkin verran yksinkertaistettu XML:aa varten (esim. elementin lopputunniste on aina pakollinen). Dokumenttia, joka on hyvinmuodostettu ja joka lisäksi noudattaa rakennesääntöjä, kutsutaan validiksi [18]. Luvun 3.2 puhelinnumeroesimerkin dokumenttityyppi voitaisiin määritellä DTD:llä seuraavasti (oletetaan, että tiedot ovat yhdessä tiedostossa):

```
<!-- elementit                                                    -->
<!ELEMENT puhelinluettelo (henkilot,puhelinnumerot)>
<!ELEMENT henkilot        (henkilo+)> <!-- +: 1 tai useampi elementti -->
<!ELEMENT puhelinnumerot  (nro+)>
<!ELEMENT henkilo         (nimi,puh+)>
<!ELEMENT puh              EMPTY>     <!-- ei alielementtejä          -->
<!ELEMENT nimi            (#PCDATA)>  <!-- #PCDATA: jäsennetty teksti -->
<!ELEMENT nro             (#PCDATA)>

<!-- attribuutit                                                  -->
<!ATTLIST henkilo id   ID    #REQUIRED>   <!-- yksilöllinen tunniste  -->
<!ATTLIST puh     xref IDREF #REQUIRED>   <!-- viittaus ID-arvoon     -->
<!ATTLIST nro     id   ID    #REQUIRED
                  type (koti|tyo|gsm) #REQUIRED> <!-- lueteltu tyyppi -->
```

Uudempi tapa rakennesääntöjen määrittelyyn on XML Schema -kieli, jossa säännöt voidaan määritellä XML-kielisellä syntaksilla. XML Schema soveltuu erityisesti datakeskeisiin dokumentteihin, koska siinä voidaan määritellä omia tietotyyppejä ja dokumentin sisällölle voidaan määrätä rajoitteita huomattavasti DTD:tä tarkemmalla tasolla [56]. Esimerkiksi puhelinnumeroesimerkin `<puh>`-elementin `xref`-viitekenttä voi DTD-määrityksen mukaan viitata joko henkilöön tai puhelinnumeroon, vaikka viittaukset tulisi sallia vain puhelinnumeroon. DTD:llä ei voida myöskään määritellä puhelinnumerolle tarkkoja tietotyyppejä. XML Schemalla voitaisiin määritellä ainakin erilliset tietotyypit ja rajoitteet niiden ID-kenttien muodon suhteen (esim. henkilön tunnisteen täytyy alkaa `h`-merkillä), joten suositus soveltuu datakeskeisille dokumenteille DTD:tä paremmin. Tekstidokumenttien kuvaamiseen DTD tarjoaa kuitenkin riittävän tarkkuuden. Toistaiseksi DTD-määritykset ovat myös huomattavasti XML Schema -määrityksiä yleisempiä. Empiirisessä osuudessa rakennemäärityksissä on hyödynnetty vain DTD:tä.

Dokumentin rakennetta analysoimalla voidaan verrata eri dokumenttien rakenteellista samanlaisuutta. Tiettyjä rakennehahmoja voidaan tunnistaa jopa DTD:stä riippumatta, elementtien yleisten rakenteellisten suhteiden mukaisesti. Arkkitehtuuriset muodot (*architectural forms*) ovat malleja, joita voidaan sovittaa eri dokumenttityyppeihin [17, sivut 80-83]. Tyypillisiä arkkitehtuurisilla muodoilla tunnistettavissa olevia rakenteita ovat listat, lomakkeet ja taulukot. Empiirisessä osuudessa on hyödynnetty yksinkertaisempaa lähestymistapaa, jossa sovellus konfiguroidaan jokaiselle käytettävälle dokumenttityypille erikseen. Konfiguroinnin osana käytetään XPath-kieltä [28] muistuttavia sääntöjä, joilla voidaan ilmaista mm. dokumentin otsikkoelementit, linkkiattribuutit ja elementit, jotka jätetään kokonaan huomiotta. Muilta osin dokumentin sisältöteksti indeksoidaan rakenteettomana. Lisätietoa konfiguroinnista on luvussa 6.3.1.

3.3.2 Hyperlinkit

Ehkä tärkein syy WWW:n suosiolle on mahdollisuus linkittää dokumentteja toisiinsa. Linkit ovat yleistetty ja yksinkertaistettu vastine tieteellisten artikkelien kirjallisuusviitteille. HTML:ssä ja XML-pohjaisissa kielissä on useita tapoja viittauksen ilmaisuun. Dokumentin sisällä voidaan käyttää ID- ja IDREF-tyyppisiä attribuutteja, mutta dokumenttien välillä tarvitaan yhteisesti sovittu nimeämiskäytäntö, joka WWW-ympäristössä on URI (*Universal Resource Identifier*). Keskitetyissä hypertekstijärjestelmissä linkit ovat yleensä kaksisuuntaisia, mikä takaa niiden yhtenäisyyden vaikka jokin dokumentti siirretään tai poistetaan. Tämä yhtenäisyyden vaatimus jätettiin tarkoituksella pois WWW:stä, mikä on sallinut dokumenttien ja linkkien määrän rajoittamattoman kasvamisen ja hajautetun arkkitehtuurin [11].

W3C:n suosittelema tapa linkkien ilmaisemiseen XML-dokumenteista on Xlink-kieli [43]. Jos linkin kohteena on myös XML-dokumentti, siihen voidaan osoittaa käyttämällä XPointer-kielistä osoitusta URI:n osana [42]. XLink on WWW:n hyperlinkkien yleistys, joka määrittelee joukon vakiotavalla nimettyjä attribuutteja omaan XML-nimiavaruuteensa (esim. viittausosoitetta kuvataan attribuutilla `xlink:href`). Näitä attribuutteja voidaan käyttää uusissa XML-pohjaisissa kielissä ja XLink-kieltä ymmärtävät ohjelmat voivat tunnistaa linkit muista kielen elementeistä riippumatta. Linkkiin voidaan määritellä kohteen ja kuvauksen lisäksi omia rooleja ja näyttötapaan liittyviä tietoja. On myös mahdollista määritellä ns. laajennettuja linkkejä, jotka sallivat usean eri tietokohteen linkityksen yhteen samanaikaisesti. XPointer on yleistys HTML-dokumenttien osatunnisteille. Se mahdollistaa sijaintiin perustuvan osoituksen yksittäisen elementin tai attribuutin tarkkuudella. Menettely on joustava, koska tällöin viitattavia elementtejä ei tarvitse merkitä dokumenttiin ID-attribuuteilla [17, sivut 143-154]. On myös mahdollista viitata pelkkien rakennetietojen perusteella, jolloin ID-arvoista ei tarvitse välttämättä edes tietää. Lisäksi XPointer-osoitus voi osoittaa samanaikaisesti moneen kohtaan dokumenttia.

XLink- ja XPointer -suositukset ovat monipuolisia ja ilmaisuvoimaisia, mutta valitettavasti vain harvat sovellukset tukevat niitä. XLinkin ongelma on, että monissa XML-pohjaisissa kielissä määritellään oma linkkimekanismi, joka ei ole yhteensopiva XLink-määrityksen kanssa (joko eri nimiavaruuden tai erinimisen attribuutin takia). Näin on jopa muutamien W3C:n suositusten osalta, tärkeimpänä XHTML. Empiirisessä osuudessa linkkejä käsitellään arkkitehtuuristen muotojen tapaan: konfigurointitiedostossa määritellään, minkä niminen elementti tai attribuutti edustaa linkkitietoa, mutta muuten sovellus ei ota kantaa linkkikieleen. Tämä mahdollistaa sekä yksinkertaisten XLink-linkkien että XHTML-linkkien käsittelyn. XPointer-kieltä ei tueta, vaan rajoitutaan HTML-tyylisiin URI-viittauksiin mahdollisine osatunnisteineen.

Linkkejä analysoimalla voidaan arvioida tietyn dokumentin luotettavuutta tai linkitettyjen dokumenttien samanlaisuutta [106]. Lisäksi uudemmat web-hakukoneet (esim. Google) hyödyntävät linkkejä arvioidessaan hakutulosten relevanssia [19]. Esimerkiksi, jos useat sivut viittaavat samoille sivustoille, viittaavat sivut ovat todennäköisesti aiheeltaan samankaltaisia. Toisaalta sivusto, johon monet samanaiheiset sivut viittaavat, sisältää todennäköisesti merkittävää tietoa viittaavien sivujen aiheesta. Linkkitiedosta on siis hyötyä sekä tiedonhaussa että dokumenttien klusteroinnissa. Linkkien analysointia käsitellään tarkemmin luvussa 5.2.

3.3.3 Metatieto

Metatieto (*metadata*) on tietoa tiedosta. Sen tarkka määrittely riippuu määrittelijästä: kirjastoalalla metatieto merkitsee indeksejä, lyhennelmiä, luokitussääntöjä ja yleensäkin tietokannoissa olevaa, kirjastojen kokoelmia kuvaavaa tietoa. Toisaalta, tietokantojen kannalta skeemat ovat metatietoa [50, sivut 4-5]. HTML-dokumenteissa metatieto on `<meta>`-elementeissä (joiden väärinkäytön johdosta eräät hakukoneet tosin jättävät metatiedot kokonaan indeksoimatta) [22, 11-12], semanttisessa webissä metatieto on RDF-kielisiä kuvauksia mistä tahansa URI-osoitteella viitattavissa olevassa kohteesta [104]. Klassisen ja äärimmäisen yleisen määritelmän antaa Gilliland-Swetland [68], jonka mukaan metatieto on *kokonaismäärä kaikesta, mitä informaatio-objektista voidaan sanoa millä tahansa koostetasolla*. Informaatio-objekti tarkoittaa tässä mitä tahansa, mikä on ihmisen tai järjestelmän viitattavissa ja käsiteltävissä omana yksikkönään (jos viittaus voidaan tehdä URI-tunnisteella, tämä vastaa RDF-terminologiassa *resurssia*). Metatieto sijaitsee yleensä kuvaamansa kohteen ulkopuolella, mutta toisaalta jopa XML-dokumenttien attribuutit voidaan tulkita metatietona. Metatieto on käsitteenä huomattavasti monimuotoisempi kuin intuitiivinen määritelmä antaa ymmärtää – ei vähiten useiden sovellusalueiden takia. Vaihtoehtoisten määritelmien lisäksi erilaisia luokitteluja on runsaasti. Metatietoa on analysoinut dokumenttien hallinnan näkökulmasta mm. Lyytikäinen [102, sivut 13-22].

Metatiedon automaattinen hyödyntäminen edellyttää tietoa metatietorakenteiden merkityksistä. Yleisessä käytössä oleva standardi on esim. Dublin Core[14], joka tarjoaa 15 määrämuotoista kenttää ”minkä tahansa” dokumentin kuvaamiseen. Dublin Coren lisäksi metatiedon määrittelyyn on saatavilla runsaasti[15] sovellusaluekohtaisia rakennekuvauksia, joita kutsutaan ontologioiksi. Gruber [72] määrittelee ontologian olevan *formaali, eksplisiittinen määrittely yhteisestä käsitteistöstä* tietämyksen kuvaamiseen. Yksinkertaisimmillaan tämä tarkoittaa sanalistaa, yleensä käsite- tai tyyppihierarkiaa; kehittyneimmillään kyse on loogises-

[14] http://dublincore.org/

[15] Esim. DAML:n ylläpitämä ontologiakirjasto sisälsi 27.10.2004 282 ontologiaa. Ks. http://www.daml.org/ontologies/

ta sovellusalueen teoriasta, jossa käsitteet – metatietotyypit suhteineen, yksittäiset kentät ja niiden sisällöt – määritellään formaalisti. Tämä mahdollistaa koneellisen päättelyn metatietokuvausten perusteella, mikä edelleen on edellytys semanttiselle webille [12]. Intuitiivisesti käsitettynä ontologiat ovat ”skeemoja metatiedolle”.

WWW-ympäristössä tärkein yleiskäyttöinen tapa metatiedon kuvaamiseen on W3C:n määrittelemä RDF-kieli (*Resource Description Framework*) [104]. RDF mahdollistaa HTML-kielen `<meta>`-elementtejä monipuolisemman tiedon liittämisen resursseihin; jopa niin, että tiedon kuvaamisessa käytettävät käsitteet voivat olla eri lähteistä. RDF-kuvaukset ovat kolmikkoja $(Resurssi, Ominaisuus, Arvo)$, joista jokainen voi edelleen olla resurssi. Kielitieteen käsittein metakuvaukset ovat lauseita, joissa resurssi on subjekti, ominaisuus predikaatti ja arvo objekti. Predikaattilogiikassa kolmikko vastaa kaksipaikkaista predikaattia. Resurssikuvausten joukko voidaan tulkita myös suunnattuna, tyypitettynä graafina (poiketen XML:stä, jonka tietomalli on oleellisesti puumainen). RDF-kielessä on oma linkitystoimintonsa, jolla kuvaukset liitetään resursseihin. Kielessä on vakiona yksinkertaisia säiliöluokkia, kuten jonot ja laukut. Käyttäjät voivat myös määritellä omia rakenteisia tietotyyppejä (yksinkertaisia ontologioita) RDFS-skeemakielen avulla. Varsinaisten resurssikuvausten lisäksi RDF tukee konkretisointia (reification), väitteiden tekemistä muista RDF-väitteistä. Kieltä varten on määritelty kaksi XML-syntaksia (tavallinen ja lyhennetty) sekä erityinen N-triples-notaatio, joka soveltuu XML-syntaksia paremmin ihmisen luettavaksi. Merkittävää RDF-kielessä ei ole kuitenkaan syntaksi, vaan monimuotoiset metakuvaukset mahdollistava tietomalli. [104]

Rakenteisuuden ja hyperlinkkien lisäksi myös metatietoa on mahdollista käyttää apuna tiedonhaussa. Shah *et al.* [121] esittävät semanttisen- ja kokotekstihaun yhdistämistä siten, että dokumenttitekstistä eristetyt käsitteet muunnetaan RDF-kuvauksiksi ja yhdistetään saatavilla olevan metatietoon. Tuloksena saatavasta indeksistä olisi mahdollista hakea indeksitermien ohella metatiedossa olevia semanttisia suhteita. Empiirisessä osuudessa on rajoituttu tapaukseen, jossa metatieto esitetään joukkona nimettyjä tekstikenttiä (katso luku 5.5), mutta kentillä ei ole sisäistä rakennetta tai formaalisti määriteltyjä suhteita. Tämä mahdollistaa esim. Dublin Core -määritysten mukaisen, RDF-kielellä tai `<meta>`-elementeillä merkityn metatiedon käytön, mutta ei rakenteisia tyyppejä tai käsitehierarkiaa sisältävien ontologioiden hyödyntämistä eikä automaattista päättelyä.

3.4 Dokumenttien esikäsittely

Riippumatta käytetystä haku- tai klusterointimenetelmästä edellytys toimivalle tekstianalyysijärjestelmälle on oleellisen tiedon erottelu dokumentista indeksiä varten (tarkastellaan tässä yksinkertaista hakumallia, jossa dokumentit esitetään indeksitermien kokoelmana). Dokumenttien esikäsittely on prosessi, jossa kontrolloidaan indeksitermien määrää ja laatua. Tällöin indeksin koko pienenee ja haun tarkkuus paranee. Esikäsittelyn taustalla on intuitiivisesti järkevä oletus, että kaikki sanat eivät ole merkityssisältönsä kannalta samanarvoisia. Dokumentin esitys kaikkien sanojensa samanarvoisena joukkona on yleisesti käytetty, mutta epätarkka esitys, jota esim. web-hakukoneet käyttävät nopeus- ja helppokäyttöisyyssyistä. Kokotekstihaku voi parantaa saantia, mutta haun tarkkuus heikkenee. Peruskäyttäjälle saattaa silti olla helpompaa käyttää kokotekstihakua kontrolloidun sanaston sijaan, jos sanasto ei ole käyttäjälle tuttu. Baeza-Yates & Ribeiro-Neto [6, sivut 163-173] jakavat dokumenttien esikäsittelyn seuraaviin vaiheisiin:

- **Leksikaalinen analyysi** on prosessi, jossa merkkivirta (dokumenttiteksti) muunnetaan sanavirraksi (termikandidaatit). Välilyöntien lisäksi analyysissa on otettava huomioon numerot, tavutus, välimerkit sekä mahdollinen isojen ja pienten kirjainten käsittely. Leksikaalinen analyysi on suoraviivainen operaatio, jossa dokumentteihin sovelletaan yhtenäistä sanojen erottelukäytäntöä. Poikkeukset voidaan ilmaista esim. säännöllisinä lausekkeina.

- **Sulkusanojen poisto**. Jopa 80% dokumenteissa olevista sanoista on niin yleisiä, että ne ovat käyttökelvottomia tiedonhaun kannalta. Näitä sulkusanoja (*stopwords*) ovat kielestä riippuen esim. artikkelit, prepositiot ja konjunktiot. Ne suodatetaan yleensä pois termikandidaattien joukosta. Tämä pienentää indeksin kokoa ja tehostaa hakua. Sulkusanojen poisto saattaa heikentää saantia, jos hakulause koostuu pääasiassa sulkusanoista.

- **Stemmaus** (*stemming, vartalointi*) on kieliriippuvainen prosessi, jossa sanojen taivutusmuodot pyritään palauttamaan perusmuotoon. Ilman stemmausta indeksiin voi päätyä monikko- ja eri sijamuotojen takia monta muotoa samasta sanasta. Stemmaus pienentää indeksin kokoa ja tehostaa hakua, joskin kirjallisuudessa on ristiriitaisia tuloksia stemmauksen toimivuudesta käytännön hakutehtävissä [6, sivu 168]. Jälkiliitteen poisto on yksinkertaisin ja tärkein stemmausmenetelmä, jota mm. suosittu Porterin algoritmi käyttää.

- **Indeksitermien valinnalla** pyritään löytämään kandidaattitermien joukosta merkityksellisimmät. Jos indeksoitavien sanojen sanaluokka pystytään tunnistamaan, kannattaa

keskittyä substantiiveihin, koska ne sisältävät eniten tietoa dokumentin aiheesta. Peräkkäisiä substantiiveja voidaan ryhmitellä, koska monet käsitteet ilmaistaan sanojen yhdistelmällä.

- **Tesaurus** (asiasanasto) on rakenteinen esitystapa tietyn sovellusalueen käsitteistölle. Se on kontrolloitu sanasto, joka sisältää tietoa termien välisistä suhteista. Termit ovat yleensä (tarvittaessa adjektiiveilla tarkennettuja) substantiiveja. Manuaalisesti määritellyssä rakenteessa termeillä voi olla myös luonnollisella kielellä annetut määrittelyt. Tesaurus tarjoaa standardisanaston indeksointiin ja hakuun, auttaa käyttäjää löytämään kyselyyn oikeat termit ja mahdollistaa haun muokkaamisen tarkemmaksi tai yleisemmäksi. Tesaurusten ongelma on joustamattomuus dynaamisen sisällön tai lyhyellä aikavälillä muuttuvien termien suhteen. Lisäksi kokemattomalla käyttäjällä saattaa olla vaikeuksia löytää haun kannalta oleellisimpia termejä laajasta tesauruksesta.

Edellä esitetyt esikäsittelyvaiheet pätevät myös rakenteisten dokumenttien käsittelyssä, mutta ennen niiden soveltamista on huomioitava rakenteen esittäminen indeksissä (tästä tarkemmin luvussa 5.3.1) ja sisältötekstin erottaminen rakenteesta. XML-dokumenttien lukeminen on helppoa esim. SAX[16]- tai DOM[17]-rajapintaa noudattavalla jäsentimellä. XML-jäsentimiä on saatavilla useimmille ohjelmointikielille ja esim. Javan viimeisimmissä versioissa ne kuuluvat kielen standardikirjastoon. XML:n tavoitteiden mukaan myöskään oman jäsentimen toteuttamisen ei pitäisi olla ylivoimaista [18].

[16] http://www.saxproject.org/
[17] http://www.w3.org/DOM/

4 Klusterointi

Klusterointi on tiedonlouhinnan perusmenetelmä, joka ryhmittelee havaintoaineiston toisiaan muistuttavista alkioista koostuviin klustereihin. Koneoppimisen kannalta klusterointi kuuluu ohjaamattoman (*unsupervised*) oppimisen menetelmiin, tilastotieteen näkökulmasta se on monimuuttujamenetelmiin kuuluvaa ryhmittelyanalyysia [76, sivut 335-336]. Muihin tiedonlouhinnan menetelmiin verrattuna klusterointi muistuttaa lähinnä luokittelua. Menetelmien suhdetta käsitellään luvussa 4.1, mutta muilta osin luokittelua ei käsitellä tarkemmin. Lisäksi muutamat klusterointimenetelmät käyttävät assosiaatioiden louhintaa algoritmin osana.

Tiedonlouhinnan termein ilmaistuna klusterit ovat tietojoukossa piilossa olevia hahmoja, jotka klusteroinnin tuloksena esitetään käyttäjälle yhtenäisenä mallina. Intuitiivisesti klusterin käsite on selkeä, mutta sen formaali määrittely on hankalaa. Havaintoaineistosta muodostetut klusterit eivät ole välttämättä yksikäsitteisiä, vaan voivat riippua esim. datapisteiden käsittelyjärjestyksestä. Lisäksi eri algoritmeilla muodostetut klusterit poikkeavat toisistaan muodoltaan, määrältään ja tarkkuudeltaan. Estivill-Castron [54] mukaan klusterointialgoritmien moninaisuus johtuu toisaalta useista mahdollisista malleista, joilla klustereita voidaan esittää, toisaalta erilaisista optimointifunktioista, joiden perusteella klusterointitehtävän tavoitteet määritellään. Klusterointimenetelmien vaatimukset vaihtelevat sovellusaloittain. Useimpiin klusterointimenetelmiin kuitenkin pätee periaate, jonka mukaan klustereiden sisällä olevien pisteiden etäisyydet on minimoitava ja klusterien väliset etäisyydet maksimoitava [76, sivu 25].

Dokumenttien klusterointiin liittyvät samat haasteet kuin tiedonlouhintaan yleensäkin. Beil *et al.* [8] mainitsevat näytteiden (tässä tapauksessa dokumenttien) suuren määrän ja piirteiden korkean dimension (runsaasti indeksitermejä). Erityisesti tekstitiedolle ominaisia tekijöitä ovat harvat piirrevektorit (yksittäiset dokumentit sisältävät vain murto-osan kaikissa dokumenteissa esiintyvistä sanoista) ja epästandardit jakaumat (mallipohjaisessa klusteroinnissa usein käytetyt normaalijakaumat eivät yleensä sovellu dokumenttidatalle). Erityisenä yksityiskohtana poikkeamat saattavat tiedonlouhinnassa olla juuri ”kiinnostavia hahmoja”, joita etsitään. Tällöin niitä ei pidä poistaa, kuten data-analyysissa yleensä [125, sivut 11-12]. Han & Kamber [76, sivut 21-28] ovatkin nostaneet poikkeamien analyysin omaksi tiedonlouhinnan lajikseen (katso luku 2.2.3). Strehl [125, sivut 11-12] lisää sovellusriippuvaisiksi haas-

teiksi myös mahdolliset aiemmin generoidut tai dokumenttikokoelman osajoukolle tehdyt klusterimallit (miten integroida olemassaolevat mallit uusiin?) sekä hajautetun datan (esim. WWW-ympäristö, monta tietovarastoa). Käytännön sovellusten kannalta oleellisin klusterointiin liittyvä ongelma on klusterien esittäminen käyttäjälle ymmärrettävällä tavalla [8].

Perusmenetelmät suuren näytemäärän käsittelyyn ovat inkrementaalinen klusterointi, klusteroinnin kuluessa laskettavat tunnusluvut datasta sekä käsiteltävän datajoukon pienentäminen näytteistämällä [10]. Niiden tarkempi käsittely sivuutetaan tässä. Ulottuvuuksien vähentämismenetelmiä käsitellään luvussa 4.4.1. Klusterien esitystapoja käydään läpi empiirisen osuuden kannalta ja yleisemminkin luvussa 6.2.2. Tutkielman sovellusalueen erityisvaatimuksena on rakenteisten dokumenttien esittämisen hankaluus, joka yleisimmässä muodossaan vaatii mutkikkaita indeksirakenteita ja pelkästään indeksitermien osalta tuhansia ulottuvuuksia. Tästä johtuen empiirisessä osuudessa on käytetty vain metrisessä avaruudessa toimivia klusterointialgoritmeja. Tällöin algoritmi on riippumaton dokumenttien sisäisestä esitystavasta ja on helposti vaihdettava. Klusterointimenetelmän kannalta riittää tieto samanlaisuusmitasta, jolla dokumentteja verrataan toisiinsa.

4.1 Luokittelu ja klusterointi

Klusterointi muistuttaa luokittelua, mutta on sitä oleellisesti vaikeampi ongelma. Luokittelu on ohjattua (*supervised*) oppimista, missä luokittelijaa koulutetaan aluksi opetusdatalla, jossa alkioiden luokat ovat tiedossa. Luokittelussa jokainen luokka voidaan kuvata sille tunnusomaisilla piirteillä, kun taas klusteroinnissa datasta tai yksittäisten piirteiden tärkeydestä ei välttämättä tiedetä mitään (onhan klusteroinnin yksi päätarkoitus antaa *yleiskuva* datasta jatkokäsittelyä varten) [140, sivu 80]. Datan piirteiden tyyppi, valittu etäisyysmitta ja klusterointimenetelmä vaikuttavat luonnollisesti lopputulokseen ja sisältävät väistämättä joitakin oletuksia. Useimmat klusterointimenetelmät vaativat lisäksi käyttäjältä syöteparametreja, joista tyypillisin on klusterien määrä.

Klusterointia kutsutaan joskus automaattiseksi luokitteluksi, joskin tämä pätee lähinnä mallipohjaiseen klusterointiin. Tällöin datasta saadaan tilastollinen malli, jonka perusteella uusia datapisteitä voidaan luokitella. Tyypillinen esimerkki mallista on ennalta annetusta määrästä n-ulotteisia normaalijakaumia koostuva sekatiheysmalli (*mixture model*). Klusteroinnin aikana mallille estimoidaan odotusarvovektorit ja kovarianssimatriisit. Yleisesti klusterointi ja luokittelu ovat kuitenkin eri asioita. Luokittelun näkökulmasta klusterointi saattaa tuntua tarpeettoman hankalalta tehtävältä ja voidaan kysyä, onko datajoukolle edes olemassa mitään ”luontaista” ryhmittelyä – luokittelukirjallisuudessa vallalla olevan käsityksen mukaan

ei ole [140, sivu 80]. Duda *et al.* [46, sivut 517-518] suosittelevat kuitenkin klusteroinnin (tai ylipäänsä ohjaamattoman oppimisen käyttöä) luokittelun sijaan seuraavissa tilanteissa:

- **Luokittelijan suunnittelu**. Opetusdatan luokittelu käsin voi olla huomattavasti sen keräämistä työläämpää. Klusteroinnin avulla opetusjoukko saadaan esiluokiteltua automattisesti.
- **Automaattinen luokittelu**. Klusteroidaan data ja nimetään löydetyt ryhmät manuaalisesti.
- **Dynaamiset luokat**. Ohjaamattomalla oppimisella voidaan mukauttaa luokittelua, jos luokkien määritykset muuttuvat ajan edetessä.
- **Piirteiden etsintä**. Klusterointia voidaan käyttää sopivien piirteiden etsimiseen. Dokumenttien klusteroinnin ohella myös termien klusterointi on eräs tiedonhaun osatehtävä.
- **Yleiskuva datasta** data-analyysin alkuvaiheessa. Tämä on tavallisin klusteroinnin käyttökohde.

Kun tietojoukko koostuu dokumenteista tai muusta tekstitiedosta, luokittelua kutsutaan usein tekstin kategorisoinniksi. Termi *automaattinen tekstin luokittelu* ei ole täsmällinen, koska sillä voidaan tarkoittaa ainakin seuraavia asioita [120]:

1. Luokittelua ennalta määrättyihin kategorioihin.
2. Kohdassa 1 mainittujen kategorioiden etsimistä.
3. Kategorioiden etsimistä ja ryhmittelyä (siis dokumenttien klusterointia).
4. Mitä tahansa tekstijoukon sijoittelua ryhmiin sisältäen sekä dokumenttien klusteroinnin että tekstin kategorisoinnin.

Tekstin kategorisointi sivuaa tiedonhakua mm. web-aihehakemistojen muodostuksen osalta. Yahoon[1] ja dmozin[2] kaltaiset aihehakemistot ovat käsin muodostettuja, joten niitä voidaan käyttää opetusdatana luokiteltaessa muita WWW-sivuja [120] tai validoitaessa web-dokumenttien klusterointia. Web-aihehakemisto on myös standardi tapa kuvata erilaisia web-sivuja samaan tapaan kuin asiasanasto kuvaa yksittäiset sanat [140, sivut 101-102]. Aihehakemistoja voidaan pitää verkkoon siirrettynä ja laajennettuna versiona kirjastoissa käytetyistä luokitusjärjestelmistä [140, sivu 60]. Hakutulosten luokittelu aihehakemiston mukaan on vaihtoehtoinen tapa hakutulosten esittämiseen klusteroinnin sijaan. Molemmat tavat ovat

[1] `http://dir.yahoo.com/`
[2] `http://www.dmoz.org/`

tämän kirjoittajan mielestä perinteistä järjestettyä listaa havainnollisempia, mutta jatkossa keskitytään lähinnä hakutulosten klusterointiin.

4.2 Klusterointi tiedonhaussa

Tiedonhaussa voidaan klusteroida joko dokumentteja tai indeksitermejä [129, sivu 23]. Dokumenttien klusterointia voidaan hyödyntää seuraavissa tiedonhakuprosessin vaiheissa [141, sivut 35-37]:

- **Dokumenttikokoelman esiklusterointi**. Klusterointia on varhaisimmissa tiedonhakujärjestelmissä käytetty parantamaan haun *suorituskykyä*. Kun dokumentit klusteroidaan esikäsittelyvaiheessa, kyselyä ei tarvitse verrata kaikkiin dokumentteihin, vaan vertailu klustereiden prototyyppeihin riittää. Hakutulokset saadaan yksittäisten dokumenttien sijaan klustereina [129, sivu 24]. Menetelmää kutsutaan klusteripohjaiseksi hauksi. Klusterointi toteutetaan yleensä hierarkkisesti, jolloin täsmäytys etenee joko klusteripuun huipulta tai yksittäisistä dokumenteista lähtien. Myöhemmissä tutkimuksissa on ehdotettu, että klusteripohjaisella haulla voitaisiin parantaa suorituskyvyn lisäksi myös haun *laatua* [136]. Tämän perusteena on kiistelty klusterointihypoteesi, jota käsitellään tarkemmin edempänä. Klusterihaun kehittyneemmässä versiossa kyselyyn parhaiten sopivien klusterien dokumentit täsmäytetään erikseen, mikä parantaa haun laatua pelkkien klusterien palautukseen verrattuna [131].
- **Klusteripohjainen selaus**. Klusteripohjaista hakua on pidetty yleisesti ottaen pettymyksenä. Hierarkkisten klusterointialgoritmien laskennallinen vaativuus on vähintään kertaluokkaa $\Theta(n^2)$, mikä ei ole hyväksyttävää suurilla dokumenttikokoelmilla edes esikäsittelyssä. Lisäksi useissa tutkimuksissa on todettu, ettei klusterihaku ole parantanut haun laatua tai voi jopa olla huonompi kuin tuloslistoihin perustuva haku. Tämän takia 1990-luvun alussa esitettiin klusterointia avuksi dokumenttikokoelman selaukseen. Tällöin käyttäjä tekee relevanssiarviot järjestelmän puolesta klusterien lyhennelmien perusteella ja voi navigoida klusteroidussa dokumenttikokoelmassa. Käyttökokemukset ovat olleet positiivisia verrattuna klusteripohjaiseen hakuun [140, sivu 67-70]. Klusteripohjaisen selauksen ideaa ovat kehittäneet Cutting *et al.* [37] Scatter/Gather-järjestelmällään. Järjestelmässä klusterointi on iteratiivinen prosessi, jossa dokumenttikokoelma tai hakutulokset ”hajotetaan” aluksi klustereihin. Käyttäjä valitsee relevantit klusterit, jonka jälkeen järjestelmä ”kokoaa” ne uudeksi dokumenttikokoelmaksi ja hajottaa sen hienommalla granulariteetilla. Scatter/Gather-järjestelmän ideat muistuttavat empiirisessä osuudessa kehitettyä hakumallia, jota käsitellään luvussa 6.2.

- **Hakutulosten klusterointi** yleistyi 1990-luvun lopulla. Tekniikalla voidaan esim. selkeyttää web-hakukoneiden laajoja tuloslistoja. Zamirin [141, sivu 9] Grouper oli todennäköisesti ensimmäinen WWW:ssä toimiva järjestelmä, joka klusteroi muista hakukoneista saatuja hakutuloksia. Lähestymistapa soveltuu erityisesti web-ympäristöön, koska nykyisten hakukoneiden tuloslistoissa on merkittäviä eroja – eri hakukoneet indeksoivat eri dokumentteja [140, sivut 131-136]. Hakukoneiden tulosten yhdistely on esimerkki fuusiohausta, jota käsitellään tarkemmin luvussa 5.5. Hakutuloksia voidaan ryhmitellä myös mahdollisen esiklusteroinnin mukaan, jolloin ryhmittelyä voisi verrata lähinnä web-aihehakemiston avulla tehtyyn (staattiseen) luokitteluun. Tulosjoukon mukaan suoritettu klusterointi on kuitenkin yleensä esiklusterointia laadukkaampaa, koska relevantit dokumentit saattavat hajaantua moneen eri klusteriin staattisessa klusterimallissa [141, sivut 35-37]. Jälkiklusterointi mukautuu kyselykohtaisesti ja lieventää dokumenttien synonyymi- ja kieliasuongelmia. Jos hakusana esiintyy dokumenteissa useammassa eri merkityksessä, eri aiheita edustavat dokumentit sijoitetaan todennäköisesti eri klustereihin. Hakutulosten klusterointi on klusteripohjaisen selauksen ohella osa empiirisen osuuden hakumallia.

Tärkein tiedonhaussa käytetty peruste klusteripohjaiselle haulle on ns. klusterointihypoteesi: *Samanlaiset dokumentit ovat relevantteja samoilla kyselyillä*. Toisin sanoen tietyn kyselyn yhteydessä palautetut dokumentit muistuttavat toisiaan enemmän kuin poisjätetyt dokumentit. Klusterointihypoteesin voimassaoloa voidaan van Rijsbergenin mukaan testata laskemalla samanlaisuuksien jakaumat dokumenttipareille, joista molemmat ovat relevantteja, ja toisaalta pareille, joista vain toinen on relevantti. Jos jakaumat poikkeavat merkittävästi toisistaan, klusterointihypoteesi on voimassa annetulla kyselyllä [129, 30-31]. Voorheesin [131] eri standardikokoelmilla tekemien testien perusteella klusterihaun laatu ei riipu merkittävästi klusterointihypoteesin voimassaolosta. Tulosten laatua saattoi tosin heikentää klusteroinnissa käytetty hierarkkinen lähimmän pisteen menetelmä.

Klusterointihypoteesi on herättänyt runsaasti kritiikkiä ja keskustelua. Vieläkään ei ole selvää, onko hypoteesista hyötyä yleisessä tiedonhaussa. Shaw *et al.* [123] esittävät voimakasta kritiikkiä erityisesti klusteripohjaista hakua kohtaan. Kritiikkiä perustellaan 13:n standardikokoelman testiaineistolla. Tuloksista havaittiin, että täysin satunnaisesti muodostetuista klustereista saadaan laadultaan samaa luokkaa olevia hakutuloksia kuin standardimenetelmillä muodostetuista. Klusterihaun huono laatu voi johtua yhdestä tai useammasta seuraavista tekijöistä:

1. Klusterointihypoteesi ei pidä paikkaansa dokumenttikokoelmalla (vertaa Voorheesin tuloksiin).

2. Klusterointialgoritmi ei sovellu käsiteltävään dokumenttikokoelmaan, parametrit ovat väärät tai dokumenteilla ei ole ”luonnollista” ryhmittelyä.

3. Tiedonhakumalli ei osaa palauttaa relevanteimpia klustereita.

Shaw toteaa, että suurimmassa osassa klusterihakua käsittelevää tutkimusta on mukana oletus, että dokumenttien aiheiden samankaltaisuudesta seuraa relevanssisuhteita. Kiinnostavaa on, että tämä oletus on täysin vastakkainen probabilistisen tiedonhaun perusoletusten kanssa (katso luku 5.1.2). Shaw myöntää, että yksittäisten dokumenttien relevanssiarvot huomioonottava ”adaptiivinen” klusterointi saattaa toimia esiklusterointia paremmin, mikä on yhtäpitävää Voorheesin tulosten ja hakutulosten klusteroinnin ideoiden kanssa. Klusteripohjainen selaus ja hakutulosten klusterointi ovatkin antaneet tukea klusterointihypoteesille täydennettynä idealla, että relevanttien dokumenttien klusterointi on kyselyriippuvaista [140, sivut 131-136]. Hearst [81, sivu 352] oli havainnut Scatter/Gather-järjestelmällä, että hakutuloksia klusteroitaessa relevantit dokumentit sijaitsivat tyypillisesti yhdessä tai kahdessa klusterissa. Zamirin [141, sivut 2-3] esittämän *käyttäjän klusterointihypoteesin* mukaan hakutulosten klusterointi kuvastaa käyttäjän mielessä olevaa mallia tulosdokumenttien aiheista ja niiden välisistä suhteista. Siksi klusterit selkeyttävät hakutulosten selausta. Tuoreen näkökulman klusterien käyttöön ovat esittäneet myös Zhang *et al.* [142] yhdistämällä klusteripohjaisen haun fuusiohakuun. Mallia käsitellään luvussa 5.5.

Tämän kirjoittajan mielestä klusterointihypoteesin paikkansapitävyys ja hyödyllisyys riippuu mittakaavasta, jolla dokumenttikokoelmaa tarkastellaan. Karkealla tasolla hypoteesi pitää yleensä paikkansa, jos haku kohdistuu johonkin yleiseen aiheeseen. Hakua tarkennettaessa (tai esim. hierarkkisen klusteroinnin dendrogrammissa edetessä) tulee kuitenkin tilanne, jossa klusterointimalli ei kykene erottamaan relevantteja dokumentteja epärelevanteista. Erityisesti tämä pitää paikkansa esiklusteroidussa kokoelmassa, mutta myös hakutuloksista saatujen ”relevanttien” klusterien *kaikki* dokumentit eivät ole relevantteja. Syynä tähän voi olla liian karkea dokumenttien esitystapa tai liian yleinen samanlaisuusmitta. Tästä johtuen klusterointia käyttävät hakumallit tarvitsevat tuekseen myös tuloslistan, joka järjestää dokumentit relevanssiarvion mukaan klusterien sisällä tai globaalisti. Tätä tukevat myös Voorheesin [131] ja Hearstin [81, sivu 352] tulokset. Klusteripohjaisen haun relevanssia olisi siis mielekästä arvioida yleisenä *aiherelevanssina*, kun taas klusterien sisällä olevat dokumentit kuuluvat *instanssirelevanssin* piiriin [139]. Myös käyttäjän antamat kyselytermit vaikuttavat klusteroinnin onnistumiseen erityisesti hakutulosten klusteroinnin osalta (esim. jos hakutermit liittyvät toisiinsa, suurin osa hakutuloksista jää todennäköisesti samaan klusteriin).

Tiedonhaussa termien klusteroinnin tarkoituksena on laajentaa kyselyä termeillä, joista käyttäjällä ei ole ollut aiemmin tietoa [136]. Termien klusterointia on käytetty myös piirteiden

erottelutekniikkana tekstin kategorisoinnissa tai dokumenttien klusteroinnissa, tavoitteena löytää dokumentista ”oleelliset” ja edustavimmat termit [136] (katso luku 4.4.1). Kummassakaan tapauksessa termien klusterointi ei kuitenkaan ole yleensä johtanut merkittävästi parantuneisiin haku/luokittelutuloksiin. Yksinkertainen tapa termien klusterointiin on verrata dokumenttien esiintymistiheyksiä termien suhteen, jolloin dokumenttivektorien sijaan klusteroidaan ”termivektoreita” (katso luku 5.1.1). Kehittyneempi tapa olisi käyttää jotain semanttista mittaa termien samanlaisuuden arviointiin, kuten etäisyyttä WordNet[3]-luokituksessa [105]. Tällöin ongelmana on lähinnä sovellusalue- ja kieliriippuvuus. Lisäksi termien täytyy olla tarkasti perusmuotoon palautettuja.

Dokumenttien klusteroinnissa on perinteisesti keskitytty joko sanoihin tai linkkeihin [136], mutta näitä molempia yhdistävät etäisyysmitat ovat jääneet vähemmälle huomiolle aivan viimeisiä vuosia lukuunottamatta. Rakenteiset ja hyperlinkitetyt dokumentit vaativat tekstidokumentteja monimuotoisempia esitystapoja. Rakenteista indeksointia käsitellään tarkemmin luvussa 5.3 ja yhdistettyjä hakutapoja luvussa 5.4. Tässä luvussa ei pääsääntöisesti oteta kantaa dokumenttien esitystapaan tai siihen, mitä dokumenttien tietoja klusteroidaan. Valittu klusterointialgoritmi aiheuttaa kuitenkin rajoitteita dokumenttien esittämiseen. Esimerkiksi K-means vaatii syötteenään vektorimuotoista tietoa, mutta hierarkkiset klusterointialgoritmit käyttävät pelkästään samanlaisuusmittaa.

4.3 Klusterointimenetelmien jaottelutapoja

Jain *et al.* [83] jakavat klusterointiprosessin *hahmojen esittämiseen* (piirteiden valinta ja erottelu sekä algoritmin mahdollisten parametrien valinta), *samanlaisuusmitan valintaan* (useimpien menetelmien osalta), *ryhmittelyyn* sekä valinnaisiin *abstrahointiin* (klusterien esittäminen) ja *tulosten arviointiin* (validointi). Tiedonlouhintamenetelmänä klusterointi voidaan sovittaa osaksi KDD-prosessia (katso luku 2.2.1). Jos oletetaan, että datan valinta ja esikäsittely on suoritettu, hahmojen esittäminen ja samanlaisuusmitan valinta kuuluvat *muunnosvaiheeseen* (vaihe 3), ryhmittely vastaa *tiedonlouhintaa* (vaihe 4) ja viimeiset vaiheet kuuluvat *tulkintaan* (vaihe 5). Mahdollinen parametrien, etäisyysmitan tai piirteiden hienosäätö tekee klusterointiprosessista interaktiivisen ja iteratiivisen. Huomattavaa on, että *hahmolla* on Jainin terminologiassa eri merkitys kuin tiedonlouhinnassa yleisesti. Tässä hahmot viittaavat klusteroinnin syötteisiin, kun yleensä hahmoilla tarkoitetaan tiedonlouhinnan tuloksia.

Useimpiin klusterointimenetelmiin liittyy tavalla tai toisella tietoalkioiden vertaamiseen käytetty mitta, jolla alkioiden samanlaisuus määritellään. Poikkeuksia tähän ovat eräät ruudus-

[3] http://www.cogsci.princeton.edu/~wn/

topohjaiset, tilastollisiin jakaumiin tai assosiaatiosääntöihin perustuvat menetelmät. Usein käytettyjä etäisyysmittoja ovat esimerkiksi L_p-etäisyydet

$$d_{L_p}(a,b) = \| \ a - b \ \|_p = (\sum_{i=1}^{n} |a_i - b_i|^p)^{1/p},$$

missä a ja b ovat verrattavien alkioiden piirrevektoreita. Erikoistapauksena L_2-etäisyys on tuttu euklidinen etäisyys. Numeeristen vektorien lisäksi mittoja on kehitetty myös muille tietotyypeille (esim. luokiteltu muuttuja), mutta tässä rajoitutaan käsittelemään numeerisia muuttujia. Menetelmästä riippuen klusterointialgoritmi tarvitsee joko *samanlaisuus*mitan tai *etäisyys*mitan, mutta jos näytteet tai mitta on normalisoitu välille $[0, 1]$ käsitteet ovat keskenään vaihdettavia.

Klusterointimenetelmiä on kehitetty erittäin runsaasti eri sovellusalueille. Menetelmät poikkeavat toisistaan mm. syötetiedon muodon, tavoitteen (voidaan menetelmästä riippuen määritellä formaalisti optimointifunktiona), taustalla olevan matemaattisen mallin (tilastotiede, sumea logiikka...) tai muodostettavien klusterien tyypin suhteen. Algoritmien moninaisuudesta ja erilaisista tavoitteista johtuen kirjallisuudessa ei ole yksimielisyyttä edes klusterin formaalille käsitteelle, joten kaikkien algoritmien luettelu ei ole järkevää tai edes käytännössä mahdollista. Sen sijaan voidaan tehdä erilaisia jaotteluja, joita on myös runsaasti saatavilla. Yksimielisyyttä ”oikeasta” jaottelustakaan ei ole, joten tässä on tyydytty hakemaan erilaisia näkökulmia menetelmien jaottelun perustaksi. Tällöin klusterointimenetelmiä voidaan luokitella näkökulmien perusteella, joista tärkeimmät ovat *klusterien laji*, *klusteroitava tieto* ja *matemaattinen malli*. Lista ei ole tyhjentävä eikä täysin formaali, mutta auttoi ainakin tämän kirjoittajaa hahmottamaan klusterointialgoritmien moninaisuutta.

Perinteisin (mutta tämän kirjoittajan mielestä ei missään tapauksessa kattavin) tapa on jakaa klusterointimenetelmät osittaviin (*partitional*) ja hierarkkisiin menetelmiin [83]. Nimitys viittaa klusterimallin tyyppiin, joka edellisellä on lähtöavaruuden (tai näytteiden) ositus ja jälkimmäisellä puurakenne. Tätä yleisempi jaottelu on mm. Zhongilla & Ghoshilla [144], jotka jakavat klusterointialgoritmit *mallipohjaisiin* ja *diskriminatiivisiin*. Näistä edellinen pyrkii oppimaan datasta jonkin (tyypillisesti tilastollisen) mallin, jota voidaan käyttää myöhemmin luokittelun tapaan. Jälkimmäinen osittaa havaintoaineiston samanlaisuusmitan perusteella, mutta ei tee datasta mallia (muuten kuin määrittämällä jokaisen pisteen kuuluvuuden tiettyyn klusteriin – tätäkin voitaisiin hyödyntää luokittelussa ainakin K-lähimmän naapurin menetelmää käyttäen). Estivill-Castron [54] kolmitasoinen jaottelu koostuu *mallista*, *induktiivisesta periaatteesta* ja *algoritmista*. Tässä *malli* on Zhongin vastaavaa käsitettä yleisempi vastaten lähinnä luvussa 4.3.2 käsiteltyjä klusterien lajeja. Induktiivinen periaate on formalisoitu tavoite, jonka mukaan klusterointi etenee ja kuuluu tässä mainituista näkö-

kulmista luvussa 4.3.4 käsiteltyyn teoreettiseen malliin. Jokaista mallia vastaa useita induktiivisia periaatteita. Edelleen on useita algoritmeja, jotka approksimoivat tiettyä periaatetta.

4.3.1 Algoritminen näkökulma

Luvussa on esitetty nykyisin yleisesti käytetty perusjaottelu algoritmien toiminnan mukaan Hania & Kamberia [76, sivut 346-348] mukaillen. Jako ei ole kattava eikä täysin täsmällinen, mutta esitetään perusteellisuuden vuoksi. Monet uudemmat menetelmät yhdistelevät eri perustyyppejä (esim. aluksi havaintoavaruuden ositus ruudustomenetelmällä ja klusterointi tiheysmenetelmällä [10]) ja uusia ryhmittelyyn sopimattomia algoritmeja esitetään. Tässä tutkielmassa painopisteenä ovat aliluvuissa 4.3.2, 4.3.3 ja 4.3.4 esitetyt näkökulmat, jotka tarjoavat moniulotteisen tavan menetelmien luokitteluun takertumatta algoritmisiin yksityiskohtiin.

- **Osittavat** menetelmät perustuvat havaintoavaruuden iteratiiviseen jakamiseen. Klusterien paikat muuttuvat algoritmin edetessä, mutta klusterien määrä on yleensä sidottu. Tyypillinen esimerkki osittavasta algoritmista on K-means-algoritmi, joka todennäköisesti on myös yleisin kaikista olemassaolevista klusterointialgoritmeista. Aluksi käyttäjä määrittelee klusterien määrän, joiden prototyypeille arvotaan alustavat paikat. Jokainen havainto sijoitetaan kuuluvaksi lähimpään klusteriin, jonka paikkaa korjataan arvojensa keskiarvon perusteella. K-meansin ongelmia ovat herkkyys poikkeamille ja riippuvuus aloituspisteistä. Algoritmin löytämät klusterit ovat ympyrämäisiä. K-meansin merkittävin etu on sen laskennallinen tehokkuus, joka on kertaluokkaa $\Theta(n)$. Hieman K-meansia robustimpia (mutta laskennallisesti vaativampia) ovat medoidimenetelmät, joissa klustereiden prototyypit ovat keskiarvopisteiden sijaan mediaanipisteitä. K-spatialmedians [96] on esimerkki robustista menetelmästä, jossa puuttuvat arvot tai poikkeamat eivät vaikuta merkittävästi prototyypin laskentaan.

- **Hierarkkiset** menetelmät jakautuvat kokoaviin ja jakaviin menetelmiin. Datasta muodostetaan puurakenne, jonka solmut edustavat klustereita tietyllä tarkkuustasolla. Kokoavissa menetelmissä puu muodostetaan aloittamalla yksittäisistä havaintoalkioista ja yhdistelemällä lähimpiä alkioita, jakavissa menetelmissä koko havaintoaineisto tulkitaan alussa yhdeksi klusteriksi, jota jaetaan. Hierarkkisten menetelmien etuna on mahdollisuus tarkastella havaintoaineistoa monella tarkkuustasolla. Ongelmia ovat lopetuskriteerin määritys, mallin staattisuus jo muodostettujen klusterien suhteen ja etäisyysmatriisin laskennasta johtuen vähintään kertaluokkaa $\Theta(n^2)$ oleva aika- ja tilavaativuus. Klassisia kokoavia menetelmiä ovat lähimmän pisteen (*single linkage*), keskipisteen (*group average*) tai yhteisetäisyyden (*complete linkage*) menetelmät [136].

Nimet viittaavat kriteeriin, jolla klusterien välisiä etäisyyksiä mitataan. Lähimmän pisteen menetelmässä klustereita edustavat niiden toisiinsa nähden lähimmät pisteet, yhteisetäisyyden menetelmässä kauimmaiset pisteet ja keskipisteen menetelmässä etäisyyksien keskiarvot. Yhteisetäisyyden menetelmä tuottaa pieniä ja koherentteja klustereita, lähimmän pisteen menetelmä tuottaa vastaavasti löyhiä klustereita, joiden ongelmana voi olla klusteroinnin ”väärien” pisteiden liittämisestä johtuva ”ketjuvaikutus”. Klassisten menetelmien yleinen ongelma on lähekkäin olevien pisteiden liittäminen samaan klusteriin, vaikka globaalin mallin kannalta niiden kuuluisi olla eri klusterien reunoilla. Hierarkkisen klusteroinnin tuloksia havainnollistetaan usein dendrogrammilla, joka on tyylitelty binääripuu.

- **Tiheyteen** (*density*) perustuvien menetelmien taustalla on ajatus, että klusteroituneiden pisteiden tiheys on ympäröivien pisteiden tiheyttä suurempi. Tällöin lähellä toisiaan olevat pisteet luokitellaan samaan klusteriin. Tiheyteen perustuvat algoritmit soveltuvat mielivaltaisen muotoisten klustereiden etsintään ja kestävät hyvin kohinaa. Ongelmana on, että klustereiden mallipohjainen tulkinta voi olla hankalaa. Klusterit voidaan muodostaa yksittäisten pisteiden paikallisten ominaisuuksien perusteella. Tiheyteen perustuva DBSCAN-algoritmi (*Density Based Spatial Clustering of Applications with Noise*) käy kertaalleen kaikki havaintopisteet läpi ja luokittelee ne ydinpisteisiin, reunapisteisiin tai kohinaksi (poikkeama). Klusterit muodostuvat ydin- ja reunapisteistä. Käsiteltävälle pisteelle haetaan ϵ-parametrin säteellä oleva naapurusto, jota kasvatetaan iteratiivisesti niin kauan, kuin ympäristöstä löytyy $minpts$-parametrin verran lähipisteitä. Ydinpisteillä on aina vähintään $minpts$ lähipistettä, reunapisteet kuuluvat jonkin ydinpisteen naapurustoon, mutta niillä on vähemmän reunapisteitä. Loput pisteet ovat kohinaa [53].

- **Ruudustoon** (*grid*) perustuvissa menetelmissä on päinvastainen lähestymistapa hierarkkisiin ja osittaviin menetelmiin verrattuna. Kun edellisissä keskityttiin klustereiden muodostamiseen havaintopisteiden perusteella, ruudustoon perustuvissa menetelmissä lähdetään havaintoavaruudesta ja sen osittamisesta. Klusterit määritetään tarkimman havaintoavaruuden osituksen ja ”valittujen ruutujen” perusteella. Menetelmät eivät tällöin ole riippuvaisia havaintoalkioiden valinnan järjestyksestä, mutta klusteroinnin tarkkuus riippuu osituksen tarkkuudesta. CLIQUE (*CLustering In QUEst*) [2] on esimerkki ruudustopohjaisesta menetelmästä, joka huomioi myös alkioiden tiheyden klustereita muodostettaessa. Lähtökohtana on n-ulotteinen vektoriavaruus, joka jaetaan säännöllisiin ξ-pituisiin yksiköihin (ruutuihin, hyperkuutioihin). Algoritmi etsii yksiköt, joissa on yli τ alkiota (ns. *tiheät yksiköt*). Klusteri määritellään joukkona vierekkäisiä tiheitä yksiköitä. Tiheät yksiköt etsitään Apriori-algoritmia (katso luku

2.2.3) muistuttavalla menetelmällä, jossa ensin lähdetään 1-ulotteisista projektioista ja kasvatetaan niitä ulottuvuuksittain vain sellaisten aliavaruuksien osalta, joissa yksiköt ovat riittävän tiheitä. Lisäksi aliavaruudet, jotka sisältävät suhteessa pienen määrän tiheitä yksiköitä karsitaan pois. Tuloksena saadut klusterit kuvataan disjunktiivisessa normaalimuodossa olevilla loogisilla lausekkeilla, joista kukin rajaa alkuperäisen avaruuden aliavaruudessa olevan hyperkuution. Johtuen klustereiden määrityksistä monelle eri aliavaruudelle CLIQUE:n muodostamat klusterit ovat limittäisiä (pehmeitä) ja niitä voidaan tarkastella monesta näkökulmasta (eri aliavaruudet) [10].

Han & Kamber [76, sivut 346-348] lisäävät perusluokitukseen myös mallipohjaisen klusteroinnin. Estivill-Castro [54] kritisoi tätä jaottelua todeten, että jokainen klusterointialgoritmi generoi mallin (erityisesti myös diskriminatiivisen klusteroinnin tulos on tässä mielessä *diskreetti rakenteinen malli*). Myös Zhong & Ghosh [144] mainitsevat, että sekä mallipohjainen että diskriminitiivinen klusterointi voidaan toteuttaa joko osittavalla tai hierarkkisella algoritmilla. Mallipohjainen klusterointi ei siis sovi *algoritmiseksi* jaottelukriteeriksi. Berkhin [10] lisää edellä mainittujen perusluokitusten lisäksi *kategorisen datan yhteisesiintymiin* perustuvat menetelmät (jotka tässä jaottelussa liittyvät enemmän klusteroitavan tiedon tyyppiin tai etäisyysmittaan), rajoitepohjaisen klusteroinnin (esim. esteiden määrittely havaintoalueella, mikä voitaisiin liittää klusteroitavaan avaruuteen) sekä luokittelemattomia koneoppimisen menetelmiä, kuten neuroverkot ja geneettiset algoritmit (jotka liittyvät teoreettisen mallin näkökulmaan). Skaalautuvat algoritmit ja korkeadimensioiseen dataan keskittyvät algoritmit on mainittu erikseen, mutta tämän kirjoittajan mielestä ne ovat ennemmin klusterointialgoritmin laatuvaatimuksia kuin luokittelukriteerejä.

Jain *et al.* [83] luettelevat muutamia algoritmisia jaottelukriteerejä, jotka jätetään pääosin tämän käsittelyn ulkopuolelle. *Monoteettisyys/polyteettisyys* kuvaa, käyttääkö algoritmi yhtä vai kaikkia piirteitä kerrallaan klusteroinnin edetessä. *Deterministisyys/stokastisuus* kuvaa satunnaisuuden käyttöä. Tiedonlouhinnan kannalta merkittävin *inkrementaalinen* klusterointi tarkoittaa klusterointia dynaamisessa ympäristössä, jossa näytteet lisääntyvät ajan kuluessa. Inkrementaalinen klusterointi soveltuu erinomaisesti tietovarastoihin tai esim. web-ympäristöön. Inkrementaalista klusterointia ei ole kuitenkaan tutkittu yhtä laajasti kuin tavanomaista klusterointia, joten sen tarkempi käsittely jätetään tässä ulkopuolelle. Mainittakoon, että dokumenttien klusteroinnin yhteydessä inkrementaalinen klusterointi vaatisi myös inkrementaalista indeksin päivitystä, joka menee laajuudeltaan tämän tutkielman ulkopuolelle. Jatkotutkimusaiheena inkrementaalinen indeksointi ja klusterointi on kuitenkin ensiarvoinen.

Hybridiklusteroinnilla tarkoitetaan yleensä kahden eri klusterointimenetelmän yhdistämistä niin, että ensimmäisen algoritmin tuloksena saatua mallia hyödynnetään toisessa algorit-

missa. Käyttäjälle hybridiklusterointi näkyy yhtenäisenä algoritmina. Tyypillinen esimerkki hybridiklusteroinnista on Tantrumin algoritmi, joka yhdistää mallipohjaisen ja diskriminatiivisen klusteroinnin [127, sivut 65-70]. Ensimmäisessä vaiheessa muodostetaan mallipohjaisella hierarkkisella klusteroinnilla sekatiheysmalleista koostuva puu. Toisessa vaiheessa merkittävästi päällekkäisiä jakaumia edustavat puun solmut yhdistetään niin, että mallin uskottavuus vähenee mahdollisimman vähän. Hybridiklusterointi voi olla toteutustekniikka yhteisklusteroinnissa (katso luku 4.4.1). Myös osittavan algoritmin muuntamista hierarkkiseksi siten, että osituksen tuloksiin sovelletaan uudelleen samaa algoritmia, voidaan pitää hybridiklusterointina. Tämän algoritmityypin samaistaminen varsinaisiin hierarkkisiin klusterointimenetelmiin on kuitenkin epäselvää. Zhongin & Ghoshin [144] mukaan osittavien algoritmien rekursiivisen ajamisen tuloksena saatavat hierarkkiset klusterit eivät välttämättä ole rakenteellisessa suhteessa toisiinsa. Toisaalta Strehlin [125, sivu 7] mukaan mikä tahansa hierarkkinen algoritmi on seurausta sarjasta tasaisia malleja, joten tasainen klusterointi on perustavanlaatuisin.

4.3.2 Klusterien lajit

Näkökulma kuvaa rakenteellisia tyyppejä, joita klusteroinnin tuloksena saadaan. Se on lähellä Estivill-Castron [54] jaottelun *malleja*, josta esimerkkeinä ovat diskriminatiivisesta klusteroinnista saatavat diskreetit rakenteiset mallit, hierarkkisesta klusteroinnista saatavat puumallit, K-meansin sukuisten osittavien algoritmien *prototyyppipohjaiset* mallit tai ”mallipohjaisen” klusteroinnin tilastollisiin jakaumiin perustuvat mallit. Näkökulmassa on oleellista riippumattomuus induktiivisesta periaatteesta (tai teoreettisesta mallista), jonka pohjalta klusterimalli muodostetaan. Esimerkiksi K-means-tyylisiä algoritmeja voidaan muodostaa erilaisten optimointifunktioiden pohjalta, joita käydään läpi luvussa 4.3.4. Lähdemateriaalissa mainittujen klusterointialgoritmien pohjalta päädyttiin seuraaviin toisistaan riippumattomiin komponentteihin, joista klusterien lajit muodostuvat.

- **Hierarkkisuus**. Algoritmisen jaottelun osittavat, tiheyspohjaiset ja ruudustoon perustuvat menetelmät muodostavat tasaisen klusterointimallin, hierarkkiset menetelmät tai rekursiivisesti sovelletut osittavat menetelmät puolestaan hierarkkisen mallin. Tämä komponentti vastaa kutakuinkin useimmissa lähteissä mainittua perusjakoa osittaviin ja hierarkkisiin menetelmiin. Klusteripohjaisessa tiedonhaussa on perinteisesti suosittu hierarkkisia malleja, mutta myöhemmissä hakutulosten klusterointisovelluksissa myös tasaiset mallit ovat yleistyneet. Suurin osa hierarkkisista menetelmistä on luonteeltaan diskriminatiivisia, mutta esim. Tantrum [127, sivut 11-13] esittää hierarkkisen, tilastollisiin malleihin perustuvan menetelmän.

- **Alkioiden kuuluvuus klustereihin**. Vaihtoehtoja ovat kova (*crisp*), limittäinen (*soft*, *pehmeä*) tai sumea (*fuzzy*) klusterointi. Useimmat tutkielmassa mainitut algoritmit osittavat datan kovalla tekniikalla. Kovassa klusteroinnissa tiedetään varmasti, kuuluuko alkio klusteriin vai ei. Pehmeässä klusteroinnissa alkio voi kuulua useaan klusteriin samanaikaisesti. Esimerkki pehmeästä klusterointialgoritmista on Zamirin suffiksipuuklusterointi (*STC*) [141]. Jos limittäistä klusterointia sovelletaan hierarkkiseen algoritmiin, tuloksena on puun sijaan suunnattu syklitön verkko. Esimerkki pehmeästä hierarkkisesta algoritmista on Beilin *et al.* [8] indeksitermien assosiaatioiden louhintaan perustuva HFTC-algoritmi (*Hierarchical Frequent Term-based Clustering*). Pehmeä klusterointi on erityisen merkittävää tekstitiedonlouhinnassa, koska dokumentit voivat edustaa useita eri aiheita [81, sivu 335]. Sumeassa klusteroinnissa jokaiselle pisteelle määritetään välillä $[0, 1]$ oleva kuuluvuusarvo jokaiseen klusteriin. Tunnetuin sumea klusterointialgoritmi on *c-means*, joka on K-meansin sumea versio [83]. Pehmeää klusterointia ei pidä sekoittaa sumeaan klusterointiin, vaikka ilmauksia käytetäänkin ajoittain kirjallisuudessa synonyymeina.

- **Klusterien esittäminen**. Vaikka esitystapa on osittain riippuvainen mallin muodostamisessa käytetystä matemaattisesta mallista ja klusteroitavan tiedon tyypistä, tässä otetaan kantaa vain klusterimallin esittämiseen, ei sen muodostamistapaan. Useimmat tutkielmassa mainitut algoritmit muodostavat absoluuttisen mallin, joissa jokainen piste sijoitetaan yhteen (tai pehmeän klusteroinnin tapauksessa useampaan) klusteriin, mutta luokittelutilanteessa uusien pisteiden sijoitusta ei voi suoraan päätellä olemassaolevista. Absoluuttinen malli vastaa likimäärin Zhongin diskriminatiivista klusterointia tai Estivill-Castron diskreettiä rakenteista mallia. K-means-tyyppiset osittavat algoritmit kuvaavat klusterit prototyypeillä, jotka ovat tulkinnasta ja algoritmista riippuen esim. klusteroitavien datapisteiden mediaaneja tai approksimoivat klusteria kuvaavan jakauman odotusarvoja. Prototyypin esitystapa riippuu piirteiden tyypistä: esimerkiksi rakenteisia dokumentteja voidaan esittää graafeina tai puina. Klustereiden tilastolliseen mallintamiseen soveltuvat sekatiheysmalli kovalle klusteroinnille ja MCMM (*Multiple Case Mixture Model*) [113] pehmeälle klusteroinnille [22, sivut 108-109]. On epäselvää, kuinka sumeat joukot voidaan mallintaa tilastollisesti, koska todennäköisyyden käsite ei ole sama kuin joukkoon kuulumisen aste. Tiedonhaun puolella Fuhr [63] on kuitenkin näyttänyt, kuinka sumeaan logiikkaan perustuvat (ja itse asiassa lähes kaikki muutkin) hakumallit voidaan esittää tilastollisessa kehyksessä. Klusterien esitystapoja käsitellään tarkemmin luvussa 6.2.2.

- **Kohinan huomiointi**. Useimmat klusterointimenetelmät määrittävät kaikki havaintoavaruuden pisteet kuuluvaksi johonkin klusteriin. Datassa saattaa kuitenkin olla poik-

keamia, jotka sovellusalueesta riippuen eivät välttämättä kuuluisi klustereihin (tai sitten niiden tulisi muodostaa omia klustereitaan). KDD-prosessin kannalta poikkeamapisteiden poisto kuuluisi *esikäsittelyyn* (vaihe 2), mutta tämä ei ole aina käytännöllistä. Tällöin voidaan käyttää klusterointimenetelmää, joka havaitsee kohinapisteet heuristisesti ja jättää ne pois mallista. Tällaisia algoritmeja ovat esim. DBSCAN [53] tai Ertözin *et al.* [52] jaettujen lähimpien naapurien (*Shared Nearest Neighbors, SNN*) algoritmi. Robustit menetelmät ovat luonnostaan neutraaleja kohinan suhteen [96].

4.3.3 Klusteroitava tieto, avaruus ja etäisyysmitta

Klusteroinnissa käytetyn syötetiedon tyyppi, tästä riippuvainen klusteroitava avaruus ja käytetty etäisyysmitta aiheuttavat perustavanlaatuisia rajoitteita käytettävälle klusterointialgoritmille. Lisäksi varsinainen klusterointi voidaan suorittaa syötetiedolle, piirteille tai molemmille, mahdollisesti samanaikaisesti (yhteisklusterointi). Omaan luokkaansa voisi myös lukea havaintoavaruuden ositukseen perustuvan klusteroinnin havaintopisteiden sijaan. Algoritmisessa jaottelussa tämä luettiin ruudustopohjaisiin menetelmiin, joskin menetelmä voi toimia myös toisen klusterointialgoritmin yhtenä vaiheena. Näin on toimittu esim. tiheys- ja ruudustopohjaisia menetelmiä yhdistävässä CLIQUE-algoritmissa [2].

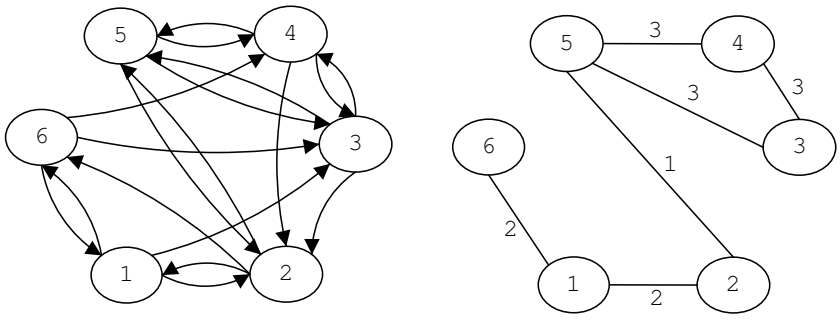
Luonnollinen tapa vertailla klusterointialgoritmeja on vertailla niissä käytettävien syötetietojen tyyppiä. Mikäli algoritmi käyttää hyväkseen vain samanlaisuusmittaa, se on neutraali syötetietojen tyypin suhteen. Tällaisten algoritmien kompleksisuus on kuitenkin yleensä vähintään kertaluokkaa $\Theta(n^2)$ johtuen työläästä samanlaisuusmatriisin laskennasta. Tämä ei ole hyväksyttävää suurilla tietojoukoilla, joten suurin osa kirjallisuudessa esitetyistä algoritmeista ottaa kantaa syötetiedon muotoon. Useimmiten oletetaan, että klusterointi tapahtuu euklidisessa vektoriavaruudessa. Tällöin klusteroitavat alkiot esitetään piirrevektoreina, joiden komponentit ovat reaalilukuja.

Han & Kamber [76, sivut 338-346] jakavat muuttujatyypit niiden mittausasteikkojen mukaan binäärisiin, luokkamuuttujiin eli kategoriseen dataan (nominaaliasteikko), järjestettyihin luokkamuuttujiin (ordinaaliasteikko) ja välimatka-asteikollisiin, jotka voivat edelleen olla jatkuvia tai diskreettejä. Muuttujien arvoja voidaan skaalata esim. logaritmimuunnoksella tai normalisoimalla ne välille $[0, 1]$. Tämäkin luokittelu olettaa, että havainnot muodostuvat tiettyä asteikkoa noudattavista vektoreista. Jos piirteitä on useita eri tyyppejä, ne voidaan yrittää normalisoida yhteismitallisiksi (tosin luokkamuuttujat vaativat joka tapauksessa oman etäisyysmitan). Luokittelu sivuuttaa kokonaan rakenteisten ja hypertekstidokumenttien vaatimat esitystavat, jotka ovat kehittyneemmillä indeksirakenteilla puita tai graafeja.

Yksinkertainen menetelmä samanlaisuuden laskemiseen kategoriselle datalle on käyttää ns. *yksinkertaista täsmäystä*, jossa lasketaan kahden alkion yhteisten muuttuja-arvojen määrä verrattuna muuttujien kokonaismäärään. Kehittyneempi menetelmä on käyttää ns. jaettujen lähimpien naapurien (*SNN*) samanlaisuutta, joka soveltuu mille tahansa muuttujatyypille kategorinen data mukaanlukien. Datajoukko hahmotetaan täydellisenä verkkona, jossa kaarien painot ovat alkioiden samanlaisuusarvot jonkin alustavan samanlaisuusmitan perusteella. SNN-pohjainen etäisyysmitta tunnetaan myös keskinäisenä naapurien etäisyytenä (*Mutual Neigbor Distance, MND*) [83]. SNN-samanlaisuus lasketaan seuraavasti [52]:

1. Lasketaan suunnattu lähimpien naapurien graafi, jossa jokaisesta yksittäistä alkiota kuvaavasta solmusta on linkit n:ään lähimpään solmuun (jokaiseen solmuun liittyy siis vähintään n kaarta). Lähimpien naapurien graafi on harvennettu versio alkuperäisestä samanlaisuusgraafista.

2. Muodostetaan suuntaamaton jaettujen lähimpien naapurien graafi siten, että jokaiselle alkioparille merkitään yhteisten naapurien määrä lähimpien naapurien graafista. SNN-graafiin otetaan mukaan vain pisteet, jotka ovat toistensa lähimpien naapurien joukossa. Jos esimerkiksi alkion 1 naapurit ovat $\{2, 3, 6\}$ ja alkion 2 naapurit $\{1, 5, 6\}$ ($n = 3$), niiden välinen samanlaisuus on 2 (1 ja 2 ovat toistensa naapureita, lisäksi niillä on yhteinen naapuri 6). Arvo voidaan normalisoida jakamalla se n:llä. Kuvassa 4.1 on esimerkki lähimpien naapurien graafista sekä SNN-graafista.



Kuva 4.1: 3-lähimmän naapurin graafi ja sen pohjalta muodostettu SNN-graafi.

Reaaliarvoisten muuttujien joukko voidaan esittää $\mathbb{R}^n$-vektoreina, mikä lienee yleisin datan esitystapa klusterointisovelluksissa. Esimerkiksi K-means-algoritmi on suunniteltu juuri tähän tapaukseen. Käytetty samanlaisuusmitta vaikuttaa jonkin verran algoritmin ajoon ja tulosten tulkintaan. Perinteinen euklidinen etäisyys soveltuu helposti K-meansin kanssa ajettavaksi, mutta esim. dokumenttien klusterointiin vektorimallin avulla käytetty kosinimitta (vektorien välinen kulma, katso luku 5.1.1) ei sovellu suoraan K-meansiin. Syy tähän on, että kosinimittaa käytettäessä dokumentit ovat etäisyysmitan kannalta n-ulotteisen pallon pinnalla. Tällöin tavallinen keskiarvon laskeminen ei johda etäisyysmitan kannalta jou-

kon keskimmäiseen alkioon. Yhtenä ratkaisuna tähän Modha & Spangler ovat kehittäneet K-meansin muunnoksen, joka lähtökohtaisesti toimii pallopinnoista johdetussa avaruudessa (katso luku 5.4). Euklidisen etäisyyden ja kosinimitan lisäksi suosittu samanlaisuusmitta on laajennettu Jaccardin mitta, joka geometrisesti asettuu euklidisen etäisyyden ja kosinimitan välimaastoon ottaen huomioon sekä vektorien välisen kulman että läheisyyden [125, sivut 94-98]:

$$sim_J(a,b) = \frac{(a|b)}{\|a\|^2 + \|b\|^2 - (a|b)},$$

missä $(a|b)$ on vektorien a ja b sisätulo.

Metrisessä avaruudessa alkioiden väliset etäisyydet tiedetään, mutta niille ei voida osoittaa yksikäsitteistä koordinaattien avulla ilmaistavaa paikkaa. Formaalisti määriteltynä *etäisyysfunktio* eli d on tyyppiä $\mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$, missä $\mathbf{X}$ on näytteiden joukko. Funktiolla on seuraavat ominaisuudet [24]:

1. $\forall x, y \in \mathbf{X}, d(x,y) \geq 0$ (positiivisuus)
2. $\forall x, y \in \mathbf{X}, d(x,y) = d(y,x)$ (symmetrisyys)
3. $\forall x \in \mathbf{X}, d(x,x) = 0$ (refleksiivisyys)

Jos lisäksi on voimassa kolmioepäyhtälö

4. $\forall x, y, z \in \mathbf{X}, d(x,y) \leq d(x,z) + d(z,y)$,

funktiota d sanotaan *metriikaksi* ja pari $(\mathbf{X}, d)$ on metrinen avaruus. Tästä huolimatta jotkut metrisen avaruuden klusterointimenetelmät käyttävät vain ehdot 1-3 toteuttavaa etäisyysfunktiota tai (mahdollisesta normalisoidusta muodosta johdettua) samanlaisuusmittaa. Tarkasti ottaen kyseessä on tällöin *pseudometrinen avaruus*. Jotkut menetelmät simuloivat vektoriavaruuden käyttöä projisoimalla datapisteet johonkin ”sopivaa” dimensiota edustavaan vektoriavaruuteen, jolloin voidaan käyttää tavanomaisia klusterointimenetelmiä [24]. Projektiot ovat vain approksimaatioita, koska kaikkia metriikoita ei voida esittää ekvivalentisti euklidisessa avaruudessa. Esimerkiksi BUBBLE [67] käyttää moniulotteista skaalausta (katso luku 6.3.3) klusteroinnin aikana. Pelkästään projektiota käyttävät menetelmät eivät kuitenkaan yleensä sovi dokumenttien klusterointiin, koska projisoitujen dokumenttien suhteet toisiinsa vääristyvät liikaa [125, sivu 16]. Empiirisen osuuden kannalta tärkeitä metrisissä avaruuksissa toimivia klusterointialgoritmeja käsitellään tarkemmin luvussa 6.3.2.

Graafit (verkot) ovat selkeä tapa hahmottaa metrisessä avaruudessa olevia olioita. Näytteistä muodostetaan aluksi täydellinen painotettu verkko, jossa kaarien painot kuvaavat etäisyyksiä. Tämän jälkeen graafi ositetaan, minkä tuloksena saadut yhtenäiset verkon komponentit ovat klustereita. Vanha perusmenetelmä graafin ositukseen on laskea sen pienin virittävä puu,

jonka jälkeen poistetaan yksi kerrallaan pisimmät kaaret. Tuloksena on jakava hierarkkinen klusterointialgoritmi, joka on ekvivalentti lähimmän pisteen hierarkkisen klusteroinnin kanssa, joskin klusterien muodostamisjärjestys on käänteinen [46, sivut 566-567]. Boleyn *et al.* [16] assosiaatiosääntöjen louhintaan perustuva ARHP-algoritmi (*Association Rule Hypergraph Partitioning*) mallintaa näytejoukon hypergraafeina. Erona tavallisiin graafeihin *hyperkaari* voi yhdistyä useampaan kuin kahteen solmuun. Toinen algoritmin erikoisuus on, että assosiaatiosääntöihin perustuvasta tekniikasta johtuen se ei tarvitse etäisyysmittaa. Algoritmi etsii sanamatriisista dokumenttijoukkoja, joilla on riittävän paljon yhteisiä termejä. Nämä yhteiset dokumenttijoukot ovat graafin hyperkaaria. Klusterit saadaan hypergraafin osituksena, osituskriteerinä on kaarien leikkauksen minimointi (katso luku 4.3.4).

Graafeja voidaan käyttää metrisen avaruuden hahmotuksen lisäksi yksittäisten näytteiden kuvaamiseen. Erityisesti rakenteiset dokumentit vaativat tietyillä indeksirakenteilla puu- tai graafiesityksen. Tämä on numeerista vektoria ilmaisuvoimaisempaa ja huomioi sisäkkäiset rakenteet ja hyperlinkit. Dokumenttikohtaisia esitystapoja ja samanlaisuusmittoja käsitellään tarkemmin luvussa 5.3. Puiden ja graafien vertailu on ongelmallista, koska täsmäysongelma on yleisessä muodossaan NP-täydellinen [119]. Tämän takia algoritmit käyttävät heuristiikkoja, jotka pyrkivät löytämään kompromissin vertailun tarkkuuden, ilmaisuvoiman ja laskennallisen tehokkuuden välillä. Yleinen graafien rakenteellisessa vertailussa käytetty samanlaisuusmitta on maksimaaliseen yhteiseen aliverkkoon (*maximal common subgraph*) perustuva MCS-mitta, joka määritellään seuraavasti [118]:

$$sim_{MCS}(A, B) = \frac{|mcs(A, B)|}{\max\{|A|, |B|\}},$$

missä osoittaja on graafien A ja B suurimman yhteisen aliverkon solmujen määrä ja nimittäjä graafien maksimaalinen solmumäärä.

4.3.4 Teoreettinen malli

Teoreettinen malli käsittää klusterointimenetelmän taustateorian, induktiivisen periaatteen ja mahdolliset oletukset. Taustateoria tarjoaa viitekehyksen ja perusterminologian, jonka pohjalta menetelmää voidaan kehittää. Lisäksi taustateoria vaikuttaa klusterien esittämiseen. Tyypillisiä taustateorioita ovat tilastotiede (jolloin klusterointi voidaan tulkita esim. sekatiheysmallin etsinnäksi) ja verkkoteoria (jolloin klusterointi on graafin ositus). Toisaalta monille diskriminatiivisille algoritmeille ei ole helposti osoitettavissa mitään mielekästä taustateoriaa, jolloin algoritmin perustana on vain heuristiikka. Induktiivinen periaate eli klusterointikriteeri [54] määrittelee formaalisti klusteroinnin tavoitteen. Sen avulla voidaan verrata

klusteroinnin vaihtoehtoisten tulosten laatua (tätä voidaan pitää myös sisäisenä validointikriteerinä, katso luku 4.4). Oletukset kuvaavat taustatietoa, jota datajoukosta on saatavilla. Osa oletuksista voidaan kuvata klusterointimenetelmän parametreina, osa seuraa suoraan valitusta menetelmästä. Oletuksia voivat olla esim. klusterien määrä, tieto piirteiden riippumattomuudesta tai vaatimukset klusterien yleiselle muodolle (ympyrämäiset, ellipsoidit vai mielivaltaisen muotoiset) ja esitystavalle (katso luku 4.3.2).

Tilastotiede on tärkein taustateoria mallipohjaisten klusterointimenetelmien kehittämiseen. Tilastotieteellisen näkökulman mukaan klusterit voidaan kuvata jakaumista koostuvina malleina, joista tyypillisin on sekatiheysmalli [46, sivut 518-519]. Tällöin jakaumien yleinen muoto ja lukumäärä yleensä tiedetään, mutta ei parametreja. Sekatiheysmallin yleinen muoto on seuraava:

$$P(x|\Phi) = \sum_{j=1}^{c} P(x|\omega_j, \Phi_j)P(\omega_j),$$

missä Φ on parametrivektori, $P(x|\omega_j, \Phi_j)$ ovat mallin komponenttien tiheysjakaumat ja prioritjakaumia $P(\omega_j)$ sanotaan sekoitusparametreiksi (painottavat yksittäisiä komponentteja). Sekatiheysmalliin perustuva klusterointi on parametrien estimointia, missä keskeiset induktiiviset periaatteet ovat suurimman uskottavuuden (*maximum likelihood*) menetelmä ja bayesilainen parametrien estimointi. Edellisen tavoitteena on etsiä malliin sellaiset parametrit, jotka suurimmalla todennäköisyydellä generoivat havaitun datan. Parametrit oletetaan kiinteiksi mutta tuntemattomiksi. Jälkimmäisessä parametreja estimoidaan satunnaismuuttujina [46, sivut 84-85]. Sekatiheysmalli kuvaa lähinnä tilannetta, jossa klusterimalli on tasainen ja ositus on ”kova”. Luvussa 4.3.2 on mainittu myös monipuolisempiin tilanteisiin soveltuvia malleja, mutta niiden estimointia ei käsitellä tässä tarkemmin.

Klassinen lähestymistapa sekatiheysmallin estimointiin on EM (*Expectation-Maximization*) -algoritmi (oikeastaan pitäisi puhua menetelmästä, koska määritys on varsin yleisellä tasolla ja algoritmilla on useita eri toteutusmahdollisuuksia), joka yleisessä muodossaan määritellään seuraavasti: olkoon Φ estimoitavan mallin P parametrit, y näytedata ja pareille (Φ, Φ') määritelty funktio $Q(\Phi|\Phi') = E(\log P(x|\Phi')|y, \Phi)$. Nyt voidaan määritellä iteraatio $\Phi_{(p)} \rightarrow \Phi_{(p+1)}$:

1. *(E-askel)* Laske $Q(\Phi|\Phi_{(p)})$
2. *(M-askel)* Valitse $\Phi_{(p+1)}$:lle arvo, joka maksimoi $Q(\Phi|\Phi_{(p)})$:n.

Parametrien valinnan tavoite on maksimoida $\log P(x|\Phi)$. Koska funktio ei ole etukäteen tiedossa, maksimoidaan sen tämänhetkinen odotusarvo datan y ja iteroitavien parametriestimaattien $\Phi_{(p)}$ suhteen. EM-algoritmia sovelletaan yleisimmin suurimman uskottavuuden kehyksessä, mutta sen voi laajentaa myös bayesilaiseen parametrien estimointiin [41]. Yleisistä

klusterointialgoritmeista K-means approksimoi EM-algoritmia, jos sekatiheysmalli koostuu normaalijakaumista, tuntemattomat parametrit ovat pelkästään odotusarvoja eivätkä mallin komponentit ole merkittävästi päällekkäin [46, sivu 528].

Klusterointi voidaan tulkita optimointiongelmana, jos klusteroinnin tavoite on määriteltävissä eksplisiittisenä funktiona, joka kertoo mallin ”hyvyyden” [46, sivu 542]. Yleinen klusterointiongelma on NP-täydellinen [125, sivut 7-8], minkä takia optimoinnissa on käytettävä approksimointialgoritmeja (tämä on myös yksi syy siihen, miksi algoritmeja on niin runsaasti). Optimointifunktioita (koneoppimisen alueella käytetään myös nimeä kohdefunktio) on yleisimmin käytetty tasaisten, osittavien algoritmien laadun arvioinnissa ja erityisesti mallipohjaisessa klusteroinnissa. Optimointifunktioita voidaan soveltaa myös hierarkkiseen klusterointiin, mistä Zhao & Karypis [143] näyttävät lukuisia esimerkkejä. Optimointifunktiot ovat eksplisiittinen esitys Estivill-Castron induktiivisesta periaatteesta, jollainen voidaan löytää miltä tahansa klusterointialgoritmilta. Tosin diskriminatiivisilla algoritmeilla tämä periaate voi olla hankala määritellä eksplisiittisesti – erityisesti tiedonlouhintaan keskittyvässä kirjallisuudessa validointikysymykset on muutenkin jätetty vähemmälle huomiolle. Tämä on ristiriitaista, koska menetelmä, jossa klusteroinnin tavoitetta ei ole määritelty tarkasti, saattaa johtaa (vertaa KDD-prosessin määritelmään luvussa 2.2.1) *triviaaliin* prosessiin, joka tuottaa datasta uusia, ymmärrettäviä ja *epäpäteviä* malleja [54].

Optimoinnilla muodostetun klusterimallin laatua voidaan arvioida kustannus- eli kriteerifunktioilla, jotka Zhao & Karypis [143] jakavat niiden vaikutusalueen mukaan seuraavasti:

- **Sisäiset** (*internal*) funktiot, jotka keskittyvät yksittäisten klusterien sisällä oleviin näytteisiin. Yksinkertaisin ja yleisimmin käytettävä kustannusfunktio on ns. neliövirhekriteeri, jonka tarkoituksena on minimoida etäisyyksien neliöt klusterin keskipisteestä. Mm. K-means pyrkii minimoimaan tämän funktion [54]. Neliövirhekriteeri määritetään euklidista etäisyyttä käytettäessä seuraavasti [46, sivu 542]:

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \|x - m_i\|^2,$$

  missä c on klusterien määrä, m_i on klusterin D_i keskiarvopiste $\frac{1}{n_i} \sum_{x \in D_i} x$, ja n_i on näytteiden määrä klusterissa. Kriteeri kuvaa etäisyyksien neliösummia klusterien keskiarvopisteistä.

- **Ulkoiset** (*external*) funktiot, joiden painopisteenä on klusterien erottaminen toisistaan. Esimerkki ulkoisesta kustannusfunktiosta on klusterien välisen hajontamatriisin (*scatter matrix*) jäljen (*trace*) maksimointi. Hajontamatriisi S_b voidaan muodostaa seuraavasti [46, sivut 544-545]:

$$S_b = \sum_{i=i}^{c} n_i (m_i - m)(m_i - m)^T,$$

missä muuttujat ovat kuten edellä neliövirhekriteerissä, m on koko datajoukon keskiarvopiste. Matriisin jälki on sen diagonaalilla olevien arvojen summa. S_b:n jälki saadaan euklidista etäisyyttä käytettäessä myös funktiolla $tr(S_b) = \sum_{i=i}^{c} n_i \|m_i - m\|^2$.

- **Hybridit** funktiot yhdistävät useiden kriteereiden ominaisuuksia. Dunnin indeksi (katso luku 4.4.2) on esimerkki hybridistä kriteeristä, koska se pyrkii samanaikaisesti minimoimaan klusterien sisäiset ja maksimoimaan klusterien väliset etäisyydet.

Zhao & Karypis lisäävät omaksi ryhmäkseen myös graafipohjaiset funktiot, mutta tämän kirjoittajan mielestä ne voidaan ryhmitellä yhtä lailla osaksi edellä mainittuja kriteerejä (joskin niiden taustateoria on tilastotieteen sijaan verkkoteoria). Esimerkki graafipohjaisesta funktiosta on kaarien leikkaus (*edge cut*), joka on ositetun graafin kaarien painojen summan suhde täydellisen verkon yhteenlaskettuun kaarien painoihin [125, sivu 104]. Huomattavaa on, että kaikki edellä mainitut kustannusfunktiot kuuluvat validoinnin kannalta *sisäisiin* kriteereihin. Ulkoinen kustannusfunktio ei siis ole sama asia kuin ulkoinen validointikriteeri. Kustannusfunktiot eivät ole täysin sama asia kuin optimointifunktiot, koska kustannusfunktioihin sisältyy oletuksena tietyllä tavalla muodostettu klusterimalli. Kustannusfunktion pohjalta pystyttäisiin muodostamaan myös optimointifunktio, jos D_i ja c lisätään estimoitaviin muuttujiin (ja huomioidaan m_i:n riippuvuus näistä).

Vaihtoehtoisia taustateorioita, joiden pohjalta klusterointialgoritmeja voidaan muodostaa, ovat mm. [83] sumea logiikka, neuroverkot, geneettiset algoritmit sekä [10] signaalinkäsittelyyn kuuluvat fourier- ja aallokeanalyysi. Kategorista dataa varten on olemassa lisäksi erityisiä käsitteellisiä (*conceptual*) klusterointimenetelmiä [75], joissa voidaan hyödyntää esim. relaatiotiedonlouhintaa. Neuroverkkojen ja muiden erikoismenetelmien käsittely sivuutetaan kuitenkin tässä, koska niiden analyysista saisi tutkielman itsessään.

4.4 Klusteroinnin laatu

Klusterointialgoritmien laatua voidaan arvioida seuraavien yleisten vaatimusten [76, sivut 335-338] pohjalta:

1. **Tehokkuus** ja **skaalautuvuus**. Tiedonlouhinnassa käsitellään suuria tietomassoja, joten tämä on ilmeinen vaatimus. Esimerkiksi hakutulosten klusterointi on suoritettava jopa reaaliaikaisesti.

2. **Erilaiset attribuuttityypit**. Useimmat klusterointimenetelmät on suunniteltu vektorimuotoista dataa silmälläpitäen, mutta käytännön sovelluksissa tarvitaan myös muita tietotyyppejä (katso luku 4.3.3).

3. **Klustereiden muoto**. Keinotekoiset testijoukot saadaan helposti noudattamaan jotain standardia jakaumaa, mutta näin ei voida olettaa mielivaltaisesta datajoukosta. Lisäksi klusterit saattavat olla mielivaltaisen muotoisia tai sisäkkäisiä.

4. **Riippumattomuus taustatiedosta**. Käyttäjältä ei pidä vaatia sovellusaluekohtaista tietoa (esim. klustereiden määrää tai etäisyystietoa). Jos taustatietoa tarvitaan, tulosten täytyy olla ainakin selkeästi validoitavissa.

5. **Robustius**. Havaintoaineisto voi sisältää poikkeamia, jotka vääristävät klustereita. Tämä on huomioitava varsinkin tietokannoissa, joita ei ole suunniteltu data-analyysia varten (katso luku 2.1.2). Toisaalta poikkeamat saattavat olla myös etsittäviä hahmoja.

6. **Havaintojen järjestys**. Algoritmin tulos ei saisi riippua syötealkioiden järjestyksestä. Tällä on erityisesti merkitystä, jos klusterointialgoritmi on inkrementaalinen.

7. **Moniulotteisuuden käsittely**. Erityisesti tekstidokumentteja käsiteltäessä datan dimensio voi nousta tuhansiin ulottuvuuksiin, vaikka luotettavien läheisyysmittojen kannalta ulottuvuuksien määrän ei pitäisi nousta yli 15:n (katso luku 4.4.1).

8. **Rajoitteiden huomiointi**. Spatiaalista dataa sisältävä aineisto voi sisältää fyysisiä rajoitteita, jotka vaikuttavat klusteroinnin tulokseen. Tämän kirjoittajan tulkinnan mukaan myös rakenteisen hakukyselyn pohjalta tehty klusterointi voidaan tulkita rajoitepohjaisena klusterointina.

9. **Tulkittavuus ja käyttökelpoisuus**. Kuvailevana menetelmänä klusteroinnin edellytetään havainnollistavan datajoukkoa käyttäjälle. Käytännössä tähän vaikuttavat klusterien esittäminen (katso luku 6.2.2) ja visualisointi (katso luku 6.3.3).

Useimmat klusterointialgoritmit eivät täytä kaikkia laatuvaatimuksia. K-means-tyylisten osittavien algoritmien tulokset riippuvat prototyyppien arvotuista aloituspisteistä ja stokastiset toteutukset riippuvat lisäksi alkioiden käsittelyjärjestyksestä. Useimmat algoritmit vaativat taustatietoja sovellusalueesta parametrien muodossa. Erityisesti vanhemmissa menetelmissä ei ole huomioitu moniulotteisuuden vaatimuksia. Hierarkkisia algoritmeja pidetään yleisesti osittavia laadukkaampina, mutta laskennallisesta vaativuudesta johtuen ne eivät skaalaudu suurille tietojoukoille: reaaliaikainen samanlaisuusmatriisin laskenta ei tule kyseeseen $\Theta(n^2)$-kertaluokan aika- ja tilavaativuuden vuoksi. Esikäsittelyvaiheessa matriisin voi laskea kerralla, mutta tällöin sen tehokas käyttö edellyttää säilyttämistä keskusmuistissa. Tätä voidaan kiertää luvussa luvussa 2.2.3 mainitulla tiedonhallintastrategialla, mikä tosin monimut-

kaistaa toteutusta. Koska ei ole olemassa kaikille sovellusalueille sopivaa yleismenetelmää, algoritmin valinnassa on aina otettava huomioon datan esitystapa, etäisyysmitta ja datajoukkoon mahdollisesti sisältyvät oletukset. KDD-prosessin loppuun sijoittuvalla validoinnilla voidaan rajoitetusti verrata eri algoritmien laatua. Nyrkkisääntönä todettakoon, että aluksi kannattaa analysoida sovellusalueen erityispiirteet ja määritellä tarkasti ratkaistava ongelma. Vasta tämän jälkeen on mielekästä valita sopiva klusterointimenetelmä [75].

4.4.1 Moniulotteisuuden käsittely

Tyypillinen tapa dokumentin esittämiseen on kuvata se vektorina, jonka jokainen ulottuvuus vastaa tiettyä indeksitermiä. Tiedonhaun kannalta piirteiden valinta käsittää tavallisesti sulkusanojen poiston ja perusmuotoon palautuksen (katso luku 3.4), mutta tämänkin jälkeen dokumenttivektorit ovat erittäin harvoja ja sisältävät mahdollisesti tuhansia ulottuvuuksia. Vektoriavaruuden avulla voidaan esittää kompaktissa muodossa suurikin joukko muuttujia, mutta dimension kasvaessa klusteroinnin laatu huononee. Tämä on intuitiivinen kuvaus kuuluisalle ”dimensiokiroukselle” (*curse of dimensionality*). Beyer *et al.* [13] ovat osoittaneet teoreettisesti, että jo yli 15-ulotteisessa avaruudessa lähimmän naapurin käsitteestä tulee epästabiili: lähes kaikki pisteet näyttävät olevan yhtä kaukana toisistaan. Tämä ei ole pelkästään vektoriavaruudessa toimivien menetelmien ongelma, vaan liittyy kaikkiin etäisyysmittoihin, jotka lasketaan korkeadimensioisen datan pohjalta huomioimatta sovellusalueen erityisominaisuuksia.

Moniulotteisuuden käsittelyyn on klusteroinnissa käytetty seuraavia perusmenetelmiä [10]:

1. **Ulottuvuuksien vähentäminen** kuuluu KDD-prosessin *muunnosvaiheeseen* (vaihe 3, katso luku 2.2.1) ja se voidaan jakaa piirteiden valintaan ja erottamiseen. Tiedonhaussa käytetty sulkusanojen poisto (katso luku 3.4) on piirteiden valintaa, LSI (katso luku 5.1.1) tai pääkomponenttianalyysi ovat piirteiden erottelua. Kehittyneempi tapa piirteiden valintaan on käyttää jotain informaatioteoreettista mittaa, kuten keskinäisinformaatiota (katso luku 5.1.2). Piirteiden erottelun ongelma on sovellusalueesta riippuen se, etteivät esim. pääkomponenttianalyysin jälkeen muunnetut piirteet ole helposti tulkittavissa [2]. Lisäksi piirteitä saattaa ulottuvuuksien vähentämisenkin jälkeen olla liikaa ”dimensiokirouksen” kannalta (Esim. jos dokumentit on alunperin esitetty parillatuhannella ulottuvuudella, LSI-muunnoksen jälkeenkin ulottuvuuksia voi olla satoja) [125, sivut 48-49].

2. **Aliavaruuksien erottelu** tarkoittaa algoritmin suunnittelua niin, että se huomioi rajattuun piirrejoukkoon (tai ulottuvuuksiin – vektoriavaruuden ollessa kyseessä puhu-

taan yleensä aliavaruuksista) keskittyvät klusterit. Projisoimalla avaruus dimensioltaan pienempiin aliavaruuksiin ”dimensiokirouksen” vaikutus vähenee ja klustereita voidaan löytää luotettavammin. Lisäksi ulottuvuuksilla on sama tulkinta kuin alkuperäisessä avaruudessa, eli piirteiden muuntamisesta aiheutuvia tulkintaongelmia ei ole. CLIQUE-algoritmi [2] on perusesimerkki aliavaruudet huomioivasta klusterointimenetelmästä.

3. **Piirteiden klusterointi**, jonka tarkoituksena on toisaalta ulottuvuuksien vähentäminen piirteiden erotusvaiheessa, toisaalta näytteen selkeämpi kuvaaminen. Jos syötedata on numeerisessa matriisimuodossa, piirteiden klusterointi voidaan periaatteessa suorittaa transponoimalla havaintomatriisi (tosin klusteroitavat vektorit pitää mahdollisesti normalisoida erikseen). Tiedonhaussa tai tekstin kategorisoinnissa piirteiden klusterointia on sovellettu termeihin. Tätä on käsitelty tarkemmin luvussa 4.2. Yhteisklusteroinnissa (*co-clustering*) saadaan yhtenäinen malli, jossa ovat klusteroituna sekä näytteet että piirteet [10]. Jokaisesta näyteklusterista saadaan kompakti esitys piirreklustereina. Yhteisklusterointia voidaan käyttää tiedonhaun erityistekniikkana. Esimerkkinä tällaisesta on Slonimin & Tishbyn algoritmi [124], joka etenee vaiheittain: aluksi klusteroidaan indeksitermit, jonka jälkeen klusteroidaan dokumentit. Algoritmia kuvataan tarkemmin luvussa 5.1.2.

Luokittelussa ulottuvuuksien vähentäminen on klusterointiakin tärkeämpää, koska monet koneoppimisalgoritmit eivät skaalaudu korkeisiin ulottuvuuksiin. Lisäksi ulottuvuuksien vähentäminen vähentää ylioppimisen riskiä luokittimessa [120]. Klusteroinnissa ja tiedonhaussa mahdollinen ratkaisu ”dimensiokiroukseen” on etsiä sopiva (sovellusaluekohtainen) samanlaisuusmitta, joka kuvaa mahdollisimman täsmällisesti datapisteiden keskinäiset erot (siirretään ongelma piirreavaruudesta samanlaisuusavaruuteen) [125, sivut 46-53]. Tekstidokumenttien osalta tällainen mitta lienee dokumenttivektorien välinen kulma (kosinimitta), joka on osoittautunut käytännössä toimivaksi jo vuosikymmenien ajan.

4.4.2 Olemassaolotestit ja validointikriteerit

Jain *et al.* [83] jakavat klusteroinnin laadun arviointimenetelmät klusteroinnin olemassaolotesteihin (*clustering tendency*) ja varsinaisiin validointimenetelmiin, joilla voidaan arvioida klusteroinnin onnistumista algoritmin suorituksen jälkeen. Validointimenetelmien jaottelussa on hieman ristiriitaisuuksia, mutta pääjako voidaan tehdä sisäisten ja ulkoisten arviointikriteerien välillä [125, sivu 102-110]. Sisäinen arviointi käyttää hyväkseen tietoa vain klusteroitavasta datasta, kun taas ulkoinen arviointi hyödyntää jotain ulkopuolista tietoa, kuten ”ideaalista” klusterimallia, johon tulosta verrataan. Toinen näkökulma on Halkidilla *et al.*

[75], jotka sallisivat vain varsinaiset tilastolliset testit kuuluviksi sisäisiin ja ulkoisiin kriteereihin. Esimerkiksi algoritmin ajo parametrien eri arvoilla ja optimointifunktion (katso luku 4.3.4) käyttö validointikriteerinä kuuluisi *suhteellisiin* kriteereihin, vaikka esim. Zhong & Ghosh [144] luokittelevat tämän selvästi sisäisiin kriteereihin. Toisaalta Jain *et al.* määrittelevät suhteellisten kriteerien olevan jonkinlaisia sisäisten ja ulkoisten kriteerien pohjalta muodostettuja yhdistelmämittoja. Tässä käsiteltävät validointimenetelmät on selkeyden vuoksi jaoteltu vain ulkoisiin ja sisäisiin menetelmiin. Klusteroinnin olemassaolotestit käsitellään erillään. Tässä mainittujen testien lisäksi mm. Berkhin [10] ja Halkidi *et al.* käyvät läpi muita validointikriteerejä.

Keskeisen tiedonlouhintaa kritisoivan argumentin mukaan mistä tahansa tietojoukosta voidaan tietyllä todennäköisyydellä löytää säännönmukaisuuksia, vaikka niitä ei datassa todellisuudessa olisi (katso luku 2.2.4). Klusteroinnin osalta tähän pyritään vastaamaan olemassaolotesteillä. Jos tutkittavassa datassa ei näy rakennetta, sitä ei ole mielekästä edes yrittää klusteroida. Tiedonhaulle ominainen olemassaolotesti on luvussa 4.2 mainittu klusterointihypoteesin testaus, joka tosin vaatii etukäteen relevanttien dokumenttien erottelun kokoelmasta. Toinen mahdollinen testi perustuu satunnaisgraafien teoriaan. Dokumenttikokoelma mallinnetaan graafiksi lähimmän pisteen hierarkkista klusterointia käyttäen (huom. varsinainen klusterointi voidaan tehdä muullakin menetelmällä – tässä klusterointia käytetään vain joukon satunnaisuuden selvittämiseen). Mallin satunnaisuutta tutkitaan jollakin tilastollisella testillä, jossa nollahypoteesina on satunnainen graafi. Testikriteerinä voidaan käyttää esim. pienintä kaarien määrää, joka yhtenäistää graafin [136]. Esim. Shaw'n *et al.* [123] klusteripohjaisen haun kritiikissä käytettiin satunnaisgraafipohjaista arviointia. Testin ongelmana voidaan pitää riippuvuutta lähimmän pisteen menetelmästä. Muihinkin olemassaolotesteihin liittyy väistämättä oletuksia – mikä tekee havaintojoukosta satunnaisen? Yleispätevien olemassaolotestien kehittäminen on siis vaikeaa.

Ulkoisia validointikriteerejä käytettäessä oletetaan, että on saatavilla jotain datajoukosta riippumatonta lisätietoa, jota voidaan hyödyntää klusteroinnin arvioinnissa. Viime kädessä ulkoiset kriteerit ovat luokitteluvirheen arviointia, jolloin klusterointi on automaattista luokittelua. Ulkoisten kriteerien ongelma erityisesti dokumenttien klusteroinnissa on, että klusteroinnin tulos liittyy tiedon käyttökelpoisuuteen käyttäjälle. Tämän seurauksena tiedonhaun validointi on lähtökohtaisesti subjektiivista [77, sivu 452]. Toisaalta ulkoiset kriteerit mahdollistavat erilaisten klusterointialgoritmien yhteismitallisen vertailun, mikä ei ole mielekästä sisäisiä kriteerejä käytettäessä [125, sivut 103-110].

Yksinkertainen luokittelun laatua kuvaava ulkoinen validointimitta on Randin indeksi, joka voidaan määritellä seuraavasti [75]: Olkoon $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ klusterointialgoritmin tuloksena saatu ositus ja $\mathbf{P} = \{P_1, P_2, \ldots, P_s\}$ ulkoisen kriteerin perusteella tiedossa oleva

”todellinen” ositus. Tarkasteltaessa nyt kaikkia datajoukon X alkioista muodostettuja pisteparej a saadaan seuraavat joukot:

- $\mathbf{R}_1 = \{a, b \in X | a \in C_i, b \in C_j, a \in P_k, b \in P_l, i = j, k = l\}$
- $\mathbf{R}_2 = \{a, b \in X | a \in C_i, b \in C_j, a \in P_k, b \in P_l, i = j, k \neq l\}$
- $\mathbf{R}_3 = \{a, b \in X | a \in C_i, b \in C_j, a \in P_k, b \in P_l, i \neq j, k = l\}$
- $\mathbf{R}_4 = \{a, b \in X | a \in C_i, b \in C_j, a \in P_k, b \in P_l, i \neq j, k \neq l\}$

Randin indeksi on $\frac{|\mathbf{R}_1|+|\mathbf{R}_4|}{|\mathbf{R}_1|+|\mathbf{R}_2|+|\mathbf{R}_3|+|\mathbf{R}_4|}$, mikä intuitiivisesti on klusteroinnin ja testijoukon samoin luokittelemien pisteparien määrä jaettuna pisteparien kokonaismäärällä [118].

Tiedonhaussa usein käytetty haun laadun mitta on tarkkuudesta ja saannista (katso luku 3.1.1) johdettu F-mitta, jonka esitti alunperin van Rijsbergen [129, 119-135]. Yleisesti F-mitta määritellään seuraavasti [140, sivu 97]:

$$F_\beta = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R},$$

missä P on tarkkuus, R saanti ja β parametri, joka kuvaa tarkkuuden ja saannin suhteellista tärkeyttä. F_1-mitta tunnetaan myös Dicen kertoimena ja se voidaan tulkita tarkkuuden ja saannin harmonisena keskiarvona [6, sivu 82]. Mittaa voidaan soveltaa suoraan myös klusterointiin, kunhan tarkkuutta ja saantia tarkastellaan klusterien tasolla yksittäisten dokumenttien sijaan [125, sivu 109].

Useimmat klusterointimenetelmät vaativat parametreja, jotka vaikuttavat muodostettavien klusterien määrään joko eksplisiittisesti (esim. K-meansin K-parametri) tai epäsuorasti (esim. DBSCAN-algoritmin ϵ ja $minpts$-parametrit). Kokoavat hierarkkiset menetelmät eivät itsessään vaadi parametreja, mutta klusterien esittäminen käyttäjälle saattaa vaatia jonkin lopetuskriteerin. Tyypillinen (ja hyvin epätehokas) tapa parametrien estimointiin on ajaa klusterointialgoritmia moneen kertaan eri parametreilla ja valita malli, joka tuottaa parhaan tuloksen. Tähän voidaan käyttää luvussa 4.3.4 mainittuja optimointifunktioita tai yleisempiä mallin- ja parametrien valintakriteerejä. Yleisesti kyse on sisäisistä validointimenetelmistä. Itse asiassa on perusteltua kysyä, onko mallipohjaisten klusterointialgoritmien yhteydessä edes mielekästä puhua erikseen validoinnista ja optimointifunktiosta – miksi ei käytettäisi validointikriteerinä suoraan maksimoitavaa optimointifunktiota [54]. Tässä käsiteltävien kriteerien perusteena on, että kaikkia klusterointialgoritmeja ei voida validoida optimointifunktioita käyttäen eikä ulkoisia luokittelukriteerejä ole aina mahdollista käyttää. Tämä pätee erityisesti hakutulosten klusterointiin.

Yksinkertainen ja kaikille klusterointialgoritmeille soveltuva sisäinen validointimitta on ”pahinta tapausta” mittaava Dunnin indeksi. Mitta formalisoi luvun alussa mainitun intuitiivisen tavoitteen minimoida klusterien sisäiset ja maksimoida klusterien väliset etäisyydet [75]:

$$D = \frac{\min_{i,j \in C}\{d_{ext}(C_i, C_j)\}}{\max_{k \in C}\{\max_{x,y \in C_k}\{d_{int}(x, y)\}\}},$$

missä osoittaja on pienin mahdollinen klusterien välinen etäisyys d_{ext} ja nimittäjä suurin mahdollinen klusterien sisällä oleva pisteiden välinen etäisyys d_{int}. C on klusterien joukko.

Klusterien määrän estimointi on olennaisesti parametrien valintaa, jolla on tärkeä merkitys hahmontunnistuksessa ja koneoppimisessa yleisemminkin. Yleisimmässä muodossaan parametrin (tai jopa mallin) valinta voidaan muotoilla J. Rissasen kehittämän MDL (*Minimum Description Length*) -periaatteen muodossa. MDL-periaate pyrkii löytämään datalle informaatioteoreettisesti mahdollisimman tiiviin koodauksen. Klusterointiin sovellettuna tämä merkitsee, että samaan klusteriin kuuluvat alkiot tulisi pystyä pakkaamaan mahdollisimman tehokkaasti [92]. Luvussa 5.1.2 esitellään klusterointimenetelmä, joka hyödyntää MDL-tyyppistä ”universaalia etäisyysmittaa”. MDL on käsitteellisesti lähellä laskennan teoriasta tunnettua Kolmogorov-kompleksisuutta K. $K(x)$ on lyhimmän sellaisen tietokoneohjelman pituus, joka tulostaa x:n ja lopettaa. Olennaisesti K kuvaa, kuinka paljon x on algoritmisesti pakattavissa. Kummatkin ovat edelleen tulkittavissa formalisoituina versioina klassisesta Occamin partaveitsestä [30, sivut 168-182].

5 Tekstitiedon analysointitekniikat

Luvussa on kuvattu kaikki keskeiset dokumenttien klusterointiin ja tiedonhakuun liittyvät tekniikat, jotka tutkimuksen edetessä on löydetty. Niitä käsitellään yhdessä, koska empiirisen osuuden sovelluksessa haku ja klusterointi on integroitu samaan hakuprosessiin. Molemmat käyttävät samaa indeksirakennetta. Käsittelyn kohteena ovat tekstitieto, linkkitieto ja dokumentin rakenne, jotka ovat eri näkökulmia dokumenttiin. Käytännössä näkökulmia ei voida erottaa kokonaan toisistaan, vaan parhaan hakutuloksen saamiseksi niitä täytyy yhdistää ja painottaa hakutehtävästä ja sovellusalueesta riippuen.

5.1 Tekstianalyysi

Luvussa käsitellään rakenteettomasta tekstistä koostuvien dokumenttien esittämistä sekä tiedonhaun että klusteroinnin näkökulmasta. XML-dokumentteja voidaan käsitellä tekstinä joko sellaisenaan (jolloin elementit ja attribuutit sekoittuvat sisältöön) tai elementtitunnisteiden poiston jälkeen (jolloin kaikki rakennetieto häviää). Rakenteisten dokumenttien kannalta puhdas teksti on rajoittunut esitystapa, koska se ”latteistaa” dokumentin. Vastaavasti tiedonlouhinnassa käytetty datamatriisi voi olla monen toisistaan jo riippuvan tietokantataulun ”latteistettu” liitos. Rakenteisilla dokumenteilla indeksirakenne on yleensä tekstidokumentteja monimutkaisempi. Indeksoinnissa voidaan käyttää hyväksi esim. termien sijaintitiheyksiä dokumenttipuun eri osissa, mutta paikallisesti (dokumenttipuun lehdissä) rakenteiset dokumentit ovat oleellisesti tekstidokumentteja.

Merkittävin tekstidokumenttien esitystapa sekä tiedonhaussa että klusteroinnissa on vektorimalli. Myös vaihtoehtoisia tapoja dokumenttien laskennalliseen mallintamiseen ja klusterointiin käsitellään lyhyesti. Dokumenttien mallintamisen teoreettisena perustana voidaan käyttää mm. tilastollisia malleja, verkkoteoriaa tai informaatioteoriaa. Käsittelyn ulkopuolelle jätetään varsinaiset luonnollisen kielen käsittelyn menetelmät, kuten mm. Aunimon [5] tarkastelema tekstin semanttisten suhteiden analyysi.

5.1.1 Vektorimalliin perustuvat menetelmät

Vektorimalli on Saltonin *et al.* [116] kehittämä tiedonhakumalli, joka perustuu dokumenttien ja kyselyjen esittämiseen matemaattisina vektoreina. Malli on suosittu erityisesti webhakukoneissa, koska se on helppo toteuttaa, nopea käyttää ja mahdollistaa hakutulosten järjestämisen sekä osittaiset täsmäykset. Mallia on kritisoitu mm. täsmällisen teoreettisen perustan puutteen vuoksi ja siksi, että dokumenttien termit oletetaan toisistaan riippumattomiksi [6, sivut 27-30]. Yksinkertaisuudestaan huolimatta vektorimalli on kuitenkin osoittautunut käyttökelpoiseksi käytännön sovelluksissa.

Formaalisti malli voidaan määritellä seuraavasti: Olkoon $\mathbf{T} = \{1..n\}$ termien indeksijoukko ja $\mathbf{D} = \{1..m\}$ dokumenttien indeksijoukko. Looginen näkymä dokumenttikokoelmaan on sanamatriisi $W_{n\times m}$, jonka sarakkeet

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})^T \in \mathbb{R}^n,\ j \in \mathbf{D},$$

ovat dokumenttivektoreita ja rivit

$$t_i = (w_{i1}, w_{i2}, \ldots, w_{im}) \in \mathbb{R}^m,\ i \in \mathbf{T},$$

edustavat tietyn termin painoja dokumenteissa. Matriisin elementti w_{ij} on siis termin i paino dokumentissa j. Dokumenttien tapaan myös kysely esitetään painotettuna vektorina q [115]. Jos sanamatriisia tarkasteltaisiin relaatiotietokannan tauluna tai tilastollisena aineistona, sen rivit ja sarakkeet esitettäisiin tavallisesti transponoidusti (W^T). Tällöin havainnot olisivat riveillä ja muuttujat sarakkeilla. Ensiksi kuvattu tapa on kuitenkin tiedonhaun kirjallisuudessa yleisemmin käytetty, joten sitä noudatetaan tässä.

Painot merkitsevät termin suhteellista tärkeyttä dokumentissa (tai kyselyssä) ja ne normalisoidaan yleensä välille $[0, 1]$ siten, että painolla 0 termi ei esiinny dokumentissa lainkaan ja 1 tarkoittaa mahdollisimman hyvin tiettyä dokumenttia kuvaavaa termiä. Painotus on yleistys vanhasta Boolen hakumallista, jossa 0 ja 1 ovat ainoat mahdolliset painot [6, sivut 25-27]. Termien painotukseen on useita eri tapoja, joista yleisimmät pohjautuvat termifrekvenssiin (*term frequency, tf*) ja käänteiseen dokumenttifrekvenssiin (*inverted document frequency, idf*). Näissä $tf \times idf$-malleissa sanamatriisin arvot määritellään kaavalla $w_{ij} = tf_{ij}idf_i$ [22, sivut 56-57]. Termifrekvenssi on

$$tf_{ij} = \frac{n(i,j)}{\max_{t\in\mathbf{T}}\{n(t,j)\}},$$

missä $n(t, j)$ on termin t esiintymien määrä dokumentissa j. Käänteinen dokumenttifrekvenssi termille i on

$$idf_i = log\frac{|\mathbf{D}|}{|\mathbf{D}_i|}, \mathbf{D}_i = \{j \in \mathbf{D} | n(i, j) > 0\},$$

missä $|\mathbf{D}_i|$ on niiden dokumenttien lukumäärä, joissa termi i esiintyy. idf:ssä käytetään logaritmia, koska se vähentää mitan riippuvuutta dokumenttien kokonaismäärästä $|\mathbf{D}|$ [77, sivu 463].

Frekvenssien tarkat määritelmät vaihtelevat kirjallisuudessa jonkin verran painotustavasta riippuen. Lisäksi kyselyvektorissa voidaan käyttää dokumenttivektoriin nähden toisenlaista painotustapaa tai sitä ei painoteta ollenkaan. Salton & Buckley [115] ovat vertailleet perusteellisesti erilaisia $tf \times idf$-tyylisiä painotustapoja. Vaihtoehtoinen lähestymistapa on käyttää painotukseen esim. termien entropiaa [47]. Tarkasta kaavasta riippumatta termifrekvenssin perusideana on, että usein tietyssä dokumentissa olevat termit kuvaavat kyseistä dokumenttia hyvin. Käänteisen dokumenttifrekvenssin idea on, että dokumentit voidaan erotella toisistaan parhaiten termeillä, jotka esiintyvät vain pienessä määrässä dokumentteja. Keskeinen indeksirakenne $tf \times idf$-painotuksen ja sanamatriisin toteutukseen on käänteistiedosto (*inverted file*), joka liittää jokaiseen kokoelman termiin listan dokumenteista, joissa se esiintyy [6, sivut 192-195].

Hakutulokset täsmäytetään vertaamalla kyselyvektoria dokumenttivektoreihin. Yleisimmin käytettävä vertailufunktio on vektorien pituuden suhteen normalisoitu sisätulo, joka on samalla vektorien d_j ja q välisen kulman kosini. Tästä syystä samanlaisuusmittaa kutsutaan usein kosinimitaksi.

$$sim_{cos}(d_j, q) = \frac{(d_j|q)}{\|d_j\|\|q\|} = \frac{\sum_{i=1}^{n} w_{ij}q_i}{\sqrt{(\sum_{i=1}^{n} w_{ij}^2)(\sum_{i=1}^{n} q_i^2)}}.$$

Muita mahdollisuuksia on käyttää vertailuun esim. euklidista etäisyyttä, laajennettua Jaccardin mittaa, Dicen kerrointa tai Pearsonin korrelaatiota [125, sivut 92-99]. Dokumenttien ja kyselyjen vertailun lisäksi dokumentteja voidaan verrata myös keskenään, mikä mahdollistaa niiden klusteroinnin. Etäisyysmitan valinta ei vaikuta merkitsevästi klusterointiin, kunhan dokumenttien pituudet normalisoidaan [129, 24-28]. Klusteroinnissa voidaan käyttää samoja termien painotuksia kuin tiedonhaussa, mutta vain yksittäisissä dokumenteissa esiintyvistä termeistä ei ole hyötyä vertailussa. Tiedonhaun kannalta niillä on kuitenkin maksimaalinen kuvausarvo. Eri painotustavoilla ei ole merkitsevää eroa klusteroinnin laatuun [136].

Vektorimallissa indeksitermejä esittävät vektorit ovat ortogonaalisia, mikä tekee niistä lineaarisesti riippumattomia. Yleistetyssä vektorimallissa [138] täsmäytyksessä huomioidaan indeksitermien väliset riippuvuudet (korrelaatiot, yhteisesiintymät eri dokumenteissa) ja esi-

tetään menetelmä niiden estimointiin. Riippuvuuksista saadaan symmetrinen matriisi $T_{n\times n}$, jonka arvot vaihtelevat välillä $[-1, 1]$ siten, että arvolla 0 verrattavat termivektorit ovat ortogonaaliset. Huomattavaa on, että termit voivat olla toisistaan lineaarisesti riippumattomia, vaikka ne eivät olisi ortogonaalisia. Ortogonaalisuus edellyttää, että vektorien sisätulo on 0 (eli niiden välillä on suora kulma), mutta riippumattomuuteen riittää, ettei mitään termivektoria voida muodostaa toisten vektorien lineaarisena kombinaationa. Täsmäytyksessä käytettävää samanlaisuusmittaa voidaan laajentaa niin, että riippuvuusmatriisin arvot otetaan huomioon. Jos oletetaan, että vektorit on normalisoitu, saadaan dokumentin ja kyselyn välille seuraava samanlaisuusmitta (jos riippuvuusmatriisi oletetaan identiteetiksi, saadaan kaavasta erikoistapauksena vektorimallissa käytetty sisätulo):

$$sim_{gvsm}(d_j, q) = \sum_{i,k=1}^{n} w_{ij} q_k t_i t_k.$$

Latentti semanttinen indeksointi (*Latent Semantic Indexing, LSI*) on tiedonhakumalli, jonka tarkoituksena on hyödyntää dokumenttien ja termien välillä olevaa implisiittistä semanttista (latenttia) rakennetta. Yleistetyn vektorimallin tapaan tavoitteena on löytää relevantteja dokumentteja, jotka sisältävät käyttäjälle tuntemattomia termejä. Tämä on erityisen tärkeää haettaessa tietoja monikielisestä dokumenttijoukosta [84]. Menetelmän kehittäjien [40] mukaan semanttinen rakenne voidaan löytää diagonalisoimalla sanamatriisi, eli etsimällä sen singulaariarvohajotelma. Diagonalisointi on lineaarialgebran perustekniikka, jonka avulla sanamatriisi W voidaan hajottaa osiin siten, että seuraava yhtälö pätee [22, sivu 97]:

$$W_{n\times m} = U_{n\times r} \Sigma V^T_{r\times m},$$

missä r on semanttisen avaruuden ulottuvuuksien määrä, U ja V ovat kannanvaihtomatriiseja, joille pätee $U^T U = V^T V = \mathbb{I}$. Σ on matriisi, jonka diagonaalilla ovat W:n ominaisarvot laskevassa järjestyksessä. Täsmäytyksessä dokumentit ja kyselyt muunnetaan semanttiseen avaruuteen, jonka jälkeen niitä voidaan verrata kuten vektorimallissa. Vain k suurinta ominaisarvoa huomioidaan, jolloin mallin dimensio pienenee. Tämä voidaan tulkita niin, että sanamatriisista poistetaan ”kohinaa” [22, sivu 98]. Nimestään huolimatta LSI:ssä ei ole mitään semanttista, vaan kyseessä on lineaarialgebran perusmenetelmän soveltaminen ilman vahvoja teoreettisia perusteluja. LSI vastaa oleellisesti tilastollisessa hahmontunnistuksessa ulottuvuuksien vähentämiseen käytettyä pääkomponenttianalyysia. LSI vähentää sanamatriisin dimensiota projisoimalla dokumenttivektorit lähimpään k-ulotteiseen lineaariseen aliavaruuteen, pääkomponenttianalyysi projisoi ne lähimpään affiiniin aliavaruuteen (origo siirtyy dokumenttivektorien keskiarvoon) [127, sivut 7-9].

Baeza-Yates & Ribeiro-Neto [6, sivut 41-45] luokittelevat yleistetyn vektorimallin ja LSI:n omiksi tiedonhakumalleikseen, mutta tiedonlouhinnan näkökulmasta ne ovat vain erilaisia tapoja piirteiden valintaan ja muuntamiseen. Täsmäytysfunktio ja dokumenttien vertailu perustuvat kaikissa malleissa dokumentti- ja kyselyvektoreihin. Jiang & Littman [84] esittävät yleisen mallin vektoripohjaisten hakumallien kuvaamiseen, joka kattaa klassisen ja yleistetyn vektorimallin sekä LSI:n. Jos oletetaan, että vektorit on normalisoitu ja vertailufunktiona käytetään sisätuloa, saadaan seuraava yleinen samanlaisuusmitta:

$$sim_{JL}(d_j, q) = ((Pd_j)|(Pq)),$$

missä P on muunnosmatriisi. Klassisessa vektorimallissa P on identtinen matriisi, yleistetyssä mallissa termien riippuvuuksista johdettu muunnosmatriisi T ja LSI:ssa ominaisarvohajotelmasta johdettu $\mathbb{I}_k U^T$. Tässä $\mathbb{I}_k$ on identiteetti, jossa vain diagonaalin k ensimmäistä arvoa ovat ykkösiä. U on edellä esitetyn singulaariarvohajotelman kannanvaihtomatriisi.

Vektorimallilla esitettyjen dokumenttien klusterointi on yksinkertaista, koska piirteet esitetään vektoreina ja samanlaisuusmitalla on selkeä geometrinen tulkinta. Klusterointi voidaan suorittaa esim. K-meansin kaltaisella perustekniikalla valittu samanlaisuusmitta huomioiden. Perinteisesti kokoavaa hierarkkista klusterointia on pidetty osittavaa laadukkaampana (lukuunottamatta ketjuuntumista aiheuttavaa lähimmän pisteen menetelmää) [136], joskin samanlaisuusmatriisin laskenta heikentää algoritmin sovellettavuutta suurilla tietojoukoilla (katso luku 4.4). Riippuvuus pelkästä etäisyysmitasta mahdollistaa vektorimuotoa rikkaammat dokumentin esitystavat, mikä on välttämätöntä rakenteisuuden hyödyntämiseksi. Samanlaisuusmitan täytyy tällöin sopeutua erilaisiin kyselyihin. Zhao & Karypis [143] ovat myöhemmin vertailleet eri klusterointialgoritmeja ja arvioineet, että myös eräät osittavat algoritmit tuottavat dokumenteille hyviä tuloksia pienemmällä laskennallisella vaativuudella. Ei voida siis osoittaa yhtä ”parasta” klusterointialgorimia dokumenteille.

5.1.2 Muita menetelmiä

Vektorimallin ohella teoreettisesti tärkeäksi, mutta käytännössä harvemmin käytetyksi malliksi ovat nousseet probabilistiset hakumallit. Tarkoituksena on mallintaa *todennäköisyys* kunkin dokumentin relevanssille tiettyä kyselyä vastaan (poiketen esim. vektorimallista, jossa dokumentin relevanssi määrätään suoraan vertaamalla sen esitystä kyselyn esitykseen). Tilastollisten hakumallien pohjalla on oletus, jonka mukaan *dokumentin relevanssi on riippumaton muista kokoelmassa olevista dokumenteista*. Tämän oletuksen pohjalta on muotoiltu tilastollinen järjestysperiaate (*Probability Ranking Principle, PRP*). Sen mukaan *todennä-*

köisen relevanssin mukaan järjestetty tuloslista on optimaalinen edellyttäen, että todennäköisyydet on estimoitu parhaalla mahdollisella tarkkuudella käytettävissä olevista tiedoista. Järjestysperiaate ei ole täysin ongelmaton: sen pohjalla oleva riippumattomuusoletus on täysin päinvastainen klusterointihypoteesin kanssa [129, sivut 87-93], eikä periaate kerro, *miten* todennäköisyydet voitaisiin estimoida [6, sivut 30-34]. Lisäksi periaatteelle on löydetty vastaesimerkkejä käytännön tiedonhakujärjestelmistä, joten oikeastaan pitäisi puhua järjestyshypoteesista [29]. PRP:n eduksi voidaan kuitenkin lukea intuitiivinen tulkinta dokumentin käyttökelpoisuudesta ja mahdollisuus tulkita tiedonhaku yhtenäisessä teoreettisessa kehyksessä.

Tärkeimmät tiedonhakumallit voidaan tulkita tilastollisina, vektorimalli mukaan lukien. Tässä mielessä probabilistiset mallit ovat yleistys vanhemmille hakumalleille [63]. Esimerkkinä yksinkertaisesta tilastollisesta hakumallista käsitellään binäärinen riippumattomuusmalli (*Binary Independence Retrieval, BIR*), joka on tilastollinen tulkinta Boolen hakumallista. Dokumentti d esitetään bittivektorina, jonka kukin komponentti vastaa vektorimallin tapaan yhtä termiä. Määritellään poissulkevat tapahtumat R *dokumentti on relevantti* ja $\overline{R}$ *dokumentti ei ole relevantti*. Nyt Bayesin kaavan perusteella saadaan dokumentin d relevanssin todennäköisyys kyselyn q suhteen kaavalla

$$P(R|d) = \frac{P(d|R)P(R)}{P(d)}$$

ja $\overline{R}$ vastaavasti. $P(R)$ on priori-todennäköisyys mielivaltaisen dokumentin relevanssille ja $P(d) = P(d|R)P(R) + P(d|\overline{R})P(\overline{R})$. Koska malli olettaa termit riippumattomiksi toisistaan, saadaan $P(d|R)$:lle (vast. myös $P(d|\overline{R})$) kaava

$$P(d|R) = \prod_{t \in \mathbf{T}_d} P(t|R) \prod_{t \in (\mathbf{T} \setminus \mathbf{T}_d)} (1 - P(t|R)),\ P(t|R) \in]0, 1[,$$

missä $\mathbf{T}_d$ on niiden termien joukko, jotka esiintyvät dokumentissa d ja $P(t|R)$ on todennäköisyys sille, että termi t esiintyy relevantissa dokumentissa [6, sivut 30-34]. $P(t|R)$:n estimointiin on useita menetelmiä, mutta niitä ei käsitellä tässä tarkemmin. Bayesilaisen päättelyn tuloksena dokumentti d sisällytetään relevantteihin hakutuloksiin, kun epäyhtälö $P(R|d) > P(\overline{R}|d)$ pätee [129, 89-93].

Myös dokumenttien klusterointiin voidaan soveltaa probabilistista (tai mallipohjaista) lähestymistapaa. Tällöin klusterointi määritellään sellaisen satunnaisprosessin ja siihen liittyvien parametrien etsimisenä, joka todennäköisimmin generoi annetun dokumenttikokoelman. Yksinkertaisin malli probabilistiseen klusterointiin on muodostaa malli dokumentissa olevien termien mukaan siten, että termit oletetaan riippumattomiksi toisistaan eikä lukumäärällä

ole väliä. Tällöin saadaan ns. binäärimalli, jossa kunkin klusterin c parametrit Φ_c kuvaavat todennäköisyyksiä kunkin termin t esiintymiselle klusterissa: $\phi_{ct}, t \in \mathbf{T}$, missä $\mathbf{T}$ on termien indeksijoukko. Todennäköisyys dokumentin d generointiin klusterissa c on

$$P(d|\Phi_c) = \prod_{t \in \mathbf{T}_d} \phi_{ct} \prod_{t \in (\mathbf{T} \setminus \mathbf{T}_d)} (1 - \phi_{ct}),\ \phi_{ct} \in]0, 1[,$$

missä $\mathbf{T}_d$ on niiden termien joukko, jotka esiintyvät dokumentissa d [22, sivut 99-114]. Nyt klusterointi palautuu sekatiheysmallin estimoinniksi, joka voidaan ratkaista esim. EM-algoritmilla (katso luku 4.3.4). Klusterien määrä on tiedettävä etukäteen, mutta sitä voidaan estimoida erilaisten validointimenetelmien avulla. Rakenteellinen yhtäläisyys probabilistisiin hakumalleihin on huomattava. Tiedonhaku voitaisiinkin tulkita dokumenttien klusterointina (oikeastaan luokitteluna, mutta mallipohjaisen klusteroinnin ja kahden klusterin ollessa kyseessä ero ei ole oleellinen) relevanttiin ja epärelevanttiin joukkoon kyselyn suhteen. Probabilistisia malleja voidaan laajentaa myös käsittämään termien lukumäärät ja esim. linkkitieto (katso luku 5.4). Lisäksi haku- ja klusterointitoiminnot olisi todennäköisesti mahdollista kuvata yhtenäisellä teoreettisesti perustellulla mallilla. Aiheen laajuuden ja vaativuuden vuoksi sen tarkempi käsittely jätetään kuitenkin tämän tutkielman ulkopuolelle ja hakumalli muodostetaan hyväksi havaittujen heuristiikkojen pohjalta.

N-grammit ovat n:n peräkkäisen kirjaimen tai sanan muodostamia yksiköitä, joita voidaan käyttää indeksoinnissa termien sijaan. Indeksointi suoritetaan liu'uttamalla n:n yksikön mittaista "ikkunaa" dokumentin yli. Koska käytetty aakkosto on äärellinen, kaikkien käytettävissä olevien n-grammien määrä G on laskettavissa. Dokumentti voidaan esittää G-ulotteisena vektorina (käytännössä esim. hajautustauluna), jonka painot saadaan kaavalla

$$x_i = \frac{m_i}{\sum_{j=1}^{G} m_j},$$

missä $\sum_{j=1}^{G} m_j = 1$ ja m_j on j:nnen n-grammin lukumäärä dokumentissa. Tämän jälkeen dokumenteista voidaan hakea tietoja ja niitä voidaan klusteroida kuten vektorimallissa. N-grammipohjaisen indeksointimallin eduiksi termipohjaiseen verrattuna on esitetty kieliriippumattomuutta (esim. sulkusanoja ei tarvitse poistaa esikäsittelyvaiheessa) ja soveltuvuutta kohinaiseen (esim. kirjoitusvirheet) tekstiin [38]. Zamir [141, sivut 98-101] on verrannut haun laatua sanatason n-grammeihin ja lauseista koottuihin suffiksipuihin perustuvilla indekseillä. Tulosten laatu oli samaa tasoa suffiksipuiden kanssa.

Myös informaatioteoriaa voidaan käyttää teoreettisena perustana dokumenttien klusterointiin. Yksittäisten termien esiintymien sijaan voidaan verrata niiden jakaumia. Jokaiselle dokumentille $d \in \mathbf{D}$ ja termille $t \in \mathbf{T}$ voidaan määritellä

$$P(t|d) = \frac{n(t,d)}{\sum_{t \in \mathbf{T}} n(t,d)},$$

missä $n(t,d)$ on termin t esiintymien määrä dokumentissa d (huomaa samankaltaisuus termifrekvenssin määritelmän kanssa). Keskinäisinformaatio (*mutual information*) $I(X;Y)$ on kahden satunnaismuuttujan välillä määriteltävä samanlaisuusmitta, joka kuvaa, paljonko X sisältää Y-muuttujaa koskevaa informaatiota. Keskinäisinformaatio voidaan määritellä seuraavasti [30, sivut 18-19]:

$$I(X;Y) = \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} = \sum_{x \in X, y \in Y} P(x)P(y|x) \log \frac{P(y|x)}{P(y)}.$$

Jakaumiin perustuvaa klusterointia voidaan soveltaa termeihin tai dokumentteihin (tai jopa molempiin samanaikaisesti). Dokumentteja klusteroitaessa pyritään säilyttämään mahdollisimman paljon informaatiota termeistä. Jakaumien keskinäisinformaation tulisi siis pienentyä jokaisella klusterointiaskeleella mahdollisimman vähän. [124]

Keskinäisinformaatio on yleiskäyttöinen samanlaisuusmitta eri jakaumien välillä, mutta se vaatii todennäköisyyksiin perustuvan esitystavan. Yleisin tapa mitata etäisyyksiä dokumenttien (tai minkä tahansa muun tiedon) välillä perustuu Kolmogorov-kompleksisuuteen K. Koska K ei ole algoritmisesti laskettavissa, sitä on approksimoitava sopivalla pakkausohjelmalla C. Cilibrasi & Vitanyi [27] esittävät ”universaalina etäisyysmittana” normalisoidun pakkausetäisyyden

$$d_{NCD}(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}},$$

missä $C(xy)$ on x:n ja y:n katenoidun yhdistelmän pakattu koko ja $C(x)$, $C(y)$ vastaavasti x:n ja y:n pakatut koot. Etäisyysmitta on normalisoitu välille $[0,1]$. Klusteroinnissa ei tarvitse (tai edes voi) huomioida piirteitä, kunhan pakkausohjelma on valittu ”sopivasti” (kohteena olevan datan toisiaan muistuttavien alkioiden pitäisi pakkaantua mahdollisimman tiiviisti). Rakenteisten dokumenttien osalta suurin hyöty saadaan, jos algoritmi ottaa huomioon dokumentin rakenteen eikä pidä elementtejä ja attribuutteja samanarvoisena datana kuin sisältöä. XML-dokumenttien pakkausta on tutkinut mm. Cheney [25].

5.2 Linkkianalyysi

Linkkianalyysia voidaan käsitellä ainakin verkkoteorian, informaatiotutkimuksen, hypertekstijärjestelmien tai WWW:n näkökulmasta. Dokumenttien välisten suhteiden mittaamiseen on kehitetty lukuisia eri mittoja, jotka voidaan jakaa verkkoteoreettisiin, merkittävyyttä mit-

taaviin, samanlaisuusmittoihin, hakumittoihin, käyttöä mittaaviin ja informaatioteoreettisiin [44]. Tässä käsitellään lähinnä samanlaisuus- ja hakumittoja.

Dokumenttikokoelman linkkirakenne esitetään suunnatulla graafilla, joka edelleen voidaan esittää yhteysmatriisina $C_{m\times m}$. Tässä m on dokumenttien lukumäärä ja matriisin alkion $c_{ij} \in \{0, 1\}$ arvo kuvaa, onko dokumentista i linkkiä dokumenttiin j. Matriisin i:nnen rivin arvot kuvaavat siis dokumentista i lähteviä linkkejä ja j:nnen sarakkeen arvot vastaavasti dokumenttiin j tulevia linkkejä.

Luvussa käsitellään haun ja klusteroinnin kannalta keskeisimpiä samanlaisuus- ja merkittävyysmittoja, jotka hyödyntävät pelkästään linkkitietoa tai linkkien selitetekstejä. Sekä tekstiettä linkkitietoa käyttäviä yhdistettyjä hakumenetelmiä käydään läpi luvussa 5.4.

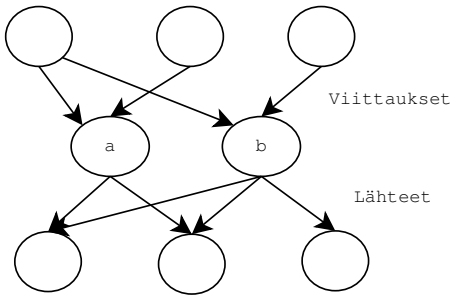
5.2.1 Bibliometriset samanlaisuusmitat

WWW-sivujen ja muiden rakenteisten dokumenttien välisten samanlaisuussuhteiden arviointi pohjautuu jo vuosikymmeniä informaatiotutkimuksen alaan kuuluneeseen bibliometriikkaan (myös yleisempää nimitystä *informetriikka* käytetään). Osareh [109] määrittelee bibliometriikan tarkoituksena olevan tieteellisen dokumentaation, tiedon ja viestinnän parantamisen kirjastojen kokoelmien ja palvelujen määrällisellä analyysilla. Lisäksi hän esittää lukuisia eri ajankohtina annettuja tarkempia määrittelyjä, mutta tämän tutkielman kannalta bibliometriikka on matemaattisten ja tilastollisten menetelmien soveltamista kirjallisuusviitteisiin. Viiteanalyysin avulla voidaan arvioida dokumenttien samanlaisuutta tai tietyn dokumentin vaikuttavuutta (olettaen, että viitettä toiseen dokumenttiin pidetään myönteisenä merkkinä). Suomen kielessä dokumentissa olevaa lähdeviitettä (lähtevää linkkiä) kutsutaan lähteeksi (*reference*) ja dokumentin toisesta dokumentista saamaa viitettä (tulevaa linkkiä) kutsutaan nimellä viittaus (*citation, sitaatio*) [95, sivut 12-13].

Bibliometriikassa käytettyjä klassisia samanlaisuumittoja ovat lähdeanalyysiin kuuluva bibliografinen kytkentä (*bibliographic coupling*) ja viittausanalyysiin kuuluva yhteisviittaus (*cocitation*). Bibliografinen kytkentä dokumenteille a ja b on niiden dokumenttien määrä, joihin molemmat viittaavat. Yhteisviittaus on niiden dokumenttien määrä, jotka viittaavat kumpaankin dokumenttiin. Mitat voidaan normalisoida jakamalla tulevien (tai vastaavasti lähtevien) solmujen kokonaismäärällä. Samanlaisuusmittoja on havainnollistettu kuvassa 5.1.

Normalisoitu kytkentä voidaan laskea linkkimatriisin avulla seuraavasti:

$$sim_{bibc}(a, b) = \frac{\sum_{k=1}^{m} \min\{c_{ka}, c_{kb}\}}{\sum_{k=1}^{m} \max\{c_{ka}, c_{kb}\}}.$$



Kuva 5.1: Esimerkki yhteisviittauksista ja kytkennästä. Dokumenttien a ja b normalisoitu yhteisviittausarvo on $\frac{1}{3}$ ja kytkentä $\frac{2}{3}$.

Yhteisviittaus voidaan esittää vastaavasti:

$$sim_{cocit}(a,b) = \frac{\sum_{k=1}^{m} \min\{c_{ak}, c_{bk}\}}{\sum_{k=1}^{m} \max\{c_{ak}, c_{bk}\}}.$$

Jos lähteiden kokonaismäärä on 0, määritellään kytkennäksi 0. Vastaavasti yhteisviittaus määritellään 0:ksi, jos viitteiden määrä on 0. Bibliometrisia mittoja voidaan hyödyntää haussa tai klusteroinnissa edellyttäen, että kokoelma sisältää kunnollisen linkkirakenteen. Tällöin käyttäjän ei tarvitse tietää sovellusalueen hakutermejä – vaikka ne olisivatkin tiedossa, dokumenteissa olevat termit voivat vaihdella. Linkkitieto on neutraalia termien ilmiasun suhteen [89]. Toisaalta dokumentin laatija päättää enemmän tai vähemmän subjektiivisesti lähteistään, jolloin esim. viittausten määrä omiin julkaisuihin saattaa olla perusteettoman suuri.

Kytkeytyminen ja yhteisviittaukset kuvaavat samaa asiaa eri näkökulmista. Tuntuu luontevalta yleistää mittaa niin, että molemmat suunnat huomioidaan samanlaikaisesti. Kochtanek [89] toteaa, että pelkkien yhteisviittausten käyttö heikentää vanhojen dokumenttien merkitystä ja kytkeytyminen jättää huomiotta uudet samanaiheiset dokumentit. Yhdistetty samanlaisuusmitta vähentää riippuvuutta dokumentin julkaisuajankohdasta. Lisäksi mm. Fox *et al.* [59] toteavat, että tiedonhaun laatu paranee yhdistämällä eri hakumenetelmiä, tässä tapauksessa bibliometrisia mittoja. Toisaalta Savoy [117] huomauttaa, että dokumenttikokoelman kasvaessa yksittäisten dokumenttien kytkeytyminen pysyy samana, mutta yhteisviittausten määrä saattaa kasvaa uusien dokumenttien viitatessa vanhoihin. Tästä syystä yhteisviittaus on pitkällä aikavälillä kytkeytymistä informatiivisempi. Mitoista on esitetty myös useamman askeleen päähän ulottuvia rekursiivisia versioita [44], mutta ne jätetään tämän käsittelyn ulkopuolelle. Huomattakoon, että WWW:n kaltaisessa dynaamisessa ympäristössä myös kytkeytyminen voi muuttua olemassaolevien web-sivujen päivitysten myötä.

Thelwall & Wilkinson [128] ovat tutkineet akateemisten web-sivustojen samanlaisuutta mitaten yhteisviittausten ja kytkeytymisen lisäksi suoria linkkejä. Tarkkuustasona olivat yksit-

täiset web-palvelimet (yliopistojen laitokset). Tuloksissa todettiin, että suorat linkit olivat paras keino samanlaisuuden arviointiin, mutta suurin osa odotetuista yhteyksistä jäi löytämättä linkkien vähäisen määrän vuoksi. Yhteisviittauksista ja kytkeytymisestä oli myös marginaalista hyötyä. Toisessa tutkimuksessa Cristo *et al.* [33] tutkivat etäisyysmittoja web-sivujen luokittelun kannalta ja havaitsivat yhteisviittaukset parhaiten erottelevaksi piirteeksi. Dokumenttikokoelmasta riippuen myös yhdistetyllä mitalla päästiin hyviin tuloksiin. WWW:ssä olevista tieteellisiä artikkeleita indeksoivista hakukoneista ainakin CiteSeer[1] hyödyntää yhteisviittauksia.

5.2.2 WWW:n suosiomittarit

Luultavasti merkittävin yksittäinen web-hakukoneiden laatua parantanut tekijä on ollut linkkitiedon hyödyntäminen relevanssiarvion laskennassa. Tärkeimmät perusmenetelmät ovat Brinin & Pagen esittämä ja Google-hakukoneessa käytetty PageRank [19] sekä Kleinbergin HITS (Hyperlink Induced Topic Search) -algoritmi [88]. Molemmat algoritmit ovat kirjallisuudessa runsaasti viitattuja. Erityisesti HITS-algoritmiin on tehty lukuisia parannuksia ja jatkotutkimusta, jota esim. Chakrabarti [22, 209-242] on koonnut yhteen. Tässä käsitellään tiiviyden vuoksi vain alkuperäiset menetelmät. Teoreettinen perusta linkkialgoritmien toiminnalle on WWW:n hahmottaminen mittakaavattomana verkkona (vastakohtana satunnaiselle verkolle). Tällöin verkossa olevien linkkien määrä ei ole tasaisesti jakautunut dokumenttien kesken, vaan noudattaa potenssilakia. Ilmiö on havaittavissa monissa verkkorakenteissa, kuten vertaisverkoissa [130], Internetin reitittimissä, solukemiallisissa reaktioissa ja eräissä sosiaalisissa verkostoissa [7].

PageRank on globaali approksimaatio tietyn WWW-sivun painoarvolle. Painoarvo määritellään rekursiivisesti sivuun tehtyjen viittausten ja niiden painoarvojen perusteella. Olkoon $\mathbf{I_a}$ sivulle a viittaavien sivujen indeksijoukko ja $d \in [0, 1]$ vaimennuskerroin. PageRank voidaan määritellä sivulle a seuraavasti linkkimatriisin avulla:

$$PR(a) = (1 - d) + d(\sum_{k \in \mathbf{I_a}} \frac{PR(k)}{\sum_{j=1}^{m} c_{kj}}),$$

missä nimittäjässä oleva summa on sivulta k lähtevien linkkien määrä. Intuitiivisesti kaava mallintaa ”satunnaisen surffaajan” todennäköisyyttä päätyä sivulle a. Oletetaan, että surffaajalle annetaan satunnainen sivu, jonka jälkeen hän klikkaa satunnaisesti jotain sivulla olevaa linkkiä ja jatkaa edelleen satunnaisesti, kunnes kyllästyy todennäköisyydellä d. Tällöin hän aloittaa uudestaan satunnaiselta sivulta. PageRank-arvoista muodostettu vektori approksimoi

[1] http://citeseer.ist.psu.edu/

normalisoidun linkkimatriisin ensimmäistä ominaisvektoria [19]. d:n merkitys on kompensoida WWW:n linkkigraafin harvaa rakennetta: kaikki sivut eivät ole yhteydessä toisiinsa [22, sivu 211].

PageRankista poiketen HITS-algoritmin arvot riippuvat käyttäjän syöttämästä kyselystä. HITS-algoritmin tavoitteena on erotella ja arvottaa tulossivut auktoriteetteihin (*authorities*) ja napoihin (*hubs*). Auktoriteetit ovat sivuja, joihin monet navat viittaavat. Navat ovat puolestaan sivuja, jotka sisältävät paljon linkkejä auktoriteetteihin. Algoritmin pohjana on *juurijoukko*, alustavat hakutulokset. Juurijoukkoa laajennetaan sivuilla, jotka viittaavat johonkin juurijoukon sivuun, sekä sivuilla, joihin juurijoukon sivut viittaavat. Näin saadaan *perusjoukko*, jonka pohjalta voidaan muodostaa kyselyriippuvainen graafi $G_q = (D_q, E_q)$, missä $D_q \subset D$ on perusjoukko ja E_q on niiden linkkien joukko, joiden alku- ja loppusolmu ovat perusjoukossa D_q. Jokaisella perusjoukon sivulla on omat auktoriteetti- ja napa-arvot, jotka voidaan kirjoittaa vektoreina a ja h siten, että vektorin i:s komponentti on perusjoukon i:ttä dokumenttia vastaava arvo. Arvot lasketaan rekursiivisesti yhtälöiden $a = E^T h$ ja $h = Ea$ avulla. E on G_q:n matriisiesitys (linkkimatriisin C D_q-joukkoon liittyviä dokumentteja vastaavat sarakkeet ja rivit). Huomattavaa on, että yhtälöt riippuvat toisistaan. a-vektori approksimoi $E^T E$:n pääominaisvektoria ja h-vektori vastaavasti EE^T:n pääominaisvektoria [88]. Vektoreilla on myös kiinnostava yhteys tekstianalyysin puolelta tuttuun LSI-malliin: osoittautuu, että HITS-algoritmin soveltaminen linkkimatriisiin on yhtäpitävä operaatio linkkimatriisin diagonalisoinnille, samaan tapaan kuin LSI diagonalisoi sanamatriisin. Tämä mahdollistaa muunnetun linkkimatriisin käytön myös klusteroinnissa [22, sivu 212-216].

Pelkän linkkirakenteen lisäksi myös dokumenttiin viittaavan linkin läheisyydessä olevaa tekstiä voidaan käyttää kuvaamaan dokumenttia. Chakrabarti *et al.* [23] kutsuvat kuvausta ankkuri-ikkunaksi (*anchor window*) ja käyttävät sitä parantamaan hakutulosten järjestämistä HITS-tyylisellä algoritmilla. Ankkuri-ikkunan käyttöä samanlaisuuden arvioinnissa puoltaa myös Gloverin *et al.* [69] web-dokumenttien luokittelua koskeva tutkimus. Paras luokittelutulos saatiin, kun dokumenttitekstin ja viittaavan linkkitekstin lisäksi hyödynnetään kappaletta, jossa linkkiteksti sijaitsee, ns. dokumentin kontekstia. Kontekstikappaleet osoittautuivat ratkaisevaksi luokittelussa, koska yksistään niiden avulla saatiin jopa parempi luokittelutulos kuin dokumenttien varsinaisella tekstillä. Tiedonlouhinnan näkökulmasta ankkuri-ikkunat ovat tekstimuotoisia piirteitä, jotka voi yhdistää painotetusti varsinaisesta dokumenttitekstistä koostettuun indeksiin tai käyttää sen rinnalla. Myös Google-hakukone hyödyntää ankkuritekstiä, mikä mahdollistaa tiedonhaun myös sivuista, joihin hakukone ei suoraan pääse käsiksi (tietokantahaut, kirjautumisen vaativat tai poistetut sivut) [19]. Tämä voi aiheuttaa myös väärinkäyttöä, jos suuri joukko sivuja linkittyy samaan sivuun mahdollisella asiattomalla linkkitekstillä. Tällöin kohdesivu päätyy ensimmäiseksi hakutuloksek-

si[2]. Tekniikkaa kutsutaan nimellä *Googlebombing* ja sen keksi ensimmäisenä Adam Mathes [82]. Myös HITS-algoritmia voidaan väärinkäyttää vastaavalla tekniikalla, joskin algoritmin kehittyneemmät versiot pystyvät suodattamaan keinotekoisia linkkejä pois [22, 219-225].

5.3 Rakenneanalyysi

Luvussa käsitellään rakenteista indeksointia, hakumalleja ja XML-rakenteiden samanlaisuusmittoja. Tarkastelun kohteena ovat lähinnä elementit ja rakenteinen teksti; attribuuttien ja linkkien käsittely sivuutetaan tässä yksinkertaisuuden vuoksi. Esimerkeissä mainitut rakenteiset hakulausekkeet on kuvattu XPath-kielellä.

5.3.1 Rakenteinen indeksointi

Rakenteisten dokumenttien indeksointitekniikat voidaan jakaa tekstipohjaisiin (ei ota huomioon rakennetta), puolirakenteisiin (ei ota huomioon skeemaa) tai rakenteisiin (ottaa huomioon skeeman, käytetään esim. tallennettaessa XML-dokumentteja relaatiotietokantaan) [101]. Tässä käsiteltävät indeksointitekniikat ovat pääasiassa puolirakenteisia: dokumentin rakenne otetaan indeksissä huomioon, mutta tiettyyn skeemaan ei sitouduta. Linkit jätetään yksinkertaisuuden vuoksi huomiotta – oletetaan, että dokumentit ovat puita. Lisäksi oletetaan, että dokumentit ovat jossakin puolirakenteisessa tietokannassa, jonka yksittäisiin elementteihin (solmuihin) voidaan viitata solmun tunnisteella. Indeksirakenteesta riippuen tunniste voi olla yksinkertainen järjestysnumero tai sitten tietue, jonka avulla päästään navigoimaan muihin solmuihin. Joillakin numerointikäytännöillä voidaan myös päätellä, onko tietty tunniste elementtihierarkiassa toisen yläpuolella. Puolirakenteinen tietokanta on loogisesti yksi puu tai graafi, joka sisältää kaikki fyysiset dokumentit [133, sivut 34-36]. Fyysisesti dokumentteja voidaan säilyttää suoraan tiedostojärjestelmässä ja pitää ainoastaan indeksi tietokannassa. Tällöin solmujen tunnisteiden täsmääminen fyysisiin XML-elementteihin saattaa tosin aiheuttaa lisätyötä.

Indeksointi- ja hakumalleja voidaan tarkastella yleisellä tasolla Weigelin *et al.* [134] esittämän PTN-luokittelun (*Path/Term/Node*) avulla. PTN-luokittelussa mallia kuvataan P- (polku), T- (termi) tai N-komponenteilla (solmu) tai niiden yhdistelmillä sen mukaan, mitä tarkkuustasoa malli tukee sisällön tai rakenteen suhteen. P-komponentti kuvaa kyselyn rakenteista osaa (esim. luvun 3 puhelinnumeroesimerkin `<henkilo>`-elementtiin voisi viitata XPath-

[2]Esimerkiksi haettaessa Googlella 12.11.2004 `miserable failure` hakutulosten kärjessä oli *George W. Bushin* elämäkertasivu.

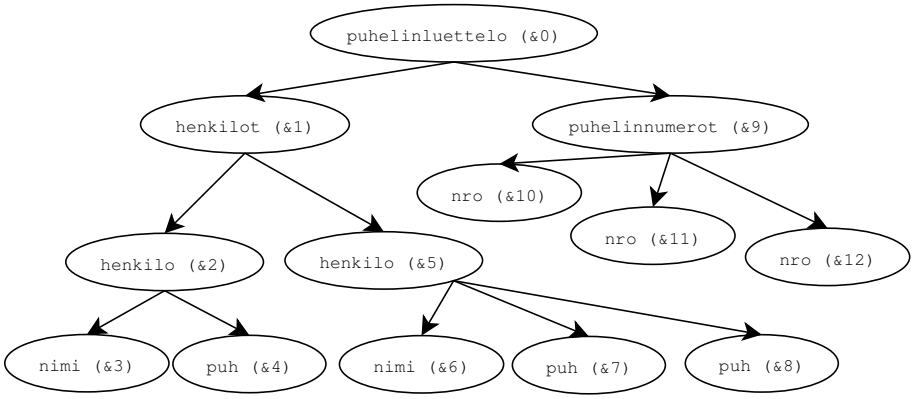
lausekkeella `/puhelinluettelo/henkilot/henkilo`) ja T vastaavasti sisältöosaa. N kuvaa dokumentteja tai niiden osia (solmuja), jotka palautetaan. Tämän luokittelun perusteella vektorimalli $tf \times idf$-tyylisellä painotuksella noudattaa {TN,T,N}-tasoa: yksittäisten dokumenttien termifrekvenssit ovat {TN}-tasolla, käänteiset dokumenttifrekvenssit ovat {T}-tasolla ja termifrekvenssin laskennassa käytetty dokumenttikohtainen termien maksimimäärä, $\max_{t \in \mathbf{T}}\{n(t,j)\}$ (katso luku 5.1.1) on {N}-tasolla. Toisaalta vektorimallin yhtenä indeksirakenteena käytetty painottamaton sanamatriisi noudattaa vain {TN}-tasoa, koska pelkkää matriisia käyttämällä idf:n tai termien maksimimäärän laskeminen vaatisi matriisin tietyn rivin tai sarakkeen täydellistä läpikäyntiä.

Weigel [133, sivut 39-40] jakaa indeksointityypit niiden mutkikkuuden mukaan

1. perusindekseihin,
2. polkuindekseihin ja
3. puuindekseihin.

Perusindeksit ovat assosiaatiotauluja tai taulukoita, yleistyksiä tekstidokumenttien indeksoinnissa käytetyille käänteistiedostoille [133, sivut 39-46]. Perusindeksejä ovat sanalista eli *käänteissolmulista* [134] (liittää jokaiseen termiin solmujoukon, jossa termi sijaitsee), arvolista (liittää attribuuttien arvot solmujoukkoihin), elementtilista (liittää elementtityypit niitä edustaviin solmujoukkoihin – ei kuitenkaan polkuja) ja vanhempi/lapsi-indeksi (indeksisolmujen hierarkiasuhteet). Sanalista noudattaa PTN-luokittelussa {TN}-tasoa, muut indeksit eivät sovellu luokitteluun. Myös Luk'n *et al.* [101] puolirakenteisiin indeksointitekniikoihin luokittelema kenttäpohjainen indeksi voidaan laskea perusindekseihin. Kenttäpohjaisessa indeksoinnissa dokumentti kuvataan puun sijasta joukkona kenttiä, joista kukin voidaan indeksoida esim. vektorimallin mukaisesti. Kenttäpohjainen indeksi soveltuu dokumenteille, joissa ei ole syvää hierarkiaa tai jossa karkea rakenteen jako (esim. otsikko, metatietokentät, leipäteksti) riittää. Kenttäpohjaisen indeksin etuna on lähinnä sen yksinkertaisuus, koska mutkikkaita puurakenteita ei tarvitse toteuttaa. Empiirisessä osuudessa on käytetty kenttäpohjaista indeksiä (katso luku 6.2). Luvussa 3.2.1 esitetyn XML-tiedoston osa on esitetty tietokannan solmuina kuvassa 5.2 (solmuihin merkitty järjestysnumerot, attribuutit jätetty huomiotta). Tietokannan käänteissolmulista ja elementtilista on esitetty taulukoissa 5.1 ja 5.2.

Pelkästään perusindeksejä käyttämällä on mahdollista tehdä hakujärjestelmä, joka tukee samanaikaisia rakenne- ja tekstihakuja. Järjestelmän suorituskyky on kuitenkin alhainen, koska kysely jouduttaisiin käsittelemään monessa vaiheessa. Esim. puhelinnumeroesimerkkiin tehtävässä XPath-kyselyssä `/puhelinluettelo/henkilot/henkilo/[nimi='A. A.']` täytyisi käydä läpi sanalistasta "`A. A.`"-arvon sisältävät solmut ja yhdistää ne elementtilistan



Kuva 5.2: Puhelinnumeroesimerkki XML-tietokantana.

Nimi	*Solmujoukko*
A.A	$\{3\}$
N.N	$\{6\}$
435-345345	$\{10\}$
11-343255	$\{11\}$
435-365552	$\{12\}$

Taulukko 5.1: Sanalista

`<nimi>`-elementin sisältäviin solmuihin. Jos `<nimi>`-elementti voisi olla jonkin muun polun varrella kuin kyselyssä mainitun (tässä esimerkissä ei ole, mutta jos skeema ei ole tiedossa, tästä ei olisi mitään takeita), `nimi`-elementtilistan solmujen polut pitäisi vielä tarkistaa hakemalla `henkilo`-elementtilistan solmut ja vertaamalla niitä `<nimi>`-solmujen vanhempiin (indeksirakenteesta riippuen vanhempi/lapsi-listaa läpikäymällä tai suoraan tunnistenumeroilla) – mahdollisesti rekursiivisesti vielä ylemmille tasoille solmuhierarkiassa. Tästä heikkoudesta huolimatta perusindeksit (erityisesti sana- ja arvolistat) soveltuvat käytettäväksi kehittyneempien indeksimekanismien apuindekseinä.

Polkuindeksit käyttävät dokumentin polkuja haun perusyksikköinä solmujen sijaan. Kaikista hakuyksiköistä tallennetaan polku kokonaisuudessaan. Tällöin polkua ei tarvitse rakentaa

Nimi	*Solmujoukko*
puhelinluettelo	$\{0\}$
henkilot	$\{1\}$
puhelinnumerot	$\{9\}$
henkilo	$\{2, 5\}$
nro	$\{10, 11, 12\}$
nimi	$\{3, 6\}$
puh	$\{4, 7, 8\}$

Taulukko 5.2: Elementtilista

Nimi	*Polkujoukko*
A.A	$\{puhelinluettelo/henkilot/henkilo[0]/nimi\}$
N.N	$\{puhelinluettelo/henkilot/henkilo[1]/nimi\}$
435-345345	$\{puhelinluettelo/puhelinnumerot/nro[0]\}$
11-343255	$\{puhelinluettelo/puhelinnumerot/nro[1]\}$
435-365552	$\{puhelinluettelo/puhelinnumerot/nro[2]\}$

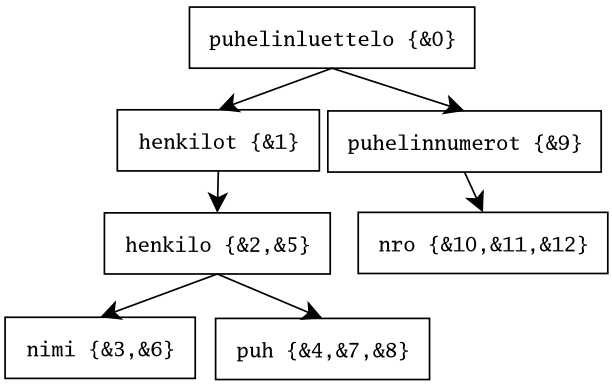
Taulukko 5.3: Käänteispolkulista

haun aikana, kuten perusindekseillä. Polkuindeksit liittävät tyypillisesti hakusanoihin ne polut, joissa hakusana esiintyy. Joissakin indeksirakenteissa on mahdollista liittää myös polku hakusanoihin, mutta olennaisinta on rakenteen taulukkomaisuus – polkujen juuri- ja yläsolmuja toistetaan indeksin sisällä. Henkilön nimi voidaan hakea puhelinnumeroesimerkin dokumentista vakioajassa, mutta jokerimerkkejä tai (rakenteesta riippuen) pelkkiä rakennekriteerejä sisältävien kyselyjen käsittely on hitaampaa [133, sivut 39-40]. Esimerkkinä polkuindeksistä tarkastellaan elementtipaikanninmallia (*element locator scheme*) [133, sivut 47-50], jonka esittivät alunperin Sacks-Davis *et al.* [112]. Indeksi on toteutettu assosiaatiotauluna, jossa termeihin liitetään solmujen paikallisilla järjestysnumeroilla varustetut polut. Koska malli ei tarvitse solmujen tunnisteita, se on toteutettavissa helposti tapauksessa, jossa XML-dokumentit ovat tiedostojärjestelmässä (tällöin polkuihin on liitettävä myös tiedoston tunniste). Malli noudattaa PTN-luokittelussa{PT}-tasoa. Esimerkki elementtipaikanninmallin mukaisesta käänteispolkulistasta on esitetty taulukossa 5.3.

Puuindeksit (tai navigationaaliset indeksit) ovat rakenteisista indeksirakenteista mutkikkaimpia, mutta samalla ilmaisuvoimaltaan, suorituskyvyltään ja yleensä myös tilavaativuudeltaan parhaita. Niiden tietorakenteena on suunnattu graafi tai puu, joka noudattaa muodoltaan dokumenttikokoelman skeemaa ja on näin dokumentin rakenteinen lyhennelmä (tai deskriptiivinen skeema). Tietokannan solmujen tiedot voidaan merkitä suhteessa tähän rakenteeseen, jolloin polkuindeksien kaltaisia moneen kertaan merkittyjä polkuja ei tule. Lisäksi kyselyn rakenteinen osa voidaan sovittaa dokumentteihin puun syvyyteen verrannollisessa ajassa [133, sivut 39-46]. Puuindeksit on useimmiten suunniteltu puhtaasti rakenteisia kyselyjä silmälläpitäen. Tekstihakua varten puuindeksin apuna voidaan käyttää esim. käänteissolmulistaa, jolloin kyselyä suoritettaessa puuindeksin ja käänteissolmulistan tulokset on yhdistettävä (ns. *content/structure-join*) [133, sivut 28-29].

DataGuide on perusesimerkki rakenteisesta puuindeksistä. Se soveltuu sekä puu- että verkkomaisten dokumenttien kuvaamiseen. DataGuide on suunnattu graafi, jonka solmuja ovat dokumentin elementtityypit järjestettynä niin, että jokainen elementeistä koostuva absoluuttinen polku (esim. `/puhelinluettelo/henkilot/henkilo`) voidaan käydä siinä deterministisesti läpi. Jokainen graafin solmu sisältää joukon dokumenttitietokannan solmujen tun-

nisteita: elementit, jotka täsmäävät solmua vastaavaan polkuun. DataGuide ei ole yksikäsitteinen, vaan se voi olla mikä tahansa dokumentin todellista skeemaa vastaavan epädeterministisen äärellisen automaatin deterministinen esitys (tulkittaessa DataGuide automaattina tilat vastaavat elementtityyppejä ja siirtymät mahdollisuutta tietyn tyyppiseen alielementtiin) [70]. Jos dokumentit eivät ole puita (sisältävät syklejä), DataGuiden solmuja täytyy kopioida huomioiden vaihtoehtoiset reitit. Pahimmassa tapauksessa tämä voi johtaa eksponentiaaliseen graafin koon kasvuun, mutta käytännössä rakenne on havaittu toimivaksi erilaisilla dokumenttikokoelmilla [133, sivut 69-75]. PTN-luokittelussa DataGuide on {P,PN}-tasolla [134]. Kuvassa 5.3 on esitys puhelinnumeroesimerkistä DataGuide-muodossa.



Kuva 5.3: Puhelinnumeroesimerkin rakenne DataGuide-muodossa.

Kehittyneimmät puuindeksit sisältävät rakennetietojen lisäksi myös tietoja solmujen sisällöstä, jolloin hakuvaiheessa ei tarvitse tehdä rakenne- ja sisältötulosten liitosta. Tällaisia rakenteita ovat mm. Weigelin *et al.* Content-Aware DataGuide [134], BUS-hakumallin indeksi sekä IndexFabric-tietorakenne [133, sivut 28-29]. Niiden tarkempi käsittely sivuutetaan.

5.3.2 Rakenteinen haku

Sacks-Davis *et al.* [112] listaavat perusteellisesti vaatimuksia, joita XML (artikkelin kirjoituksen aikana SGML) -kyselykieli voisi täyttää:

1. Sanahaku (kokotekstihaku)
2. Järjestetyt hakutulokset.
3. Määrättyihin dokumentin osiin kohdistuva haku (esim. XPath-polulla ja termeillä rajattu kysely, joka palauttaa koko dokumentin).
4. Dokumentin osien (alipuiden) haku (esim. datakeskeisillä hakukielillä).
5. Kysely rakenteen mukaan (esim. XPath-lausekkeet, ei termejä).

6. Samanaikainen haku monesta eri dokumenttityypistä.
7. Haku attribuuttien, entiteettien tai muiden XML-spesifisten ominaisuuksien mukaan.
8. Yhdistetty haku XML-dokumenttien lisäksi muista tietojoukoista (esim. ulkoinen metatieto).

Kokotekstihaku ja järjestetyt hakutulokset ovat perinteisesti kuuluneet tekstitiedonhakuun, kun taas kohdat 3-5 ovat tietokannoille ominaisia. Niiden yhdistelmä voidaan hoitaa indeksoinnin osalta puuindeksillä, jota on mahdollisesti täydennetty käänteissolmulistalla. Kohta 6 on mahdollinen, jos koko dokumenttikokoelma mallinnetaan yhtenä graafina tai jos pitäydytään pelkässä tekstihaussa. Attribuuttihakua (esim. XPath) lukuunottamatta tämän kirjoittaja ei löytänyt kohtaan 7 soveltuvia kyselykieliä. Yhdistetyt hakutavat (katso luku 5.4) pyrkivät vastaamaan kohtaan 8.

Bunemanin [20] mainitsemat lähestymistavat XML-kyselykielten määrittelyyn (SQL:n laajennus tai puolirakenteisen mallin soveltaminen) vastaa melko tarkasti XML-pohjaisten kielten jakoa data- ja dokumenttikeskeisiin. Datakeskeisten dokumenttien käsittelyyn soveltuu esim. W3C:n XQuery-kieli [14], joka sallii XPath-kieltä monipuolisempien rakenteisten ja kenttäpohjaisten ehtojen muodostamisen SQL-kielen tapaan. Hakutulosten järjestely relevanssin mukaan tai kokotekstihaku eivät kuitenkaan ole oletuksena tuettuja (kieleen on suunniteltu laajennuksia, mutta ne sivuutetaan tässä). Dokumenttikeskeisiä kieliä varten tarvitaan tiedonhaun käsitteiden pohjalta kehitetty hakukieli, joita voidaan edelleen jakaa rakenteeseen tai sisältöön keskittyviin [64]. Yksittäisiä XML-kyselykieliä on runsaasti, eikä tässä ole mahdollista keskittyä minkään yksittäisen kielen syntaksiin muuten kuin edellisessä luvussa käytettyjen XPath-lausekkeiden tasolla (poikkeuksena hakumallit, joilla kysely voidaan antaa suoraan dokumenttifragmenttina). Sen sijaan käydään läpi yleisiä hakumalleja, termien ja elementtien painotustapoja sekä relevanssin arviointitapoja. Kuvauksessa on käytetty alkuperäisten lähteiden lisäksi apuna Luk'n *et al.* [101] kattavaa kirjallisuuskatsausta.

Helposti toteutettava, mutta ilmaisuvoimaltaan rajoittunut tapa rakenteiseen tiedonhakuun on käyttää Foxin *et al.* [59] laajennettua vektorimallia. Standardissa vektorimallissa dokumentit ja kyselyt esitetään vektorina, laajennetussa ne ovat vektorimonikoita, joista kukin vektori kuvaa tiettyä näkökulmaa (näkökulmia voivat olla esim. sanamatriisi, linkkimatriisi, otsikkotekstit tai metatietokentät) dokumenttiin. Esimerkiksi sana- ja linkkimatriisia käyttävä dokumentti j voitaisiin esittää seuraavana vektoriparina:

$$d_j = ((w_{1j}, w_{2j}, \ldots, w_{nj}), (c_{1j}, c_{2j}, \ldots, c_{mj})).$$

Foxin artikkelissa indeksoinnin kohteena oli bibliografinen tietokanta, mutta ainakin Crouch *et al.* [35] ovat käyttäneet laajennettua vektorimallia myös XML-dokumenttikokoelman indeksointiin ja hakuun. Hakuvaiheessa jokaista vektoria verrataan mahdollisesti omalla etäisyysmitallaan ja tulokset yhdistetään painotettuna lineaarikombinaationa. Edellisen dokumenttivektorin tapauksessa täsmäytysfunktion R arvoksi kyselyllä q tulee

$$R(d_j, q) = \sum_{k=1}^{2} w_k sim_k(r_k(d_j), r_k(q)),$$

missä r_k on rajoitefunktio, joka erottaa dokumentti- ja kyselymonikoista k:nnen vektorin, sim_1 ja sim_2 ovat etäisyysmitat termi- ja linkkivektoreita varten ja w-vektori sisältää komponenttien painot siten, että niiden summa on 1. Laajennettu vektorimalli soveltuu erityisesti kenttäindeksin kanssa käytettäväksi, mutta muitakin indeksirakenteita on mahdollista hyödyntää. Mallia voidaan käyttää esim. painotettuihin polkukyselyihin, jos käytettävissä on polkuindeksi [101]. Ideaa voi yleistää eri lähteistä saatavien ja esitystavaltaan vaihtelevien etäisyysmittojen yhdistelyyn, jolloin malli soveltuu myös metriseen klusterointiin. Tällöin kahden alkion välinen etäisyys on painotettu summa niiden indeksikomponenttien välisestä etäisyydestä ja lasketaan kuten laajennetun vektorimallin täsmäytysfunktio R.

Varhaisimmat hierarkkista tiedonhakua koskevat tulokset on esitetty hypertekstikirjallisuudessa: haulla pyritään etsimään paras ”aloitusdokumentti” kokoelman selaamiseen. Rakenteinen dokumenttikokoelma tulkitaan hypertekstikantana. Tällöin varsinaisten linkkien ohella myös sisäkkäiset elementit tulkitaan linkittyneiksi solmuiksi. Hypertekstimallien yleinen ongelma on, että haun tarkkuus heikkenee merkittävästi liian pienillä dokumenttiyksiköillä [66]. Ongelma on erityisen vakava, jos jokaista elementtiä pidetään omana dokumenttinaan ja dokumenttityyppi sisältää muotoiluspesifisiä elementtejä. Ongelmaa yritti ensimmäiseksi korjata Frisse [62], jonka sovelluksessa lääketieteellinen käsikirja jaettiin hierarkkisiin ”kortteihin”. Kukin kortti vastasi kirjan tiettyä kappaletta tai lukua. Jokainen kortti sisälsi edelleen linkit omiin ”alikortteihinsa”. Korteilla oli omat otsikot ja tekstit, mutta asiayhteyden puolesta ne eivät olleet itsenäisiä dokumentteja (mallin samankaltaisuus XML-dokumenttiin on ilmeinen, jos ”kortti” tulkitaan elementiksi). Kortit indeksoitiin tavallisella $tf \times idf$-painotuksella, mutta täsmäytyksessä huomioitiin korttien hierarkia. Olk. $\mathbf{S}_j$ dokumentin j alidokumenttien indeksijoukko. Tällöin täsmäytysfunktion R arvo kyselyllä q on

$$R(d_j, q) = sim(d_j, q) + \frac{\sum_{k \in \mathbf{S}_j} R(d_k, q)}{|\mathbf{S}_j|},$$

missä sim on vertailufunktio. Täsmäytys aloitetaan dokumenttipuun lehdistä, jonka jälkeen edetään rekursiivisesti juuridokumenttiin asti. Jakotermin $|\mathbf{S}_j|$ tarkoitus on vähentää yksittäi-

sen alidokumentin merkitystä, jos alidokumentteja on suuri määrä. Tulokset pitäisi näyttää käyttäjälle muuten relevanssijärjestyksessä, mutta jos tuloslistaan on jo merkitty korkeammalla relevanssiarvolla tämänhetkisen kortin ”ylikortti”, se voidaan jättää pois (muuten tuloslista saattaisi täyttyä hierarkiassa hyvin lähellä toisiaan olevista korteista).

Weigelin [133] terminologiaa käyttäen Frissen malli tarvitsee ”lattean” sanamatriisin lisäksi vanhempi/lapsi-indeksin täsmäytysfunktion laskennassa. Fuller *et al.* [66] kehittävät Frissen ideoita edelleen SGML-dokumenteille, mutta perusidea painojen laskemisesta on sama. Uudemmista XML-hakumalleista XIRQL [64] muistuttaa Frissen mallia siinä mielessä, että dokumentit on ositettu itsenäisiksi solmuiksi, joista kuhunkin on koottu sopivaksi katsottu dokumentin alipuu. Termit on tallennettu käänteissolmulistaan, joka on painotettu $tf \times idf$-mallilla siten, että jokaista solmua pidetään dokumenttina. XIRQL:n täsmäytys perustuu kuitenkin Frissen mallista poiketen probabilistiseen päättelymaliin [101].

XPRES [137] on rakenteinen laajennus luvussa 5.1.2 mainitulle probabilistiselle hakumallille (laajennuksen johtaminen sivuutetaan). Dokumentin rakenne kuvataan käyttäjälle joukkona *rakenteisia rooleja*, joiden tarkkuus voi vaihdella karkeasta kenttäpohjaisesta jaottelusta yksittäisiin XPath-polkuihin. Jokainen rooli voi sisältää useita dokumentin elementtejä ja toisaalta jokainen elementti voi kuulua useampaan rooliin. Kyselyt esitetään $(termi, rooli)$-parien joukkona. XPRES käyttää käänteissolmulistaa termien hakuun ja puuindeksiä rakenteen (elementtien ja roolien) esittämiseen. Tietyn elementin e relevanssiarvo termistä t_i ja roolista s_k koostuvaan kyselyyn voidaan esittää seuraavasti:

$$R(e, (t_i, s_k)) = (C + ief(t_i, s_k)) \left(K + (1 - K) \frac{f((t_i, s_k), e)}{maxfreq_e} \right),$$

missä C ja K ovat mallin parametrivakioita, $maxfreq_e$ on suurin mielivaltaisen termin esiintymismäärä elementissä e tai sen jossain alielementissä, $f((t_i, s_k), e)$ on termin t_i esiintymien yhteenlaskettu määrä elementissä e ja sen niissä alielementeissä, jotka kuuluvat rooliin s_k. ief on käänteinen elementtifrekvenssi (*inverted element frequency*), idf-mitan rakenteinen yleistys. Se voidaan laskea kaavalla

$$ief(t_i, s_k) = log \frac{N_{s_k} - n_{t_i, s_k}}{n_{t_i, s_k}},$$

missä N_{s_k} on niiden elementtien määrä, jotka kuuluvat rooliin s_k, ja n_{t_i, s_k} on niiden elementtien määrä, joissa on termi t_i ja jotka kuuluvat rooliin s_k.

Schliederin & Meussin [119] puumalli yhdistää rakenteiset kyselyt todennäköisen relevanssin mukaan järjestettyihin tuloksiin ja dynaamisiin dokumentteihin, joiden granulariteetti riippuu kyselystä. Kyselyt annetaan XML-dokumenttifragmentteina. Dokumenttikokoelma

tulkitaan yhdeksi puuksi, jonka alipuita sanotaan *loogisiksi dokumenteiksi*. Puun solmuja ovat dokumentin elementit, attribuutit ja varsinainen teksti. Merkkijonot paloitellaan niin, että jokainen sana muodostaa oman solmunsa. Palautettavan dokumentin tyyppi riippuu kyselypuun juuresta: jokainen looginen dokumentti, jolla on samanniminen juuri kuin kyselypuulla, voidaan palauttaa hakutuloksena. Näitä dokumentteja kutsutaan soveltuviksi dokumenteiksi (*admissible documents*).

Hakupuun sovitus dokumenttipuuhun on järjestämättömien puiden täsmäytysongelma. Täsmäytys on funktio hakupuulta dokumenttipuulle, mille pätevät seuraavat ehdot kaikilla hakupuun solmuilla u,v:

1. u:n nimi on sama kuin $f(u)$:n nimi.
2. Jos u:sta on polku v:hen, niin myös $f(u)$:sta on polku $f(v)$:hen.

Kysely täsmää dokumenttiin, jos on olemassa täsmäytysfunktio kyselystä dokumenttiin. Täsmäys on osittainen, jos jokin kyselyn alipuu täsmää kohteena olevaan dokumenttiin. Sovituksessa on haettu kompromissia laskennallisen tehokkuuden ja kielen ilmaisuvoiman välillä.

Yhdistetyn mallin keskeinen käsite on *rakenteinen termi*, joka tarkoittaa mitä tahansa dokumentin tai kyselyn alipuuta. Rakenteinen termi on yleistys vektorimallin termille, joka yleensä viittaa vain yhteen sanaan. Olkoon nyt T rakenteinen termi. Sen esiintymät dokumenteissa ja kyselyissä määritellään seuraavasti:

1. Jos on olemassa T:n kanssa isomorfinen rakenteinen termi T' siten, että T' on kyselyn Q alipuu, T *esiintyy* Q:ssa. T':n juuri on T:n *esiintymä* Q:ssa.
2. Jos T täsmää dokumenttiin D, T *esiintyy* D:ssä. Täsmäytysfunktion tuloksena saatu D:n alipuu on T:n *esiintymä* D:ssa.

Rakenteisille termeille voidaan vektorimallin termien tapaan määritellä termifrekvenssi ja käänteinen dokumenttifrekvenssi. Olkoon D looginen dokumentti, $freq_T(D)$ rakenteisen termin T esiintymien määrä D:ssä ja $maxfreq(D)$ mielivaltaisen rakenteellisen termin maksimaalinen esiintymien määrä D:ssä (käytännössä suurin D:n samannimisten solmujen määrä). Termifrekvenssi määritellään seuraavasti:

$$tf_{TD} = \frac{freq_T(D)}{maxfreq(D)}.$$

Käänteinen dokumenttifrekvenssi määritellään kyselykohtaisesti soveltuville dokumenteille, koska kysely kohdistuu vain juureltaan samannimisiin dokumentteihin. Olkoon T rakenteinen termi ja t dokumenttityyppi. $|D_t|$ on t-tyyppisten dokumenttien määrä, n_T on T:n kanssa täsmäävien dokumenttien määrä kokoelmassa. Tällöin käänteinen dokumenttifrekvenssi on

$$idf_{Tt} = \log \frac{|D_t|}{n_T} + 1.$$

Termifrekvenssi ilmaisee, kuinka hyvin tietty termi kuvaa dokumenttia. Käänteinen dokumenttifrekvenssi kuvaa, kuinka hyvin tämä termi erottaa dokumentin muista. Termit voidaan painottaa vektorimallin tapaan $tf \times idf$-mallilla.

Kysely ja dokumentit esitetään painotettuina vektoreina, jonka komponentit ovat kaikki dokumenttikokoelmassa esiintyvät toisistaan poikkeavat rakenteiset termit. Kyselyn painotusten avulla käyttäjä voi ilmaista eri termeille suhteellisen tärkeyden. Relevanssiarvion perustana oleva dokumentin ja kyselyn samanlaisuus lasketaan vektorimallin tapaan, esim. kosinimitalla. Mallin termien välillä vallitsee selkeä rakenteellinen riippuvuus, koska jokin rakenteinen termi voi olla toisen termin alipuu. Vektorimallia voidaan simuloida asettamalla kaikkien kyselyn rakenteisten solmujen painoksi 0. Vastaavasti pelkkää puiden sovitusta voidaan simuloida asettamalla tekstisolmujen painoksi 0.

Malli käyttää puuindeksiä rakenteen kuvaukseen ja käänteissolmulistaa termien liittämiseen solmuihin. Solmujen tunnisteet on nimetty niin, että voidaan tarkistaa vakioajassa, onko jokin solmu toisen alisolmu. Kyselyt käsitellään Frissen mallin tapaan lehdistä juuriin. Painot lasketaan täsmäytyksen edetessä. Mallin termien ja alirakenteiden painotus, osittaiset täsmäykset ja dokumenttifragmenttien haku muistuttavat Wolffin XPRES-mallia. Kyselykieli ja relevanssiarvio ovat kuitenkin erilaiset.

5.3.3 Rakenteiset samanlaisuusmitat

Perinteiset klusterointimenetelmät eivät yleensä ota huomioon dokumentin rakennetta [132]. Edellisissä luvuissa mainittujen tekstiin ja linkkirakenteisiin perustuvien samanlaisuusmittojen ohella dokumentteja voidaan klusteroida niiden rakenteeseen perustuvin kriteerein. Pelkästään rakenteet huomioonottavia etäisyysmittoja voidaan käyttää esim. klusteroitaessa eri dokumenttityyppejä noudattavia tai skeemattomia dokumentteja niiden todennäköisen tyypin mukaan. Osa rakenteisista mitoista ottaa huomioon myös dokumenttien tekstisisällön, mikä mahdollistaa kokonaisvaltaisen rakenteisten dokumenttien klusteroinnin. Rakenteen ja tekstin yhdistäminen klusteroinnissa on epätriviaalia, koska mahdollisia esitystapoja on runsaasti ja ne vaihtelevat ilmaisuvoimaltaan ja laskennalliselta vaativuudeltaan. Erityisesti puhtailla tekstidokumenteilla käytettyä, sekä tiedonhakuun että klusterointiin soveltuvaa vektorimallin kaltaista yhtenäistä ”standardimallia” ei ole toistaiseksi näköpiirissä. Lupaavalta tavalta tiedonhaun ja klusteroinnin yhdistämiseen vaikuttaa Schliederin & Meussin [119] puumalli, koska siinä kyselyt ja dokumentit kuvataan yhtenäisessä kehyksessä. On kuitenkin

epäselvää, mikä on mallin suorituskyky suurilla dokumenttikokoelmilla ja onko puiden sovituksessa käytetty heuristiikka yleispätevä. Samaa sisältöhaun periaatetta voisi käyttää myös muilla luvussa käsiteltävillä samanlaisuusmitoilla.

Perusmenetelmä puhtaasti rakenteiseen klusterointiin saadaan toisesta tiedonlouhinnan menetelmäperheestä: assosiaatiosääntöjen louhinnasta. Wang & Liu [132] esittävät algoritmin, jolla dokumenteista saadaan eroteltua ”tyypilliset” rakenteet. Tyypilliset rakenteet esitetään puulausekkeilla, jotka ovat oleellisesti dokumenttifragmentteja ilman sisältötekstejä. Leen *et al.* [98] malli hyödyntää peräkkäisten sääntöjen etsintää. Dokumentti mallinnetaan determinisoituna ja minimoituna äärellisenä automaattina, jossa siirtymät kuvaavat elementin sisältymistä toiseen. Automaatin lopputilaa kuvaavat vain tekstiä sisältävät elementit. Käymällä läpi verrattavista dokumenteista generoitujen automaattien tilasiirtymät saadaan dokumenttien mahdolliset polut. Vertaamalla polkuja toisiinsa voidaan löytää maksimaaliset n:n elementin pituiset ketjut, jotka ovat yhteisiä kummallekin dokumentille. Yksinkertainen tapa rakenteellisen samanlaisuuden vertailuun on laskea maksimaalisten yhteisten polkujen suhde kaikkiin löydettyihin polkuihin.

Graafit ovat luonnollinen tapa rakenteisten dokumenttien kuvaamiseen. Dokumenttia ei välttämättä tarvitse muuntaa useimmissa muissa menetelmissä käytettyyn vektorimuotoon ja klusteroinnissa voidaan käyttää verkkoteoreettisia menetelmiä ja graafien etäisyysmittoja, kuten maksimaalista yhteistä aliverkkoa (katso luku 4.3.3). Schenkerin *et al.* [118] mallissa dokumentin jokainen sulkusanoista eroteltu termi muunnetaan verkon solmuksi. Dokumentin rakenne ja sisältö huomioidaan määrittelemällä linkkityyppejä, jotka yhdistävät tietyissä elementeissä olevat sanasolmut niiden esiintymisjärjestyksen mukaan. Linkkityyppejä voivat olla esim. otsikko, linkkiteksti tai sisältöteksti. Schenkerin malli on suunniteltu lähinnä tekstidokumentille ja muistuttaa kenttäpohjaista indeksointia (katso luku 5.3.1), mutta se voidaan myös yleistää XML-dokumentteihin esim. määrittelemällä linkkityypit XPath-poluiksi. Dokumenttityypistä riippuen linkkityyppien määrä saattaisi tosin kasvaa kohtuuttoman suureksi, jolloin malli muistuttaisi lähinnä käänteispolkulistaa. Puhtaasti rakenteinen lähestymistapa graafipohjaiseen klusterointiin on Lianilla *et al.* [100]. Heidän mallissaan jokainen dokumentti mallinnetaan DataGuide-tietorakennetta muistuttavana rakennegraafina (*s-graph*). Klusteroinnin etäisyysmittana käytetään maksimaalista yhteistä aliverkkoa.

5.4 Yhdistetyt hakutavat

Yhdistetyllä hakutavalla tarkoitetaan tiedonhaussa tai klusteroinnissa käytettävän samanlaisuusmitan muodostamista monen erityyppisen (tai erilaisia näkökulmia kuvaavien) piirtei-

den pohjalta. Näitä näkökulmia voivat olla dokumentin tekstitieto, hyperlinkit, rakennetieto tai metatieto. Edelleen yksittäisiä piirteitä voidaan esittää tai mitata eri tavoilla, kuten edellisissä luvuissa on lukuisin esimerkein näytetty. Piirteiden esitystavasta riippuen yhdistetyllä hakutavalla kuvattu dokumentti voidaan esittää esimerkiksi vektorina, vektorimonikkona, matriisina, puuna, verkkona tai näiden yhdisteenä. Yleisimmässä mahdollisessa tapauksessa piirteiden esittämisestä ei voida olettaa mitään – tiedossa ovat ainoastaan dokumenttien väliset etäisyydet määrätyllä samanlaisuusmitalla. Tällöin klusterointi on suoritettava jollakin etäisyyspohjaisella menetelmällä tai dokumentit on projisoitava samanlaisuusmatriisin perusteella johonkin vektoriavaruuteen.

Varhaisimmat perustelut yhdistetylle hakutavalle esittivät Fox *et al.* [59] laajennetun vektorimallin yhteydessä ns. yhdistämisen periaatteena (*principle of combination*), jonka mukaan *tehokas informaatiolajien yhdistäminen johtaa parempaan tiedonhakuun*. Foxin mallissa käytettiin bibliografista tietokantaa, jossa omat vektorinsa muodostivat kirjoittaja, ACM-luokitus, julkaisupäivä, indeksitermit, kytkeytyminen, linkit ja yhteisviittaukset. Laajennetun vektorimallin ongelmana on lähinnä riippuvuus vektoripohjaisesta esityksestä, mikä ei välttämättä ole luonnollisin tapa rakenteisten dokumenttien esittämiseen. Yhdistettyä hakutapaa kutsutaan myös klusterointia käyttävien hakumenetelmien yhteydessä hybridiklusteroinniksi [140, sivut 73-79], joskin nimityksen yleisempi merkitys lienee kahden eri klusterointialgoritmin yhdistäminen siten, että ensimmäisen tulokset ovat toisen syötteitä [127, sivut 65-66] (esimerkiksi luvussa 4.3.1 esitetty Tantrumin algoritmi). Yhdistetty hakutapa ja hybridiklusterointi voidaan lukea myös osaksi fuusiohakua (erityisesti paradigmafuusiota) [140, sivut 127-130], mutta koska fuusiohaku on muilta osin yhdistettyä hakutapaa laajempi käsite, se käsitellään erillään luvussa 5.5.

WWW-ympäristössä yhdistetystä hakutavasta on erityistä etua, koska hakutulosten luotettavuus paranee. Luvussa 5.2 mainittu väärinkäyttö ei onnistu yhtä helposti, kun dokumenttia analysoidaan eri näkökulmista. Yang [140, sivu 73] kiteyttää puhtaan linkkipohjaisen (tai vastavasti minkä tahansa vain yhteen paradigmaan perustuvan) klusteroinnin ongelmat:

> Link-based clustering can suffer from the very nature of the links that it attempts to exploit.

Hypertekstikirjallisuudessa Frisse (katso luku 5.3.2) esitti varhaisimman mallin hierarkkisista solmuista koostuvien dokumenttien hakuun. Savoy [117] yleistää Frissen mallia yhdistettyjen hakutapojen vaatimuksia vastaavaksi niin, että jokaisella dokumenttisolmulla voi olla oma painonsa. Savoyn mallilla täsmäytysfunktion R arvo kyselyllä q dokumentille j on

$$R_{(c+1)}(d_j, q) = R_{(c)}(d_j, q) + \sum_{k \in \mathbf{S}_j} \alpha_{jk} R_{(c)}(d_k, q),\ c < t$$

missä $\mathbf{S}_j$ on dokumentin j alidokumenttien indeksijoukko, α_{jk} on linkin $j \rightarrow k$ ”vahvuus”, $R_{(0)}(d_j, q) = sim(d_j, q)$ (sim on samanlaisuusfunktio) ja t on rekursion syvyys. Jos α_{jk}:n arvoksi asetetaan $\frac{1}{|\mathbf{S}_j|}$ ja t valitaan dokumenttipuun syvyyden mukaisesti, kaavasta saadaan erikoistapauksena Frissen hakumalli. Toinen Savoyn mallin yleistys on idea, ettei ”linkin” tarvitse välttämättä tarkoittaa suoraa viittausta dokumentista toiseen, vaan ne voidaan muodostaa esim. kytkennän, yhteisviittauksen tai tekstipohjaisen samanlaisuusvertailun tuloksena saatavina *lähimmän naapurin linkkeinä*. Linkkien ei tarvitse olla hierarkkisia, vaan syklit voivat olla joillakin linkkityypeillä mahdollisia (tämän takia täsmäytysfunktiossa tarvitaan t-vakiota). Savoyn malli on heuristiikka Croftin & Turtlen probabilistiselle päättelymallille [34], jossa dokumentit ja niiden väliset linkit mallinnetaan Bayes-verkkona.

Klusterointia käsittelevässä kirjallisuudessa varhaisimmat ideat erityyppisten piirteiden yhdistämisestä ovat Yangin [140, sivut 73-79] mukaan Pirollin *et al.* ja Weissin *et al.* järjestelmät. Pirollin mallissa dokumentti esitetään vektorina, jonka komponentit on johdettu linkeistä, indeksitermeistä, käyttäjädatasta ja muusta metatiedosta. Klusterointi tehdään kullekin piirretyypille erikseen, jolloin tuloksena on useita rinnakkaisia ryhmitelmiä [110]. Lähempänä varsinaista yhdistettyä hakutapaa on Weissin HyPursuit-hakukone, joka käyttää hierarkkista klusterointia yhteisetäisyyden menetelmällä. Klusteroinnissa käytetty etäisyysmitta muodostetaan teksti- ja linkkikomponenttien maksimiarvosta. Tekstikomponentti on standardi kosinimitta $tf \times idf$-painotuksella. Linkkikomponentti muodostuu dokumenttien lyhimmän polun sekä yhteisten vanhempi- ja lapsidokumenttien lukumäärän lineaarisesta kombinaatiosta [135].

Modhan & Spanglerin [106] malli on esimerkki uudemmasta yhdistettyä hakutapaa käyttävästä klusterointimenetelmästä, joka perusteiltaan on tosin laajennetun vektorimallin mukainen. Lähtökohtana on tietty hakutulosjoukko $\mathbf{Q}$, joka on osa laajempaa dokumenttikokoelmaa. $\mathbf{Q}$:n dokumentit esitetään (D, F, B)-vektorikolmikkona, missä D on $tf \times idf$-painotettu d-ulotteinen sanavektori, F on *lähdevektori*, jonka komponentit ovat ne $\mathbf{Q}$:n ja $\mathbf{Q}$:sta lähtevien linkkien päässä olevat dokumentit, joihin on ainakin kaksi viittausta $\mathbf{Q}$-joukkoon kuuluvista dokumenteista (f kappaletta). b-ulotteinen *viittausvektori* B muodostetaan vastaavasti joukkoon $\mathbf{Q}$ tulevien linkkien perusteella. Kukin vektorikomponentti normalisoidaan yksikköpituuteen. Samanlaisuusmittana käytetään painotettua sisätuloa:

$$sim_{MS}(a, b) = \alpha_d D_a^T D_b + \alpha_f F_a^T F_b + \alpha_b B_a^T B_b,$$

missä α-kertoimien summan on oltava 1. Malli voidaan tulkita geometrisesti niin, että kukin komponenttivektori sijaitsee d, f tai b-ulotteisen yksikköpallon pinnalla. Tällöin dokumenttivektorikolmikko on yksikköpallojen tuloavaruudessa, joka on $d + f + b$ -ulotteinen torus. Tämän perusteella johdetaan K-means-algoritmin muunnelma, joka toimii euklidisen ava-

ruuden sijasta toruspinnalla. Kiinnostavaa Modhan mallissa on myös klusterien esitystapa käyttäjälle, joka ottaa huomioon sekä prototyypit että klustereihin yleisesti liittyvät piirteet. Klusterien esittämistä käsitellään tarkemmin luvussa 6.2.2.

Yhdistettyjen hakutapojen haaste on dokumenttien hierarkkisen rakenteen ja linkkien huomiointi oikealla granulariteetilla suhteessa niiden sisältöön. Hypertekstipohjaiset järjestelmät huomioivat tyypillisesti dokumenttien tekstin ja linkit (jolloin yksittäinen dokumentti tulkitaan monesta hierarkkisesta solmusta koostuvaksi ”hyperdokumentiksi”). Toisaalta rakenteiset hakukielet huomioivat yksittäisten dokumenttien rakenteen, mutta linkeillä ei yleensä ole erityisasemaa, vaan ne tulkitaan tavanomaisena attribuuttitietona. Monipuolisin lähdemateriaalista löydetty malli oli Yangilla [140, sivut 138-143], jossa tiedonhaussa hyödynnettiin teksti- ja linkkitiedon lisäksi web-aihehakemistoihin pohjautuvaa dokumenttien luokittelua. Jos samanlaisuusmitta koostuu monesta painotetusta komponentista, tämän tutkielman puitteissa avoimeksi kysymykseksi jää, mikä on ”paras” tapa painottaa komponentit (vai onko sitä – katso luku 5.5). Ongelma on ollut esillä Foxin *et al.* alkuperäisestä artikkelista [59] lähtien, mutta sen käsittely sivuutetaan tässä.

5.5 Fuusiohaku ja klusterointi

Fuusiohaku on yleistys yhdistetyille hakutavoille. Se on samalla yleinen viitekehys, jolla voidaan hahmottaa lähes kaikki tässä tutkielmassa mainitut algoritmien tai toiminnallisuuden yhdistämisideat. Fuusiohaun perusidea on havainto, että erilaisten (jopa yksinään heikoiksi tiedettyjen) tiedonhakustrategioiden yhdistäminen parantaa haun laatua lähes riippumatta siitä, mitä strategioita yhdistetään (vastaava periaate on havaittu myös mm. luokittelijoita ja neuroverkkoja yhdistettäessä [79, sivut 351-391]). Fuusiohaun tulos on yleensä parempi kuin minkään yksittäisen hakumallin käyttö, eli kokonaisuus on enemmän kuin osiensa summa [140, sivut 106-109]. Foxin *et al.* [59] *yhdistämisen periaate* on fuusiohaun lähtökohta. Näkökulmaero yhdistettyjen hakutapojen ja fuusiohaun välillä on, että edellinen käyttää samaa hakumallia monimuotoisten piirteiden yhdistämiseen, kun taas jälkimmäinen yhdistää eri hakumallien (tai hakukoneiden) tuloksia (käsittäen myös yhdistetyt hakutavat).

Fuusiohakua voidaan tarkastella eri näkökulmista riippuen siitä, mitä yhdistetään ja miten. Karkea jako voidaan tehdä hakutulosten yhdistämisajankohdan suhteen, joka voidaan tehdä ennen relevanssiarvion laskentaa (jolloin yhdistämisen kohteena on samanlaisuusmitta) tai hakukoneiden tuottamien relevanssiarvioiden perusteella [140, sivu 165]. Yhdistetyt hakutavat ja varhaisimmat fuusiohakusovellukset kuuluvat edelliseen ryhmään. Uudemmat sovellukset käyttävät samanlaisuusmittojen sijaan yleensä relevanssiarvioiden yhdistämistä, koska

metahakusovelluksissa tämä lienee yksinkertaisin lähestymistapa. Yang [140, sivut 106-130] jakaa fuusiohaun tutkimuksen seuraaviin ryhmiin niiden kohdealueen mukaan:

- **Datafuusio** tarkoittaa dokumentin ja/tai kyselyn esittämistä useilla rinnakkaisilla tavoilla ja hakutuloksen muodostamista niiden yhdistelmänä. Haku kohdistuu vain yhteen dokumenttikokoelmaan. Paradigmafuusiosta poiketen haun komponentit ovat käsitteellisesti samanlaisia (esim. useita indeksitermien painotusmalleja yhdistävä sanahaku on datafuusiota, mutta teksti- ja linkkitietoja käyttävä yhdistetty hakutapa kuuluu paradigmafuusioon). Esimerkki ennen hakutulosten laskentaa tehtävästä datafuusiosta on dokumentin esittäminen otsikolla ja avainsanoilla, joista kumpaakin verrataan kyselyyn esim. vektorimallilla ja tuloksena saadaan kaksi samanlaisuusmittaa. Toisaalta haku samasta kokoelmasta vektorimallia ja Boolen hakumallia käyttävillä koneilla on hakutulosten jälkeistä datafuusiota, koska yhdistettävänä on kaksi tuloslistaa.

- **Kokoelmafuusio** on yhdistelmä erillisiä tai osittain samoja kokoelmia indeksoivien hakukoneiden tuloksista. Datafuusiosta poiketen kokoelmafuusion painopisteenä on hajautettujen ja muodoltaan vaihtelevien tuloslistojen tehokas yhdistäminen ja uudelleenjärjestely, jonka ansiosta käyttäjä näkee yhtenäisen kokoelman [140, sivu 124]. Metahakukoneet, kuten Clusty ja KartOO[3] ovat esimerkkejä kokoelmafuusiosta.

- **Paradigmafuusio** on useiden eri hakuparadigmojen yhdistämistä. Yang [140, sivu 127-130] laskee paradigmafuusioon lähinnä teksti- ja linkkihaun sisältävät yhdistetyt hakutavat, mutta yhtälailla luokittelun ja klusteroinnin. Tässä mielessä siis useimmat luvussa 5.4 mainitut haku- ja klusterointimenetelmät kuuluvat paradigmafuusioon. Tämän kirjoittajan mielestä myös rakennehaun (katso luku 5.3.2) pitäisi kuulua paradigmafuusioon, koska rakennetta hyödyntävä dokumenttien esittäminen ja indeksointi vaatii uudenlaisia esitystapoja tekstidokumentteihin verrattuna. Paradigmafuusiolle ei tällä hetkellä ole tiedossa formaalia mallia tai edes määritelmää, mikä on ymmärrettävää yhdistämistapojen moninaisuudesta johtuen. Useimmiten eri paradigmoja sovelletaan peräkkäisinä (esim. hybridiklusterointi), rinnakkaisina (esim. laajennettu vektorimalli) tai integroituna jollakin menetelmäkohtaisella tekniikalla [140, sivu 128].

Varhainen lähestymistapa fuusiohakuun on läheisyysmittojen yhdistäminen: kukin hakukone vertaa kyselyä dokumenttiin omalla hakumallillaan ja tuloksena saadaan samanlaisuusarvio väliltä $[0, 1]$. Foxin & Shaw'n [60] esittämät mitat ovat samanlaisuusarvojen maksimi, minimi, mediaani sekä summa – joko sellaisenaan, jaettuna tai kerrottuna positiivisten samanlaisuusarvojen määrällä. Samanlaisuusmitan valinta riippuu hakukoneiden keskinäisestä laadusta ja dokumenttikokoelmasta. Esimerkiksi minimimitta minimoi todennäköisyyden sille,

[3] http://www.kartoo.com

että epärelevantti dokumentti saa hyvän tuloksen, kun taas maksimimitta korostaa relevanttien dokumenttien tulosta. Mittoja voidaan käyttää myös haun jälkeiseen tuloslistojen yhdistämiseen. Tällöin kaavoissa käytetään samanlaisuuden sijaan relevanssiarviota (joka esim. vektorimallissa on myös samanlaisuusarvio) tai suoraan dokumentin paikasta tuloslistassa johdettua arvoa. Monipuolisempi tapa tulosten yhdistämiseen on käyttää laajennetun vektorimallin tapaan painotettua lineaarista summaa, jossa summan komponentit ovat hakukoneiden palauttamat relevanssiarviot [140, sivu 167-171]. Kuten laajennetussa vektorimallissa, ”parhaiden” painojen määrittäminen on epätriviaali tehtävä ja sen käsittely sivuutetaan.

Fuusiojärjestelmän toimivuutta voidaan intuitiivisesti perustella sillä, että tiedonhakuprosessi on liian monimutkainen ja epävarma millekään yksittäiselle lähestymistavalle. Mallien yhdistäminen tuo erilaisia näkökulmia hakuprosessiin ja parantaa sen laatua. Erityisesti datafuusiossa perusteluna on käytetty sitä, että eri hakukoneiden palauttamat hakutulokset ovat ”todistusaineistoa” tietyn dokumentin relevanssista [140, sivu 109]. Fox *et al.* [60] havaitsivat, että parhaat hakutulokset saatiin, kun yhdistetään keskenään mahdollisimman *erilaisia* hakumalleja tai esitystapoja. Lee [97] on selittänyt fuusiojärjestelmien suorituskykyä sillä, että hakujärjestelmät palauttavat samoja relevantteja dokumentteja, mutta toisistaan eroavia epärelevantteja dokumentteja. Tämä voidaan testata empiirisesti Leen päällekkäiskertoimen avulla, joka määritellään seuraavasti n:llä hakukoneella:

$$R_{overlap} = \frac{R \times n}{\sum_{k=1}^{n} R_k},$$

missä R on relevanttien dokumenttien kokonaismäärä ja R_k on k:nnen hakukoneen palauttamien relevanttien dokumenttien määrä. Päällekkäiskerroin voidaan määritellä vastaavasti epärelevanteille dokumenteille. Kertoimilla voidaan arvioida fuusiojärjestelmän laatua samaan tapaan kuin F-mitalla (katso luku 4.4) arvioidaan tiedonhakujärjestelmän laatua.

Zhang *et al.* [142] ilmaisevat fuusiohaun tavoitteen ns. fuusiohypoteesin muodossa: *Eri järjestyslistoissa relevanttien dokumenttien päällekkäisyys on suuri*. Hypoteesi muistuttaa tarkoituksellisesti klusterointihypoteesia (katso luku 4.2), koska heidän mallinsa pyrkii yhdistämään fuusiohaun klusteripohjaiseen hakuun. Aluksi jokaisen fuusiossa mukana olevan hakujärjestelmän tulokset klusteroidaan erikseen. Klusterointihypoteesin perusteella relevantit dokumentit keskittyvät vain muutamaan klusteriin, ja ne pyritään löytämään fuusiohypoteesia hyödyntäen. Kullekin klusterille määritetään reliabiliteetti, joka on sitä suurempi, mitä enemmän klusterissa on yhteisiä dokumentteja toisen hakukoneen tuottaman klusterin kanssa. Tämän jälkeen kussakin klusterissa olevien dokumenttien relevanssiarvioita muokataan reliabiliteetin mukaan. Lopuksi hakukoneiden tulokset yhdistetään jollakin tyypillisellä fuusiokaavalla, kuten relevanssien summalla.

6 ExtMiner-sovellus

Luvussa kuvataan tutkielman empiirinen osuus: ExtMiner-haku- ja klusterointisovellus, joka on osittain kehitetty Knowledge Mining -projektin yhteydessä vuoden 2004 aikana. Sovellus jäi projektin aikana prototyyppitasolle, mutta sen perustoiminnot toimivat ja sitä on käytetty tiedonhakuun useista dokumenttikokoelmista. Sovelluksessa on käytettävien algoritmien, indeksointi- ja näytintekniikoiden suhteen avoin arkkitehtuuri. Lisäksi sovelluksessa on oma haku- ja klusterointimalli, joka perustuu erilaisten menetelmien yhdistämiseen.

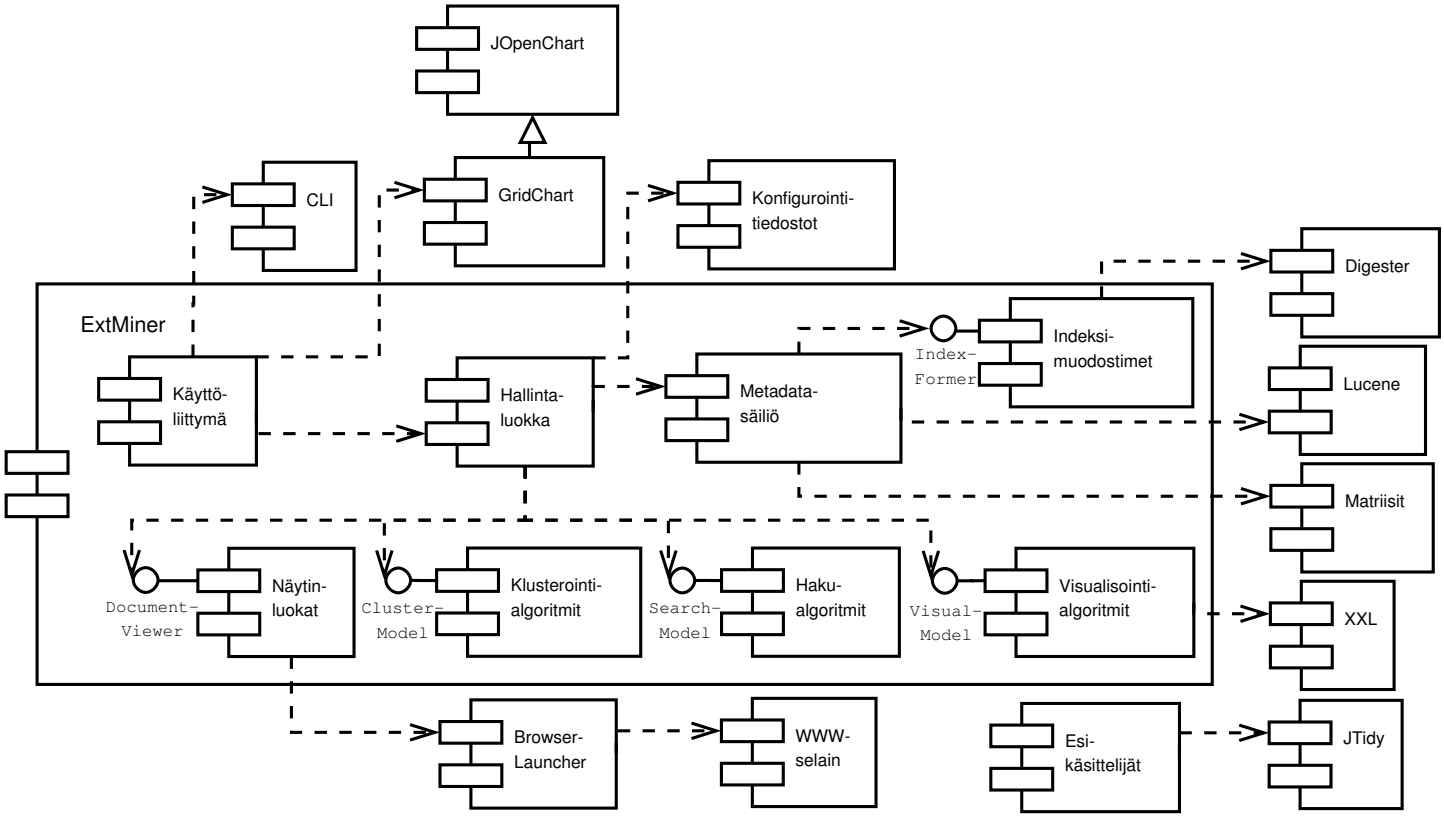
6.1 Arkkitehtuuri

ExtMiner-sovellus on julkaistu avoimen lähdekoodin ohjelmistona (myös *vapaaohjelma*-nimitystä käytetään [85]) MIT/X-lisenssin alaisuudessa (poikkeuksena LGPL-lisenssillä julkaistu GridChart-komponentti). Lisenssi on listattu Open Source Initiativen[1] avoimen lähdekoodin lisensseissä ja on lisäksi GPL-yhteensopiva Free Software Foundationin[2] kriteerien mukaan. MIT/X sallii sovellukselle lähdekoodeineen lähes minkä tahansa käytön, muokkauksen ja jopa uudelleenlisensoinnin. Motivaationa mahdollisimman sallivalle lisenssille on tavoite yleishyödyllisestä sovelluksesta, joka saavuttaa ainakin teoriassa mahdollisimman laajan käyttäjä- ja kehittäjäkunnan – kaupalliset sovellukset mukaanlukien. Ohjelmisto hyödyntää merkittävästi muita avoimen lähdekoodin komponentteja, joten on asianmukaista julkaista myös omat tuotokset vastaavasti. Lisäksi tämän kirjoittajan mielestä on tärkeää, että ohjelmien lähdekoodi on saatavilla ja että käyttäjät saavat muuttaa ja levittää ohjelmiaan. Tämä helpottaa kehittäjiä uudelleenkäytön muodossa ja palvelee loppukäyttäjiä mahdollistamalla omaan käyttöön tehdyt korjaukset. Myös yritykset hyötyvät avoimista ohjelmistoista uusien liiketoimintamallien muodossa. MIT/X-lisenssi on liitteenä B.

Sovelluksen arkkitehtuurista on pyritty tekemään mahdollisimman modulaarinen niin, että se sopeutuu eri sovellusalueisiin. Keskeisiä käytön aikana vaihdettavia komponentteja ovat käytetty haku-, klusterointi- ja visualisointialgoritmi. Jokaiselle dokumenttikokoelmalle voidaan määritellä oma indeksimuodostin, joka vastaa XML:n jäsennyksestä ja indeksoitavista kentistä. Lisäksi dokumenttien näyttämistä varten voidaan määritellä oma näytinluokka, joka

[1] http://www.opensource.org/
[2] http://www.fsf.org/

voi vaihdella teksti-ikkunasta järjestelmän web-selaimeen. Modulaarisuuden lisäksi järjestelmä on jaettu yleisellä tasolla kerroksiin siten, että käyttöliittymä, konfigurointi- ja hallintatoiminnot sekä indeksin käsittely ovat eristettyjä toisistaan. Toteutuksen tasolla järjestelmä nojaa hyvin vahvasti Java-kielen rajapintoihin. Arkkitehtuuri on esitetty komponenttitasolla kuvassa 6.1.



Kuva 6.1: ExtMiner-järjestelmän arkkitehtuuri.

Järjestelmässä on hyödynnetty seuraavia avoimen lähdekoodin komponentteja:

- **BrowserLauncher**[3] on Eric Albertin kehittämä komponentti, joka mahdollistaa järjestelmän WWW-selaimen avaamisen Java-sovelluksesta. Osa ExtMinerin näytinluokista käyttää komponenttia. BrowserLauncher on julkaistu MIT/X-tyylisellä sallivalla lisenssillä.
- **CLI**[4] on Apache Foundationin Jakarta-projektin kehittämä kirjasto, joka helpottaa komentoriviparametrien käsittelyä. Kirjasto on käyttöliittymäluokkien käytössä. CLI on julkaistu Apache 2.0 -lisenssillä.

[3] http://browserlauncher.sourceforge.net/
[4] http://jakarta.apache.org/commons/cli/

- **Digester**[5] on myös Jakarta-projektin kehittämä XML-jäsennin, joka yksinkertaistaa SAX-rajapinnan käyttöä. ExtMinerin indeksimuodostimet käyttävät jäsennintä dokumenttien indeksointivaiheessa. Digester on julkaistu Apache 2.0 -lisenssillä.

- **Lucene**[6] on Jakarta-projektin kehittämä hakukone, joka on suunniteltu tekstidokumenttien käsittelyyn, mutta mahdollistaa myös muiden dokumenttien kenttäpohjaisen indeksoinnin. Tällä hetkellä suurin osa ExtMinerin hakutoiminnoista hyödyntää Lucenea. Myös klusteroinnissa käytettävät matriisit muodostetaan Lucenen indeksin perusteella (joskin $tf \times idf$-painotus lasketaan sovelluksen puolella). Muiden Jakarta-komponenttien tapaan myös Lucene on Apache 2.0 -lisenssin alainen.

- **JOpenChart**[7] on Sebastian Müllerin kehittämä komponentti käyrien, pistekuvioiden ja muiden diagrammien esittämiseen. Käyttöliittymään sisältyvä visualisointinäkymä käyttää komponentista kehitettyä interaktiivista GridChart-ruudukkoa. JOpenChart on LGPL-lisenssin alainen.

- **JTidy**[8] on Java-versio Dave Raggettin kehittämästä HTML-tiedostojen siistimisohjelmasta, HTML Tidysta. Se ei ole varsinaisesti ExtMiner-järjestelmän käytössä, vaan sitä käytetään esikäsittelyvaiheessa WWW-sivuista koostuvilla dokumenttikokoelmilla. JTidy on julkaistu MIT/X-tyylisellä lisenssillä.

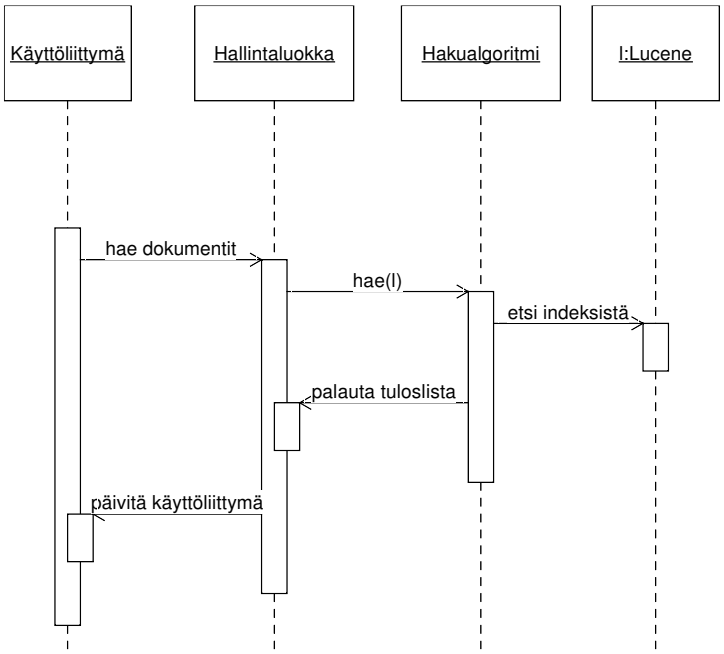
- **XXL**[9] on Marburgin Philipps-yliopistossa kehitetty laaja kirjasto tietokantojen ja erityisesti kyselyjen käsittelyyn. Minimaalinen osa kirjastosta sisältää myös FastMap-projektioalgoritmin, jota sovellus käyttää klusterimallin visualisoinnissa. XXL on julkaistu LGPL-lisenssillä.

Seuraavat komponentit ja osajärjestelmät on suunniteltu ja toteutettu itse:

- **Esikäsittelijät**. Kyseessä ei ole varsinaisesti komponentti, vaan joukko dokumenttikokoelmariippuvaisia sekalaisia mekanismeja, joiden tarkoituksena on muokata kohdedokumentit indeksimuodostimien käsiteltävään muotoon. Esikäsittelijät voivat olla esim. JTidya käyttäviä Java-luokkia, XSLT-tyylitiedostoja tai komentojonoja, joilla ohjataan keskitetysti merkkijono-operaatioita esim. sed-sovelluksella. Esikäsittelijät ovat täysin erillään varsinaisesta sovelluksesta, joskin jatkokehityksen aikana ne liitetäneen ylläpitosyistä osaksi indeksimuodostimia.

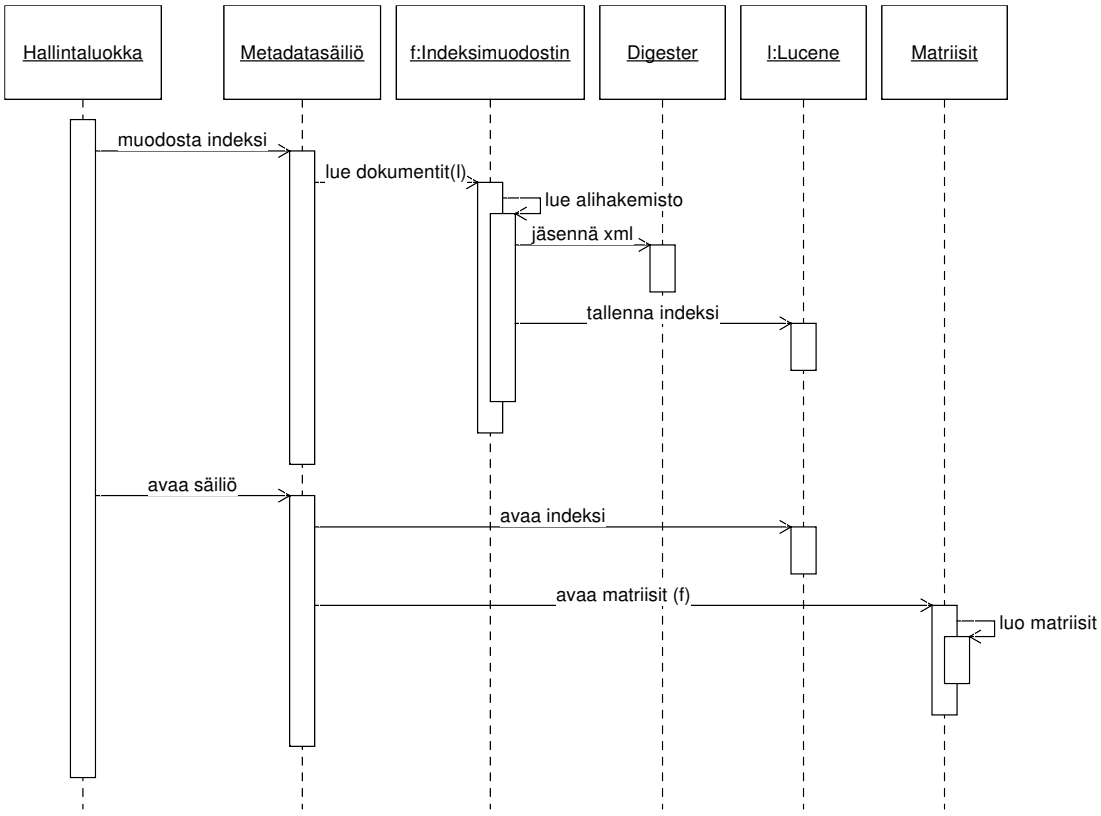
[5] http://jakarta.apache.org/commons/digester/
[6] http://jakarta.apache.org/lucene/
[7] http://jopenchart.sourceforge.net/
[8] http://jtidy.sourceforge.net/
[9] http://dbs.mathematik.uni-marburg.de/research/projects/xxl/

- **GridChart** on JOpenChart-komponentin pohjalta kehitetty interaktiivinen ruudukko, joka tukee alueiden valintaa ja zoomausta. Lisäksi ruudukossa esitettävien alkioiden värit määritellään automaattisesti tietojoukkojen määrästä riippumatta. GridChartiin on kopioitu hieman JOpenChartin koodia ja se on kiinteästi riippuvainen alkuperäisestä komponentista, joten tämän kirjoittajan LGPL-tulkinnan mukaan se on julkaistava muusta sovelluksesta erillään alkuperäisen komponentin lisenssillä.
- **Hakualgoritmit** ovat toteutuksia rajapinnalle, joka ottaa vastaan hakulausekkeen ja palauttaa relevanssiarvioilla varustetun tuloslistan. Tämän kirjoitushetkellä sovellukseen on toteutettu ainoastaan Lucene-kenttähaku, mutta rajapinta mahdollistaisi myös esim. linkkipohjaisen haun. Hakutoiminto on esitetty yleisellä tasolla kuvassa 6.2.



Kuva 6.2: Sekvenssikaavio hausta.

- **Hallintaluokka** on käyttöliittymäluokkien näkymä järjestelmään. Sen päätarkoitus on eristää käyttöliittymä indeksidatan tallennusmekanismista, jolloin sisäisen hakukoneen voisi ainakin teoriassa vaihtaa Lucenesta joksikin muuksi. Suurin osa luokan toiminnoista on lähinnä viestien ohjaamista muille luokille.
- **Indeksimuodostimet** ovat järjestelmän keskeisin mekanismi eri dokumenttityyppien käsittelyyn. Indeksimuodostin ottaa vastaan tiedoston ja palauttaa Lucene-dokumentin (käytännössä kenttäpohjaiset indeksitiedot). Tärkein tapa indeksin muodostukseen on sääntöjen määritys Digester-jäsentimelle (XML-elementtien liittäminen eri kenttiin), mutta yhtä lailla indeksimuodostin voisi lukea esim. tekstitiedostoja. Indeksin muodostusprosessi on esitetty yleisellä tasolla kuvassa 6.3.

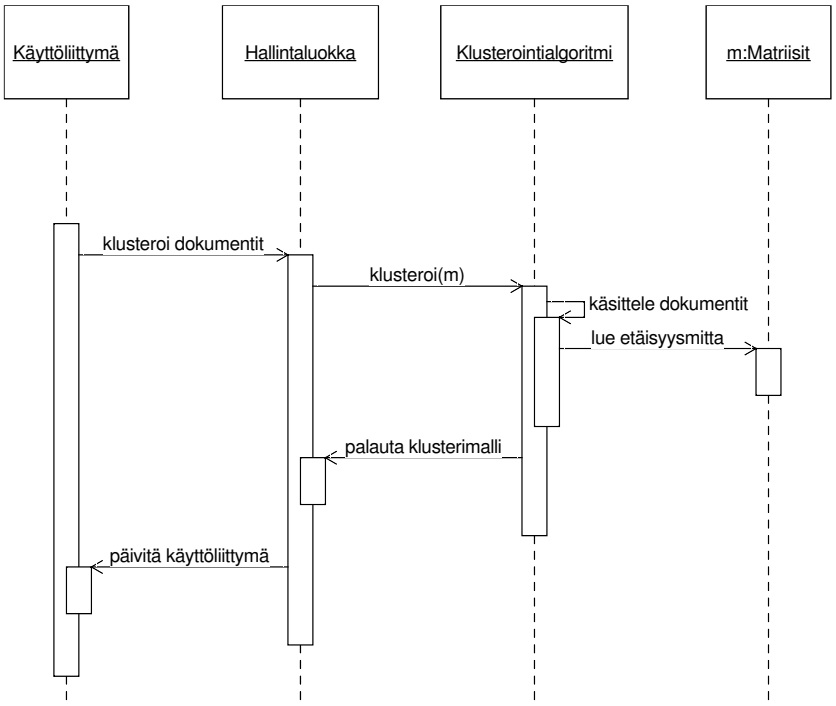


Kuva 6.3: Sekvenssikaavio indeksin muodostuksesta.

- **Klusterointialgoritmit** ovat toteutuksia rajapinnalle, joka määrittelee potentiaalisesti hierarkkisen klusterimallin. Tämän kirjoitushetkellä toteutettuja algoritmeja ovat DBSCAN ja keskipisteen menetelmään perustuva hierarkkinen klusterointi (katso luku 4.3.1). Hakualgoritmien tapaan käyttäjä voi määritellä, mikä osajoukko dokumenttikokoelmasta klusteroidaan. Klusterointitoiminto on esitetty yleisellä tasolla kuvassa 6.4.

- **Konfigurointimekanismi**. Sovelluksen asetukset määritellään konfigurointitiedostolla, jota hallintaluokan eristämä komponentti lukee. Tiedostossa määritellään hakemistot dokumenteille ja indekseille, nimikriteeri luettaville tiedostoille, määrityksiä dokumenttien paikallisuuden ja näyttötavan suhteen, klusterointialgoritmin parametrit, indeksimuodostinluokka ja näytinluokka. Jälkimmäiset ilmaistaan suoraan Java-luokkien niminä, mikä mahdollistaa järjestelmän laajentamisen uusilla muodostin- ja näyttömekanismeilla. Tarkempi esimerkki konfigurointimekanismista on luvussa 6.3.1.

- **Käyttöliittymä** on toteutettu Swing-pohjaisena sisältäen klusteripuun, tuloslistan ja visuaalisen esityksen dokumenttikokoelmasta. Käyttöliittymän eri osa-alueita käsitel-



Kuva 6.4: Sekvenssikaavio klusteroinnista.

lään seuraavissa luvuissa. Hallintaluokka mahdollistaa järjestelmän käyttämisen periaatteessa monella eri käyttöliittymällä.

- **Matriisit** ovat järjestelmän esitystapa klusteroitavalle tiedolle, joista tärkeimpiä ovat termi- linkki- ja samanlaisuusmatriisit. Matriisit lasketaan indeksointivaiheen yhteydessä ja tallennetaan pysyvästi Java-kielen sarjallistamista käyttäen, jolloin ne ovat valmiina muistissa ohjelman eri käyttökerroilla.
- **Metadatasäiliö** on abstraktio sovelluksen indeksointidatalle, jota sijaitsee Lucene-hakukoneen indekseissä ja järjestelmän omissa matriiseissa. Lisäksi metadatasäiliö vastaa yksittäisten dokumenttiesitysten palauttamisesta muille kerroksille ja dokumenttien uudelleenindeksoinnista.
- **Näytinluokat** vastaavat dokumentin avaamisesta käyttöliittymässä. Mahdollisia näytinluokkia ovat esim. teksti-ikkuna, XML-puunäyttö tai web-selaimen avaaja. Indeksimuodostimen tapaan näytinluokan voi määrittää konfigurointitiedostossa.
- **Visualisointialgoritmit** ovat toteutuksia rajapinnalle, joka ottaa vastaan dokumenttikokoelman samanlaisuusmatriisin ja projisoi dokumentit sen perusteella 2-ulotteiseen avaruuteen. Tällä hetkellä visualisointialgoritmeista on käytössä ainoastaan FastMap-projektio.

6.2 Haku- ja klusterointimalli

ExtMiner-sovelluksen hakumallin johtoajatukset ovat integraatio ja iteratiivisuus. Toisaalta malli yhdistää tuloslistoihin perustuvan haun klusteripohjaiseen selaukseen, toisaalta itse haku- ja klusterointialgoritmien laskennan tulos kootaan sovellusalueesta riippuen monesta eri komponentista. Iteratiivisuus tarkoittaa tässä yhteydessä mahdollisuutta haun ja klusterimallin vaiheittaiseen ja rinnakkaiseen tarkentamiseen.

Sovelluksen haku- ja klusterointitoiminnot ovat data- tai paradigmafuusion sovelluksia. Klusteroinnissa käytettävät piirteet yhdistetään laajennetun vektorimallin avulla yhdeksi samanlaisuusarvoksi. Myös hakualgoritmeista saadut relevanssiarviot yhdistetään painotetulla lineaarisella summalla yhdeksi relevanssiksi. Haku- ja klusterimallien pohjalla on yhteinen kenttäpohjainen indeksi, jonka komponentteina voivat olla indeksitermit, valinnaisista dokumentin elementeistä koostetut tekstikentät (esim. otsikot, kuvatekstit), linkkimatriisi tai metatietokentät. Malli mahdollistaa useiden haku- ja klusterointialgoritmien integroinnin järjestelmään. Klusterointialgoritmin osalta ainoa vaatimus sovelluksen kannalta on toiminta metrisessä avaruudessa. Hakualgoritmin täytyy palauttaa järjestetty hakutuloslista käyttäjän kyselyn pohjalta. Lisäksi hakua tai klusterointia täytyy pystyä tarvittaessa soveltamaan mielivaltaiseen dokumenttikokoelman osaan, mikä mahdollistaa yhdistetyn hakuprosessin.

Lineaarisia malleja ei painoteta sovelluksessa automaattisesti, vaan ne jätetään käyttäjälle haun apuvälineiksi. Sekä laajennetun vektorimallin että lineaariseen summaan perustuvan fuusiohaun keskeinen ongelma on ollut ”oikeiden” painojen etsiminen jollakin optimointitekniikalla. Tämän kirjoittajan näkökulmasta ei välttämättä ole mielekästä kysyä, mikä on paras ”yleinen” painotustapa. Kiinnostavampaa on, miten painotukset suhtautuvat käyttäjän tietotarpeisiin ja dokumenttikokoelman rakenteisuusasteeseen. Dokumenttien klusterointi ensisijaisesti tekijää ja julkaisuvuotta painottaen vastaa eri kysymykseen kuin klusterointi avainsanojen mukaan. WWW:ssä ja tieteellisissä artikkeleissa linkit ovat ensiarvoisen tärkeää semanttista lisätietoa, mutta linkkien painottaminen ei ole mielekästä, jos niitä ei ole useimmissa dokumenteissa tai dokumenttiformaatti ei niitä tue. Rikkaan rakenteisen kuvauskielen sisältävä dokumenttikokoelma hyötyy ilmaisuvoimaisesta XPath-tyylisestä kyselykielestä, kun taas XHTML-kielisille dokumenteille saattaa riittää yksinkertaisempi otsikoista, kokotekstistä, ankkuri-ikkunoista ja linkkirakenteesta koottu yhdistelmämitta. Täydellisessä maailmassa jokainen tietoyksikkö merkittäisiin tietysti myös RDF-metakuvauksilla jonkin asianmukaisen ontologian mukaan, jolloin edellisten lisäksi tarvittaisiin myös metatietoon kohdistuvaa semanttista hakua [12] päättelyineen.

6.2.1 Hakutulosten klusterointia vai hakua klustereista?

ExtMiner-sovellus yhdistää joitakin klusteripohjaiseen hakuun liittyviä ideoita, jotka ovat esiintyneet erillisinä aiemmissa hakusovelluksissa. Klusteroinnin käyttämistä tiedonhaussa yleisesti on käsitelty luvussa 4.2.

1. **Iteratiivinen haku- ja klusterointiprosessi.** Sanahaku ja dokumenttien klusterointi ovat perusoperaatioita, joita voidaan suorittaa iteratiivisesti ja tarkentaen sopivaksi katsottuun alueeseen. Sovellusta käynnistettäessä käyttäjä näkee alustavan klusterimalliin perustuvan näkymän, mutta tämän jälkeen haku voidaan kohdistaa esim. tiettyyn klusteriin tai muodostaa uusi klusterimalli sanahaun tuloslistan pohjalta. Prosessia voidaan jatkaa ja tarkentaa tarvittaessa yksittäisiin käyttäjän valitsemiin dokumentteihin asti. Alkuperäinen idea haun ja klusteroinnin yhdistämisestä esitettiin Scatter/Gather-järjestelmässä: käyttäjän hakustrategia nähdään osana jatkuvaa mallia, jonka toisessa ääripäässä on klusteripohjainen selaus ja toisessa sanahaku [37]. Selaus soveltuu tilanteeseen, jossa käyttäjällä ei ole selkeää päämäärää, vaan tarkoituksena on tutustua dokumenttikokoelmaan yleisesti. Haku soveltuu tilanteeseen, jossa käyttäjä tietää tarkasti mitä haluaa. Tämän yhtenä mahdollisena edellytyksenä on kokoelman aiempi selailu, joka auttaa formuloimaan kyselyjä. Alkuperäinen Scatter/Gather keskittyi lähinnä klusteripohjaiseen selaukseen, mutta myöhemmissä tutkimuksissa [81, sivu 352] järjestelmää on sovellettu myös hakutulosten klusterointiin.

2. **Interaktiivinen klusterimalli.** Käyttäjä voi valita dokumentteja mistä tahansa sovelluksen näkymästä, joita ovat tuloslista, klusteripuu ja visuaalinen näkymä dokumenttikokoelmaan. Valinnat päivittyvät kaikkiin näkymiin ja käyttäjä näkee samalla koosteen valitusta dokumentista. Klusteripuu on interaktiivinen: käyttäjä voi merkitä tietyn klusterin ”kohinaksi” tai yhdistää tietyn klusterin aliklusterit. Tämä on käytännöllistä erityisesti tarkasteltaessa hierarkkista mallia, jossa klustereita on alustavasti liikaa käyttäjän hahmotettavaksi. Samankaltaisia ideoita on Allenin *et al.* klusterointijärjestelmän [4] käyttöliittymässä, joka samalla on luultavasti varhaisin hakutulosten klusterointiin keskittyvä järjestelmä. Hakutulokset klusteroitiin hierarkkisella algoritmilla, jonka tuloksia pystyy tarkastelemaan ”interaktiivisella dendrogrammilla”. Tuloslistojen sijaan järjestelmän painopiste on klusterien selailussa, joskin järjestelmä pystyy näyttämään visuaalisesti kokoelman dokumenttien samanlaisuuden valittuun dokumenttiin.

3. **Samanaikaiset klusteri- ja listanäkymät.** Kirjallisuudessa on esitetty runsaasti tutkimuksia, joissa on verrattu klusteripohjaisen selauksen tehokkuutta tuloslistoihin [139]. Jo Scatter/Gather-järjestelmän hakumallin yhteydessä todettiin, että tuloslistat ja klusterit tukevat erilaisia hakutehtäviä. Pelkkää hakutuloslistaa käyttämällä relevantit do-

kumentit voivat mm. moniselitteisten termien takia hajaantua ympäri listaa. Klusterointihypoteesin perusteella relevanttien dokumenttien pitäisi sijoittua samoihin klustereihin, mutta käytännön tulokset ovat tältäkin osin ristiriitaiset ja riippuvat luonnollisesti käytetyistä algoritmeista. Käyttäjän kannalta joustavin vaihtoehto on, kun järjestelmä esittää molemmat näkymät. Tämä tapa on valittu ExtMiner-sovellukseen. Leuskin & Allanin LightHouse-käyttöliittymä [99] on esimerkki varhaisemmasta klusterit ja tuloslistat yhdistävästä järjestelmästä. Järjestelmä tarjoaa käyttäjälle lista- ja klusterinäkymän lisäksi vaihtoehtoisia tuloslistoja, jotka voidaan muodostaa alkuperäisen tuloslistan, klusterimallin ja käyttäjän palautteen pohjalta.

6.2.2 Dokumenttien ja klusterien esittämisestä

Klusterien esittämisellä käyttäjälle on ratkaiseva merkitys tiedonlouhinnan käyttökelpoisuuden ja mallin ymmärrettävyyden kannalta. Hakutulosten klusterointi ja kokoelman selaus korostavat klusterointia kuvailevana menetelmänä (katso luku 2.2.3). Tällöin täsmällistä matemaattista mallia merkittävämpää on saada selkeä kuvaus klusterista. Klusterointimalli saattaa olla käyttäjän kannalta ainutkertainen, vain edellisestä hakusanasta tai hakuprosessin kuluessa valituista dokumenteista riippuva. KDD-prosessin (katso luku 2.2.1) puitteissa klusterien esittäminen ja visualisointi kuuluvat *tiedonlouhinnan* ja *tulkinnan* (vaiheet 4 ja 5) välimaastoon. Klusterien esitykset abstrahoivat tietoa ja vaikuttavat klusteroinnin jälkeiseen päätöksentekoon.

Perinteisiä tapoja klusterien esittämiseen erityisesti vektoriavaruudessa ovat menetelmästä riippuen tilastollisten jakaumien tunnusluvut ja prototyypit (esim. klusteria edustavat dokumentit). Teksti- ja rakenteisten dokumenttien osalta voidaan hyödyntää myös sovellusaluekohtaista lisätietoa (esim. esitettäessä dokumentit vektorimallilla tiedetään, että jokainen piirre vastaa dokumentin termiä), mikä helpottaa klusterien tulkintaa. Tekstipohjaisten esitysten lisäksi myös erilaiset visualisointimenetelmät tukevat klusterien esittämistä. Visualisointia käsitellään luvussa 6.3.3. Klusterien esitystavat voidaan jakaa yleisellä tasolla seuraavasti (kohdat 1-3 Jainin *et al.* [83] luettelosta):

1. **Prototyypit**, jotka ovat klusterin ”edustavia” alkioita. Vektoriavaruudessa toimivilla menetelmillä prototyypin ei tarvitse kuulua näytejoukkoon, vaan se on klusterin keskiarvo- tai mediaanipiste. Monet osittavat algoritmit ovat prototyyppipohjaisia. Periaatteessa prototyyppi voi edustaa mitä tahansa tietotyyppiä, kunhan etäisyysmitan ja keskipisteen käsitteet on määritelty. Metrisissä avaruuksissa prototyyppiä ei ole yleensä määritelty. Prototyypeillä voidaan kuvata vain symmetrisiä (tarkempi muoto riippuu etäisyysmitasta, tyypillisesti ympyrämäisiä) klustereita, jos muuta lisätietoa ei ole

käytettävissä. Yhden keskipisteen sijaan klusteri voidaan kuvata myös usealla hajallaan olevalla prototyypillä, mikä mahdollistaa muodoltaan monipuolisemmat klusterit. Näin on tehty esim. hierarkkisessa CURE-algoritmissa (*Clustering Using REpresentatives*) [73].

2. **Luokittelupuu** soveltuu tilanteeseen, jossa klusterit voidaan kuvata loogisilla lausekkeilla tai todennäköisyysehdoilla. Luokittelupuu on päätöspuun yleistys, jossa loogisten ehtojen tilalla voi olla myös todennäköisyyksiä. Luokittelupuulla voidaan kuvata käsitteellisen klusteroinnin tuloksia [76, sivu 376].

3. **Loogiset lausekkeet** joko konjunktiivisessa tai disjunktiivisessa normaalimuodossa. Käytetään lähinnä käsitteellisessä klusteroinnissa vaihtoehtoisena notaationa luokittelupuulle. Myös CLIQUE-algoritmin tulos esitetään loogisina lausekkeina.

4. **Tilastolliset jakaumat ja tunnusluvut** ovat luonnollinen tapa klusterien esittämiseen kaikille mallipohjaisille menetelmille. Tyypillisiä jakaumamalleja ovat sekatiheysmalli kovan ja MCMM pehmeän klusteroinnin yhteydessä. Mallissa käytetyistä jakaumista ja taustaoletuksista (esim. ovatko jakaumat ympyrämäisiä vai ellipsoideja) riippuu, millaisia tunnuslukuja käytetään; tavallisia ovat keskiarvovektorit ja kovarianssimatriisit. Joidenkin heuristisen algoritmien klusterimalli on tulkittavissa myös tilastollisessa kehyksessä, merkittävimpänä K-means (katso luku 4.3.4).

5. **Sovellusaluekohtaiset mallit**, jotka voivat olla algoritmikohtaisia ja jotka eivät suoraan sovellu muihin ryhmiin. Monet teksti- ja web-dokumenttien klusterointiin suunnitellut algoritmit käyttävätkin omia erikoistuneita esitystapojaan, joista käsitellään muutamia esimerkkejä edempänä.

Useimmat tekstidokumenttien klusterointimenetelmät esittävät dokumentit termivektoreina. Muutamissa menetelmissä tämä viedään klusterimallin tasolle asti: tällöin klusterin kuvaus on sille ominaisten termien tai fraasien (sanaryhmien tai lauseiden) joukko. Sanatasolla toimiva esimerkki tällaisesta algoritmista on Beilin *et al.* [8] HFTC-algoritmi, joka perustuu assosiaatiosääntöjen louhintaan. Verrattaessa algoritmia Boleyn *et al.* [16] ARHP-algoritmiin (katso luku 4.3.3) sanamatriisia käsitellään transponoidusti: ARHP etsii dokumenttijoukkoja, joilla on yhteisiä termejä, kun taas HFTC etsii mahdollisimman monessa dokumentissa esiintyviä termijoukkoja. Kuvauksesta saadaan hierarkkinen (huipulla pienemmät termijoukot) ja pehmeä (eri klusterit voivat jakaa samoja termejä). Lausetasolla vastaava esitystapa on Zamirin STC-klusteroinnissa [141, sivut 40-64], jossa klusterit kuvataan niille ominaisilla fraaseilla. Dokumentti kuuluu niihin klustereihin, joita edustavia fraaseja se sisältää.

Rakenteiset ja linkitetyt dokumentit tuovat uusia mahdollisuuksia klusterien esittämiseen, joskaan yhtä standarditapaa ei voida vielä tällä hetkellä esittää. Ilmaisuvoimaisin tapa on

käyttää jotain sanalistan ja rakennepuun yhdistelmää, esimerkiksi Schliederin & Meussin malli [119] (katso luku 5.3.2) vaikuttaa lupaavalta. On kuitenkin epäselvää, miten näin raskasta esitystä voitaisiin klusteroida tehokkaasti. Yksinkertaisempi tapa puhtaasti rakenteisen klusteroinnin kuvaukseen on esimerkiksi assosiaatiosääntöjen louhiminen hieman ARHP-algoritmia muistuttavalla tavalla. Leen *et al.* [98] algoritmissa (katso luku 5.3.3) etsitään polkujoukkoja, jotka ovat yhteisiä mahdollisimman monelle dokumentille.

Jos klusterointia sovelletaan hakutuloksiin, niiden kuvauksessa voidaan käyttää hyväksi kyselyn ominaisuuksia. Varhainen tämänsuuntainen idea oli Voorheesilla [131], joka ehdotti, että klusteripohjaisen haun tuloksena saatujen relevanttien klustereiden dokumentit voisi täsmäyttää erikseen. Hakutulosten klusterointiin sovellettuna klusterit voisi kuvata niiden relevanteimmilla dokumenteilla. Tällöin prototyyppi ei olisikaan välttämättä keskimmäinen alkio, vaan käyttäjän kyselyyn todennäköisesti sopivin dokumentti. Jos käytössä on yhdistetty hakutapa, mahdollisuudet monipuolistuvat entisestään. Linkitetyllä dokumenttikokoelmalla voitaisiin esimerkiksi kehittää Google-hakukoneen inspiroima hakumalli, jossa klusterointi suoritetaan termien perusteella, mutta klustereissa olevat dokumentit näytetään käyttäjälle niiden PageRank-arvon perusteella.

ExtMiner-sovelluksen klusterien ja dokumenttien esitystapa nojaa laajennettuun vektorimalliin. Indeksitermit ja linkit esitetään vektoreina, muut dokumentista erotetut elementit (esim. otsikot) sekä metatieto esitetään kenttäkohtaisina merkkijonoina. Yksittäisen dokumentin, klusterin tai käyttäjän valitseman dokumenttijoukon välillä ei tehdä eroa: yhden dokumentin ollessa kyseessä järjestelmä näyttää sen indeksitermit $tf \times idf$-painojen mukaisessa järjestyksessä, tulevat ja lähtevät linkit muihin dokumentteihin sekä dokumenttityypistä riippuvat kentät – kaikissa dokumenttityypeissä on vähintään jonkinlainen otsikko. Jos dokumentteja on valittuna useampia, järjestelmä laskee eri dokumenttien piirteiden pohjalta keskimääräiset tai ”tyypillisimmät” arvot. Varsinaisia prototyyppidokumentteja ei ole käytössä, joskin hakutuloksena saatuja dokumentteja voidaan tarkastella relevanssijärjestyksessä. Esitystavan etuna on riippumattomuus yksittäisestä haku- tai klusterointialgoritmista (tai jopa niiden antamista tuloksista). Käyttäjän kannalta esitystapa ei kuitenkaan ole kovin informatiivinen.

Dokumenttien esitystapa muistuttaa myös Scatter/Gather-järjestelmän esitystä, jossa klusterit esitetään prototyyppien sanavektoreina laajennettuna keskeisten klusterissa olevien dokumenttien otsikoilla [81, sivu 355]. Toinen sovellukseen vaikuttanut esitysmalli on Modhan & Spanglerin yhdistetyn hakutavan klusterointimenetelmä (katso luku 5.4), jossa klusteri esitetään seuraavilla teksti- ja linkkipohjaisilla piirteillä eli annotoinneilla:

1. **Lyhennelmä** on dokumentti, jolla on kaikkein tyypillisin sanavektori klustererin dokumenteista. Se vastaa tekstipohjaisten menetelmien prototyyppiä.

2. **Läpimurto** (*breakthrough*) on dokumentti, jolla on kaikkein tyypillisin viittauslinkkien vektori klusterin dokumenteista. Viittaukset voivat tulla klusterin ulkopuolelta.
3. **Yleiskatsaus** (*review*) on dokumentti, jolla on tyypillisin lähdelinkkien vektori. Lähdelinkit voivat mennä klusterin ulkopuolelle.
4. **Avainsanat** ovat klusterin termikomponentin prototyyppivektorin voimakkaimmin painotetut termit. Ne kuvaavat klusterin tyypillisimpiä ja kuvaavimpia sanoja.
5. **Viittaukset** ovat klusterin viittauskomponentin prototyyppivektorin voimakkaimmin painotetut linkit. Ne kuvaavat tyypillisimpiä dokumentteja, jotka viittaavat klusteriin.
6. **Lähteet** ovat klusterin lähdekomponentin prototyyppivektorin voimakkaimmin painotetut linkit. Ne kuvaavat tyyppillisimpiä dokumentteja, joihin klusterista viitataan.

6.3 Toteutusratkaisut

Sovelluksen toteutuskieleksi valittiin Java 1.4.2 ja kehitysvälineeksi NetBeans 3.6. Motivaationa valinnalle oli Javan käyttöjärjestelmäriippumattomuus ja helppo siirrettävyys erityisesti käyttöliittymän osalta. Lisäksi Java-kielelle oli saatavilla runsaasti avoimen lähdekoodin komponentteja, joita voitiin käyttää kehityksen tukena. Avainasemassa oli erityisesti Lucene-hakukone. Java 5 julkaistiin kehityksen aikana, mutta sitä ei otettu käyttöön, koska siirrettävyys haluttiin pitää mahdollisimman laajana eikä sovelluksen kaikilla testaajilla ollut julkaisuhetkellä käytössään viimeisintä versiota.

6.3.1 Indeksointi ja haku

Järjestelmä olettaa indeksointivaiheessa, että dokumentit ovat saatavilla paikallisella levyllä. Konfigurointitiedostossa määritellään kriteeri luettavien tiedostojen valintaan, sekä lukuhakemisto, josta indeksointi aloitetaan. Indeksointia jatketaan rekursiivisesti kaikille lukuhakemiston alihakemistoille. Ohessa on esimerkki HTML- ja PDF-dokumenteista koostuvan kokoelman konfigurointitiedostosta.

```
inputdir = testcollection
indexdir = testcollection
formerClass = repository.HtmlIndexFormer

viewerClass = ui.BrowserWrapper
extension = .*\.xml
viewExtensions = .pdf|.html|
```

```
localfiles = true
showAsLocal = false
overrideDTD = true
eps=0.48
minpts=6
```

Konfigurointitiedosto on muodoltaan standardi Javan `props`-tiedosto, joka muodostuu seuraavista $(avain, arvo)$-pareista:

- `inputdir` ja `indexdir` ovat syötetietojen ja indeksin hakemistojen nimet (suhteessa ohjelman omaan hakemistoon).
- `formerClass` on käytössä oleva indeksimuodostin (Java-luokan nimi).
- `viewerClass` on käytössä oleva näytinluokka (myös Java-luokan nimi; tässä tapauksessa näytin avaa järjestelmän web-selaimen).
- `extension` kuvaa säännöllisen lausekkeen, johon täsmäävät tiedostot indeksoidaan (tässä tapauksessa nimet, jotka päättyvät merkkijonoon ”`.xml`”).
- `viewExtensions` määrittelee näytettävien tiedostojen tarkenteet pystyviivalla erotettuna listana. Ohjelma olettaa, että näytettävä tiedosto on muuten samanniminen kuin indeksoitava tiedosto, mutta sen tarkenne vaihdetaan. Jos eritarkenteisia tiedostoja on olemassa useampia, valitaan nimi listan ensimmäisen tarkenteen mukaan.
- `localfiles` määrittää, käsitteleekö indeksimuodostin dokumentteja paikallisina vai otoksena WWW-sivustosta. Jos tiedostot tulkitaan WWW-sivustoksi, järjestelmä osaa muuntaa sopivasti nimetyt hakemistot URL:ksi hakusovellusta ajettaessa. Jos esimerkiksi lukuhakemiston alihakemistot on nimetty `www.cc.jyu.fi` ja `~minurmin`, se tulkitaan indeksoinnissa osoitteeksi `http://www.cc.jyu.fi/~minurmin/`. Tämä mahdollistaa järjestelmän käytön indeksoinnin jälkeen ilman alkuperäisiä dokumentteja.
- `showAsLocal` kuvaa, näytetäänkö käyttäjälle indeksoitu (yleensä XML) tiedosto vai `viewExtensions`- ja `localfiles`-tietojen perusteella muodostetun mahdollisen URL-osoitteen viittaama (esim. HTML-muotoinen) tiedosto.
- `overrideDTD` on ohje XML-jäsentimelle jättää huomiotta DTD-tiedosto. Tämä nopeuttaa tiedostojen jäsennystä, koska indeksointivaiheessa tiedostoja ei tarvitse enää validoida.
- `eps` ja `minpts` ovat parametreja DBSCAN-algoritmille (jatkokehityksessä eri algoritmeille annettavien parametrien syöttömekanismia on syytä yleistää, mutta tässä versiossa yksinkertainen ratkaisu on riittävä).

Indeksimuodostinolio vastaa dokumenttien jäsennyksestä, tietojen lukemisesta ja viennistä Lucene-hakukoneelle sekä edelleen ohjelman omille indeksimatriiseille. Lisäksi yksittäisten dokumenttien konkreettiset esitysoliot ovat indeksimuodostimen vastuulla. Järjestelmään on kirjoitettu valmiiksi yleiset indeksimuodostimet, jotka lukevat XHTML-dokumentteja tai mitä tahansa XML-tietoa. XHTML-dokumenteista erotellaan tekstin lisäksi otsikot, kuvatekstit ja linkit. Yleisestä XML-tiedosta ei erotella tekstin lisäksi mitään ylimääräisiä kenttiä, koska mielivaltaisen XML-dokumentin elementtien merkityksistä ei voida olettaa mitään. Ohjelman koekäyttöä varten kirjoitettiin lisäksi kaksi erikoistunutta muodostinta, joita käsitellään luvussa 6.4.1. Ideaa eri elementtien käytöstä omina kenttinään ovat käyttäneet myös Cutler *et al.* HTML-dokumenteille [36]. Perusmekanismin Digesterin ja Lucenen yhdistämisestä on esittänyt Gospodnetic [71]. Ohessa osa `HtmlRuleSet`-luokkaa, jolla määritellään Digesterille lukusäännöt HTML-tiedostoja varten ja joka on `HtmlIndexFormer`-luokan käytössä. HTML-tiedostoista indeksoidaan tekstin lisäksi otsikot ja kuvien `alt`-tekstit omiin kenttiinsä. Lisäksi `<script>` ja `<style>`-elementtien sisältö jätetään huomiotta.

```
public void setContent(String s) {
  if (s.length()>1) currentDoc.add(Field.Text("content",s));
}

public void setHeader(String s) {
  currentDoc.add(Field.Text("head",s));
  currentDoc.add(Field.Text("content",s));
}

public void setAlt(String s) {
  currentDoc.add(Field.Text("content",s));
}

public void setHref(String s) {
  currentDoc.add(Field.UnIndexed("outlink",s));
}

public void assignRules(Digester d) {
  d.clear();
  d.setRules(new RegexRules(new StandardRegexMatcher()));

  d.addSetProperties(".*/img","alt","alt");
  d.addSetProperties(".*/a","href","href");

  SetTextSegmentRule r = new SetTextSegmentRule("setContent");
  d.addRule("html/body/.*", r);
```

```
    addExclCallMethod(d,".*/(h1|h2|h3|h4|h5)","setHeader",0);
    addNullMethod(d,".*/(script|style)");
    addExclCallMethod(d,"html/head/title","setHeader",0);
  }
```

Lucene-indeksin muodostuksen jälkeen luodaan klusteroinnin apuna käytettävät matriisit: termimatriisi, linkkimatriisi, ominaisuusmatriisi ja tärkeimpänä samanlaisuusmatriisi. Näistä kolme ensimmäistä matriisia sisältävät oleellisesti samat tiedot kuin Lucenekin (ja mahdollisesti optimoidaan jatkokehityksen aikana pois), mutta kenttiä on painotettu eri tavoin. Samanlaisuusmatriisi puolestaan käyttää muita matriiseja apunaan dokumenttien samanlaisuuden laskennassa. Matriisit ovat harvoja, minkä vuoksi ne on toteutettu joukkoja sisältävinä assosiaatiotauluina. Tällöin hukkatilaa jää mahdollisimman vähän, mutta tietoja voidaan hakea dokumentin tunnistenumeron mukaan $\Theta(log\, n)$-ajassa (hajautustaulu on Javan `TreeMap`, joka on toteutettu punamustana puuna). Samanlaisuusmatriisi on tämän lisäksi symmetrinen, eli kaksoiskappaleita ei tallenneta. Matriisit tallennetaan Javan sarjallistamista käyttäen.

Lucene-hakukoneen indeksi on kenttäpohjainen. Sen pohjalta voidaan tehdä kokoteksti- ja kenttähakuja, mutta linkkipohjaisen relevanssin (esim. PageRank) laskemista varten tarvitaan omia hakualgoritmeja. Jos hakualgoritmeja on käytössä useita, ne yhdistetään lineaarisena summana käyttäjän määrittämillä painoilla (katso luku 5.5). Käyttäjä näkee vain yhden tuloslistan relevanssiarvioineen. Hakukielenä käytetään Lucenen omaa syntaksia, jossa oletuksena on kokotekstihaku ja termien yhdistäminen TAI-operaattorilla. $+$-merkillä voidaan määritellä jokin kenttä pakolliseksi ja :-merkillä voidaan erottaa kenttä haettavasta termistä. Indeksin muodostuksessa ja hakukyselyn luvussa käytetään oletuksena Porterin stemmausalgoritmia, joka pyrkii palauttamaan sanat perusmuotoon. Se toimii tyydyttävästi vain englanninkielisellä materiaalilla.

Sovelluksessa käytetty hakukoneiden tulosten yhdistäminen muistuttaa jonkin verran Ben-Aharonin *et al.* [9] XML-hakujärjestelmää. Järjestelmä käsittää dokumenttitasolla toimivan $tf \times idf$-painotusta hyödyntävän sanaindeksin, sanojen etäisyysindeksin sekä painotukset mahdollistavan elementti-indeksin. XPath-kyselystä generoidaan XSL-tyylitiedosto, jonka avulla potentiaalisista dokumenteista erotellaan rakenteiseen kyselyyn sopivat fragmentit. Potentiaaliset dokumentit valitaan sanaindeksin perusteella. Yksittäisiä elementtejä voidaan painottaa solmuindeksin perusteella, jonka jälkeen fragmentit syötetään arvostelufunktioille. Relevanssiarviot yhdistetään järjestämällä ne (leksikografiseen) ”aakkosjärjestykseen” siten, että kullakin arvostelufunktiolla on etukäteen määrätty paino. Aakkosjärjestyksessä ensimmäiseksi määritellyllä hakukoneella on ratkaiseva merkitys relevanssin määrityksessä. Järjestelmässä kiinnostavaa on mahdollisuus rakenteisten hakujen käyttöön, vaikka indeksi ei pääosin tue XML-tiedostojen rakennetta. Vastaavaa järjestelyä voisi jatkokehityksen aikana

harkita myös ExtMiner-sovellukseen, koska myös sen indeksi toimii kenttäpohjaisuudesta huolimatta vain dokumenttitasolla. Sen sijaan hakutulosten yhdistämisessä ExtMinerin lineaarinen painotettu summa on aakkosjärjestystä joustavampi.

6.3.2 Metriset klusterointialgoritmit

Metrisissä avaruuksissa (katso luku 4.3.3) toimivien klusterointialgoritmien on huomioitava useita tekijöitä, joilla vektoriavaruudessa ei yleensä ole merkitystä. Näitä ovat ainakin seuraavat (kohdat 1-3 Gantin *et al.* [67] esittämiä):

1. Klusterin prototyypin käsitettä ei voida määritellä täsmällisesti. Sitä voidaan kuitenkin approksimoida havaintopisteellä, jonka etäisyys muihin klusterin pisteisiin on minimaalinen. Ganti *et al.* kutsuvat tätä klustroidiksi.

2. Etäisyysfunktion arvon laskeminen voi olla huomattavasti vektoriavaruuksissa käytettyjä etäisyysmittoja vaativampaa. Esimerkiksi m- ja n-pituisten merkkijonojen editointietäisyyden (Levenshteinin etäisyys) laskennallinen vaativuus on kertaluokkaa $\Theta(mn)$, kun taas n-ulotteisten vektorien euklidinen etäisyys voidaan laskea $\Theta(n)$-ajassa.

3. Klusteroinnin sovellusalueriippuvuus aiheuttaa vaatimuksia, joihin on hankalaa vastata vain yhdellä algoritmilla. Keinoja tähän ovat hybridiklusterointi tai etäisyysmitan kokoaminen monesta eri komponentista (kuten ExtMiner-sovelluksessa on tehty).

4. Metrisen avaruuden suorituskykyinen hyödyntäminen vaatii erityisiä indeksirakenteita, joiden käsittely sivuutetaan tässä. Lisätietoa metrisistä indeksirakenteista on esim. Chávez'n *et al.* [24] kirjallisuuskatsauksessa.

Useimmat metrisessä avaruudessa toimivat algoritmit ovat diskriminatiivisia. Syynä tähän lienee se, että tilastolliset jakaumat vaativat vektorimuotoista tietoa, jolloin niiden käyttäminen ei tule kyseeseen. Metrisen avaruuden alkioiden klusteroinnin taustateoriaksi sopiikin paremmin verkkoteoria, jolloin yksittäisiin muuttujiin tai ulottuvuuksiin ei tarvitse ottaa kantaa. Diskriminatiivisuus ei ole ExtMinerissa ongelma, koska klusterimalleja ei käytetä uudelleen. Hakutulosten tai valitun alueen klusterointi luo aina uuden ryhmittelyn. Mallipohjaisuudesta olisi etua lähinnä luokittelussa. Käytettäessä klusterointia haun ja selauksen apuvälineenä on oleellisinta tarjota käyttäjälle selkeät ja intuitiiviset kuvaukset klustereista. Lisäksi klusterimallin täytyy olla muokattavissa käyttäjän etsimästä aiheesta riippuen. Tällä hetkellä mukauttaminen onnistuu sekä klusteroitavan dokumenttijoukon että etäisyysmitan painotusten suhteen. Lisäksi useimpia algoritmeja (esim. DBSCAN) voidaan edelleen parametrisoida.

ExtMiner-sovelluksen pääasialliseksi klusterointimenetelmäksi on valittu DBSCAN (katso luku 4.3.1). Syitä valintaan olivat algoritmin toiminta metrisessä avaruudessa, perustavanlaatuisuus (useampi tiheyspohjainen klusterointimenetelmä on rakennettu DBSCAN:in päälle) ja tasainen klusterointimalli (käyttäjälle huomattavasti hierarkkista mallia yksinkertaisempaa). Lisäksi algoritmi oli kohtuullisen helppo toteuttaa. Kun asetustiedostoon onnistutaan vielä valitsemaan sopivat parametrit, myös klusteroinnin laatu on kohtuullinen (tosin kohinaksi luokiteltujen pisteiden määrä on lähes poikkeuksetta suuri, eräillä kokoelmilla jopa yli puolet kaikista dokumenteista). Ongelmaton algoritmi ei kuitenkaan ole. Johtuen dokumenttien suuresta dimensiosta on kyseenalaista, onko DBSCAN-algoritmin tiheyden käsite mielekäs [125, sivu 27]. Dimensio selittää osaltaan myös kohinapisteiden suuren määrän. Lisäksi algoritmin vaatimien ϵ ja $minpts$-parametrien valinta on epätriviaalia. Parametrien valintaa varten tehtiin testiohjelma, joka käy parametriavaruuden läpi vakiotarkkuudella ja kirjoittaa tiedostoon lyhennelmät klusterimalleista. Tämän laskenta on kuitenkin aikaavievää ja lyhennelmät täytyy tässä ohjelman versiossa käydä manuaalisesti läpi.

Toinen sovellukseen toteutettu klusterointimenetelmä on hierarkkinen keskipisteen menetelmä. Syitä valintaan olivat algoritmin toiminta metrisessä avaruudessa, ainakin lähimmän pisteen menetelmää parempi laatu ja tarve kokeilla hierarkkisen mallin toimivuutta käytännössä hakusovelluksen käyttöliittymällä. DBSCAN:in tapaan algoritmi oli varsin helppo toteuttaa ja tarvittaessa vaihdettavissa hyvin pienellä työllä lähimmän pisteen tai yhteisetäisyyden menetelmäksi. Menetelmän etuna on parametrittomuus ja klusteroinnin hyvä laatu (edellyttäen, että käyttäjä jaksaa muokata klusterimallin näkymän sopivaksi). Willet [136] on kritisoinut keskipisteen menetelmää pienten poikkeamaklusterien generoinnin takia, mutta empiirisessä osuudessa käytetyillä dokumenteilla tätä ongelmaa ei ollut (tai ainakaan se ei tuntunut ongelmalta). Suurempi ongelma oli hakusovelluksen käyttöliittymässä, jolla hierarkkisen mallin käsittely oli hyvin kömpelöä. Hierarkkisen klusteroinnin seurauksena hakusovellus näkee kymmeniä pieniä klustereita, joiden järkevä käyttö edellyttää mallin muokkaamista niin, että yksittäisiä klustereita samaistetaan aliklustereihinsa tai määritetään kohinaksi. Tämän muokkauksen jälkeen malli olikin laadultaan vähintään DBSCAN-klusteroinnin tasoinen. Loppukäyttäjän kannalta mallin ”minimointi” pitäisi kuitenkin automatisoida tai vaihtoehtoisesti muuttaa käyttöliittymää niin, että kaikki aliklusterit eivät näy mallista jatkuvasti.

Jatkokehityksen aikana olisi kiinnostavaa vertailla ainakin seuraavien metristen klusterointialgoritmien suorituskykyä ja klusteroinnin laatua jo toteutettuihin menetelmiin. Oletettavasti jokainen niistä klusteroi dokumenttikokoelman jo toteutettuja algoritmeja laadukkaammin (ainakin, jos menetelmien kehittäjien omiin testeihin on uskominen). Graafin ositukseen perustuva Chameleon ja DBSCAN-tyylinen SNN-menetelmä pystyvät löytämään mielivaltaisen muotoisia klustereita, BUBBLE:n etuna on skaalautuvuus ja inkrementaalisuus.

- **BUBBLE** [67] on vektorimallisen datan klusterointiin suunnitellun BIRCH-algoritmin yleistys metrisiä avaruuksia varten. Tietoa klustereita ylläpidetään CF*-puun (*Cluster Feature*) lehdissä, jotka sisältävät tiedon mm. klusterin säteestä sekä keskimmäisestä alkiosta eli klustroidista. Puun muut solmut sisältävät tietoa omassa alipuussa olevista klustereista, jotka esitetään niissä olevien alkioiden otoksena. Lisättäessä uutta alkiota klusterimalliin sen samanlaisuutta verrataan puun juurista lehtiin ja se asetetaan lähimpään klusteriin, edellyttäen, että lisäys ei heikennä klusterin laatua. Lisäystoiminto on inkrementaalinen ja mahdollistaa uusien klusterien luomisen tai vanhojen yhdistämisen klusteroinnin aikana. Osa uuden alkion lisäämisessä tehtävistä vertailuista tehdään suorituskykysyistä moniulotteisella skaalauksella (katso luku 6.3.3) projisoidussa vektoriavaruudessa, joka approksimoi metristä avaruutta. Tästä huolimatta BUBBLE on metrisessä avaruudessa toimiva menetelmä.

- **Chameleon** [86] on hierarkkinen klusterointialgoritmi, joka pyrkii korjaamaan klassisten menetelmien ongelmia. Perinteisten menetelmien yhdistämiskriteerinä on ollut joko klusterien lähimpien pisteiden etäisyys tai niiden yleinen kytkeytyminen (yhteisetäisyyden menetelmä). Kumpikin lähtökohta on väärä, jos klusteroitava data ei sovi yhdistämiskriteerin oletuksiin. Chameleon-algoritmi käyttää dynaamista yhdistämismallia, joka huomioi sekä klusterien läheisyydet että niiden kytkeytymisen. Läheisyys- ja kytkeytymismitat lasketaan normalisoituina käsiteltävien klusterien koon ja tiheyden suhteen. Algoritmin alussa havaintojoukko mallinnetaan K-lähimmän naapurin graafiksi. Tämän jälkeen graafi ositetaan vaiheittain säilyttäen tieto minimaalisista kaarien leikkauksista. Tuloksena saadun ositetun graafin komponentit ovat aloitusklustereita. Klusterit yhdistetään kokoavalla hierarkkisella menetelmällä. Yhdistämiskriteerinä käytetään parametrisoitua läheisyys- ja kytkeytymismitan yhdistelmää, joka hyödyntää osituksen aikana saatuja kaarien leikkauspainoja.

- **SNN**-menetelmä perustuu jaettujen lähimpien naapureiden samanlaisuusmittaan, johon pohjautuvia algoritmeja Ertöz *et al.* [52] kutsuvat alkuperäisten keksijöidensä mukaan Jarvis-Patrick-klusteroinniksi. He ovat määritelleet menetelmän DBSCAN-algoritmin [53] pohjalta. Algoritmin alussa muodostetaan K:n jaetun lähimmän naapurin graafi, jonka jälkeen edetään kuten DBSCAN-algoritmissa sillä erotuksella, että tiheys määritellään SNN-mitan avulla. Klustereihin kuuluvilla pisteillä on oltava vähintään ϵ-parametrin verran jaettuja lähimpiä naapureita. $minpts$-parametrilla määritellään minimimäärä ydinpisteen naapuruston pisteille. Niillä on oltava yli ϵ kappaletta jaettuja lähimpiä naapureita ydinpisteen kanssa. Muut pisteet jätetään huomiotta kohinana. Verrattuna esim. euklidiseen tai kosinimittaan määritelmän etuna on mukautuminen erilaisiin paikallisiin tiheyksiin ja korkeaulotteisiin dokumenttivektoreihin. Algo-

ritmin ongelmana on, että käyttäjän on syötettävä kolme parametria, joiden parhaiden arvojen etsiminen voi olla työlästä.

Sovelluksessa käytetty klusterointialgoritmien yhdistämistoiminto perustuu samanlaisuusarvojen yhdistämiseen laajennetulla vektorimallilla ja käyttäjän määräämillä painoilla. Varsinaisia algoritmeja ei integroida eikä klusterointi ota kantaa dokumenttien esitystapoihin. Jos haluttaisiin hyödyntää esitystapoihin sidottuja algoritmeja (esim. K-means sanavektoreille ja puiden sovitus XML-rakenteelle), voitaisiin ajaa klusterointialgoritmit rinnakkain ja yrittää yhdistää tuloksena saadut klusterimallit (tiedonhaussa tämä vastaisi relevanssiarvon jälkeen tehtävää fuusiohakua). Han & Kamber [76, sivu 345] huomauttavat kuitenkin, ettei rinnakkaisten klusterimallien yhdistäminen onnistu käytännössä, elleivät algoritmit tuota keskenään yhteensopivia malleja. Strehlin [125, sivut 124-126] mukaan kirjallisuudessa ei juuri ole tunnettuja menetelmiä tasaisten klusterimallien yhdistämiseen (jos tietoa alkuperäisten alkioiden piirteistä ei ole saatavilla). Hän esittää menetelmän, jossa klusterien yhdistäminen hahmotetaan optimointiongelmana ja ratkaistaan klusterimallien välisen keskinäisinformaation avulla. Olemassaolevista järjestelmistä klusterimallien yhdistäminen on toteutettu ainakin MSEEC-metahakukoneeseen [78]. Yhdistäminen onnistuu, koska algoritmien tulokset ovat yhtenäiset: klusterimallien dokumentit esitetään vertailtavissa olevina avainsanojen joukkona. Klustereista muodostetaan puu siten, että tietyn solmun vanhemmat sisältävät osajoukon lapsisolmun avainsanoista. Tätä voisi kutsua myös klusterien klusteroinniksi.

6.3.3 Visualisointi käyttöliittymän osana

Informaation visualisoinnin tavoitteena on antaa visuaalinen kuvaus laajasta informaatioavaruudesta. Ihmisten kyky hahmottaa kuvia ja muuta visuaalista tietoa on merkittävä – erityisesti, jos graafinen esitys on hyvin suunniteltu. Abstraktin tiedon visualisointi on kuitenkin huomattavasti fyysisten ilmiöiden esittämistä haastavampaa [6, sivut 259-261]. ExtMiner-sovelluksessa visualisoinnin kohteena ovat klusteroidut dokumentit. Abstrakteja malleja voidaan esittää monella eri tavalla, joista ei voida osoittaa yleispätevästi parasta. On otettava huomioon sovellusalue ja käyttäjien tarpeet. Klustereita voidaan visualisoida mm. seuraavilla tekniikoilla [125, sivut 34-35]:

- **Projektiot**. Tarkoituksena on projisoida korkeaulotteinen tai metrisessä avaruudessa oleva esitys kahteen tai kolmeen ulottuvuuteen. Menetelmiä tähän ovat esim. pääkomponenttianalyysi tai moniulotteinen skaalaus (*Multidimensional Scaling, MDS*). Näistä jälkimmäinen on ExtMiner-sovelluksen kannalta kiinnostava, koska se ei vaadi syötteekseen vektoriesitystä. MDS projisoi metrisen avaruuden datan vektoriavaruuteen niin, että alkioiden keskinäiset etäisyydet säilyvät mahdollisimman hyvin [22, sivut

91-96]. FastMap [57] on nopea MDS-algoritmi, joka käy datan läpi $\Theta(n)$-nopeudella projisoinnin tarkkuuden kustannuksella. Sovelluksen visualisointitoiminnoissa käytetään tällä hetkellä XXL-kirjaston FastMap-toteutusta.

- **Käyrät ja diagrammit**. Jos dokumenttikokoelma on pieni, klusterit voidaan esittää rinnakkaisina käyrinä. Myös tilastojen kuvauksessa käytettyjä diagrammeja (esim. histogrammit, pylväsdiagrammit, pistekuviot) voidaan käyttää klusterin ominaisuuksien kuvaamiseen [76, sivut 181-217].

- **Itseorganisoituva kartta** (*Self-Organizing Map, SOM*) on T. Kohosen kehittämä, K-meansia muistuttava heuristinen klusterointimenetelmä. Tavanomaisista klusterointimenetelmistä poiketen menetelmä projisoi klustereita jokaisella iteraatiolla matalaulotteiseen avaruuteen siten, että toisiaan muistuttavat klusterit pyrkivät asettumaan lähelle toisiaan. Klusterit voidaan esittää esim. säännöllisenä ruudustona tai kolmiohilana [22, sivut 90-91]. WEBSOM[10] on esimerkkisovellus SOM:n käytöstä dokumenttikokoelmien hahmotukseen.

- **Matriisiesitykset**. Yksinkertainen klusteriesitys voidaan saada suoraan harvasta datamatriisista esittämällä matriisi kuvana siten, että rivit ja sarakkeet järjestetään sopivasti uudelleen ja alkioiden arvot muunnetaan esim. kuvan kirkkausarvoiksi. Vastaavalla tekniikalla voidaan visualisoida myös samanlaisuusmatriisi, jolloin voidaan arvioida klusterien ”vahvuutta” ja niiden keskinäisiä suhteita [125, sivut 63-69].

- **Dokumenttipuut ja -verkot**. Dokumentit voidaan esittää käyttäjälle verkon solmuina, joissa kaaret esittävät dokumenttien keskinäistä samanlaisuutta. Verkko voidaan esittää käyttäjälle esim. jousimallina, jossa solmut esitetään hiukkasina ja kaarien painot kuvaavat hiukkasten välisiä vetovoimia [141, sivu 26]. Perinteinen tapa hierarkkisen klusteroinnin visualisointiin on järjestetty binääripuu, jota kutsutaan dendrogrammiksi. Verkon solmut kuvaavat klustereita ja ne on järjestetty siten, että saman klusterin aliklusterit ovat graafissa lähellä toisiaan. Kaarilla voidaan esittää myös hyperlinkkejä, jos dokumenttityyppi mahdollistaa tämän. Näin on tehty esim. web-sivustorakenteita visualisoivassa Mapuccino-järjestelmässä [6, sivu 301].

ExtMiner-sovelluksen käyttöliittymän keskeinen ominaisuus on klusteroinnin ja tuloslistojen yhdistetty käyttö dokumenttikokoelman hahmottamiseksi. Wu’n *et al.* [139] käytettävyystutkimuksen perusteella käyttäjät pitävät klusteroituja hakutuloksia tuloslistoja selkeämpinä. Toisaalta käyttäjien löytämien relevanttien dokumenttien määrä oli samaa luokkaa kummallakin menetelmällä (tarkempaa tietoa klusteroinnin käyttämisestä tiedonhaussa on luvussa 4.2). Hakulistan ja klusteripuun lisäksi käyttöliittymä sisältää visuaalisen näkymän doku-

[10] `http://websom.hut.fi/websom/`

menttikokoelmasta. Näkymän tarkoitus on osaltaan helpottaa klusterien ja hakutulosten tulkintaa. Kuvia käyttöliittymästä on liitteessä A.

Näkymän granulariteetti (ruudun koko) on skaalattavissa. Lisäksi käyttäjä voi tarkentaa haluamaansa alueeseen. Visuaalinen näkymä ei ole riippuvainen klusteroinnista tai hausta, vaan se luodaan suoraan samanlaisuusmatriisista. Kuhunkin klusteriin (tai kohinaan) liittyvät dokumentit piirretään omalla värillään. Jos näkymän yksittäisessä ruudussa on monta dokumenttia, ruudun väri määräytyy dokumenttien määrän ja eri klusterien osuuksien mukaan. Yksittäisiä dokumentteja tai klustereita voidaan valita aktiivisiksi, mikä näkyy myös hakulistassa ja klusteripuussa. Valintaa voidaan käyttää uutta klusterimallia muodostettaessa. Käyttöliittymässä voidaan myös valita, mitkä klusterit ovat näkyvissä. Näkymä tukee mielivaltaista määrää klustereita, joskin hierarkkisten klusterimallien esittäminen on tämänhetkisessä toteutuksessa hankalaa. Syynä tähän on, että näkymä käsittelee jokaista klusteripuun solmua omana klusterinaan, jotka edelleen väritetään itsenäisesti. Sovellus ei ota kantaa, millä algoritmilla dokumentit projisoidaan näkymään. Tällä hetkellä toiminnassa on vain FastMap-projektio.

Sovelluksen käyttöliittymää ovat inspiroineet erityisesti Scatter/Gather-järjestelmä [37] hakuprosessin osalta, LightHouse-käyttöliittymä [99] ja KartOO-metahakukone yleisten visualisointiominaisuuksien osalta sekä Vivísimo-yhtiön metahakukone klusterien ja tuloslistojen yhdistämisen osalta. Vivísimo oli myös ensimmäinen tämän kirjoittajan löytämä tuotantokäytössä oleva hakukone, joka hyödyntää klusterointia. Erityismaininnan ansaitsee myös CiteSeer-bibliografiahakukone, jossa käyttäjä voi eksplisiittisesti verrata dokumenttien samanlaisuutta mm. yhteisviittauksiin perustuen. Riippumatta siitä, mikä hakukone tulee olemaan ”tulevaisuuden Google”, tämän kirjoittaja pitää todennäköisenä sitä, että klusterointi integroituu hakuprosessin luonnolliseksi osaksi samalla tavalla kuin tuloslistat ovat nyt. Klusterointi ei kuitenkaan korvaa tuloslistoja, vaan täydentää ja jäsentää niitä. Myös klusterissa olevat dokumentit voidaan esittää tuloslistana.

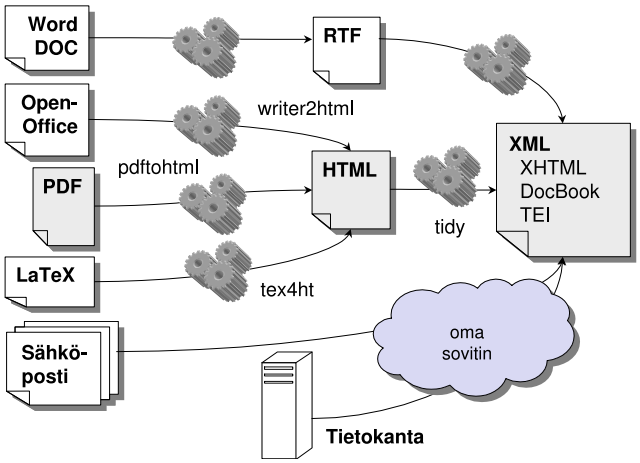
6.4 Sovelluksen arviointia

Sovelluksen kehityksen yhteydessä ilmeni useita ideoita ja mahdollisia dokumenttikokoelmia, joihin sovellusta voitaisiin käyttää. Kaikkia mahdollisia ominaisuuksia ei kuitenkaan ehditty toteuttamaan ja osa dokumenttikokoelmista odottaa klusterointiaan mahdollisen jatkokehityksen aikana. Sovellusta päästiin testaamaan kahdella todellista dataa sisältävällä kokoelmalla: oppimispäiväkirjoilla ja koulutusjärjestelmällä. Näistä edellistä testasi syksyllä 2004 kolme henkilöä ja jälkimmäistä syystalvella 2004 yksi henkilö. Lisäksi tämän kirjoit-

tajan omassa testikäytössä oli muita rajatumpia kokoelmia, joista tärkeimpänä on osa tässä tutkielmassa käytetyistä lähdeartikkeleista. Artikkelien klusterointia on kuvattu tarkemmin liitteessä A esimerkkinä sovelluksen käytöstä.

6.4.1 Dokumenttikokoelmat

ExtMiner-sovelluksen indeksimuodostimet lukevat oletuksena XML-dokumentteja. Ei kuitenkaan ole mitään periaatteellista estettä käsitellä myös muita formaatteja, kunhan sopiva muunnosohjelma on saatavilla. Sovelluksessa on kokeiltu HTML-, PDF-, teksti- ja yleisiä XML-dokumentteja. Paikallisten tiedostojen käyttäminen indeksoinnin lähteenä on indeksimuodostimista riippumaton vaatimus, joten tietokannan tai sähköpostiarkistojen suora indeksointi vaatisi jonkin verran muutoksia itse järjestelmään (indeksoinnin lisäksi myös tietueeseen osoittamisen ja sopivan näyttimen osalta). Eräitä mahdollisia lähdedokumenttiformaatteja ja muunnosohjelmia on esitetty kuvassa 6.5.



Kuva 6.5: Dokumenttien muunnosprosessi.

ExtMiner-sovellusta on testattu Knowledge Mining -projektin aikana seuraavilla dokumenttikokoelmilla:

1. **Metso Oyj:n kotisivut** olivat sovelluksen varhaisimmalla prototyypillä testattu kokoelma, joka sisälsi n. 70 HTML- ja PDF-dokumenttia. Dokumentit haettiin wget[11]-ohjelmalla Metson kotisivuilta[12] kahden rekursioaskeleen syvyydeltä. PDF-dokumentit muunnettiin HTML:ksi pdftohtml-ohjelmalla[13], jonka jälkeen kaikki dokumentit puhdistettiin ja muunnettiin XML-muotoon HTML Tidy -ohjelmalla[14]. Klusterimallis-

[11] http://www.gnu.org/software/wget/wget.html
[12] http://www.metso.com/
[13] http://pdftohtml.sourceforge.net/
[14] http://tidy.sourceforge.net/

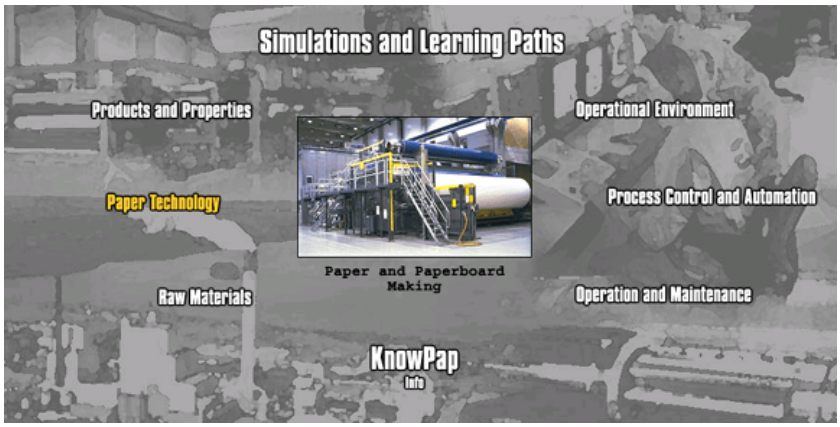
ta erottuivat ”perussivujen” lisäksi lähinnä Metson tytäryhtiöitä kuvaavat sivut sekä PDF-muodossa olevat raportit, joissa oli edelleen pientä hajontaa erikielisten esitysten välillä. Raportit olivat suomen- ruotsin- tai englanninkielisiä muiden sivujen ollessa pääosin englantia. Dokumenteista indeksoitiin tekstin lisäksi otsikot ja lähtevät linkit. Dokumenttikokoelmasta ei löytynyt juurikaan sovelluksen kannalta kiinnostavia hakutapauksia, joten haku- ja klusterointitoiminnoista ei ollut tässä erityistä hyötyä. Kokoelmaa voitiin joka tapauksessa käyttää kehitysaikaisena testijoukkona.

2. **Johdatus ohjelmistotekniikkaan -kurssi** pidettiin syksyllä 2004 Jyväskylän yliopiston Tietotekniikan laitoksella. Kurssin vastaava luennoitsija oli Hilkka Heikkilä. Kurssin suoritukseen kuului 13:n esseen kirjoittaminen, yksi kutakin luentoa varten. Kurssille oli ilmoittautunut yli 200 opiskelijaa, minkä takia kaikkien esseiden läpikäynti käsin olisi ollut työlästä. ExtMiner-sovellusta käytettiin esseiden tarkastukseen, vastausten vertailuun ja mahdollisten huijaustapausten etsimiseen. Esseet olivat tekstitiedostoja, joista kunkin tiedoston 1. rivi määriteltiin dokumentin otsikoksi. Lisäksi Korppi[15]-kurssinhallintajärjestelmästä saatiin käyttöön dokumentteihin liittyvää metatietoa, kuten dokumenttien sijainti sekä tekijöiden nimet ja pääaineet. Dokumenttien nimeämisestä päätteltiin, mihin luentoon kukin essee liittyy. Tiedot indeksoitiin omiksi kentikseen, jolloin käyttäjä pystyi hakemaan esim. tietyn opiskelijan tiettyyn luentokertaan liittyvän esseen. Kurssin loppuvaiheessa käsiteltäviä dokumentteja oli yli 1000, mikä oli prototyypin suorituskyvyn äärirajoilla. Indeksointivaiheessa samanlaisuusmatriisin (kertaluontoinen) laskeminen dokumenttikokoelmalle kesti 2.8 GHz Pentium 4 -koneella yli kaksi tuntia. Itse sovelluksen käyttö oli hitaanlaista lähinnä visuaalisen näkymän käyttöliittymän osalta. Hakuja pystyi tekemään ja uuden klusterimallin laskeminen projektioineen vei runsaat 10 sekuntia. Klusteroinnissa eri luentoihin liittyvät vastaukset saatiin erotettua kohtuullisella tarkkuudella toisistaan (joissakin tapauksissa 2-3 luentoa oli liitetty samaan klusteriin). DBSCAN-algoritmin kannalta ongelmallista oli, että eri luentojen vastausklusterien tiheydet poikkesivat toisistaan.

3. **Shakespearen näytelmät** ovat saatavilla julkisesti WWW:ssä[16] XML-muodossa. Kokoelma sisältää 37 dokumenttia. Dokumenttijoukkoa käytettiin yleiskäyttöisen XML-muodostimen testaukseen. Mitään kenttäkohtaista lisätietoa ei hyödynnetty, dokumenteista indeksoitiin ainoastaan sisältöteksti. Muodostin ei sellaisenaan ole kovin hyödyllinen, mutta sitä voidaan kokeilla nopeasti mihin tahansa XML-muotoiseen dataan ilman räätälöintiä. Näytelmistä löytyi klustereita, mutta niille ei ollut havaittavissa selkeää tulkintaa.

[15] https://korppi.jyu.fi/
[16] http://www.oasis-open.org/cover/bosakShakespeare200.html

4. **KnowPap** on UPM-Kymmene Oyj:llä käytössä oleva HTML-pohjainen koulutusjärjestelmä, joka sisältää runsaasti paperiteknologiaan liittyvää materiaalia. Esimerkki KnowPap-järjestelmän käyttöliittymästä on esitetty kuvassa 6.6. Knowledge Mining -projektin aikana n. 300 dokumentin otos järjestelmän sisällöstä klusteroitiin ExtMiner-sovelluksella. Tekstisisällön ja otsikoiden lisäksi dokumenttien linkit media-aineistoon (piirrokset, valokuvat, animaatiot, videoleikkeet) indeksoitiin selityksineen, mikä mahdollisti alkeellisen mediahaun. Mahdollisia käyttökohteita voisivat olla metahaku mahdollisesti useammasta koulutusjärjestelmästä samanaikaisesti, dokumenttikokoelman hahmotus ja navigointi usean vaihtoehtoisen näkökulman pohjalta (koulutusjärjestelmässä on myös oma navigointinsa, mutta jotkut asiakokonaisuudet ovat hajautuneet ympäri järjestelmää). Kuvahaun osalta suunniteltiin kouluttajan mediapankkia (järjestelmä sisältää hakemistohierarkiassa tuhansia kuvia ja muuta materiaalia, joka ei ole aiemmin ollut helposti haettavissa keskitetysti). Mediapankin avulla uusia koulutusmateriaaleja voisi koostaa olemassaolevien pohjalta aiempaa helpommin. UPM:n yhteyshenkilönä toimi projektipäällikkö Antti Saarinen.



Kuva 6.6: Osa KnowPap-järjestelmän aloitussivusta.

Sovelluksen kehityksen aikana ei käytetty standardeja testikokoelmia, koska erityisesti XML-dokumenteille suunniteltua julkista kokoelmaa ei löytynyt. Projektin painopisteenä on ollut avoimen ja mukautuvan arkkitehtuurin kehitys ja sovelluksen kokeilu saatavilla olleilla dokumenttikokoelmilla, joten sovellukseen ei ehditty tehdä laskennallisia validointitoimintoja. Klusteroinnin laadun arviointi on siis jäänyt visuaaliselle ja intuitiiviselle tasolle. Toisaalta käytettyjen kokoelmien klusteroinnin tavoite on ollut etsiä uusia näkökulmia dokumentteihin – ei automaattinen luokittelu. Sitäpaitsi on epäselvää, mikä edes olisi ”oikea” klusterointitulos (vai onko sitä?) esim. KnowPap-kokoelman yhteydessä. Jatkokehityksen aikana järjestelmään on kuitenkin syytä lisätä mahdollisuus sekä sisäisiin että ulkoisiin validointitoimintoihin.

6.4.2 Sovelluksen arviointia ja jatkokehitysideoita

ExtMiner-sovelluksesta kehitettiin Knowledge Mining -projektin aikana toimiva prototyyppi, joka tukee rakenteisten dokumenttien klusterointia ja kenttäpohjaista tiedonhakua. Järjestelmä on konfiguroitavissa muodoltaan vaihteleville dokumenttikokoelmille. Lisäksi erilaisia haku- klusterointi- ja visualisointimenetelmiä voidaan yhdistää joustavasti ohjelman hakuprosessin avulla. Sovellus on ollut myös koekäytössä kahden eri dokumenttikokoelman osalta, mutta nykyisessä muodossaan se ei vielä sovellu tuotantokäyttöön. Seuraavassa on esitetty kehityksen ja koekäytön aikana tunnistettuja puutteita ja hahmotettu sovelluksen jatkokehitysmahdollisuuksia.

- **Rajapinnat muihin järjestelmiin**. Sovelluksen käyttökelpoisuutta rajoittaa se, että indeksoitavien dokumenttien on oltava saman hakemistopuun alla paikallisella levyllä. Lisäksi sovellus on erillään XML-dokumenttien tuotanto- ja julkaisuprosessista, vaikka klusterointi- ja hakutoiminnot kuuluisivat ilman muuta osaksi laajempaa hallintaprosessia. Järjestelmään tarvittaisiin rajapinnat XML-tietokantoihin, tuotantoympäristöihin sekä dokumenttien hallinta- ja julkaisujärjestelmiin. Myös versionhallinta pitäisi huomioida, ellei jokin hallintajärjestelmä jo kata sitä. Indeksitietojen tallennusta varten järjestelmässä pitäisi olla rajapinta myös perinteiseen relaatiokantaan.

- **Haku**. Kenttäpohjainen indeksointi on osoittautunut riittäväksi koekäytössä olleilla dokumenttikokoelmilla. Suorituskyky- ja toteutusvaatimuksiltaan vaativamman puupohjaisen indeksin käyttö ei ehkä ole tarpeen dokumenttikeskeisillä kielillä, elleivät dokumentit ole sisällöltään hyvin monimuotoisia ja sisällä loogisia tietorakenteita (jolloin kielessä olisi siis myös datakeskeisiä piirteitä). Peruskäyttäjälle kenttäpohjainen haku on riittävän ilmaisuvoimainen ja helppo ymmärtää. Jos puupohjaista indeksiä kuitenkin käytettäisiin, tämä mahdollistaisi sisältöhaun dokumenttifragmenteilla Schliederin & Meussin mallin [119] (katso luku 5.3.2) tapaan. Tällöin käyttäjä voisi esim. uutta dokumenttia kirjoittaessaan nähdä nopeasti, millaista vastaavaa rakennetta noudattavaa sisältöä on tehty aiemmin ja uudelleenkäyttää sitä. Hakujärjestelmän tulisi myös sisältää validointitoiminto, jolla eri hakualgoritmien laatua voidaan testata standardeilla testikokoelmilla esim. F-mittaa käyttäen.

- **Klusterointi**. Nykyinen arkkitehtuuri mahdollistaa useiden eri metristen klusterointialgoritmien käytön. Suuriin dokumenttijoukkoihin skaalautumiseen ei ole kuitenkaan ole kiinnitetty huomiota, eli Handin *et al.* [77, sivut 15-18] termein sovelluksella ei ole kunnollista tiedonhallintastrategiaa. Suuret ja muuttuvat tietojoukot vaativat inkrementaalista klusterointia ja indeksointia. Samanlaisuusmatriisin käsittelyä pitäisi kehittää niin, ettei kaikkea dataa pidetä jatkuvasti keskusmuistissa. Luvussa 6.3.2 mainitut uu-

det klusterointialgoritmit tulisi liittää osaksi järjestelmää. Myös klusterien esitystapoja voisi kehittää havainnollisempaan suuntaan, esim. ottamalla käyttöön metrisessä avaruudessa toimiva muunnelma CURE-algoritmin [73] monen prototyypin esityksestä. Hakujärjestelmän tapaan myös klusterointitoimintojen osaksi pitäisi lisätä validointi. Kaikkia sisäisiä validointikriteerejä ei voida käyttää metrisen avaruuden vuoksi, mutta esim. Dunnin indeksin (katso luku 4.3.4) kaltaisten yleisluontoisten mittojen hyödyntäminen on mahdollista. Ulkoisia kriteerejä voitaisiin käyttää hakujärjestelmän validoinnin tapaan standardeilla kokoelmilla.

- **Käyttöliittymä**. Prototyypin käyttöliittymä on suunniteltu eri klusterointi- ja hakualgoritmien vertailua ja testausta varten, mutta ei loppukäyttäjiä ajatellen. Sovelluksen käyttöliittymä on erotettu sovelluslogiikasta, joten vaihtoehtoisia käyttöliittymiä on mahdollista tehdä. Mahdollisia käyttöliittymäparannuksia olisivat hakuhistoria, ohjelman ehdottamat ”tyypilliset” hakutermit, joiden pohjalta lähteä hakemaan tietoja, muokatun klusterimallin lataus ja tallennus sekä graafien käyttö esim. dokumentin välisten suhteiden tai dokumentin rakenteen visualisointiin. Tämä voitaisiin toteuttaa esim. TouchGraph[17]-graafikäyttöliittymää käyttäen.

Pidemmän aikavälin kehitystavoitteena hakujärjestelmän voisi laajentaa entistä yleiskäyttöisemmäksi rakenteisen tietämyksen hallintaympäristöksi. Hallintaympäristön tarkoituksena olisi tukea tietovirta-analyysissa ja asiakirjojen standardoinnissa määriteltyjä toimintaprosesseja. Rajapintojen avulla sovellus tukisi sekä dokumenttien julkaisua (esim. web-hakukoneen muodossa) että niiden tuotantoa (esim. koostettaessa uutta materiaalia vanhojen pohjalta). Lisäksi laajojen dokumenttikokoelmien hahmottaminen helpottuu dokumentin elinkaaren eri vaiheissa. Järjestelmä mahdollistaisi yhtenäisen, loogisen näkymän perinteisesti erillisiin järjestelmiin.

[17] http://touchgraph.sourceforge.net/

7 Yhteenveto

Tutkielmassa käsitellään rakenteisten dokumenttien tiedonhakua ja klusterointia sekä niiden yhdistämisen etuja. Yleisenä viitekehyksenä toimii tiedonlouhinta – yleistetyssä ja laajennetussa muodossaan tietämyksen muodostaminen tietojoukoista. Klusterointi on tiedonlouhinnan perusmenetelmä ja tiedonhakua voidaan pitää osana tietämyksen muodostamisprosessin alkuvaihetta. Kokonaisuutena prosessia voidaan pitää tekstitiedonlouhintana tai tietämyksen muodostamisena teksteistä. Toisesta näkökulmasta katsottuna kyse on myös tekstitiedonhausta, koska klusterointi voidaan käsittää tiedonhaun apuvälineeksi. Haku- ja klusterointitoiminnot käyttävät lisäksi samaa indeksiä ja samaa yleistä periaatetta eri algoritmien yhdistämiseen: laajennettua vektorimallia.

Puolirakenteinen tieto on joustava tietomalli tietokannoissa olevan rakenteisen tiedon ja rakenteettoman datan välimaastossa. Esimerkki puolirakenteisesta tiedosta ovat rakenteiset dokumentit, jotka sisältävät tekstin lisäksi rakennetietoja, linkkejä ja metatietoa. Rakenteisuus tuo tiedonhakuun lisätietoa, jolla haun laatua voidaan parantaa. Dokumenttien ja kyselyjen esittäminen on mutkikkampaa verrattuna puhtaaseen tekstiin, mutta hakuja voidaan tehdä hienommalla granulariteetilla ja tuloksissa pystytään esittämään dokumenttien relevanteiksi havaitut osat. Myös linkkitiedon avulla voidaan arvioida dokumenttien relevanssia tai samankaltaisuutta.

XML on tärkein tapa rakenteisten dokumenttien esittämiseen. Se valittiin käytettäväksi formaatiksi sen yleisyyden, laaja-alaisuuden ja saatavilla olevien työkalujen vuoksi. Lisäksi tavoitteena oli kehittää sovellus, jossa XML-formaatin käytöstä saadaan perusteltua lisäarvoa. Formaatin avulla saadaan edustava otos rakenteisista dokumenteista, koska mikä tahansa rakenteinen tai puolirakenteinen tieto on muunnettavissa XML-muotoon. Dokumentti voidaan hahmottaa puuna, mikä mahdollistaa monipuoliset indeksirakenteet ja hakumallit. Tulkittaessa XML-dokumenttikokoelma hypertekstijärjestelmäksi jokaista dokumentin osaa pidetään hypertekstisolmuna, joka on linkittynyt muihin saman dokumentin osiin ja mahdollisesti muiden dokumenttien solmuihin.

Sovellusalueen vaatimuksia klusteroinnille ovat datan moniulotteisuus (jokainen termi ja linkkianalyysin osalta myös jokainen dokumentti ovat oma piirteensä) sekä mielivaltaisen muotoiset ja tiheydeltään vaihtelevat klusterit. Lisäksi rakenteiset dokumentit vaativat erityisiä indeksirakenteita, joiden esittäminen vektoriavaruudessa ei ole käytännöllistä. Täs-

tä syystä oletetaan, että klusterointialgoritmi toimii metrisessä avaruudessa, jolloin se on riippumaton dokumenttien esitystavasta. Itse etäisyysmitta kootaan käyttäjän antamilla painotuksilla eri piirretyypeistä. Painotuksia muuttamalla käyttäjä voi muuttaa klusteroinnissa käytettäviä kriteereitä. Toteutetussa hakumallissa oletetaan lisäksi, että dokumenttikokoelma on homogeeninen: dokumentit ovat rakenteeltaan samaan skeemaan kuuluvia ja laajuudeltaan samaa kertaluokkaa. Indeksointivaiheessa pyritään etsimään sovellusaluekohtaisen konfiguroinnin avulla dokumenttien relevantit ja kuvaavimmat osat, mutta käytetään kuitenkin yksinkertaista kenttäpohjaista indeksiä. Tietoja voidaan hakea samanaikaisesti eri hakualgoritmeja käyttäen ja ne yhdistetään lineaarisena summana käyttäjän antamilla painotuksilla. Tämä mahdollistaa esim. teksti- ja linkkipohjaisen haun yhdistämisen.

Klusteroinnin käyttöä tiedonhaussa on perusteltu klusterointihypoteesilla, jonka mukaan samanlaiset dokumentit ovat relevantteja samoilla kyselyillä. Hypoteesi on kiistanalainen ja mm. probabilistisessa tiedonhaussa käytetty tilastollinen järjestysperiaate on sen vastainen. Tässä tutkimuksessa päädyttiin testikokoelmien tarkastelun perusteella kompromissiin, jonka mukaan hypoteesin paikkansapitävyys ja hyödyllisyys riippuu tarkasteltavasta mittakaavasta. Karkealla tasolla hypoteesi pitää yleensä paikkansa, mutta hakua tarkennettaessa törmätään usein tilanteeseen, jossa klusterointimalli ei kykene erottamaan relevantteja dokumentteja epärelevanteista. Erityisesti tämä pitää paikkansa esiklusteroidussa kokoelmassa, mutta myöskään hakutuloksista saatujen relevanttien klusterien kaikki dokumentit eivät ole relevantteja. Syynä tähän voi olla liian karkea dokumenttien esitystapa tai liian yleinen samanlaisuusmitta. Siksi klusterointia käyttävät hakumallit tarvitsevat tuekseen myös tuloslistan. Hakutuloksia klusteroitaessa relevantit dokumentit jäävät yleensä yhteen tai kahteen klusteriin kyselystä riippuen.

Empiirisessä osuudessa kehitetty ExtMiner-sovellus on yleinen alusta eri haku-, klusterointi- ja visualisointialgoritmien yhdistämiseen ja testaamiseen. Jokaiselle dokumenttikokoelmalle voidaan määritellä oma indeksimuodostin, joka vastaa XML:n jäsennyksestä ja indeksoitavista kentistä. Dokumenttien näyttämistä varten voidaan määritellä oma näytinluokka. ExtMiner-sovellus yhdistää muutamia klusteripohjaiseen hakuun liittyviä ideoita, jotka ovat esiintyneet erillisinä aiemmissa hakusovelluksissa. Tärkeimpiä näistä ovat iteratiivinen haku- ja klusterointiprosessi, interaktiivinen klusterimalli sekä samanaikaiset klusteri- ja listanäkymät. Sovellus on ollut koekäytössä kahdella eri dokumenttikokoelmalla, mutta nykyisessä muodossaan se ei vielä sovellu tuotantokäyttöön. Pidemmän aikavälin kehitystavoite on laajentaa hakujärjestelmä yleiseksi rakenteisen tietämyksen hallintaympäristöksi.

Tämän kirjoittaja pitää todennäköisenä sitä, että klusterointi integroituu hakuprosessin luonnolliseksi osaksi samalla tavalla kuin tuloslistat ovat nyt. Tätä tukevat Raon [111] ennusteet tulevaisuuden tiedonhausta:

1. Nykyistä rikkaampi informaatioavaruus, jossa kokoelmat on organisoitu hierarkkisesti ja niiden keskinäiset suhteet on merkitty linkeillä. Lisäksi metatietoa hyödynnetään.
2. Käyttäjiä tukevat uudet hakutoiminnot, kuten automaattisesti luodut käsitekartat ja lyhennelmät. Lisäksi teksteistä voi etsiä hahmoja ja suhteita tiedonlouhintasovelluksilla.
3. Avoin infrastruktuuri eri dokumenttikokoelmien, hakukoneiden ja hakutoimintojen välillä (fuusiohakua kaikilla tasoilla).
4. Tekstitiedonlouhinnan ja luonnollisen kielen käsittelyn tekniikoiden integroituminen perinteiseen tiedonhakuun ja lopulta sen korvaaminen.

Työssä tutkittiin XML-dokumenttien klusterointia ja tiedonhakua sekä niiden yhdistämisen tarjoamia mahdollisuuksia samaa skeemaa noudattavien dokumenttikokoelmien hahmottamisessa. Tutkimuksen tuloksena havaittiin, että tuloslistojen ja klusteroinnin yhdistäminen parantaa erityisesti käyttömukavuutta, mutta klusterit eivät yksiselitteisesti paranna haun laatua (katso luku 4.2). Sen sijaan haun ja klusteroinnin laatua parantavat merkittävästi yhdistetyt hakutavat ja fuusiotekniikat (katso luku 5.5). Rakenteisten dokumenttien erityisongelma on tekstidokumenttejakin hankalampi piirteiden esittäminen. Termien ja linkkien indeksointi nostaa datan dimension tuhansiin ulottuvuuksiin ja rakenteen täysimääräinen hyödyntäminen vaatii puumaisia indeksirakenteita. Selkeästi parasta ratkaisua rakenteisten dokumenttien hakuun ja klusterointiin ei löytynyt, mutta esim. Schliederin & Meussin puumalli vaikuttaa lupaavalta kandidaatilta. Monimuotoisten esitystapojen ongelma voidaan kiertää käyttämällä metrisen avaruuden klusterointimenetelmiä, mutta ”dimensiokirousta” tämäkään ei välttämättä poista. Samanlaisuusmitan täytyy pystyä erottamaan dokumentit toisistaan.

Tavoite yleishyödyllisen XML-sovelluksen luomisesta toteutui osittain. ExtMinerin kenttäpohjainen indeksi ei pysty käyttämään XML:n kaikkia piirteitä haussa ja klusteroinnissa hyödyksi. Toisaalta semanttisesti rikkaalla tavalla merkattuja julkisia dokumenttikokoelmia ei tutkimuksen aikana löytynyt. XHTML-tieto on linkkejä lukuunottamatta semanttisesti niin alhaisella tasolla, että täydestä puuindeksistä tuskin olisi ollut hyötyäkään. Kuvaavaa on, että täysin tekstimuotoisesta (yhdistettynä muutamaan metatietokenttään) kurssipäiväkirjakokoelmasta saatiin haettua tietoa monipuolisemmin kuin HTML- ja PDF-dokumenteista. Tutkimuksen edetessä tämän kirjoittajalle tuli selväksi, että XML:n käsittelyä vaikeampi ja kriittisempi ongelma on se, että suurin osa organisaatioiden dokumenteista on heterogeenisissa kokoelmissa (jos ylipäänsä ovat digitalisoituja) ja semantiikaltaan matalalla tasolla – jopa silloin, kun merkkaus on XML-kielistä. XML-muotoinen (tai mikä tahansa pitkälle rakenteistettu) tieto on hyödyllistä ja helposti haettavaa, mutta kuinka saattaa dokumentit tälle tasolle? Uusia dokumentteja kirjoitettaessa käsin tehty merkkaus ei ole käytännöllistä, joten jo editointiympäristön pitää olla pitkälle kehittynyt. Vanhojen dokumenttien rakenteistami-

nen on vielä hankalampaa, jos lähtötasona on vain joukko skannattuja kuvia. Nämä ongelmat ovat perustavanlaatuisempia kuin päätös siirtymisestä uuteen dokumenttiformaattiin.

Tutkielman keskeiset kontribuutiot ovat seuraavat:

- KDD-käsitteen ja tiedonlouhinnan sekä niistä ”periytyneiden” louhintatieteiden suhteiden pohdinta. Ehdotus KDD-termin viimeisen D:n yleistämisestä tietokannoista (*databases*) *tietojoukkoihin* (*datasets*) (luku 2.2).
- Tiedonlouhinnan ja tekstitiedonhaun erilaisten tutkimusperinteiden yhdistäminen; etäisyysmittojen, indeksirakenteiden, haku- ja klusterointimenetelmien integraatio. Tiedonhaun huomiointi osana tiedonlouhinnan iteratiivista ja interaktiivista prosessia (luku 3.1).
- Ehdotus uudesta moniulotteisesta tavasta klusterointimenetelmien jaotteluun: algoritmipohjaisen jaottelun sijaan tulisi keskittyä toisistaan riippumattomiin näkökulmiin, joita ovat klusterien lajit, syötetiedon muoto, klusteroinnissa käytettävä avaruus, etäisyysmitta sekä klusteroinnin perustana oleva taustateoria (luku 4.3).
- Kirjallisuuskatsaus rakenteisten dokumenttien indeksirakenteista ja etäisyysmitoista käsittäen teksti-, linkki- ja rakenteisen analyysin sekä yhdistetyt hakutavat. Hakumallien käsittely tavallisen tekstitiedon, hypertekstin ja rakenteisten dokumenttien näkökulmasta (luku 5).
- Edelliset tekijät huomioivan haku- ja klusterointisovelluksen prototyypin suunnittelu ja toteutus (luku 6).

Laajojen tietomassojen käsittelynä tiedonlouhintaan sisältyy omat yksityisyys- ja tietosuojariskinsä. Tässä tutkielmassa ei ole käsitelty niitä tarkemmin, koska lähestymistapa on ollut tekninen. On kuitenkin tärkeää muistaa, että mitä tahansa tekniikkaa voidaan käyttää kyseenalaisiin tarkoituksiin riippumatta siitä, miten yleishyödylliseksi se on alunperin suunniteltu. Erityisen ongelmallisia ovat mainostus- ja vakoiluohjelmat, joita käyttäjä on saattanut asentaa koneelle tietämättä niiden todellista tarkoitusta. Jopa niinkin yksinkertaista asiaa kuin WWW-sivujen evästeitä (*cookies*) voidaan käyttää tiedon keräämiseen, vaikka käyttäjä ei sitä välttämättä haluaisi. Analysoitaessa henkilökohtaisia tietoja on aina muistettava hyvät tiedonkeruukäytännöt: käyttäjän on saatava tietää, mitä tietoa hänestä kerätään ja mihin tarkoitukseen [76, sivut 476-478]. Luonnollisestikaan mitään tietoja ei pidä kerätä ilman käyttäjän suostumusta. Tämän tutkielman tarkoituksena on ollut kehittää yleishyödyllisiä menetelmiä ja avoimeen lähdekoodiin perustuvia ohjelmistoja, jotka auttavat *kaikkia* käyttäjiä tietotulvasta selviämiseen.

Lähteet

[1] M. Agosti, F. Crestani, and G. Pasi (eds.): *Lectures on Information Retrieval, Third European Summer-School, ESSIR 2000*, vol. 1980 of *Lecture Notes in Computer Science*. Springer, 2001. 121, 125

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan: *Automatic subspace clustering of high dimensional data for data mining applications*. In Haas, L. *et al.* [74], pp. 94–105. http://doi.acm.org/10.1145/276304.276314. 44, 48, 56, 57

[3] H. Ahonen, O. Heinonen, M. Klemettinen, and I. Verkamo: *Applying data mining techniques in text analysis*. Tech. Rep. C-1997-23, Helsingin yliopisto, Tietojenkäsittelytieteen laitos, 1997. http://citeseer.ist.psu.edu/ahonen97applying.html. 19

[4] R. B. Allen, P. Obry, and M. L. Littman: *An interface for navigating clustered document sets returned by queries*. In S. Kaplan (ed.): *Proceedings of the Conference on Organizational Computing Systems*, pp. 166–171. ACM Press, 1993. http://doi.acm.org/10.1145/168555.168572. 96

[5] L. Aunimo: *Tekstifragmenttien välisen semanttisen samanlaisuuden tunnistaminen*. Pro Gradu -työ, Helsingin yliopisto, Yleisen kielitieteen laitos, 2002. http://ethesis.helsinki.fi/julkaisut/hum/yleis/pg/aunimo/. 61

[6] R. Baeza-Yates and B. Ribeiro-Neto: *Modern Information Retrieval*. Addison-Wesley, 1999. 16, 17, 18, 33, 59, 62, 63, 65, 66, 107, 108

[7] A.-L. Barabási and E. Bonabeau: *Scale-free networks*. Scientific American, 288(60-69), 2003. http://www.nd.edu/~networks/PDF/Scale-Free%20Sci%20Amer%20May03.pdf. 71

[8] F. Beil, M. Ester, and X. Xu: *Frequent term-based text clustering*. In O. R. Zaïane, R. Goebel, D. Hand, D. Keim, and R. Ng (eds.): *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 436–442. ACM Press, 2002. http://doi.acm.org/10.1145/775047.775110. 35, 36, 47, 98

[9] Y. Ben-Aharon, S. Cohen, Y. Grumbach, Y. Kanza, J. Mamou, Y. Sagiv, B. Sznajder, and E. Twito: *Searching in an XML corpus using content and structure*. In Fuhr, N. *et al.* [65], pp. 46–52. http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf. 103

[10] P. Berkhin: *Survey of clustering data mining techniques*. Techn. rep., Accrue Software, 2002. http://citeseer.ist.psu.edu/berkhin02survey.html. 36, 43, 45, 54, 56, 57, 58, 135

[11] T. Berners-Lee: *WWW: Past, present, and future*. Computer, 29(10):69–77, 1996. http://dx.doi.org/10.1109/2.539724. 30

[12] T. Berners-Lee, J. Hendler., and O. Lassila: *The semantic web*. Scientific American, Toukokuu 2001. http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21. 32, 95

[13] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft: *When is "nearest neighbor" meaningful?* In C. Beeri and P. Bruneman (eds.): *Proceeding of the 7th International Conference on Database Theory*, vol. 1540 of *Lecture Notes in Computer Science*, pp. 217–235. Springer, 1999. http://www.springerlink.com/link.asp?id=04p94cqnbge862kh. 56

[14] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon: *XQuery 1.0: An XML query language, W3C working draft*. Techn. rep., W3C, 2004. http://www.w3.org/TR/2004/WD-xquery-20041029/. 26, 78, 135

[15] U. Bohnacker, L. Dehning, J. Franke, and I. Renz: *Textual analysis of customer statements for quality control and help desk support*. In K. Jajuga, A. Sokolowski, and H.-H. Bock (eds.): *Classification, Clustering, and Data Analysis. Recent Advances and Applications. Proceedings of the 8th Conference of the International Federation of Classification Societies (IFCS-2002)*, vol. 21 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pp. 437–445. Springer, 2002. 8

[16] D. Boley, M. Gini, R. Gross, E.-H. S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore: *Partitioning-based clustering for web document categorization*. Decision Support Systems, 27(3):329–341, 1999. http://dx.doi.org/10.1016/S0167-9236(99)00055-X. 51, 98

[17] N. Bradley: *The XML Companion*. Addison-Wesley, 2000. 26, 28, 29, 30

[18] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau: *Extensible markup language (XML) 1.0 (third edition), W3C recommendation*. Techn. rep.,

W3C, 2004. http://www.w3.org/TR/2004/REC-xml-20040204. 25, 28, 34

[19] S. Brin and L. Page: *The anatomy of a large-scale hypertextual web search engine*. In Enslow, Jr., P. H. and Ellis [51], pp. 107–117. http://dx.doi.org/10.1016/S0169-7552(98)00110-X. 31, 71, 72

[20] P. Buneman: *Semistructured data*. In A. Mendelzon and Z. M. Özsoyoglu (eds.): *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 117–121. ACM Press, 1997. http://doi.acm.org/10.1145/263661.263675. 21, 24, 78

[21] V. Bush: *As we may think*. The Atlantic Monthly, 176(1):101–108, 1945. http://www.ps.uni-sb.de/~duchier/pub/vbush/. 2, 5, 6, 135

[22] S. Chakrabarti: *Mining the Web - Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003. 11, 31, 47, 62, 64, 67, 71, 72, 73, 108

[23] S. Chakrabarti, B. E. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg: *Automatic resource compilation by analyzing hyperlink structure and associated text*. In Enslow, Jr., P. H. and Ellis [51], pp. 65–74. http://dx.doi.org/10.1016/S0169-7552(98)00087-7. 72

[24] E. Chávez and G. Navarro and R. Baeza-Yates and J. L. Marroquín: *Searching in metric spaces*. ACM Comput. Surv., 33(3):273–321, 2001. http://doi.acm.org/10.1145/502807.502808. 50, 104

[25] J. Cheney: *Compressing XML with multiplexed hierarchical PPM models*. In *Proceedings of the Data Compression Conference (DCC 2001)*, pp. 163–172. IEEE Computer Society, 2001. http://www.cs.cornell.edu/People/jcheney/papers/cheney-dcc2001.pdf. 68

[26] Y. Chiaramella: *Information retrieval and structured documents*. In Agosti, M. *et al.* [1], pp. 286–309. http://www.springerlink.com/link.asp?id=yx9n79wbrk3d9adt. 16, 19, 23

[27] R. Cilibrasi and P. Vitanyi: *Clustering by compression*. IEEE Transactions on Information Theory, 51(4), 2005. http://arxiv.org/abs/cs/0312044. 68

[28] J. Clark and S. DeRose: *XML Path language (XPath) version 1.0, W3C recommendation*. Techn. rep., W3C, 1999. http://www.w3.org/TR/xpath. 29, 135

[29] W. S. Cooper: *The formalism of probability theory in IR: a foundation or an encumbrance?* In W. B. Croft and C. J. van Rijsbergen (eds.): *Proceedings of the 17th An-*

nual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 242–247. Springer, 1994. http://portal.acm.org/citation.cfm?id=188562. 66

[30] T. M. Cover and J. A. Thomas: *Elements of Infomation Theory*. John Wiley & Sons, 1991. 60, 68

[31] J. Cowan and R. Tobin: *XML Information Set (second edition)*. Techn. rep., W3C, 2004. http://www.w3.org/TR/2004/REC-xml-infoset-20040204/. 27, 135, 136

[32] J. Cowie and W. Lehnert: *Information extraction*. Commun. ACM, 39(1):80–91, 1996. http://doi.acm.org/10.1145/234173.234209. 18

[33] M. Cristo, P. Calado, E. S. de Moura, N. Ziviani1, and B. Ribeiro-Neto: *Link information as a similarity measure in web classification*. In G. Goos, J. Hartmanis, and J. van Leeuwen (eds.): *String Processing and Information Retrieval*, vol. 2857 of *Lecture Notes in Computer Science*, pp. 43–55. Springer, 2003. http://www.springerlink.com/link.asp?id=aryan7c17m1b94ta. 71, 136

[34] W. B. Croft and H. Turtle: *A retrieval model incorporating hypertext links*. In R. Akscyn (ed.): *Proceedings of the Second Annual ACM Conference on Hypertext*, pp. 213–224. ACM Press, 1989. http://doi.acm.org/10.1145/74224.74242. 85

[35] C. Crouch, S. Apte, and H. Bapat: *An approach to structured retrieval based on the extended vector model*. In Fuhr, N. *et al.* [65], pp. 89–93. http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf. 79

[36] M. Cutler, Y. Shih, and W. Meng: *Using the structure of HTML documents to improve retrieval*. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97)*. USENIX, 1997. http://www.usenix.org/publications/library/proceedings/usits97/full_papers/cutler/cutler.pdf. 102

[37] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey: *Scatter/gather: a cluster-based approach to browsing large document collections*. In N. Belkin, P. Ingwersen, A. M. Pejtersen, and E. A. Fox (eds.): *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 318–329. ACM Press, 1992. http://doi.acm.org/10.1145/133160.133214. 38, 96, 109

[38] M. Damashek: *Gauging similarity with n-grams: Language-independent categorization of text*. Science, 267(5199):843–849, 1995. http://gnowledge.sourceforge.net/damashek-ngrams.pdf. 67

[39] T. H. Davenport and L. Prusak: *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, 1998. 6, 7

[40] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman: *Indexing by latent semantic analysis*. Journal of the American Society for Information Science, 41(6):391–407, 1990. http://www3.interscience.wiley.com/cgi-bin/abstract/10049585/ABSTRACT. 64

[41] A. P. Dempster, N. M. Laird, and D. B. Rubin: *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, 39(1):1–38, 1977. http://links.jstor.org/sici?sici=0035-9246%281977%2939%3A1%3C1%3AMLFIDV%3E2.0.CO%3B2-Z. 52

[42] S. DeRose, E. Maler, and R. Daniel, Jr.: *XML Pointer language (XPointer) version 1.0, W3C Candidate Recommendation*. Techn. rep., W3C, 2001. http://www.w3.org/TR/2001/CR-xptr-20010911/. 30

[43] S. DeRose, E. Maler, and D. Orchard: *XML Linking language (XLink) version 1.0, W3C recommendation*. Techn. rep., W3C, 2001. http://www.w3.org/TR/xlink. 26, 30

[44] D. Dhyani, W. K. Ng, and S. S. Bhowmick: *A survey of web metrics*. ACM Comput. Surv., 34(4):469–503, 2002. http://doi.acm.org/10.1145/592642.592645. 69, 70

[45] M. Dixon: *An overview of document mining technology*, 1997. http://citeseer.ist.psu.edu/dixon97overview.html. 19, 20

[46] R. O. Duda, P. E. Hart, and D. G. Stork: *Pattern Classification*. John Wiley & Sons, 2001. 37, 51, 52, 53

[47] S. T. Dumais: *Improving the retrieval of information from external sources*. Behavior Research Methods, Instruments, & Computers, 23(2):229–236, 1991. http://psychonomic.org/search/view.cgi?id=5145. 63, 135

[48] S. Džeroski: *Multi-relational data mining: An introduction*. ACM SIGKDD Newsletter, 5(1):1–16, 2003. http://www.acm.org/sigs/sigkdd/explorations/issue5-1/Dzeroski.pdf. 11, 23, 135

[49] J. Dörre and P. Gerstl and R. Seiffert: *Text mining: finding nuggets in mountains of textual data*. In U. Fayyad, S. Chaudhuri, and D. Madigan (eds.): *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data*

mining, pp. 398–401. ACM Press, 1999. http://doi.acm.org/10.1145/312129.312299. 11, 19

[50] R. A. Elmasri and S. Navathe: *Fundamentals of Database Systems*. Addison-Wesley, 2000. 3, 4, 6, 31

[51] P. H. Enslow, Jr. and A. Ellis (eds.): *Proceedings of the Seventh International Conference on World Wide Web 7*. Elsevier, 1998. 121

[52] L. Ertoz, M. Steinbach, and V. Kumar: *Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data*. In D. Barbara and C. Kamath (eds.): *Proceedings of the Third SIAM International Conference on Data Mining (SDM 2003)*, vol. 112 of *Proceedings in Applied Mathematics*. Society for Industrial and Applied Mathematics, 2003. http://www.siam.org/meetings/sdm03/proceedings/sdm03_05.pdf. 48, 49, 106

[53] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu: *A density-based algorithm for discovering clusters in large spatial databases with noise*. In E. Simoudis, J. Han, and U. Fayyad (eds.): *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, pp. 226–231. AAAI Press, 1996. http://www.cs.ualberta.ca/~joerg/papers/KDD-96_final.pdf. 44, 48, 106, 135

[54] V. Estivill-Castro: *Why so many clustering algorithms: a position paper*. SIGKDD Explor. Newsl., 4(1):65–75, 2002. http://doi.acm.org/10.1145/568574.568575. 35, 42, 45, 46, 51, 53, 59

[55] O. Etzioni: *The world-wide web: quagmire or gold mine?* Commun. ACM, 39(11):65–68, 1996. http://doi.acm.org/10.1145/240455.240473. 11

[56] D. C. Fallside: *XML Schema part 0: Primer*. Techn. rep., W3C, 2001. http://www.w3.org/TR/xmlschema-0/. 29

[57] C. Faloutsos and K.-I. Lin: *Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets*. SIGMOD Rec., 24(2):163–174, 1995. http://doi.acm.org/10.1145/568271.223812. 108

[58] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth: *From data mining to knowledge discovery in databases*. AI Magazine, 17:37–54, 1996. http://citeseer.ist.psu.edu/fayyad96from.html. 7, 9, 13, 14, 135

[59] E. A. Fox, G. L. Nunn, and W. C. Lee: *Coefficients of combining concept classes in a collection*. In Y. Chiaramella (ed.): *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.

291–307. ACM Press, 1988. http://doi.acm.org/10.1145/62437.62465. 70, 78, 84, 86

[60] E. A. Fox and J. A. Shaw: *Combination of multiple searches*. In *The Second Text REtrieval Conference (TREC-2)*, vol. 500-215 of *NIST Special Publication*, pp. 243–252. NIST, 1994. http://trec.nist.gov/pubs/trec2/papers/ps/vpi.ps.gz. 87, 88

[61] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus: *Knowledge discovery in databases: An overview*. AI Magazine, 13(3):57–70, 1992. http://citeseer.ist.psu.edu/frawley92knowledge.html. 7, 9, 13, 14, 135

[62] M. E. Frisse: *Searching for information in a hypertext medical handbook*. In J. B. Smith and F. Halasz (eds.): *Proceeding of the ACM Conference on Hypertext*, pp. 57–66. ACM Press, 1987. http://doi.acm.org/10.1145/317426.317433. 79

[63] N. Fuhr: *Models in information retrieval*. In Agosti, M. *et al.* [1], pp. 21–50. http://www.springerlink.com/link.asp?id=02bmlrvcaax428pn. 16, 18, 47, 66

[64] N. Fuhr and K. Großjohann: *XIRQL: a query language for information retrieval in XML documents*. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel (eds.): *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 172–180. ACM Press, 2001. http://doi.acm.org/10.1145/383952.383985. 78, 80, 135

[65] N. Fuhr, S. Malik, and M. Lalmas (eds.): *INEX 2003 Workshop Proceedings*. INEX, 2003. http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf. 120, 122

[66] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson: *Structured answers for a large structured document collection*. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 204–213. ACM Press, 1993. http://doi.acm.org/10.1145/160688.160720. 23, 24, 79, 80

[67] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French: *Clustering large datasets in arbitrary metric spaces*. In *Proceedings of the 15th International Conference on Data Engineering*, pp. 502–511. IEEE Computer Society, 1999. http://intl.ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=754966. 50, 104, 106

[68] A. J. Gilliland-Swetland: *Setting the stage*. In M. Baca (ed.): *Introduction to Metadata: Pathways to Digital Information*. Getty Research Institute, 2000. http://www.getty.edu/research/institute/standards/intrometadata/. 31

[69] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake: *Using web structure for classifying and describing web pages*. In D. Lassner, D. De Roure, and A. Iyengar (eds.): *Proceedings of the Eleventh International Conference on World Wide Web*, pp. 562–569. ACM Press, 2002. http://doi.acm.org/10.1145/511446.511520. 72

[70] R. Goldman and J. Widom: *Dataguides: Enabling query formulation and optimization in semistructured databases*. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld (eds.): *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pp. 436–445. Morgan Kaufmann, 1997. citeseer.ist.psu.edu/126680.html. 77, 135

[71] O. Gospodnetic: *Parsing, indexing, and searching XML with digester and lucene*. Techn. rep., IBM DeveloperWorks, 2003. http://www-106.ibm.com/developerworks/java/library/j-lucene/. 102

[72] T. R. Gruber: *Toward principles for the design of ontologies used for knowledge sharing*. Tech. Rep. KSL-93-04, Knowledge Systems Laboratory, Stanford University, 1993. http://citeseer.ist.psu.edu/gruber93toward.html. 31, 135

[73] S. Guha, R. Rastogi, and K. Shim: *CURE: an efficient clustering algorithm for large databases*. In Haas, L. *et al.* [74], pp. 73–84. http://doi.acm.org/10.1145/276304.276312. 98, 114

[74] L. Haas, P. Drew, A. Tiwary, and M. Franklin (eds.): *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1998. 119, 126

[75] M. Halkidi, M. Vazirgiannis, and Y. Batistakis: *On clustering validation techniques*. Journal of Intelligent Information Systems, 17(2-3):107–145, 2001. http://dx.doi.org/10.1023/A:1012801612483. 54, 56, 58, 60

[76] J. Han and M. Kamber: *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000. 12, 13, 27, 35, 43, 45, 48, 54, 98, 107, 108, 118

[77] D. Hand, H. Mannila, and P. Smyth: *Principles of Data Mining*. MIT Press, 2001. 9, 10, 12, 13, 21, 58, 63, 113

[78] P. Hannappel, R. Klapsing, A. Krug, and G. Neumann: *MSEEC - a multi search engine with multiple clustering*. In M. KhosrowPour (ed.): *Managing Information Technology Resources in Organizations in the Next Millennium: Proceedings of the 10th Information Resources Management Association International Conference*. Idea Group Publishing, 1999. http://nm.wu-wien.ac.at/research/publications/mseec.pdf. 107

[79] S. Haykin: *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. 86

[80] M. A. Hearst: *Untangling text data mining*. In *37th Annual Meeting of the Association for Computational Linguistics*, pp. 3–10. Morgan Kaufmann, 1999. http://acl.ldc.upenn.edu/P/P99/P99-1001.pdf. 18, 19

[81] M. A. Hearst: *The use of categories and clusters in information access interfaces*. In *Natural Language Information Retrieval*, vol. 7 of *Text, Speech and Language Technology*, pp. 333–374. Kluwer, 1999. http://www.sims.berkeley.edu/~hearst/papers/cats-and-clusters.pdf. 40, 47, 96, 99, 136

[82] J. Hiler: *Google time bomb*. Microcontent news, March 2002. http://www.microcontentnews.com/articles/googlebombs.htm. 73

[83] A. K. Jain, M. N. Murty, and P. J. Flynn: *Data clustering: a review*. ACM Comput. Surv., 31(3):264–323, 1999. http://doi.acm.org/10.1145/331499.331504. 9, 41, 42, 45, 47, 49, 54, 57, 97

[84] F. Jiang and M. L. Littman: *Approximate dimension equalization in vector-based information retrieval*. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 423–430. Morgan Kaufmann Publishers Inc., 2000. http://citeseer.ist.psu.edu/jiang00approximate.html. 64, 65, 135

[85] A.-J. Kaijanaho: *Open Source vai vapaaohjelma*, 2002. http://www.mit.jyu.fi/antkaij/vapaa/ossko. 89

[86] G. Karypis, E.-H. Han, and V. Kumar: *Chameleon: Hierarchical clustering using dynamic modeling*. Computer, 32(8):68–75, 1999. http://dx.doi.org/10.1109/2.781637. 106, 135

[87] T. Kilpeläinen: *Digitalization in relation to organizational communication and information overload*. Master's thesis, University of Jyväskylä, 2004. 6, 7

[88] J. M. Kleinberg: *Authoritative sources in a hyperlinked environment*. In H. Karloff (ed.): *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algo-*

rithms, pp. 668–677. Society for Industrial and Applied Mathematics, 1998. http://portal.acm.org/citation.cfm?id=315045. 71, 72

[89] T. R. Kochtanek: *Bibliographic compilation using reference and citation links*. Information Processing & Management, 18(1):33–39, 1982. http://dx.doi.org/10.1016/0306-4573(82)90049-8. 70

[90] Y. Kodratoff: *Knowledge discovery in texts: A definition and applications*. In Z. W. Ras and A. Skowron (eds.): *Foundations of Intelligent Systems, 11th International Symposium, ISMIS '99, Proceedings*, vol. 1609 of *Lecture Notes in Computer Science*, pp. 16–29. Springer, 1999. http://citeseer.ist.psu.edu/kodratoff99knowledge.html. 7, 9, 18, 20

[91] P. Koikkalainen: *Tilastollisen hahmontunnistuksen perusteet*. Luentomoniste, Jyväskylän yliopisto, Tietotekniikan laitos, 2000. http://erin.mit.jyu.fi/pako/kurssit/th2000/. 5

[92] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri: *An MDL framework for data clustering*. Tech. Rep. 2002-8, Helsinki Institute for Information Technology, 2002. http://cosco.hiit.fi/Articles/hiit-2002-8.pdf. 60

[93] R. Kosala and H. Blockeel: *Web mining research: a survey*. SIGKDD Explor. Newsl., 2(1):1–15, 2000. http://doi.acm.org/10.1145/360402.360406. 11

[94] J. H. Kroeze, M. C. Matthee, and T. J. D. Bothma: *Differentiating data- and text-mining terminology*. In J. Eloff, A. Engelbrecht, P. Kotzé, and M. Eloff (eds.): *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology*, pp. 93–101. South African Institute for Computer Scientists and Information Technologists, 2003. http://portal.acm.org/citation.cfm?id=954024. 18, 19, 135

[95] R. Kärki ja T. Kortelainen: *Johdatus bibliometriikkaan*. Informaatiotutkimuksen yhdistys, 1996. 69

[96] T. Kärkkäinen and S. Äyrämö: *Robust clustering methods for incomplete and erroneous data*. In C. A. Brebbia, N. F. F. Ebecken, and A. Zanasi (eds.): *Data Mining V: Data Mining, Text Mining and Their Business Applications: Fifth International Conference on Data Mining*, vol. 33 of *Information and Communication Technologies*. WIT Press, 2004. 43, 48

[97] J. H. Lee: *Analyses of multiple evidence combination*. SIGIR Forum, 31(SI):267–276, 1997. http://doi.acm.org/10.1145/278459.258587. 88

[98] J.-W. Lee, K. Lee, and W. Kim: *Preparations for semantics-based XML mining*. In N. Cercone, T. Y. Lin, and X. Wu (eds.): *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 345–352. IEEE Computer Society, 2001. http://dx.doi.org/10.1109/ICDM.2001.989538. 83, 99

[99] A. Leuski and J. Allan: *Lighthouse: Showing the way to relevant information*. In *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, pp. 125–129. IEEE Computer Society, 2000. http://www-ciir.cs.umass.edu/~leouski/publications/papers/ir-205.pdf. 97, 109

[100] W. Lian, D. W. lok Cheung, N. Mamoulis, and S.-M. Yiu: *An efficient and scalable algorithm for clustering XML documents by structure*. IEEE Transactions on Knowledge and Data Engineering, 16(1):82–96, 2004. http://dx.doi.org/10.1109/TKDE.2004.1264824. 83

[101] R. W. Luk, H. Leong, T. S. Dillon, A. T. Chan, W. B. Croft, and J. Allan: *A survey in indexing and searching XML documents*. Journal of the American Society for Information Science and Technology, 53(6):415–437, 2002. http://doi.wiley.com/10.1002/asi.10056. 24, 25, 73, 74, 78, 79, 80

[102] V. Lyytikäinen: *Contextual and Structural Metadata in Enterprise Document Management*. PhD thesis, University of Jyväskylä, 2004. http://selene.lib.jyu.fi:8080/vaitos/studies/studcomp/9513917835.pdf. 31

[103] H. Mannila: *Theoretical frameworks for data mining*. SIGKDD Explor. Newsl., 1(2):30–32, 2000. http://doi.acm.org/10.1145/846183.846191. 14

[104] F. Manola and E. Miller: *RDF primer, W3C recommendation*. Techn. rep., W3C, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. 31, 32, 136

[105] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller: *Introduction to wordnet: An on-line lexical database*. Journal of Lexicography, 3(4):235–244, 1990. ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.pdf. 41

[106] D. S. Modha and W. S. Spangler: *Clustering hypertext with applications to web searching*. In F. M. Shipman, III, P. J. Nürnberg, and D. L. Hicks (eds.): *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pp. 143–152. ACM Press, 2000. http://doi.acm.org/10.1145/336296.336351. 31, 85

[107] T. H. Nelson: *Structure, tradition and possibility*. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, p. 1. ACM Press, 2003. http://doi.acm.org/10.1145/900051.900053. 23

[108] C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel, and L. Seligman (eds.): *Proceedings of the Eleventh International Conference on Information and Knowledge Management*. ACM Press, 2002. 131, 133

[109] F. Osareh: *Bibliometrics, citation analysis and co-citation analysis: A review of literature I*. Libri, 46(3):149–158, 1996. 69

[110] P. Pirolli, J. Pitkow, and R. Rao: *Silk from a sow's ear: extracting usable structures from the web*. In M. J. Tauber, B. Nardi, and G. C. van der Veer (eds.): *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 118–125. ACM Press, 1996. http://doi.acm.org/10.1145/238386.238450. 85

[111] R. Rao: *From IR to search and beyond*. ACM Queue, 2(3), 2004. http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=148. 116

[112] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel: *Database systems for structured documents*. IEICE Transactions on Information and Systems, 34-D(11):1335–1342, 1995. http://citeseer.ist.psu.edu/476526.html. 76, 77, 135

[113] M. Sahami, M. A. Hearst, and E. Saund: *Applying the multiple cause mixture model to text categorization*. In L. Saitta (ed.): *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96)*, pp. 435–443. Morgan Kaufmann, 1996. http://citeseer.ist.psu.edu/sahami96applying.html. 47

[114] A. Salminen: *XML family of languages*. Techn. rep., Jyväskylän yliopisto, Tietojenkäsittelytieteiden laitos, 2004. http://www.cs.jyu.fi/~airi/xmlfamily-20040907.html. 26

[115] G. Salton and C. Buckley: *Term-weighting approaches in automatic text retrieval*. Information Processing & Management, 24(5):513–523, 1988. http://dx.doi.org/10.1016/0306-4573(88)90021-0. 62, 63

[116] G. Salton, A. Wong, and C. S. Yang: *A vector space model for automatic indexing*. Commun. ACM, 18(11):613–620, 1975. http://doi.acm.org/10.1145/361219.361220. 62

[117] J. Savoy: *An extended vector-processing scheme for searching information in hypertext systems*. Information Processing & Management, 32(2):155–170, 1996. http://dx.doi.org/10.1016/S0306-4573(96)85003-5. 70, 84

[118] A. Schenker, M. Last, H. Bunke, and A. Kandel: *Comparison of distance measures for graph-based clustering of documents*. In E. R. Hancock and M. Vento (eds.): *Proceedings of the Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop (GbRPR 2003)*, vol. 2726 of *Lecture Notes in Computer Science*, pp. 202–213. Springer, 2003. http://springerlink.metapress.com/link.asp?id=bxkjxj1vt9mlxylx. 51, 59, 83

[119] T. Schlieder and H. Meuss: *Querying and ranking XML documents*. Journal of the American Society for Information Science and Technology, 53(6):489–503, 2002. http://www.cis.uni-muenchen.de/people/Meuss/Pub/JASIS02.pdf. 51, 80, 82, 99, 113

[120] F. Sebastiani: *Machine learning in automated text categorization*. ACM Comput. Surv., 34(1):1–47, 2002. http://doi.acm.org/10.1145/505282.505283. 37, 57

[121] U. Shah, T. Finin, and A. Joshi: *Information retrieval on the semantic web*. In Nicholas, C. *et al.* [108], pp. 461–468. http://doi.acm.org/10.1145/584792.584868. 32

[122] N. Sharma: *The origin of the data information knowledge wisdom hierarchy*, 2004. http://www-personal.si.umich.edu/~nsharma/dikw_origin.htm. 6

[123] W. M. Shaw, Jr., R. Burgin, and P. Howell: *Performance standards and evaluations in IR test collections: Cluster-based retrieval models*. Information Processing & Management, 33(1):1–14, 1997. http://dx.doi.org/10.1016/S0306-4573(96)00043-X. 39, 58

[124] N. Slonim and N. Tishby: *Document clustering using word clusters via the information bottleneck method*. In E. Yannakoudakis, N. J. Belkin, M.-K. Leong, and P. Ingwersen (eds.): *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 208–215. ACM Press, 2000. http://doi.acm.org/10.1145/345508.345578. 57, 68

[125] A. Strehl: *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, 2002. http://strehl.com/download/strehl-phd.pdf. 35, 46, 50, 53, 54, 56, 57, 58, 59, 63, 105, 107, 108

[126] J. M. Tague (ed.): *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 1985. 132, 133

[127] J. Tantrum: *Model Based and Hybrid Clustering of Large Datasets*. PhD thesis, University of Washington, 2003. http://www.stat.washington.edu/tantrum/thesis.pdf. 46, 64, 84, 135

[128] M. Thelwall and D. Wilkinson: *Finding similar academic web sites with links, bibliometric couplings and colinks*. Information Processing & Management, 40(3):515–526, 2004. http://dx.doi.org/10.1016/S0306-4573(03)00042-6. 70, 136

[129] C. J. van Rijsbergen: *Information Retrieval*. London: Butterworths, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html. 17, 38, 39, 59, 63, 66

[130] M. A. Vapa, N. P. Kotilainen, A. K. Auvinen, H. M. Kainulainen, and J. T. Vuori: *Resource discovery in P2P networks using evolutionary neural networks*. In *Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA 2004)*. IEEE Computer Society, 2004. http://www.cc.jyu.fi/~npkotila/documents/NeuroSearchAISTA2004.pdf. 71

[131] E. M. Voorhees: *The cluster hypothesis revisited*. In Tague, J. M. [126], pp. 188–196. http://doi.acm.org/10.1145/253495.253524. 38, 39, 40, 99

[132] K. Wang and H. Liu: *Discovering typical structures of documents: a road map approach*. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel (eds.): *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 146–154. ACM Press, 1998. http://doi.acm.org/10.1145/290941.290982. 21, 82, 83

[133] F. Weigel: *A survey of indexing techniques for semistructured documents*. Project thesis, Institute of Computer Science, LMU, Munich, 2002. http://www.pms.ifi.lmu.de/publikationen/#PA_Felix.Weigel. 73, 74, 76, 77, 80

[134] F. Weigel, H. Meuss, K. U. Schulz, and F. Bry: *Content and structure in indexing and ranking XML*. In S. Amer-Yahia and L. Gravano (eds.): *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004*, pp. 67–72, 2004. http://webdb2004.cs.columbia.edu/papers/5-2.pdf. 73, 74, 77, 135

[135] R. Weiss, B. Vélez, and M. A. Sheldon: *HyPursuit: a hierarchical network search engine that exploits content-link hypertext clustering*. In D. Stotts (ed.): *Proceedings of the the Seventh ACM Conference on Hypertext*, pp. 180–193. ACM Press, 1996. http://doi.acm.org/10.1145/234828.234846. 85

[136] P. Willett: *Recent trends in hierarchic document clustering: A critical review*. Information Processing & Management, 24(5):577–597, 1988. http://dx.doi.org/10.

1016/0306-4573(88)90027-1. 38, 40, 41, 43, 58, 63, 65, 105

[137] J. E. Wolff, H. Flörke, and A. B. Cremers: *Searching and browsing collections of structural information*. In *Proceedings of the IEEE Advances in Digital Libraries 2000*, p. 141. IEEE Computer Society, 2000. http://dx.doi.org/10.1109/ADL.2000.848377. 80, 135

[138] S. K. M. Wong, W. Ziarko, and P. C. N. Wong: *Generalized vector spaces model in information retrieval*. In Tague, J. M. [126], pp. 18–25. http://doi.acm.org/10.1145/253495.253506. 63

[139] M. Wu, M. Fuller, and R. Wilkinson: *Using clustering and classification approaches in interactive retrieval*. Information Processing & Management, 37(3):459–484, 2001. http://dx.doi.org/10.1016/S0306-4573(00)00057-1. 40, 96, 108

[140] K. Yang: *Combining Text-, Link-, and Classification-based Retrieval Methods to Enhance Information Discovery on the Web*. PhD thesis, University of North Carolina, 2002. http://www.webir.org/resources/phd/Yang_2002.pdf. 36, 37, 38, 39, 40, 59, 84, 85, 86, 87, 88

[141] O. E. Zamir: *Clustering Web Documents: A Phrase-based Method for Grouping Search Engine Results*. PhD thesis, University of Washington, 1999. http://www.webir.org/resources/phd/zamir-thesis-pdf.zip. 38, 39, 40, 47, 67, 98, 108, 135

[142] J. Zhang, J. Gao, M. Zhou, and J. Wang: *Improving the effectiveness of information retrieval with clustering and fusion*. Computational Linguistics and Chinese Language Processing, 6(1):1–18, 2001. http://rocling.iis.sinica.edu.tw/CLCLP/Vol6-1/paper5.pdf. 40, 88

[143] Y. Zhao and G. Karypis: *Evaluation of hierarchical clustering algorithms for document datasets*. In Nicholas, C. *et al.* [108], pp. 515–524. http://doi.acm.org/10.1145/584792.584877. 53, 65

[144] S. Zhong and J. Ghosh: *A unified framework for model-based clustering*. Journal of Machine Learning Research, 4:1001–1037, 2003. http://www.ai.mit.edu/projects/jmlr//papers/v4/zhong03a.html. 42, 45, 46, 58

[145] S. Äyrämö and T. Kärkkäinen: *Data mining – principles and basic applications*. Techn. rep., Jyväskylän yliopisto, Agora Center, 2003. 8, 10

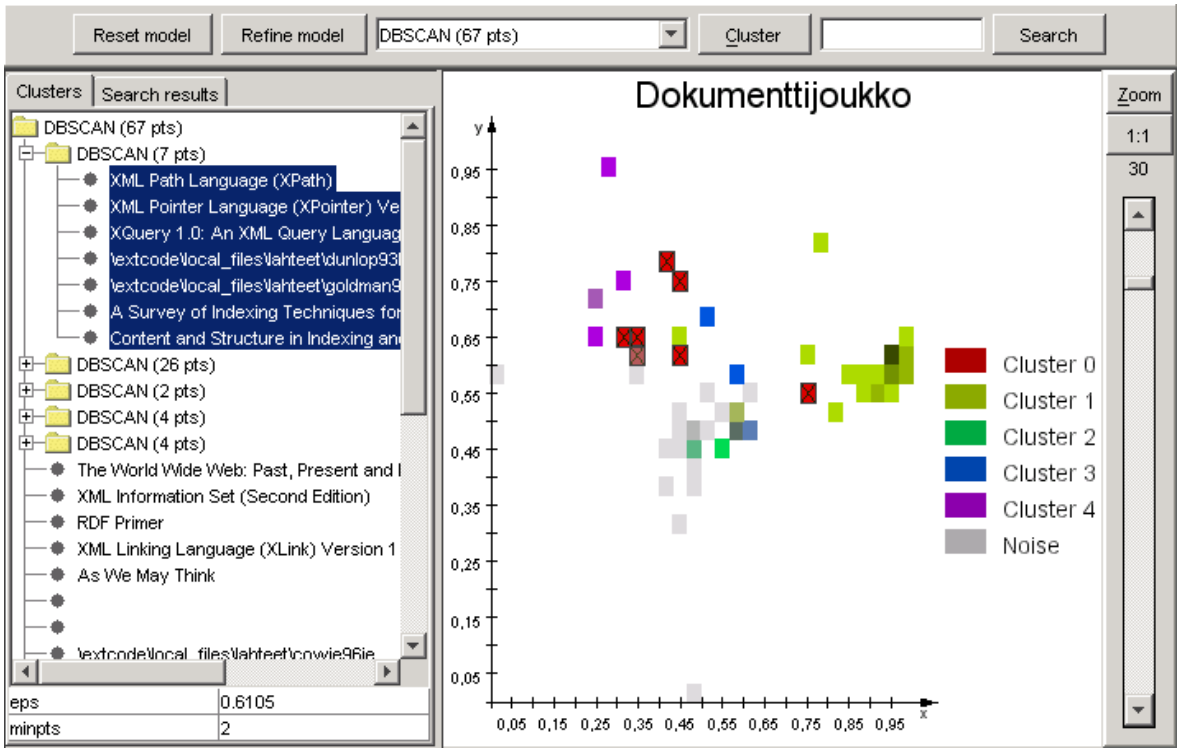
Lähteiden WWW-linkit tarkastettu 17.1.2005.

A Käyttöesimerkki: lähdeartikkelien klusterointi

Esimerkkinä ExtMiner-ohjelman käytöstä käydään läpi tämän tutkielman digitaalisessa muodossa olevien lähteiden klusterointi. Lähteistä valittiin käsittelyyn artikkelit ja opinnäytteet. Kirjat ja kokoelmateokset sivuutettiin, koska ne olisivat todennäköisesti vääristäneet hakutuloksia. Lähtötiedostoina oli 122 PDF- ja HTML-tiedostoa. Tiedostot esikäsiteltiin pdftohtml ja HTML Tidy -ohjelmilla, joista tuloksena saadut XML-tiedostot tarkastettiin. Osa PDF-tiedostoista koostui lähinnä skannatuista kuvista, jolloin XML-tiedostoihin ei saatu juurikaan järkevää sisältöä. Muutamien PDF-tiedostojen fontteja ei saatu luettua, jolloin XML-tiedostojen teksti oli lähinnä erikoismerkkien sekamelskaa. Lisäksi osassa HTML-tiedostoja merkkaus oli niin sotkuista, ettei HTML Tidykaan saanut selvitettyä niitä. Tämän manuaalisen ”datan puhdistuksen” jälkeen ohjelman syötedata oli 71 XML-dokumenttia.

XML-jäsennyksen jälkeen dokumenttien määrä putosi 69:ään, koska XML-tiedostoihin oli jäänyt lukukelvottomia merkkejä. Todennäköisesti osaan dokumenteista oli merkitty väärä merkistö niiden sisältöön nähden. Indeksoitaessa tietoja Luceneen kokeiltiin erilaisia analysoijia (Luceneen kuuluvia luokkia, joilla indeksitermejä voidaan suodattaa ja muuntaa tekstistä). Aluksi kokeiltiin yksinkertaista ”standardianalysoijaa”, joka indeksoi kaikki dokumentin sanat jättäen pois välimerkit ja muuntaen termit pienaakkostoon. Tuloksena saatiin 32000 eri termiä. Analysoijaa muokattiin niin, että myös numerot suodatetaan pois, jolloin termejä löytyi 28000. Edelleen termejä saatiin vähennettyä omalla sulkusanalistalla ja Luceneen integroidulla Porterin stemmausalgoritmilla. Termien määrä putosi nyt 20000:een, mutta Porterin algoritmia käytettäessä on huomioitava, ettei se käsittele suomenkielisiä sanoja oikein. Kokoelmasta poistettiin suomenkieliset dokumentit, joita tässä vaiheessa oli vain kaksi kappaletta. Loppujen lopuksi dokumentteja jäi 67 ja termejä ”vain” 16000. Termien lisäksi dokumenteista indeksoitiin myös otsikot, jotka olivat muodoltaan varsin vaihtelevia.

Dokumenttikokoelma klusteroitiin sekä DBSCAN-algoritmilla että hierarkkisella keskipisteen menetelmällä. Klusteroinnissa hyödynnettiin vain indeksitermejä. Tulokset on esitetty DBSCAN-algoritmin osalta kuvassa A.1 ja hierarkkisen klusteroinnin osalta kuvassa A.2. DBSCAN-algoritmin parametrit on asetettu parametrialueen likimääräisen läpikäynnin jälkeen niin, että klusterien määrä on mahdollisimman suuri. Hierarkkista klusterimallia on muokattu käsin niin, että aiheiltaan yhtenäiset samassa alipuussa olevat klusterit on yhdistetty. Tästä huolimatta klustereita jäi DBSCAN-algoritmin tuottamaa mallia enemmän.



Kuva A.1: DBSCAN-algoritmin klusterointitulos.

DBSCAN-algoritmi tuotti seuraavat klusterit:

0. **XML-klusteri** (7 pistettä, kuvassa valittuna). Sisältää W3C:n suosituksia [28] [14] sekä XML-indeksirakenteisiin liittyviä tekstejä [70] [134].

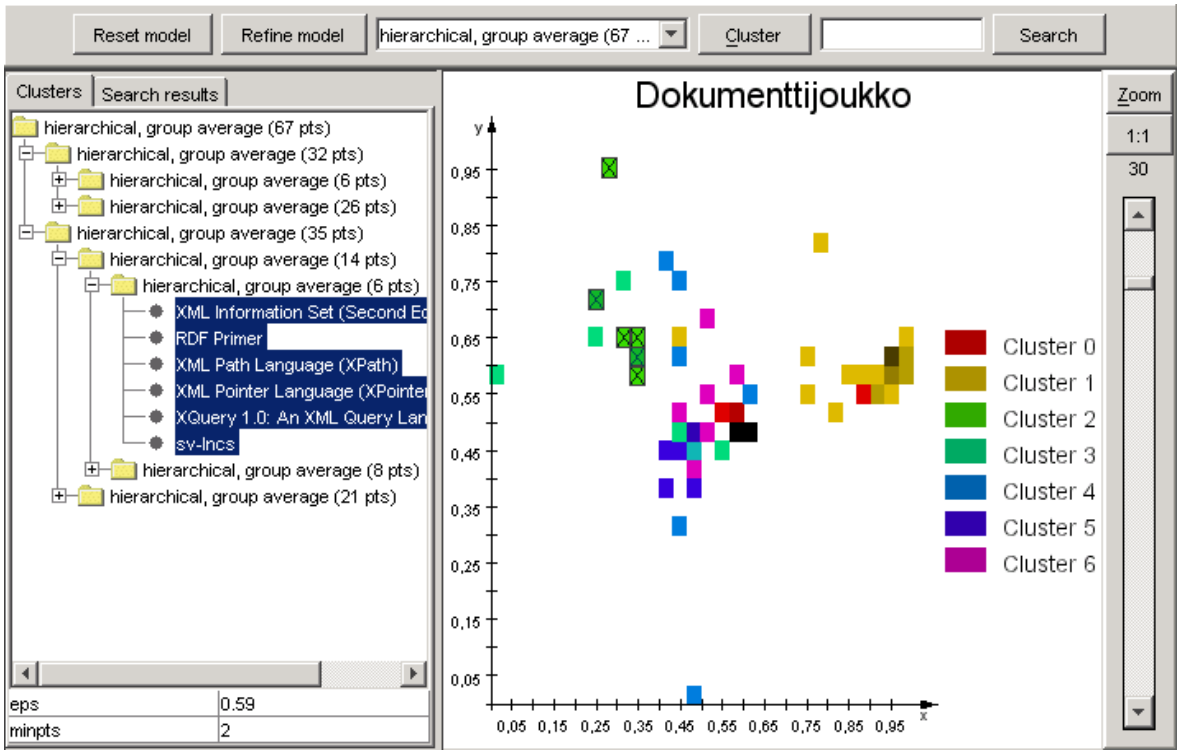
1. **Pääklusteri** (26 pistettä, kuvassa oikealla). Sisältää sekalaisia, pääasiassa klusterointiin keskittyviä artikkeleita [10] [86] [53] ja väitöskirjoja [141] [127]. Dokumentit ovat tiheässä, mistä johtuen klusteri on kooltaan suurin.

2. **LSI-klusteri** (2 pistettä, keskellä vierekkäin olevan pisteet). Klusteri on kooltaan pieni, mutta erottuu teemaltaan selkeästi muista. Kyseessä ovat indeksoidun dokumenttikokoelman ainoat artikkelit [47] [84], joissa käsitellään LSI-menetelmää.

3. **KDD-klusteri** (4 pistettä, keskialueella). Sisältää tietämyksen muodostamiseen [61], tiedonlouhintaan [58] ja tekstitiedonlouhintaan [94] liittyviä artikkeleita.

4. **XML-hakuklusteri** (4 pistettä, ylhäällä vasemmalla). Sisältää XML-hakukieliin [64] [137] ja hakumalleihin [112] liittyviä artikkeleita.

Algoritmi luokitteli 24 dokumenttia kohinaksi. Osa artikkeleista olisi teemaltaan (esim. tiedonlouhinta [48], XML [31]) kuulunut muihin klustereihin, mutta suurin osa (esim. Bushin [21] ja Gruberin [72] artikkelit) oli aiheeltaan yleisluontoisia ja kuuluivat siksi ryhmään.



Kuva A.2: Keskipisteen menetelmän klusterointitulos.

Hierarkkinen klusterointi tuotti seuraavat klusterit:

0. **Linkkiklusteri** (6 pistettä, kuvassa keskellä). Sisältää linkkianalyysin liittyviä artikkeleita [128] [33]. Hearstin [81] klusteriartikkeli kuuluisi asiallisesti (ja myös visuaalisen näkymän perusteella) klusteriin 1.
1. **Pääklusteri** (26 pistettä). Vastaa lähes tarkalleen DBSCAN-mallin klusteria 1.
2. **XML-klusteri** (6 pistettä, kuvassa valittuna). Melko lähellä DBSCAN-mallin vastaavaa klusteria, mutta sisältää myös W3C:n suosituksia, jotka DBSCAN oli luokitellut kohinaksi [31] [104].
3. **XML-hakuklusteri** (8 pistettä). Melko lähellä DBSCAN-mallin klusteria 4. Myös LSI-artikkelit (eivät liity XML:ään, mutta ovat hakumalleja) on sijoitettu klusteriin.
4. **XML-indeksointiklusteri** (7 pistettä). Rakenteiseen indeksointiin liittyviä dokumentteja, joista osa oli DBSCAN-mallin klusterissa 0.
5. **Yleisklusteri** (6 pistettä, keskialueella alavasemmalla). Sisältää aiheeltaan yleisluontoisia artikkeleita, jotka DBSCAN-algoritmi oli luokitellut kohinaksi.
6. **KDD-klusteri** (8 pistettä). DCSCAN-mallin klusteri 3 laajennettuna.

B MIT/X -lisenssi

Empiirisessä osuudessa toteutettu ExtMiner-sovellus on GridChart-komponenttia lukuunottamatta lisensoitu tällä lisenssillä. Lisäksi sovellus käyttää LGPL- ja Apache 2.0 -lisensoituja komponentteja.