Transformaatiot sisällönhallinnassa

Siru Örström-Koch (siru.orstrom-koch@helsinki.fi)

Helsinki 20.5.2001

Pro gradu -tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Nykypäivän tietotulva on valtava, tietoa löytyy loputtomiin ja sitä on saatavilla yhä useampien kanavien kautta. Informaatiovolyymi on jo kauan sitten ylittänyt käyttäjän kyvyn kahlata läpi kaikki syötetieto ja erottaa relevantit asiat tästä tulvasta [OS95]. Puhutaan mediakonvergenssista, jossa internet, radio ja televisio yhdistyvät; samaan verkkoon voidaan mennä joko tietokoneen, television tai matkapuhelimen kautta. Laadukkaan ja luotettavan tiedon helppo saatavuus onkin pelkästään hyvä asia. Merkittäväksi tekijäksi nousee sisällönhallinta, joka käsittelee tietoa siten, että käyttäjä saa tiedon kaikkein informatiivisimmassa muodossa, esimerkiksi suodattamalla ja lajittelemalla tietoa.

Tutkielmassa tutkitaan sisällönhallintaprosessia ja tietoon kohdistuvia muunnoksia eli transformaatioita prosessin eri vaiheissa. Tietolähteet ovat heterogeenisiä ja kustakin lähteestä halutaan erotella vain oleellisin ja kiinnostavin tieto. Jos etsittävä tieto sattuu löytymään ripoteltuna useaan eri lähteeseen, on toivottavaa, että tietokone tekee yhdistelyn automaattisesti. Tavoitteena on tarjota käyttäjälle tietoa ongelmanratkaisun ja päätöksenteon pohjaksi ja oleellisen tiedon edelleenjalostamiseksi. Usean tietolähteen samanaikainen seuraaminen, tietojen yhdistäminen, soveltaminen ja vertailu ovat monissa työtehtävissä tärkeitä ja niiden tulee tapahtua automaattisesti. Ongelmana ei ole tiedon olemassaolo – tietoa kyllä löytyy – vaan tiedon esittäminen ja yhdistäminen käyttäjää eniten palvelevassa muodossa.

Heterogeeniset informaatiolähteet vaativat rakenteellisen ja rakenteettoman tiedon yhdistettyä hallintaa. Tieto on mahdollisesti tallennettu eri järjestelmiin (tietokannat, tiedostojärjestelmät, multimedia-laitteet) ja tiedon jakeluun on eri muotoja (eri laitteet ja eri tarpeet), joten prosessi sisältää monia muunnoksia. Tiedon esittämismuotoon vaikuttaa esimerkiksi se, että sama tieto täytyy voida muuntaa sisällöllisesti, ulkonäöllisesti ja teknisesti eri muotoihin (esimerkiksi sama tekstisisältö lyhyempänä ja ilman kuvia WAPiin ja pidemmässä muodossa kuvien kera WWW:hen).

Luvussa 2 tutustutaan sisällönhallintaprosessiin, joka koostuu tiedon syöttämisestä, prosessoinnista, tallentamisesta ja esittämisestä. Tutkielma painottuu tiedon prosessointiin, mutta myös tiedon syöttämistä ja esittämistä käsitellään lyhyesti. Luvussa 3 kartoitetaan erilaisia tiedon esitystapoja. Luvussa 4 käsitellään varsinaisia tietoon kohdistuvia muunnoksia. Transformaatiot on tutkielmassa jaoteltu kolmeen osaan: formaatti-, tiedon poimimis- ja tiedon yhdistämistransformaatioihin. Formaattitransformaatiot nähdään tiedon esitysmuodon muunnoksina: tietosisältö säilyy samanlaisena, mutta esimerkiksi eri mediat vaativat erilaisen tiedon esitysmuodon. Tiedon poimimistransformaatioilla tarkoitetaan yhteen tietolähteeseen kohdistettavia muunnoksia: kuinka yhdestä tietolähteestä erotetaan tärkein ja käyttäjää palvelevin tieto. Yhdistämistransformaatiot tekevät muunnokset, joita tarvitaan yhdisteltäessä eri tietolähteiden tietoja.

Tietotekniikka lisää käyttäjän kognitiivisia vaatimuksia, jotka kohdistuvat erityisesti havaitsemis- ja muistitoimintoihin sekä tiedon seuraamiseen, tarkkaavaisuuteen ja ongelmanratkaisukykyyn. Vielä toistaiseksi on olemassa vain vähän tutkimustietoa siitä, miten jatkuvassa tietotulvassa työskentely vaikuttaa ihmisen toimintakykyyn ja hyvinvointiin. Tutkielman luvussa 5 käsitellään käyttäjän näkökulmaa tietojärjestelmien parissa. Luvussa esitetään ihmisen tiedonkäsittelyn periaatteita ja pohdiskellaan kuinka inhimilliset ajattelu-, muisti- ja havaitsemiskyvyt voidaan ottaa huomioon tietojenkäsittelyssä. Selvää on, että vain helppokäyttöiset ja tietotulvaa helpottavat ratkaisut pärjäävät markkinoilla. Luvussa 6 esitetään esimerkkitapauksina muutamia sisällönhallintaprosesseja kokonaisuuksina. Esimerkkitapauksia käytettiin sisällönhallintaprosessin analysointiin ja transformaatioiden etsimiseen ja ne havainnollistavat hyvin sen, kuinka monia muunnoksia sisällönhallintaprosessiin liittyy.

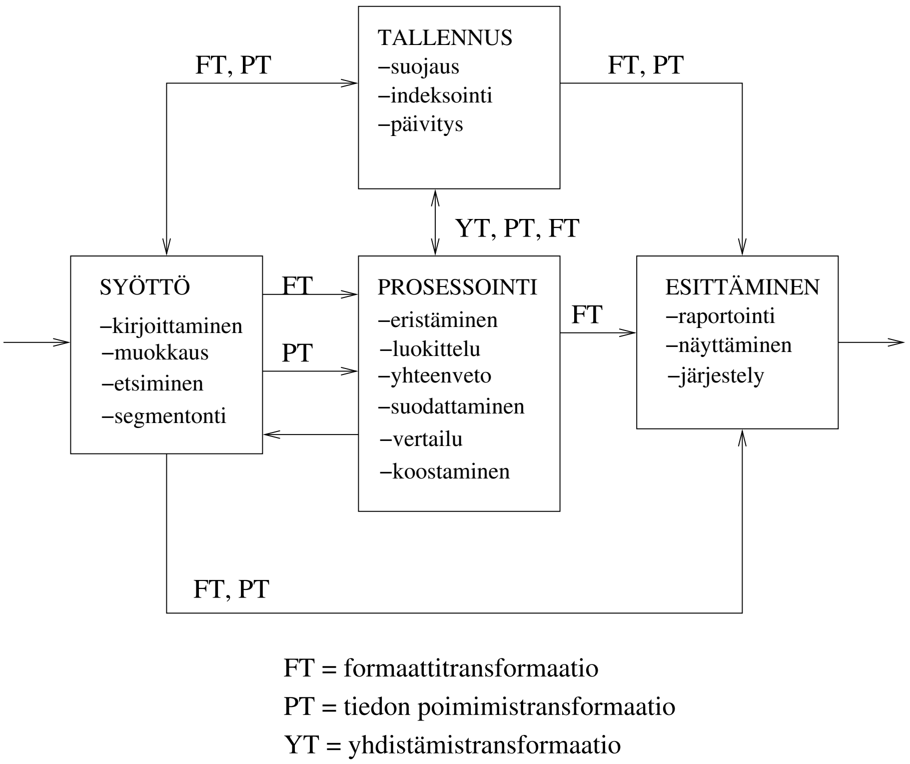
Heterogeenisten tietolähteiden yhdistämiseen on aiemmin pyritty esimerkiksi useissa WWW-tietoon keskittyvissä projekteissa, kuten projekteissa TSIMMIS [GMHI+95], Garlic, Ariadne [KMA+98, KM98] ja Informia [BBMS98].

Tutkielma on toteutettu Helsinki Institute for Information Technologyn (HIIT) tutkimusryhmässä osana Media Convergencen Core -projektia.

2 Sisällönhallintaprosessi

Sisällönhallintaprosessi tarkoittaa tiedon kulkua järjestelmässä: tiedon syöttäminen järjestelmään, prosessointi, tallentaminen ja esittäminen. Päämääränä on datan muuntaminen päätöksentekoa hyödyttäväksi tiedoksi [Hic87]. Data- ja tietokäsitteet eroavat toisistaan siten, että data on tietoa ennen kuin sitä on prosessoitu ja tieto on prosessoinnin käyttäjää hyödyttävä tulos. Tässä tutkielmassa käsitellään lähinnä tietoa, mutta jos on tarpeen viitata tietoon pelkkänä tietofaktana käytetään sanaa data. Tiedon synonyyminä käytetään sanaa informaatio. Käytännössä erot sanojen välillä riippuvat myös käyttäjästä, toisen käyttäjän data voi olla toisen tietoa ja prosessin syötedata voi olla jo jonkun toisen prosessin tulosteinformaatiota. Esimerkiksi työntekijän syöttämät tuntitiedot ovat tietoa hänen esimiehelleen. Yhtiön toimitusjohtajalle samat tiedot ovat puolestaan dataa, joka edelleen prosessoidaan ja summataan muiden työtunteihin.

Sisällönhallintaprosessi on yksinkertaistettuna esitetty kuvassa 1. Kuvassa koko sisällönhallintaprosessi on jaettu neljään pääosaan: tiedon syöttämiseen, prosessointiin, esittämiseen ja tallentamiseen. Kuvaan on merkitty tässä tutkielmassa esiteltäviä tiedon muunnostoimintoja eli erilaisia formaatti-, tiedon poimimis- tai yhdistämistransformaatioita (käytetty lyhenteitä FT, PT ja YT) prosessin eri vaiheiden välille. Formaattitransformaatiot tarkoittavat tietomuunnoksia, joissa tietosisältö pysyy samana. Tiedon poimimistransformaatioilla tarkoitetaan muunnoksia, jotka kohdistuvat yhteen tietolähteeseen ja yhdistämistransformaatiot muunnoksia useampien tietolähteiden välillä. Kuvaan on liitetty transformaatiotoimintoja, kuten eristäminen, luokittelu tai vertailu, siihen sisällönhallintaprosessin vaiheeseen, johon ne parhaiten sopivat.



Kuva 1: Sisällönhallintaprosessin vaiheet.

Transformaatioita eivät ole pelkästään tässä tutkielmassa käsiteltävät toiminnot, vaan on paljon muitakin tietoon kohdistuvia transformaatioita. Näitä kuvaamaan on vaiheiden välille merkitty lyhenteillä mahdollisia prosessista toiseen siirryttäessä ilmeneviä transformaatioita. Esimerkiksi syöttöprosessista on yhteys kaikkiin muihin osaprosesseihin. Syöttötiedosta siirtyminen suoraan esittämiseen voisi sisältää formaattitransformaatioita, kuten ohjelmointieditorissa eri datatyyppien esittäminen eri värein tai tiedon poimimistransformaatioita, esimerkiksi Wordissä välimerkkien (välilyönti, rivinvaihto, tabulointi) näyttäminen tai piilottaminen. Tiedonhallintaprosessi on hyvin iteratiivinen, yhden prosessin syöttötieto on useimmiten jonkun toisen prosessin prosessoinnin tulos, joten prosessi on hankala yksinkertaistaa lineaariseksi prosessiksi. Seuraavissa aliluvuissa käsitellään syöttö- ja prosessointitoimintoja sekä lyhyesti tiedon tallennusta ja varastointia. Tiedon esittämistä käsitellään luvussa 3.

2.1 Tiedon syöttäminen

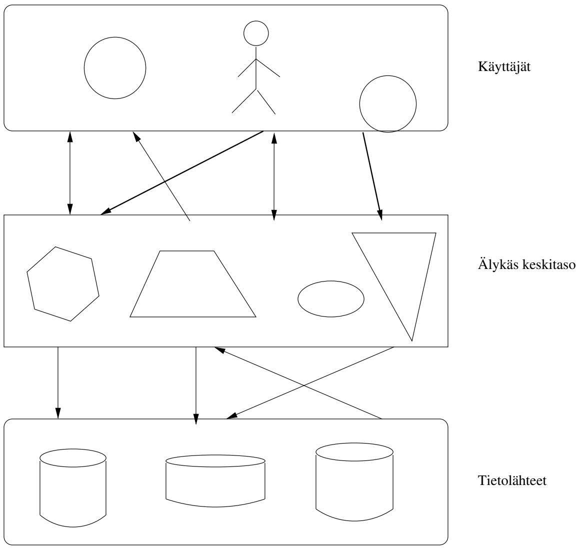
Tiedon syöttäminen järjestelmään voi tapahtua keräämällä olemassa olevaa tietoa, tuottamalla uutta tietoa tai näiden yhdistelmänä. Tietoa on jo olemassa valtavasti, joten tiedon uudelleenkäyttäminen olemassa olevista lähteistä on tärkeää ja järkevää. Tietoa voi kerätä useasta eri lähteestä, jolloin tieto on usein rakenteellisesti vaihtelevaa, sillä se voi olla peräisin esimerkiksi WWW-sovelluksista, tietokannoista, tiedostojärjestelmistä tai sähköpostista. Lähteet saatetaan yhdistää kokonaisuudessaan tai niistä voidaan poimia valitut osat yhdistettäväksi. Relevanttien asioiden poimiminen on monimutkainen tehtävä, tähän palataan myöhemmin tutkielmassa. Tiedon poimimista voi tehdä vasta kun tietolähde on löydetty (retrieved). Tiedon etsimistä on tutkittu paljon ja hakukoneet ovat WWW:ssä avainosassa. Suuresta tietomäärästä ei ole mitään apua, jos tietoa ei jälkikäteen löydetä ja voida käyttää. Tässä tutkielmassa ei paneuduta tiedon etsimiseen, vaan tietolähteet oletetaan löydetyiksi ja keskitytään käyttämään tietolähteitä lisäarvoa tuottavalla tavalla. Erityisesti usean tietolähteen seuraamiseen kiinnitetään huomiota ja pohditaan heterogeenisen tiedon yhdistämiseen liittyviä ongelmia.

Syötettävä tieto voi olla täysin uutta tietoa. Usein tiedon pohjana halutaan kuitenkin käyttää jotain esitietoa. Uuden tiedon tuottamiseen on paljon valmiita sovelluksia. Tekstinkäsittelyohjelmat, graafiset editorit ja sivustojen hallintaohjelmat (esimerkiksi Frontpage) esittävät sivuston sisällön graafisesti ja tukevat tiedon muokkaustoimintoja kuten uusien sivujen luontia, poistoa ja uudelleennimeämistä. Nämä ohjelmat tekevät automaattisesti monet formaattimuunnokset, kuten HTML-transformaatiot, mutta pohjatietojen keräämistä tai yhdistämistä ei tueta.

2.2 Prosessointivaihe

Prosessointivaihe sisältää monenlaista tiedon käsittelyä; kaikki tässä tutkielmassa käsiteltävät tiedon muuntamistoiminnot voidaan ajatella kuuluvaksi tähän vaiheeseen (jotkut myös muihin osavaiheisiin).

Heterogeenista tietoa integroivat järjestelmät perustuvat yleisesti kolmitasoarkkitehtuuriin, joka yksinkertaistettuna on esitetty kuvassa 2. Tätä voidaan kutsua mediaattoriarkkitehtuuriksi, jossa toiminnot, jotka eivät sovellu asiakkaalle eivätkä palvelimelle, on kerätty moduuleihin keskikerrokseen. Moduuleissa suoritetaan esimerkiksi useammalta palvelimelta kerätyn datan yhdistäminen ja muu prosessointi sekä transformointi muotoon, jossa asiakas sen haluaa [Wie99].



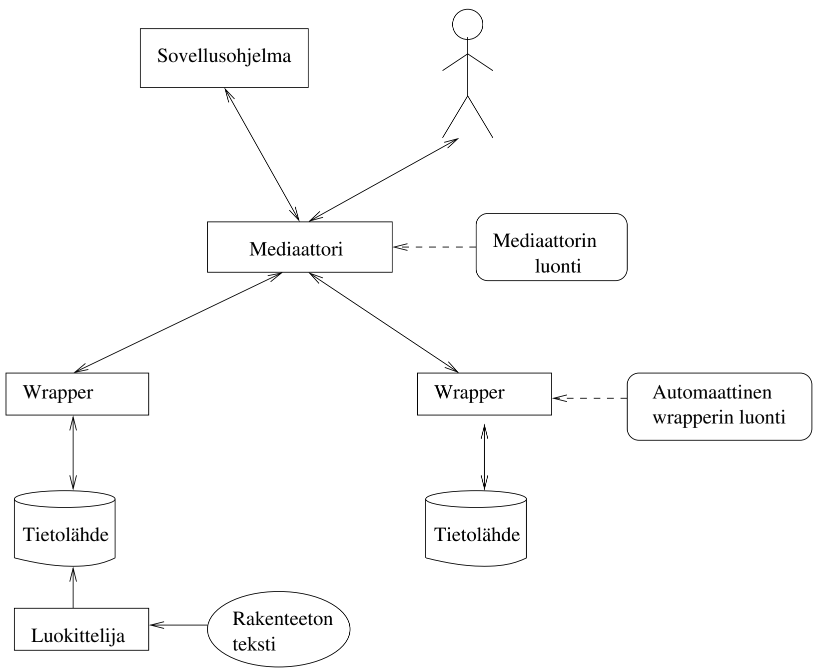
Kuva 2: Sisällönhallinnan kolmitasoarkkitehtuuri.

Mediaattori (mediator, joskus myös broker tai agent) on järjestelmän keskiosa, joka vastaanottaa pyyntöjä käyttäjältä tai sovellukselta, koordinoi pyynnön eri lähteisiin ja palauttaa tuloksen takaisin käyttäjälle [GMHI+95]. Mediaattori toimii älykkäänä tiedon yhdistelijänä tietolähteiden ja käyttäjän välissä. Sen täytyy tietää kuinka kommunikoida eri lähteiden kanssa ja ohjata kyselyt hyvin niihin vastaaviin tietolähteisiin. Toisaalta mediaattorin täytyy integroida tulokset esimerkiksi minimoimalla päällekkäisyydet, ratkaisemalla ristiriidat lähteiden välillä tai muodostamalla yhteenvetoja useista lähteistä. Mediaattorit käsittelevät tässä tutkielmassa pääosin tietojen yhdistämistransformaatioita.

Koska tietolähteet ovat heterogeenisia, tarvitaan lähteitä varten *sovittimet tai kääreet* (wrapper, translator, proxie, adapter, converter). Ne ovat tietolähdekohtaisia ja ne on räätälöity kääntämään yhden tietyn resurssin kanssa tapahtuva kommunikointi mediaattorin ymmärtämään muotoon ja päinvastoin. Sovitin siis muuntaa mediaattorin kyselyn muotoon, jota voidaan käyttää sen lähteessä, etsii ja poimii tiedon tietolähteestä ja palauttaa tiedon mediaattorille sopivassa muodossa [AK97]. Sovittimet ovat yleensä suhteellisen yksinkertaisia ja perustuvat lähinnä merkkisääntöihin.

Lisäksi järjestelmissä saattaa olla vielä *luokittelijoita* (classifier, extractor), jotka automaattisesti luokittelevat rakenteetonta tietoa ja poimivat metatiedot [GMHI$^+$95].

Mediaattori, sovitin ja luokittelija on esitetty kuvassa 3. Kuva on yksinkertaistettu versio TSIMMIS-järjestelmän arkkitehtuurista [GMHI$^+$95].



Kuva 3: Mediaattori, sovitin ja luokittelija TSIMMIS-järjestelmässä.

Dynaamisuus voi liittyä vain tietosisältöön, jolloin mediaattori kerää datan, mutta sen esitysmuoto on staattinen (tieto haetaan valmiisiin mallipohjiin), tai dynaamisuus voi ulottua myös esitykseen ja navigointiin [Fra99]. Älykkään järjestelmän oletetaan luonnollisesti mukauttavan esitysmuodon käyttäjän tarpeiden mukaan.

2.3 Tallennus ja varastointi

Varastointiin liittyy tiedon suojaaminen, indeksointi ja päivittäminen [Hic87]. Suojaamalla estetään tiedon luvaton poistaminen, muuttaminen ja käyttö. Tiedostoja voi suojata määrittelemällä käyttäjille tai käyttäjäryhmille oikeuksia, esimerkiksi vain lukuoikeudet. Yrityksillä voi olla palomuurissaan turvallisuusmediaattori, joka suodattaa kyselyn tulokset ennen kuin ne päästetään ulkopuoliseen maailmaan [Wie99]. Indeksoinnilla luodaan osoitteisiin hakemisto, joka osoittaa tiedon fyysisen sijainnin. Päivitystapahtumia ovat olemassa olevan tiedon lisääminen, poistaminen ja muuttaminen. Dokumenttien arkistoinnissa pitkäaikaista säilytystä varten on otettava huomioon dynaamisten dokumenttien muuttuminen, ja jos sisältö, rakenne ja esitystapa on erotettu toisistaan, täytyy miettiä miten arkistoida hajautettu tieto.

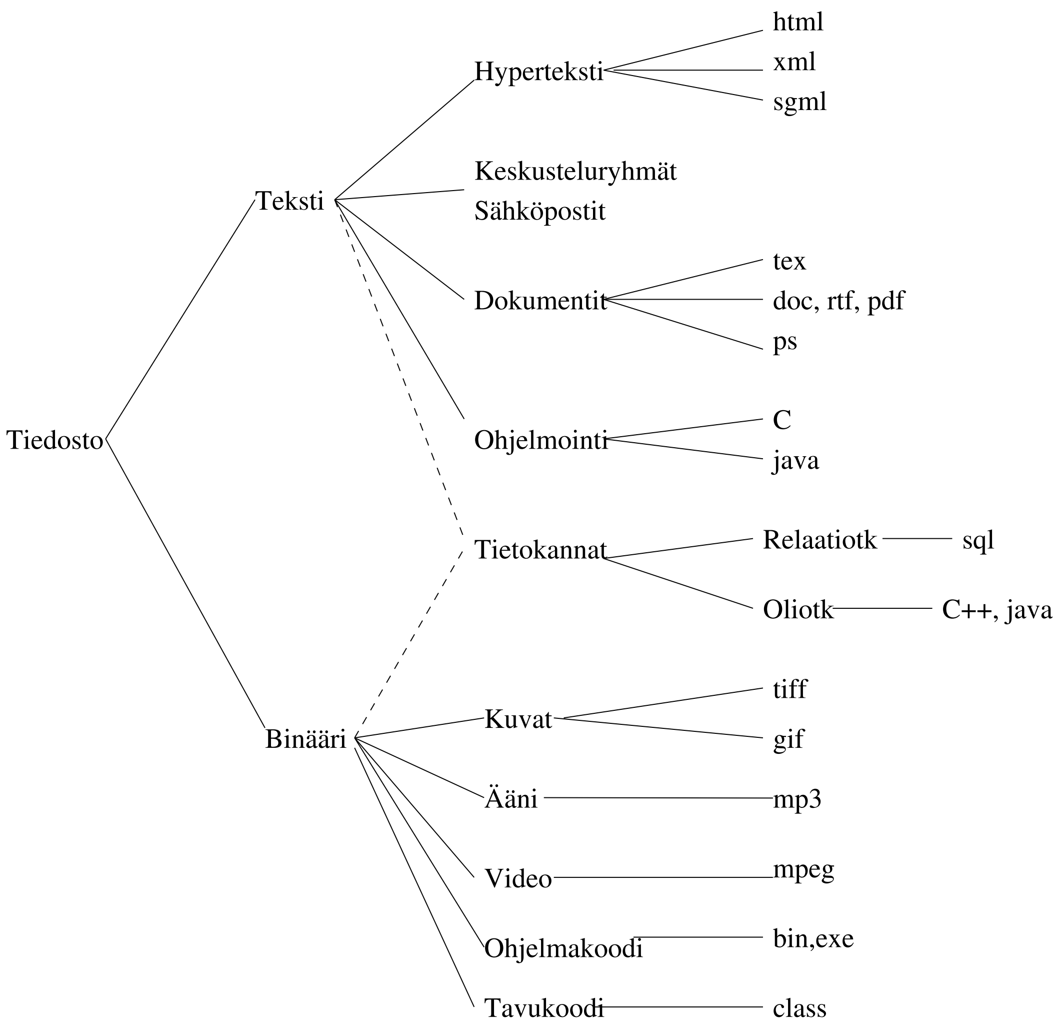
3 Tiedon esitystavat

Tietoa voidaan hakea esimerkiksi tietokannoista, tietämyskannoista (knowledge base), tiedostojärjestelmistä, digitaalisista kirjastoista, hakupalvelimista ja sähköpostijärjestelmistä. Tietojärjestelmien laaja Internet-verkkoon liittäminen on avannut lukuisat tietopalvelut WWW-selaimen avulla tavoitettaviksi. Tässä luvussa tutustutaan lyhyesti joihinkin tiedon esitysmuotoihin. Valtava esitysmuotojen kirjo kuvaa laajaa tietomäärää, joka ympärillämme on tarjolla sekä sen mukana antamiaan vaikeuksia, jota tällaisen heterogeenisen tiedon hallinnassa ja yhdistelyssä kohdataan. Hieman tarkemmin kuvataan rakenteista tietoa, SGML-pohjaisia formaatteja, koska ne näyttävät yleistyvän nopeasti ja koska rakenteinen tieto helpottaa tiedonkäsittelyyn liittyviä muita muunnoksia.

3.1 Tiedostomuotoja

Kuvassa 4 on esitetty hierarkkisesti eri muotoisten tietojen suhdetta toisiinsa. Kuvassa on ainoastaan muutamia esimerkkejä, sillä eri tiedostoformaatteja kaikenkaikkiaan on satoja tai luultavasti tuhansia.

Digitaalisen tiedon tietoyksiköksi voidaan ajatella tiedostoa. Tiedostot voidaan jakaa teksti- ja binäärimuotoisiin tiedostoihin, jotka on kuvassa yritetty luokitella vielä aliluok-



Kuva 4: Tietoelementtejä hierarkkisesti luokiteltuina. Kuvassa on ainoastaan esimerkkitapauksia.

kiin. Tekstitiedostojen yleisin muoto on ASCII tai tästä tehty laajennos ISO Latin-1. UNIX ja DOS-perustaiset käyttöjärjestelmät lukuunottamatta Windows NT-käyttöjärjestelmää käyttävät ASCII-muotoa tekstitiedostoille. Windows NT käyttää uudempaa Unicode-merkistöä. Binääritiedostot sisältävät tietokoneen tai ohjelman ymmärtämässä muodossa olevaa dataa, kuten kuvaa, ääntä tai ohjelmakoodia. Jos tiedosto tulostetaan kuvaruudulle tai kirjoittimelle, se näyttää pelkältä sotkulta, sillä tavuja ei ole tarkoitettu tulkittavaksi ASCII-merkkeinä.

Perinteinen, suoraviivainen tiedon esitystapa liittää dokumentin fyysisen ja loogisen rakenteen läheisesti toisiinsa. Esimerkiksi kirjan looginen rakenne edellyttää, että se luetaan alusta loppuun järjestyksessä. Tavallisella lineaarisella tekstillä on vain edellinen ja seuraava tekstielementti. Hypertekstillä puolestaan voi olla useita liitoksia ja tietoa voi lukea valikoiden. Hyperteksti koostuu itsenäisistä informaatioyksiköistä, solmuista, jotka voidaan linkittää asiayhteydeltään useisiin muihin katkelmiin. Hyperteksti antaa mahdollisuuden järjestää, hakea, ja esittää tietoa assosiatiivisesti, kuten ihmisajattelun voisi ajatella toimivan. Kuvassa 4 puhutaan vain hypertekstistä, mutta solmut voivat sisältää ääntä, musiikkia, animaatiota ja videota.

Keskusteluryhmät ja sähköposti on esitetty kuvassa omana haaranaan, koska ne ovat nykyään yleisimpiä viestintätapoja ja helpottavat globaalia viestintää. Ne ovat sähköisten viestien kirjoitus-, lähetys-, vastaanotto- ja arkistointijärjestelmiä ja ne voidaan osoittaa tietylle henkilölle tai ryhmälle. Usenet News [IET83] on maailmanlaajuinen keskustelu- ja uutisointijärjestelmä, jossa on kymmenisentuhatta keskusteluryhmää. Internet-sähköpostin [IET82] perustana on sopimus, protokolla, siitä, millaista tietoa (dataa) sähköpostilla voidaan siirtää ja miten sähköpostiviestejä siirretään palvelimelta toiselle. Esimerkiksi MIME-standardi [RFC93] sisältää 8-bittisten merkistöjen käytön, eli MIME-standardisessa sähköpostissa voidaan käyttää skandinaavisia merkkejä å, ä ja ö sekä liittää sähköpostiin liitetiedostoja.

Dokumenteilla tarkoitetaan tavallisia tekstidokumentteja eli asiakokonaisuutta, jonka käyttäjä jäsentää kokonaisuudeksi. Ohjelmointi kielellä kirjoitetut tiedostot on kuvassa erotettu omaksi haarakseen, sillä ne ovat ohjelmiston näkökulmasta katsottuja kokonaisuuksia. Tietokannat on yhdistetty kuvassa sekä teksti- että binäärimuotoihin, koska niiden toteutus on binäärinen, vaikka tieto itsessään on rakenteista tekstiä.

Digitaalinen kuva voidaan tallettaa tiedostoon kahdessa erilaisessa formaatissa: bittikarttana (rasterigrafiikka) tai vektorigrafiikkana. Erilaisia kuvamuotoja ovat muun muassa Graphic Interchange Format (GIF)[1], Joint Photographic Expert Group (JPEG) [ISO99], Tag Image File Format (TIFF)[2] [Ado95] ja Portable Network Graphics (PNG) [W3C96]. Ne kaikki perustuvat rasterigrafiikkaan ja ovat hieman eri tarkoituksiin suunnattuja. TIFF-muotoisen kuvan tulostusjälki on hyvää, ja GIF sopii hyvin WWW-sivuille, joissa ei käytetä laajaa väriskaalaa. JPEG-muoto puolestaan sopii nimenomaan jatkuvien sävyjen esittämiseen WWW-sivuilla, mutta se hävittää tietoa pakattuna. PNG on Internet-komitean kehittämä esitysmuoto korvaamaan GIF-muotoa, johon liittyy lisenssointiongelmia koska GIF-kuvissa käytetty pakkausmenetelmä on Unisysin omistukses-

[1] The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. The GIF standard uses LZW for compression which is patented by Unisys. http://www.unisys.com/unisys/lzw/

[2] TIFF on yksityinen standardi, jonka kehitti Aldus Corporation (nyt osa Adobe Softwarea)

sa.

Äänitiedostoista mp3 eli MPEG-1 (the Motion Picture Experts Group) Audio Layer 3 on tärkein standardoitu audiomuoto [ISO93]. Wave Audio eli .Wav on Windows ja DOS -ympäristöissä yleinen äänitiedostoformaatti. MPEG sisältää standardin myös videon esitysmuodolle.

Ohjelmakoodilla tarkoitetaan prosessorin ymmärtämää konekielikoodia, kuten .exe-tiedostot. Javan lähdekoodi käännetään laitteistoriippumattomaksi tavukoodiksi, joka on korkeantason konekoodia Javan virtuaalikoneelle. Mikä tahansa Java-sovellus toimii tällöin kaikissa laite- ja käyttöjärjestelmäympäristöissä, kunhan vaan koneeseen on asennettu Javan virtuaalikone.

Jokaiseen dokumenttiin liittyy sisältö, rakenne ja esitystapa, jotka voidaan teknisesti erottaa toisistaan. Esimerkiksi kuvan sisältö voidaan esittää bittikarttana tai vektorigrafiikkana, dokumentin rakenne HTML:n elementtien alku- ja loppumerkkien mukaisesti ja ulkoasu puolestaan CSS-formatointiominaisuuksilla. Esitysmuodon ja tietosisällön erottaminen toisistaan tarjoaa lisäarvoa erityisesti useamman median ja vaihtelevien tarpeiden ollessa kyseessä ja tähän onkin pitkälti siirrytty. Eri medioiden sisällölliset ja rakenteelliset rajoitukset ja erikoistarpeet voidaan määritellä tyylimäärittelyillä (stylesheets) sekä mediakohtaisilla transformaatioilla. Tyylimäärittelyitä voi olla useita, joten käyttäjä voi itse määrittää esitysmuodon mieltymystensä mukaiseksi ja hänen oma määrittelynsä korvaa palvelun tarjoajan tyylimäärittelyt. Näin samaa tietosisältöä ei tarvitse tallentaa useampaan kertaan ja hallita useampina versioina. Esimerkiksi teksti voidaan kääntää pistekirjoitukseksi tai 10-pisteen fontti voidaan korvata 18-pisteen fonttikoolla [Kno98]. Sisällön ja esitysmuodon erottaminen tukee myös yhteen mediaan kohdistettua useampaa esitysmuotoa. Esimerkiksi tuloste voidaan haluta useaan eri paperikokoon ja eri kokoiset näytöt voidaan huomioida tai sama tieto voidaan esittää visuaalisesti tai äänenä.

SGML (Standard Generalized Markup Language) [ISO86] on standardoitu dokumenttien metamerkkauskieli. Metamerkkauskieli tarkoittaa sitä, että SGML määrittelee vain kieliopin, jonka avulla voi määritellä omia kieliä ja ilmaista mitä tahansa sisältöä. Käyttäjä voi itse määrittää minkä nimisiä elementtejä käyttää. SGML perustuu siihen, että dokumenteilla on rakenteellisia ja semanttisia elementtejä, joita voidaan kuvata määrittelemättä kuinka ne tulisi esittää. SGML-dokumentin rakenne esitetään DTD-määrittelynä (document type definition, dokumenttityypin määrittely). DTD:ssä määritellään millaisia elementtejä voi olla olemassa ja niiden suhteet eli dokumentin hierarkkinen rakenne. DTD kuvailee dokumentin datan rakenteen, mutta DTD ei määrittele millään lailla sitä, miten datan ulkoasu tulisi esittää vaikkapa tulostettaessa tai sitä, miten informaatiosysteemien tulisi käsitellä dataa. Ennalta määritelty DTD toimii, ja sen tehtävä on toimia, ohjeena dokumentin luojille; toisin sanoen näyttää mitä rakenne-elementtejä käytetään ja missä järjestyksessä. SGML-dokumentti koostuu komesta osasta: SGML-esittelyosasta (declaration), dokumentin rakennemäärittelystä (DTD) ja dokumentti-ilmentymästä eli merkatusta sisällöstä. Merkattu sisältö on dokumentin rakenteistettu tekstiosa, joka on käyttää DTD:ssä määriteltyjä elementtien nimiä ja sisältää dokumentin tietosisällön. Ilmentymä on laillinen (valid), jos se toteuttaa luokkansa rakennekuvauksen. Lisäksi kaikki kolme osaa, SGML-esittelyosa, DTD ja dokumentti-ilmentymä tarvitaan lailliseen SGML-dokumenttiin.

Elementtien esitysmuoto saattaa vaihdella median ja tyylimieltymysten mukaisesti. SGML kuvaa dokumentin loogisen rakenteen, dokumentin visuaalinen esitysmuoto voidaan määritellä DSSSL-tyylimäärittelyin [ISO96]. DSSSL-kieli on ISO-standardi ja kehitetty SGML-dokumenttien esitysmuodon määrittelyä ja muuntamista varten. Se sisältää ominaisuuksia sekä näytöllä esitettävien dokumenttien (mm. vierityspalkit, linkit ja reunahuomautukset) että muulla tavoin tulostettavien dokumenttien esitysmuodon määrittelyyn.

Selainten käyttämä kieli, Hypertext Markup Language (HTML) [W3C99c] on esimerkki SGML-standardin mukaisesta merkkauskielestä. HTML-elementit on määritelty yhdessä DTD:ssä eli HTML:n elementit on kiinnitetty. Kiinteä määrittely tekee HTML:stä joustamattoman ja yksinkertaisen. HTML:n uusimman version XHTML 1.0:n [W3C00b], dokumenttityypit ovat XML-perustaisia.

XML (Extended Markup Language) [W3C00d] on noussut monimutkaisen ja raskaan SGML:n kilpailijaksi samantyyppisenä, mutta kevyempänä kuvauskielenä. XML on lähinnä verkkokäyttöön suunnattu joustava yksinkertaistus SGML-määrityksestä. XML-perustaiset standardit ovat muun muassa auttaneet esitysmuodon kontrollin siirtymisessä käyttäjän puolelle [Kno98]. XML-määrittelyssä puhutaan erikseen hyvinmuodostetuista (well-formed) ja laillisista (valid) dokumenteista. Hyvinmuodostettu dokumentti on XML-määrittelyn ja kieliopin mukainen. Lisäksi siinä oleville attribuuteille ja entiteeteille on asetettu yksityiskohtaisia ehtoja. Dokumentti voi olla hyvinmuodostettu, vaikka siihen ei liittyisi lainkaan DTD:tä. Tämä on merkittävä kevennys SGML-standardiin verrattuna. Laillinen XML-dokumentti sisältää XML:n mukaisen dokumenttityyppien esittelyn (XML document type declaration). Tämä esittely puolestaan joko sisältää tai viittaa dokumenttityypin määrittelyyn (XML-dokumenttityyppien määrittely tarkoittaa samaa asiaa kuin SGML-dokumenttien DTD). XML-dokumentteihin voi siis liittyä myös DTD ja monissa sovelluksissa se on välttämätöntä. Ilman DTD:tä dokumentin rakenteen yksityiskohtainen hallinta ei ole mahdollista. XML Schema [W3C01b] on W3C:n uusi, kohdetiedoston rakenteen kuvaamiseen tarkoitettu määrittely. Schemat ovat XML-kielellä kirjoitettuja, toisin sanoen XML:n avulla kuvataan XML-asiakirjan rakennetta.

XML-määritys ei sisällä muotoiluun liittyviä asioita eikä sen merkeillä ole lukkoon lyötyjä merkityksiä. Kuitenkin tieto on esitettävä ihmiselle jollakin tavalla, usein vielä erilaisena eri medialla (esimerkiksi näyttö, paperi). XML-dokumenttien näyttämiseen tarvitaan jokin menetelmä, esimerkiksi voidaan käyttää XSL-kieltä (eXtensible Style Language) [W3C00a], joka on yleisin XML:n tyylitiedostojen määrittelyyn tarkoitettu kieli. Tyylien määrittelyyn voitaisiin käyttää myös HTML:n tyylien määrittelyyn kehitettyä CSS:ää (esitetään myöhemmin), mutta XSL on kehittyneempi. XSL koostuu kahdesta erillisestä osasta; muunnoskielestä (XSLT, XSL Transformations) ja muotoilukielestä (formatting objects, XSL-FO tai pelkkä XSL). XSL Transformaatioita käsitellään tutkielman kappaleessa 4.1 Formaattitransformaatiot. XSL:n avulla voidaan määritellä fontit, marginaalit, listat, otsikoiden muodot, sivunumeroinnit, jne. XSL-kielellä voidaan tehdä myös monimutkaisempia ulkoasuun liittyviä asioita kuten alaviitteitä, sarakkeita tai taulukoita. XSL noudattaa XML -syntaksia sekä käyttää epäsuorasti hyödykseen CSS:n ominaisuuksia. XSL-tyylitiedoston (joka on dokumentti- tai dokumenttiluokkakohtainen) määrityksillä kuvataan säännöt, joilla dokumentti muunnetaan esimerkiksi XML-dokumentiksi. Eri medioiden sisällölliset ja rakenteelliset rajoitukset ja erikoistarpeet voidaan määritellä tyylimäärittelyillä (stylesheets) sekä mediakohtaisilla transformaatioilla, joihon voidaan käyttää XSL/XSLT:tä.

HTML-merkattujen dokumenttien tyylimäärittelyihin on käytettävissä kaksitasoinen ratkaisu: CSS1 ja CSS2 [W3C99a, W3C98]. CSS1 (Cascading Style Sheets, level 1) on hyvin suppea ja se on tarkoitettu selainten yhteydessä käytettäväksi. CSS2 (level 2) on paljon laajempi ja sen avulla pystyy yksityiskohtaisesti ohjaamaan dokumentin ladontaa ja erilaisia muunnoksia. CSS-tyylimäärittelyjä voi käyttää myös XML-dokumenttien muotoilussa. CSS3-versiota kehitellään parhaillaan [W3C01c].

Sisältö-, rakenne- ja esitystiedon lisäksi auttaa tiedonhallinnassa vielä erillinen tieto tiedosta eli *metadata*, joka tarkoittaa kuvausta tai määritelmää tiedosta [dCMCB99]. Se voi olla tietoa dokumentin sisällöstä, muodosta, sijainnista ja ylläpidosta. Metadatan tarkoitus on helpottaa tiedonhakua, -hallintaa ja paikantamista. Dokumentin metatietoa ovat esimerkiksi indeksit, viitteet dokumentin ulkopuolelta tai Word-dokumentin attribuutit. XML-elementin metatiedot puolestaan määritellään DTD:ssä elementin attribuutteina. Tietokannan kentän metadataa on tietokantakaavio. Erityisesti WWW:n metadata määritellään koneen ymmärtämäksi tiedoksi WWW-lähteestä ja sillä on määritelty merkitys ja rakenne. Metadatan tarkoitus on lisätä automaattisuutta WWW:ssä, esimerkiksi luetteloimalla WWW-lähteen sivujen sisältö ja sisältösuhteet helpottamalla siten tietojen jakamista ja vaihtamista. Metadatan prosessointia ja esittämistä varten on kehitetty RDF (Resource Description Framework) [W3C00c, W3C99b], joka tarjoaa mahdollisuuden sovellusten välisen tiedon vaihtamiseen.

3.2 Käsitteitä

Tiedon käsittelyyn vaikuttavia tekijöitä ovat tiedon rakenteisuus, granulariteetti ja heterogeenisuus. Nämä on otettava huomioon transformaatioita tehtäessä ja jos käsitellään useaa tietolähdettä, on tekijät yleensä yhdenmukaistettava. Tarkastellaan seuraavaksi kutakin käsitettä hieman tarkemmin.

Tiedon rakenteisuus tarkoittaa tiedossa tunnistettavia komponentteja ja se voi vaihdella rakenteettomasta täysin määriteltyyn rakenteeseen [HGMC+97]. Rakenteettomalla tiedolla tarkoitetaan tässä tekstiä, kuvia tai ääntä, jolla ei ole esitettyä rakennetta. Esimerkiksi tekstikappaleiden erottaminen toisistaan rivinvaihdoin luo jo hieman rakennetta. Sellaisenaan kyseessä on kuitenkin vasta dataa, ei älykästä, tulkittua tietoa. Rakenteetonta tietoa voi käsitellä ainoastaan kokonaisena elementtinä: jos tietoa halutaan automaattisesti analysoida ja manipuloida, se täytyy rakenteistaa. Rakenteista tietoa ovat perinteiset relaatiotietokannat ja oliotietokannat, joiden rakenne on täysin tunnettu.

Rakenteinen tieto = tieto koostuu loogisista komponenteista, jotka voidaan tunnistaa Tekninen dokumentointi AS-75.112 Dokumentti orientoinut luokitus: ?luku, kappale, ?kuva, kuvateksti, video, audio, taulukko, lista jne. Data orientoinut luokitus: ?luokka, ?opiskelija, osasto,

Suurin osa Internet-tiedosta sijoittuu rakenteettoman ja rakenteellisen tiedon välimaastoon, kuten HTML-sivut ja SGML-dokumentit. Tieto on jollain tavalla muotoiltua, sisältäen esimerkiksi otsikot eli jollakin tavalla nimetyn rakenteen. Tällaista tietoa kutsutaan puolirakenteiseksi (semistuctured). Puolirakenteisella tiedolla tarkoitetaan usein esimerkiksi XML-dataa, johon liittyy DTD, mutta jonka DTD:ssä ei kuitenkaan voida määritellä elementtien sisältöä. Puolirakenteinen tieto voidaan yleensä esittää jonkinlaisena puurakenteena, mutta siitä ei välttämättä ole mallinnettu mitään erillistä kaaviota (kuten tietokantakaavaa). Rakenne on pääteltävissä tiedon esityksestä, se ilmenee esimerkiksi sivun koodauksessa käytetystä merkkauksesta eli tageista. Puolirakenteiseen tietosisältöön ei ole asetettu vaatimuksia tai on asetettu ainoastaan löyhät vaatismukset [Bun97].

Granulariteetti eli tietoelementin raekoko määrää osittain rakenteisuutta. Suuri granulariteetti eli pienet rakeet saavat aikaan rikkaan rakenteen ja pieni granulariteetti eli suuret rakeet aiheuttavat yksinkertaisen rakenteen. Käsiteltävä tieto voi olla suuruudeltaan yhdestä attribuutista dokumenttikokoelmaan. Rakenteisessa dokumentissa raekokona on yksi elementti, tietokannoissa yksi kenttä. Granulariteetti tarkoittaa myös tiedon abstraktiotasoa: jos useampia tietoja halutaan yhdistää älykkäästi, on niiden granulariteetti yhdenmukaistettava.

Heterogeenisuudella tarkoitetaan tässä tutkielmassa pääosin tiedon rakenteen tai esitysmuodon monimuotoisuutta. Tosin myös käyttäjien, ohjelmien ja laitteistojenkin heterogeenisuus vaikuttavat käsiteltäviin asioihin.

Tiedon rakenteisuuteen ja granulariteettiin on mahdollista vaikuttaa transformaatioilla ja seuraavassa kappaleessa näistä löytyy myös esimerkkejä. Koko tutkielma on suunnattu nimenomaan heterogeenisuudesta johtuviin ongelmiin ja trasformaatiot nähdään ratkaisuna heterogeenisten tietolähteiden käsittelyssä.

4 Tiedon käsittely

Tässä luvussa käsitellään tietoon kohdistettavia muunnostoimintoja sisällönhallintaprosessissa. Prosessia on pyritty tutkimaan mahdollisimman kokonaisvaltaisesti muun muassa myöhemmin esiteltävien esimerkkitapausten avulla. Muunnokset on luokiteltu kolmeen osaan: formaattitransformaatioihin, tiedon poimistransformaatioihin ja tiedon yhdistämistransformaatioihin, joita kutakin käsitellään omissa aliluvuissaan.

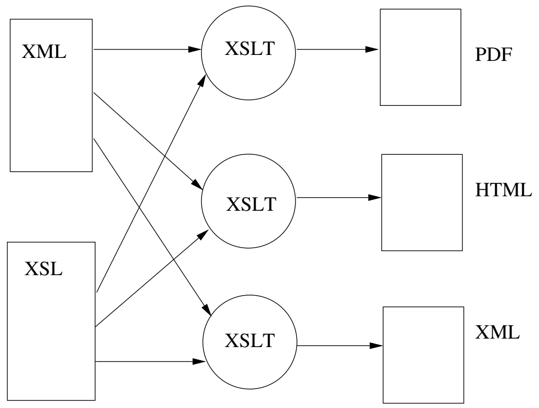
4.1 Formaattitransformaatiot

Tiedonkulkuun ja tiedonkäsittelyyn erityisesti heterogeenisiä tietolähteitä käsiteltäessä liittyy paljon transformaatioita. Formaattitransformaatiolla tarkoitetaan tässä tutkielmassa muunnoksia, joissa tietosisältö säilyy ennallaan, mutta esitysmuoto muuttuu. Periaatteessa formaattitransformaatioita voisi olla olemassa kombinatorinen määrä tiedon esitysmuodoista. Formaattitransformaatiot tarkoittavat siis pelkkää teknistä muunnosta, jotta sama tieto voidaan esittää eri esitysmuodossa. Jos tietosisältö muuttuu, jos se suppenee tai laajenee, ei kyseessä enää ole pelkkä formaattitransformaatio vaan myöhemmissä luvuissa käsiteltävä poimimis- tai yhdistämistransformaatio.

Formaattimuunnokset ovat lähinnä teknisiä, melko yksinkertaisia ongelmia, eikä niitä pidetä niin mielenkiintoisina tässä tutkielmassa. Kategoriaan kuuluu tosin erittäin vaikeitakin muunnoksia, esimerkiksi muunnokset tekstistä puheeseen. Yleistäen ja ehkä hieman uhkarohkeasti voisi ajatella, että muunnokset kuvassa 4 ovat sitä yksinkertaisempia, mitä lähempänä haarat ovat toisiaan. Samassa alihaarassa oleville tiedostoformaateille on paljon muunnoksia, esimerkiksi kuvankäsittelyohjelmilla voi tehdä muunnoksia kuvaformaatista toiseen. Valmiita muunnoskieliä tekstitiedostojen välillä on paljon, esimerkiksi html-muotoon kääntäviä työkaluja ovat LaTeX2HTML ja RTFtoHTML. Perl on tarkoitettu eritysesti tekstitiedostojen käsittelyyn ja analyysiin, joten se soveltuu hyvin formaattitransformaatioiden määrittelyyn.

Koska XML näyttää tällä hetkellä hyvin varteenotettavalta ratkaisulta heterogeenisten tietolähteiden sisällönhallintaprosessissa, esitellään lyhyesti XML-kielen käyttöä tukevia määrittelyjä ja muunnoksia.

XSLT (XSL Transformations) on XML-syntaksia noudattava muunnoskieli ja osa XSL:ää (eXtensible Style Language) [W3C00a]. XSLT-kielen avulla XML-dokumenttien rakenteita voidaan muokata tai dokumentit voidaan muuntaa kokonaan toista rakennemäärittelyä noudattaviksi, esimerkiksi HTML:ksi. XSLT:tä voidaan käyttää myös muuttamaan XML-dokumentit täysin XML:stä eroavaa kieltä noudattavaksi. Esimerkiksi XML-dokumentista voidaan poistaa merkkaus, jolloin lopputuloksena saadaan puhdas tekstidokumentti, tai dokumentti voidaan tulostusta varten muuntaa noudattamaan LaTeXin muotoilukoodeja. Kuvassa 5 on esitetty XML-dokumenttien, XSL-transformaatioiden ja tulosdokumenttien suhteita.



Kuva 5: Esimerkki XML:n ja XSLT:n käytöstä monikanavajulkaisemisessa

Rakenteistamiseen voidaan käyttää valmiita ohjelmia kuten konversio-ohjelmia Balise ja Omnimark. Omnimark on alunperin SGML-dokumenttien käsittelyyn (ja erityisesti muunnoksiin) tarkoitettu ohjelmointikieli, jota voi käyttää myös XML -dokumenttien käsittelyyn. Ohjelma kääntää esimerkiksi täysin rakenteetonta tekstiä SGML:ksi tai XML:ksi. Omnimark-kieli on samannimisen yrityksen kehittämä ja omistama[3].

Jotta XML:ää voidaan käsitellä ohjelmallisesti, täytyy XML-dokumentit jäsentää. Jäsennin tulkitsee XML-dokumentin rakenteen ja antaa rajapinnan sen käsittelyyn. DOM (Document Object Model) [W3C01a] ja SAX (Simple Application Interface for XML) [Meg00] ovat ohjelmointirajapintoja XML:lle. Ne ovat alusta- ja kieliriippumattomia. Rajapintojen avulla ohjelmat voivat lukea ja muokata dokumenttien sisältöä ja rakennetta sekä navigoida dokumentissa. DOM-määrittely esittelee puumaisen rajapinnan XML- ja HTML-dokumenttien käsittelyyn. Jotta se voi rakentaa puun ja tarjota siihen rajapinnan, täytyy koko XML-dokumentti lukea ja säilyttää muistissa. Tämän vuoksi DOM-rakenteet ovat usein raskaita ja saattavat kuluttaa paljon muistia. SAX:n lähestymistapa eroaa DOM:sta, mutta sillä voidaan tehdä lähes samat asiat XML-dokumentteihin. SAX ei luo minkäänlaista tietorakennetta XML-dokumentista, vaan se toimii tapahtumapohjaisesti. SAX käsittelee ainoastaan tietyllä hetkellä luettua tietoa ja on riippumaton dokumentin muista osista.

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

4.2 Tietoelementtien poimiminen tietolähteestä

Tiedonpoimimistransformaatiot ovat muunnoksia, jotka kohdistuvat yhteen tietolähteeseen. Niiden avulla yhdestä tietolähteestä poimitaan relevantti, käyttäjää eniten hyödyttävä tieto. Eri tietoelementtien poimimistoiminnoiksi on löydetty tiedon eristäminen, automaattinen yhteenveto, segmentointi, suodattaminen ja luokittelu, joista kutakin tarkastellaan seuraavissa aliluvuissa. Toiminnot ovat osittain päällekkäisiä, ja vaikeasti toisistaan irrotettavia, mutta ne on yritetty kategorisoida päätoimintojensa mukaisesti.

4.2.1 Eristäminen

Tiedon eristämisellä tarkoitetaan spesifien faktojen saamista dokumentista: informaatiosta poimitaan tiedonhakijan kannalta tärkeät tiedot. Valtava määrä tiedosta on vain vapaana luonnollisen kielen tekstinä ja jos sitä halutaan automaattisesti analysoida ja käyttää, se täytyy ensin kääntää rakenteisempaan muotoon. Tiedon eristäminen pureutuu tähän ongelmaan: relevantti tieto tunnistetaan luonnollisen kielen tekstistä ja poimitaan halutut faktat rakenteistetussa muodossa. Kun perinteinen tiedonhaku palauttaa listan hakukriteerit täyttävistä dokumenteista, tiedon eristäminen pyrkii palauttamaan haetut faktat valmiiksi poimittuina näistä dokumenteista. Koska tiedon eristäminen käsittelee luonnollista kieltä, prosessissa tarvitaan aina jonkin asteista kielen analyysiä. Analyysin syvyys vaihtelee järjestelmittäin, sillä analyysin kustannukset on punnittava tarkkaan hyötyyn nähden. Kielianalyysi voitaisiin määritellä myös formaattitransformaatioksi, mutta se on niin tärkeä osa eristämistä, että kielianalyysiin tutustutaan tässä luvussa. Täyteen tekstin ymmärtämiseen tähtäävä analyysi on erittäin monimutkainen ja työläs ja usein suurelta osalta turha. Niinpä tekstin syntaktinen analyysi on yleensä vain osittaista. Esimerkkejä eristämistarpeista ovat tietofaktojen poimimiset tiedelehdistä, lakipäätöksistä tai sairasraporteista, tai vaikkapa sanomalehdestä voitaisiin haluta löytää kaikki murha-tapaukset sekä näistä tiedot, kuka tapettiin, kuka tappoi, milloin ja missä.

Tiedoneristämisjärjestelmät perustuvat eristämishahmoihin eli eristämissääntöihin, jotka kertovat eristtävälle järjestelmälle, mikä tieto dokumentista halutaan erottaa [Mus99]. Pääsuuntauksia on kaksi: automaattinen oppiminen (tai koneoppiminen) ja tietämysjärjestelmään (knowledge engineering) perustuva. Automaattisella oppimisella toteutettu eristämisjärjestelmä tarvitsee opetusdataa, josta kerätään sanojen tilastollista tietoa. Aihepiirin tunteva ihminen merkitsee opetustekstiin eristettävät asiat ja eristämissäännöt johdetaan tämän perusteella automaattisesti. Tietämysjärjestelmään perustuva eristäminen tarkoittaa sääntöjen kirjoittamista manuaalisesti. Asiantuntija käy läpi aineistoa ja kirjoittaa sääntöjä, ”kielioppia”, kuinka haluttu tieto eristetään. Tämä vaatii usein paljon työtä ja iteraatiota sekä laajan testikorpuksen. Yleensä pyritään sääntöjen automaattiseen, tai puoliautomaattiseen, luomiseen esimerkkien avulla [GY00]. Toisaalta ihmisasiantuntijuuttakaan ei saa aliarvioida ja tietämysjärjestelmien suorituskyky onkin usein parempi, varsinkin jos opetusaineistoa on vähän [AI99]. Kuvassa 6 on esimerkki mallipohjaperustaisesta eristämisestä. Kuvan yläosassa on uutinen, josta on eristetty tietofaktat mallipohjaan.

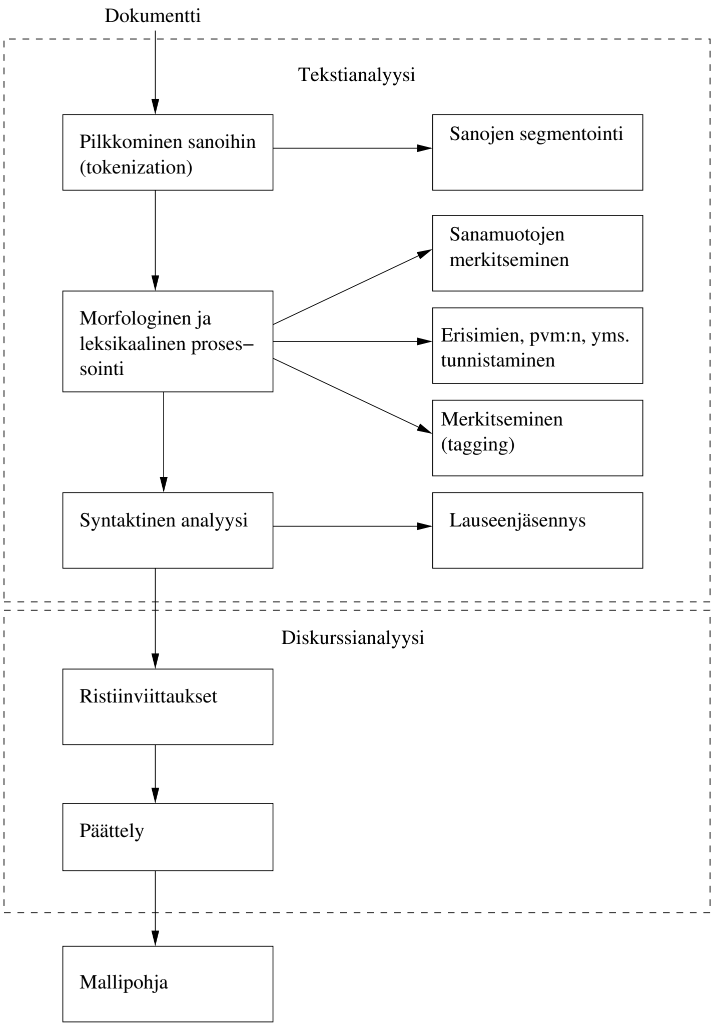
JERUSALEM – A Muslim suicide bomber blew apart18 people on a Jerusalem bus and wounded 10 in a mirror–image of an attack one week ago. The carnage by Hamas could rob Israel's Prime Minister Shimon Peres of the May 29 election victory he needs to pursue Middle East peacemaking. Peres declared all–out war on Hamas but his tough talk did little to impress stunned residents of Jerusalem who said the election would turn on the issue of personal security.

MESSAGE: ID	TST–REU–0001
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 3, 1996 11:30
PRIMSOURCE: SOURCE	
INCIDENT: DATE	March 3, 1996
INCIDENT: LOCATION	Jerusalem
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	''killed: 18''
	''wounded: 10''
PERP: ORGANIZATION ID	

Kuva 6: Uutisartikkeli ja siitä mallipohjaan eristetyt tiedot [Rad99]

Toteutetaanpa säännöt kummalla tavalla tahansa, sisältää eristämisjärjestelmä tekstin prosessointia jollain tasolla. Luonnollisen kielen käsittely aloitetaan rakenteistamalla syötedata, tunnistamalla tekstielementtejä ja suhteita. Yksinkertaisimmillaan eristäminen perustuu eristämissääntöihin, esimerkiksi jos etsitään tietoja ihmisten palkoista, voi sääntöinä toimia mallipohjat ”*person* made *amount-of-money*” tai ”*person* earned *amount-of-money*”[GY00]. Sääntöjä varten teksti on täytynyt esiprosessoida siten, että siitä on tunnistettu nimet ja rahayksiköt.

Jos tekstianalyysia katsotaan tarkemmin, se voidaan jakaa useisiin moduuleihin. Näitä ovat selaaja (tokenizer), leksikaalinen ja morfologinen prosessointi, syntaktinen analyysi sekä diskurssianalyysi, jossa selvitetään viittaukset ja päättelyt [AI99, Gri97]. Vaiheet on esitetty kuvassa 7.



Kuva 7: Tiedon eristämisjärjestelmän rakenne mukailtuna Appeltin, Israelin ja Grishmanin [AI99, Gri97] vastaavista esityksistä.

Selaaja (lexer, scanner) suorittaa tokenisoinnin eli tunnistaa merkkivirrasta yhteenkuuluvien merkkien jonot ja muodostaa niitä vastaavat tokenit. Tokenisointi on hienojakoista segmentointia, jossa erotetaan sanat ja lauseet. Tokenisointi on helppoa suomen tai englannin kaltaisissa kielissä, sillä välilyönnit erottavat sanat ja pisteet erottavat lauseet melko luotettavasti. Aivan yksinkertaista ei lauseiden erottaminen kuitenkaan ole, sillä esimerkiksi lyhenteiden, päivämäärien ja numeroyksiköiden pisteet on pystyttävä erottelemaan lauseen lopettavista pisteistä.

Tekstin morfologisessa analyysissa tekstissä esiintyvät sanat palautetaan perusmuotoon, ja kunkin sanan kaikkien tulkintojen morfologiset piirteet (vähintään sanaluokka) merkitään (sanaluokkalajittelu, tagging). Alla on morfologinen analyysi (TWOL, Two-level Model) lauseelle ”tuo juomaa”[4].

```
"<tuo>"
"tuo"DEM PRON NOM SG
"tuoda"V PRES ACT SG3
"tuoda"V IMPV ACT SG2
"tuoda"V PRES ACT NEG
"tuoda"V IMPV ACT NEG SG
"<juomaa>"
"juoma"N PTV SG
"juoma"V DV−MA N PTV SG
```

Leksikaalisessa analyysissä sanan merkitystä selvitetään leksikon (sanaston) avulla. Morfologinen ja leksikaalinen prosessointi sisältää myös erisnimien, päivämäärien ja rahayksiköiden tunnistamisen. Erisnimien tunnistaminen on erityisen tärkeää eristämisjärjestelmille, sillä tyypillisesti halutaan eristää tapahtumia, piirteitä ja suhteita johonkin erisnimeen liittyvälle objektille. Nimien tunnistukseen liitetään usein myös erilailla ilmaistujen nimien yhdistämistä toisiinsa, esimerkiksi *Bill Gates* tunnistetaan samaksi kuin *Mr. Gates* ja *Hewlett-Packard* osataan yhdistää *HP:ksi*.

Alkeellinen syntaktinen analyysi tarkoittaa esimerkiksi vain predikaatin tunnistamista, mutta syntaktinen analyysi voi jäsentää lauseen myös täydellisesti, määrittäen kaikki lauseenjäsenet ja subjekti-verbi-objekti -suhteet, sijamuodot ja aikamuodot.

Diskurssianalyysi selvittää tekstin viittaukset ja tekee päättelyitä. Viittauksia ovat anaforiset viittaukset sekä monimerkitykselliset erisnimet. Anaforisten viittauksien selvittäminen tarkoittaa esimerkiksi sen ratkaisemista keneen tekstissä esiintyvä sana *hän* viittaa. Erisnimet ovat ongelmallisia silloin, kun sana ei ole yksikäsitteinen, vaan esimerkiksi Nokia voi tarkoittaa paikannimeä tai yritystä. Päättely voi sisältää tekstin tulkintaa, esimerkiksi lauseiden *”Ahtisaari oli presidentti. Häntä seuraa Halonen”* ja *”Ahtisaari on presidentti. Hän seuraa Halosta”* ero tai suhteellisten päivämäärien tulkintaa, esimerkiksi *”viime viikolla”*.

[4]Esimerkki on Yleisen kielitieteen laitoksen Unix-kurssilta, osoitteesta http://www.ling.helsinki.fi/~mlounela/unixk/twol.html

Eristämisjärjestelmän täytyy tietenkin ottaa huomioon tekstin rakenteisuus. Järjestelmiä on rakennettu rakenteiselle, puolirakenteiselle ja vapaalle tekstille [GJ99]. Esimerkiksi AutoSlog (Riloff 1993) on eristämisalgoritmi rakenteettomalle tekstille. Sitä on käytetty terroristi-iskun kohteen poimimiseen dokumenteista. Muita järjestelmiä ovat LIEP, PALKA, CRYSTAL, CRYSTAL + Webfoot ja HASTEN [Mus99]. Näiden tekniikoiden eristämän tiedon granulariteetti vaihtelee, koska osa järjestelmistä eristää tiedon syntaktisin perustein, esimerkiksi objektin tiettyjen verbien jälkeen, osa taas eristää kokonaisia lauseita.

Erillään perinteisestä tiedon eristämisyhteisöstä on WWW-sivuihin pyritty luomaan poimimistoimintoja. Tämän seurauksena ovat syntyneet sovittimet, jotka pyrkivät tarjoamaan WWW-sivuihin kohdistettuja tietokantamaisia hakutoimintoja. Ne toimivat siis yhtenä toteutustapana tiedon eristykseen. Sovittimet ovat yleensä suoraviivaisia, yksinkertaisia poimijoita ja niidenkin generoiminen automaattisesti on usein käytännöllistä. Sovittimet eivät tee raskaita kielen analyyseja, vaan poimivat tiedon erotinmerkkien perusteella (delimiter based) [Mus99]. Sovittimet eivät sovellu täysin rakenteettomaan tekstiin, mutta niissä voidaan käyttää hyväksi joitain eristäjien komponentteja kuten esimerkiksi selaajia [GJ99]. Sovittimet ovat tietolähdekohtaisia ja ottavat vastaan kyselyitä lähteeseensä, etsivät sieltä kyselyyn sopivimmat sivut sekä poimivat sivuilta kysytyn tiedon ja palauttavat vastauksen.

Sovittimelle määritellään tekstimallein missä kohtaa HTML-sivua haluttu tieto sijaitsee ja sovittimet tunnistavat ainoastaan halutun tiedon alku- ja loppukohdat eivätkä ymmärrä sisältöä. Tällainen erottaja on tehokas ja sillä voidaan analysoida paljon tietoa, mutta jos lähteen formaatti muuttuu, täytyy määritelmä myöskin päivittää. Määrittely, joka annetaan syötteenä erottelijalle, on yksinkertainen tekstitiedosto, jota voi muokata suoraan millä tahansa tekstieditorilla. Hammerin ja Garcia-Molinan [HGMC+97] tutkimusryhmä eristi WWW-sivulta säätiedot kaupungeittain ja tämä rakenne tallennettiin oliona, jonka jälkeen siihen voi kohdistaa kyselyitä. Eristämistä on kokeiltu myös vuokra-asuntojen tietojen vertailuun, jolloin esimerkiksi makuuhuoneiden lukumäärä löydetään pseudosäännöllä *"etsi merkkijono br, brs tai bedrooms, jota seuraa tai edeltää numero"* ja hinta säännöllä *"sivuuta kaikki merkit siihen asti, kunnes löytyy dollari-merkki, jota seuraa numero"* [Mus99].

Esimerkkitapauksissa luvussa 6.2 esitelty Ariadne-järjestelmä on WWW-sovellus, joka kerää tietoa elokuvateattereista ja -näytöksistä sekä ravintoloista eri kaupungeissa

USA:ssa. Tiedot kerätään viidestä eri WWW-lähteestä, joihin on yhteensä kuusi sovitinta. Käyttäjä voi kohdistaa järjestelmään hakuja kaupungeittain esimerkiksi elokuvateatterin läheisyydessä olevista ravintoloista. Sovittimet luodaan automaattisesti siten, että sovellusasiantuntija nimeää haettavan tiedon ja maalaa esimerkkivastauksen lähteestä. Sivujen sisällöstä luodaan tietomalli, jonka perusteella sivuilta löydetään käyttäjän kaipaama tieto.

4.2.2 Automaattinen yhteenveto

Yhteenveto tarkoittaa yhdestä tai useammasta tietolähteestä oleellisimman tiedon keräämistä ja esittämistä. Yhteenvedot auttavat lukijaa päättämään, haluaako hän lukea koko artikkelin vai ei [MR95]. Perinteisesti kustakin tallennetusta tietoelementistä, esimerkiksi dokumentista tai artikkelista, on kirjoitettu erikseen ingressi, abstrakti, johdanto tai muu tiivistelmä, mutta tätä osuutta on myös pyritty automatisoimaan. Automaattinen yhteenveto on tutkielmassa kategorisoitu yhtä tietolähdettä käsittelevään lukuun, vaikka yhteenveto voidaankin käytännössä muodostaa useasta lähteestä. Yhteenveto useasta eri lähteestä on joka tapauksessa tehtävä joko muodostamalla ensin kustakin lähteestä oma yhteenveto ja yhdistämällä ne, tai yhdistämällä ensin lähteet ja tekemällä yhteenveto tästä koostetusta dokumentista. Tiedon yhdistämiseen ja vertailuun tutustutaan myöhemmin tutkielmassa.

Yhteenveto yhdestä dokumentista pyrkii yleensä olemaan noin 5-30% alkuperäisen kokoinen [HM00]. Pienet kämmenlaitteet tai yhteenveto useammasta lähteestä vaativat kuitenkin voimakkaampaa tiivistämistä. Automaattiset yhteenvedot voidaan kategorisoida tietoköyhiin (knowledge-poor) ja tietorikkaisiin (knowlegde-rich), joskin on myös molempia yhdistäviä järjestelmiä. Ensin mainitut perustuvat etukäteen määrättyihin sääntöihin, esimerkiksi avainsanoihin, sivupohjiin, lauseiden eristämiseen ja/tai tilastollisiin tekniikoihin. Jälkimmäisissä pyritään ymmärtämään tekstin sisältöä, mihinn tarvitaan laajaa taustatietoa aiheesta sekä luonnollisen kielen analysointia ja tuottamista. Tyypillinen kielen generoija sisältää kaksi komponenttia, sisältösuunnittelijan, joka valitsee informaation tietolähteestä, ja lingvistisen komponentin, joka valitsee sanat joita käytetään ja miten sanat järjestetään lauseiksi. Yksinkertaisesti sanottuna yhteenvetojärjestelmällä on kaksi perustehtävää: määritellä lähteen käsittelemät tärkeät asiat ja päättää miten tiivistää tekstin lyhyemmäksi.

Sivupohjiin eli mallipohjiin (template) perustuvassa järjestelmässä dokumentti luetaan kielen käsittelyjärjestelmällä, joka täydentää mallipohjassa kysytyt kohdat dokumentista. Esimerkiksi terrorismia koskevista uutisartikkelista saatettaisiin poimia tietolähde, julkaisupäivä, tapahtumapäivä, tapahtumapaikka, tapahtumatyyppi ja haavoittuneiden määrä [MR95]. Tämä on tiedon eristämistä, jossa yhteenvetona toimii täytetty lomake.

Yhteenveto voidaan muodostaa poimimalla lähteestä lauseet, jotka on havaittu jollain mittarilla tärkeäksi lähteessä. Poimiminen voi perustua tilastollisiin menetelmiin, jossa kullekin dokumentin lauseelle määritellään pisteytys ja yhteenvetoon valitaan suurimmat pisteet saaneet lauseet. Tilastollista menetelmää varten täytyy olla kerätty aihepiirikohtaista tietoa sanojen esiintymistodennäköisyyksistä. Pisteytys voidaan perustaa ennalta määriteltyihin semanttisiin rooleihin, retorisiin suhteisiin, tiettyihin termeihin (esimerkiksi kappale, joka alkaa *"Tämä raportti"* tai *"Yhteenvetona"*), avainsanafrekvensseihin tai sijaintiheuristiikkoihin (tärkeät lauseet ovat koko dokumentin alussa ja lopussa sekä kappaleiden ensimmäiset ja viimeiset lauseet). Kupiecin, Pedersenin ja Chenin [KPC95] yhteenvetojärjestelmä perustuu lauseiden eristämiseen ja tilastotietoihin. Tällainen poimintoihin perustuva yhteenveto vastasi 84%:sti asiantuntijan samasta dokumentista luomaa yhteenvetoa. Boguraevin ja Neffin [BN00] tiivistelmäjärjestelmällä on käytettävissään suurempi dokumenttien prosessointiin ja analysointiin kehitetty järjestelmä. Se sisältää komponentit erisnimien tunnistamiseen, teknisten termien tunnistamiseen sekä lyhenteiden käsittelemiseen. Näiden avulla muodostetaan dokumenttikokoelmalle aihepiirikohtainen taustatietokanta, jota varten jokaisesta dokumentista poimitaan kaikki merkitykselliset sanat (sanat, jotka eivät ole poistolistalla) sekä sanarykelmät, jotka järjestelmän lingvistiset komponentit tunnistavat. Nämä liitetään sanastoon perusmuodoissaan, lasketaan ja kootaan kaikki dokumentit yhdistävään aihepiirisanastoon. Tätä sanastoa tilastotietoineen käytetään lauseiden keskeisyysarvojen laskemiseen. Sanaston lisäksi järjestelmä muodostaa dokumentin rakennekuvauksen, joka sisältää muun muassa ulkoasutunnisteet, otsikot, abstraktin, osiot ja aliosiot, taulukot ja kuvat. Keskeisiksi sanoiksi katsotaan sanat, jotka esiintyvät useammin kuin kerran dokumentissa, joiden keskeisyysarvo sanastosta katsottuna ylittää tietyn raja-arvon tai joiden sijainti dokumenttirakenteessa on merkittävä. Taustatiedon pakollisuus rajoittaa järjestelmän soveltuvuutta kaiken tyyppisten dokumenttien käyttöön.

Yleisten yhteenvetojen lisäksi on käyttäjäkeskeisiä (user-focused), personalisoituja yhteenvetoja, jotka pyrkivät tyydyttämään yksilön tai ryhmän (esimerkiksi lapset) spesifimmät tarpeet kiinnostuskohteen ja tietotaidon mukaan [HM00]. Käyttäjän profiili määrittelee yhteenvedon luonteen ja sisältöön poimitaan kiinnostuskohteita käsittelevät asiat. Yhteenvedon uusia sovellusaloja ovat monikielisistä lähteistä muodostettu yhteenveto, rakenteellisesti eroavien lähteiden yhdistäminen (esimerkiksi baseball-pelaajan tilastolliset tiedot yhdistetään häntä käsittelevään uutisartikkeliin), usean lähteen yhteenvedot sekä eri medioiden yhteenvedot (audio- tai videokatkelmien sisällöt) [HM00].

Summaukset tai yhteenlaskut voidaan tavallaan ajatella automaattisiksi yhteenvedoiksi ja jotkut kielet (kuten SQL) tarjoavat keinoja ryhmittelyyn ja summauksiin. Tietovarasto (warehouse) ja tiedon louhinta (data-mining) -teknologiat ovat myös suuntautuneet tähän ongelmaan [Wie99].

4.2.3 Segmentointi

Segmentoinnilla tarkoitetaan tiedon jakamista useammaksi tietoelementiksi. Pitkiä dokumentteja on raskasta käsitellä ja ne saattavat kattaa useita eri aihealueita. Erityisesti tiedon hakuun tämä saattaa tuottaa ongelmia: vaikka pitkä dokumentti sisältäisi tarkan vastauksen käyttäjän kysymykseen, voi vastauksen frekvenssi dokumentissa kokonaisuudessaan olla niin alhainen, ettei dokumenttia pidetä relevanttina. Vaikka dokumentti löytyisikin, eivät käyttäjät varmasti miuluusti selaa läpi 30 sivun dokumenttia, jonka jollain sivulla on se erittäin tärkeä ja hyödyllinen kappale. Segmentointi voidaan tehdä katkomalla teksti suoraviivaisesti tietyn mittaisiin osiin ja käsittelemällä näitä osia, segmenttejä, omina kokonaisuuksinaan. Usein katkominen loogisiin kokonaisuuksiin lisää segmentoinnin käyttöarvoa. Loogisella kokonaisuudella tarkoitetaan, että otetaan huomioon tekstin rakenne tai sisältö: katkotaan teksti esimerkiksi kappaleittain, luvuittain tai aihealueittain. Segmentointia voidaan hyödyntää tekstin lisäksi myös ääni- ja kuvatiedojen käsittelyssä [Cho00, Rey99].

Yksinkertaisessa segmentoinnissa teksti katkotaan tietyn mittaisiin tekstilohkoihin ottamatta huomioon tekstin rakennetta tai sisältöä. Segmenttikoko voi olla esimerkiksi 200 tavua, jonka jälkeen alkaa uusi segmentti. Tällaisesta segmentoinnista voi olla hyötyä esimerkiksi tiedon haussa. Yleensä segmentoinnissa kuitenkin otetaan huomioon sanat, virkkeet tai kappaleet. Tällöin tarvitaan kielen prosessointia, sillä tietokone tunnistaa digitaalisesta datasta ainoastaan merkit ja rivit. Usein segmentointia tehdään, jotta tiedon haku helpottuisi: jotta tietomassasta löytyisi relevantimpi ja täsmällisempi vastaus kyselyyn. Tätä varten tekstit on muutettava muotoon, jossa niitä voi verrata kyselyyn. Yleinen muoto on tehdä muodostaa tekstistä termivektori. Termivektoria varten dokumentista poimitaan termejä, jotka voivat olla yksittäisiä sanoja tai useampisanaisia fraaseja. Yleensä sanan taivutusmuodolla ei ole merkitystä, vaan sanan esiintyminen jossakin muodossa dokumentissa on riittävää. Tätä varten sanoille voidaan tehdä morfologinen analyysi eli palauttaa sanat perusmuotoon (lemmatization) tai poimia sanavartalot pudottamalla sanapäätteet (stemming). Termivektoria varten isot kirjaimet muutetaan usein pieniksi ja tekstistä poistetaan välimerkit. Termivektoriin poimitaan vain merkitykselliset sanat, jotka voidaan määrätä esimerkiksi sanaluokan mukaan (verbit ja substantiivit) tai sanafrekvenssien mukaan (otetaan huomioon ainoastaan sanat, jotka esiintyvät useammin kuin kerran dokumentissa) [Cho00, Hea97, SSBM96].

Segmentoinnilla halutaan tunnistaa tekstin elementit, jotka ovat omia kokonaisuuksiaan ja jotka voidaan erottaa muusta sitä ympäröivästä tekstistä. Selkein mahdollisuus on käyttää hyvin määriteltyjä fyysisiä elementtejä, jotka on helppo erottaa. Näitä ovat taulukot, kuvat, kuvatekstit, otsikot, kirjallisuusviitteet, listat, teoreemat tai perinteisemmät tekstielementit kuten kappaleet, lauseet tai osiot [RA95, SSBM96].

Avainsanojen tai fraasien perusteella segmentin vaihtumiskohdat pyritään ilman suurempaa tekstianalyysia määrittelemään kohtiin, joissa aihepiiri vaihtuu [Rey99]. Tekstistä etsitään tiettyjä tärkeäksi ajateltuja avainsanoja ja lauseet, jotka näitä sanoja sisältävät, aloittavat tai lopettavat segmentin. Esimerkiksi uutisen loppumista ennustaa fraasi *"This is <name>"*, sillä reportteri yleensä lopettaa uutisensa näihin sanoihin. Avainsanojen käyttö on rajoittunut tiukasti aihepiirikohtaisesti. Esimerkiksi asuntoilmoitusten välinen samankaltaisuus on suuri, mutta segmentointi kiinteistöittäin onnistuu merkkeihin perustuvien sääntöjen avulla [Man98].

Kehittyneemmässä, semanttisessa segmentoinnissa otetaan huomioon tekstin osien sisällöllinen samankaltaisuus: jos samankaltaiset tekstilohkot käsittelevät samaa aihetta, ne kuuluvat samaan segmenttiin [Hea94]. Tällaisen järjestelmän täytyy olla älykäs ja "ymmärtää" tekstiä: järjestelmän täytyy tietää, käsittelevätkö vierekkäiset lauseet samaa aihetta vai eri asioita.

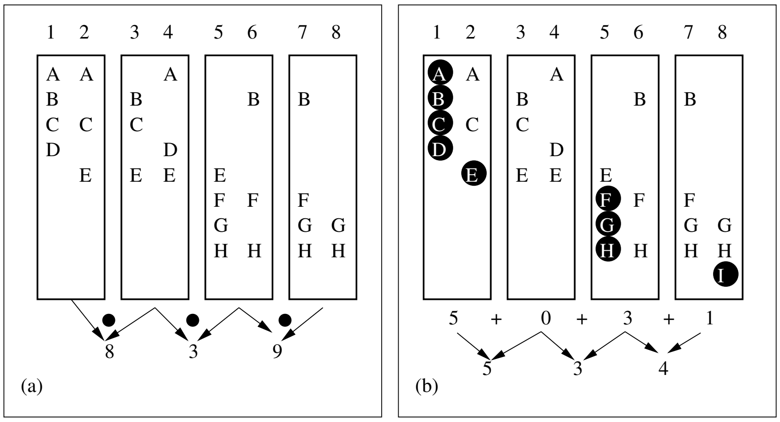
Laskennallisena perusajatuksena on, että jos teksti jatkuu käyttäen samanlaisia sanoja, puhutaan todennäköisesti vielä samasta aiheesta. Pelkkä sanafrekvenssi on joskus riittämätön, jolloin voidaan ottaa huomioon myös sanojen synonyymit ja merkityssuhteet.

Päätös segmentin katkaisukohdista tehdään laskemalla lohkoille samankaltaisuusarvot, joiden perusteella määritellään segmenttien vaihtumiskohdat. Yleisin samankaltaisuusmitta on kosinifunktio, joka lasketaan sanafrekvenssien perusteella. Jos samankaltaisuusarvo kahden lohkon välillä on korkea, tarkoittaa se, että niillä on monta yhteistä termiä, jolloin lauseiden voidaan uskoa käsittelevän samaa aihepiiriä [Hea94, Cho00]. Samankaltaisuus tai merkityssuhde voidaan ottaa huomioon tarkistamalla sanojen merkitykset ja synonyymit sanakirjasta [Koz93]. Tällöin voidaan ottaa huomioon järjestelmälliset semanttiset suhteet (punainen/väri ja punainen/vihreä) sekä ei-järjestelmälliset semanttiset suhteet (punainen/veri ja punainen/lämmin). Jälkimmäiset suhteet vaativat laajaa tietämysjärjestelmää.

Kosinimittaa voidaan kehittää painotetuilla termivektoreilla, jolloin termivektorin painot määräytyvät kunkin sanan tärkeyden ja yleisyyden perusteella [RSA97]. Yleinen kaava termien painoille on Saltonin *termifrekvenssi * käänteinen dokumenttifrekvenssi* (TF*IDF) [Sal89]. Mitta ottaa huomioon sen, että jos kaikki dokumentit sisältävät termin, ei termi ole kovin informatiivinen. Termit, jotka esiintyvät usein yhdessä dokumentissa ja harvoin koko dokumenttikokoelmassa, ovat parhaimpia kuvaamaan dokumenttia ja mitta antaa korkeimmat painot juuri tällaisille termeille. Youmans [You91] käyttää samankaltaisuusmittana sitä, kuinka monta uutta termiä lohkossa esiintyy. Hänen mukaansa uuteen aihepiiriin siirryttäessä käytettävä termistö muuttuu, joten uudet sanat kerääntyvät aihepiirien vaihtokohtaan.

Kuvassa 8 on esitetty a) sanafrekvensseihin perustuva kosinimitta ja b) uusiin sanoihin perustuva samankaltaisuusmitta. Kirjaimet kuvaavat sanoja ja yläosan numerot lauseita, joita vertaillaan kahden lauseen kokoisissa fragmenteissa. Alaosan numero kuvaa samankaltaisuusarvoa – alhainen samankaltaisuus indikoi hyvää segmentin vaihtumiskohtaa. Kuvan b)-kohdassa lasketaan uudet sanat kussakin lohkossa sekä sen ympäristössä. Tällöin laskeva uusien ja vanhojen sanojen suhde indikoi aiheen päättymistä ja nouseva suhde uuden aiheen alkua [You91].

Jokainen lohkon vaihtumiskohta on potentiaalinen segmentin vaihtumiskohta ja kuhunkin lohkoväliin lasketaan sen vierekkäisten lohkojen välinen samankaltaisuusarvo. Samankaltaisuuden lisäksi segmentin vaihtumiskohtaan saattaa vaikuttaa esimerkiksi lauseen sijainti dokumentissa.

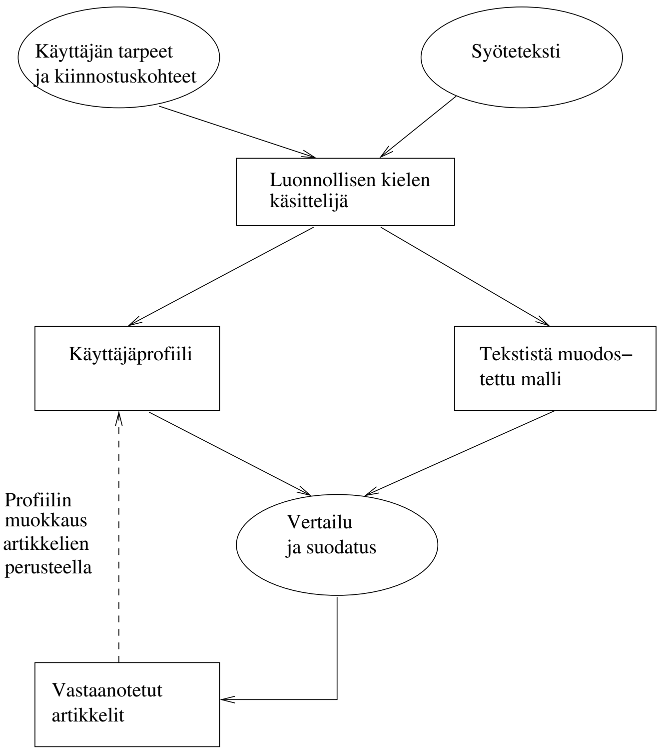


Kuva 8: Peräkkäisten lohkojen samankaltaisuuden laskutapoja [Hea97].

4.2.4 Suodattaminen

Suodattaminen on menetelmä, joka vähentää tietotulvaa ja ohjata tietoa sitä tarvitseville. Suodattaminen on samantyylinen etsimis-, reititys-, kategorisointi- ja tiedoneristämisprosessien kanssa, ne kaikkihan pyrkivät relevantin tiedon tarjoamiseen. Suodatin määritellään profiileilla, joko yksittäisen käyttäjän profiililla tai ryhmäprofiileilla. Suodatin poistaa ylimääräisiä syötetietoja, muttei varsinaisesti etsi tai poimi sopivaa tietoa. Yleisin esimerkki suodattimista on sähköpostien suodattimet roskapostien estämiseksi. Suodattimilla pystytään estämään postien tulo tietystä osoitteesta tai vertaamaan postia johonkin malliin tai sääntöihin, esimerkiksi, jos teksti sisältää sanat *"free"* tai *"call"*, se on todennäköisesti ei-toivottu. Suodattaminen lähtee liikkeelle käyttäjästä ja sillä viestitetään suhteellisen pysyviä kiinnostuksen kohteita ja ja tiedonhaluja [OS95, BC92, Paz00]. Suodattamisprosessi on esitetty yksinkertaisena kuvassa 9.

Suodattaminen eroaa kategorisoinnista siten, että suodattimet ovat käyttäjän muutettavissa ja siten dynaamisempia kuin kategorisoinnin usein staattiset luokat. Tiedon reititys tarkoittaa relevantin tiedon ohjaamista käyttäjille eli sekin sisältää dokumenttien ohjailua sisällön perusteella sopiville vastaanottajille. Mutta reititys on pikemminkin tiedon tarjoajan puolelta tehtävää palvelua, suodattaminen taas käyttäjän omia valintoja. Tiedonhaku puolestaan on kertaluontoista, eikä siinä tiedon ajankohtaisuus välttämättä ole tärkein tekijä, kun suodattamisessa on usein hyvinkin tärkeää. Tiedon eristäminen eroaa suodattamisesta siten, että siinä syötedatasta poimitaan faktoja, eikä tieto-objektista kokonaisuudessaan olla kiinnostuneita.



Kuva 9: Suodattamisprosessi

Suodattamisessa syötetekstiä verrataan annettuun profiiliin ja lasketaan todennäköisyys, että syöteteksti vastaa profiilia. Järjestelmä joutuu käsittelemään useita eri formaatteja ja ollakseen tehokas sen kielenkäsittelytekniikan on oltava suhteellisen yksinkertaista. Negatiivinen profiili on huomattavasti helpompi määritellä kuin profiili, joka kertoo mitä tietoa halutaan. Negatiivinen profiili kuvailee tiedon, jota ei haluta ja kaikki muu tieto tulee päästää suodattimen läpi käyttäjän vastaanotettavaksi. Profiilit voidaan kirjoittaa sääntöinä, järjestelmä voi oppia säännöt teksteistä (esimerkiksi perseptronilla) tai suodatettavasta aineistosta voidaan muodostaa prototyyppi, johon syötetekstejä verrataan.

Esimerkiksi INFOrmer-järjestelmä suodattaa uutisartikkeleita käyttäjän personoinnin mukaan [OS95]. Alkuperäinen käyttäjäprofiili rakennetaan automaattisesti käyttäjän osoittamista, hänen kiinnostustaan vastaavista artikkeleista ja dokumenteista. Kunkin artikkelin vastaanottamisen jälkeen käyttäjä voi arvioida sen relevanttiuden eli antaa järjestelmälle palautetta, jotta se pystyy tarkentamaan profiilia. Palautteen avulla järjestelmä seuraa myös käyttäjän muuttuvia kiinnostuksen kohteita. Käyttäjäprofiili mallinnetaan assosiatiivisella verkolla, johon kerätään termejä käyttäjää kiinnostavista teksteistä. Termit muodostavat verkkoon solmuja ja vierekkäisten termien solmut yhdistetään kaarilla. Kaarille asetetaan painoja termin esiintymistiheyden mukaan. Saapuvat syötedokumentit analysoidaan ja niistä muodostetaan dokumenttimalli, jota verrataan käyttäjäprofiiliin. Järjestelmä sisältää leksikaalisen analysoijan, poistolistan ja stemmausalgoritmin.

Suodattamisella voidaan myös tarkoittaa esimerkiksi jo vastaanotetun tiedon suodattamista eli esimerkiksi uutisartikkelien vastaanottamisen siinä tapauksessa, jos käyttäjä ei ole vielä vastaanottanut kyseistä uutista. Lisäksi esimerkiksi yrityksillä voi olla turvallisuusmediaattori palomuurissaan yksityisyyden ja turvallisuuden takaamiseksi. Se suodattaa tiedosta kaiken salassa pidettävän tiedon ennen kuin tieto päästetään ulkopuoliseen maailmaan [Wie99].

4.2.5 Luokittelu

Luokittelu helpottaa suurten tietomäärien käsittelyä jakamalla ne pienempiin osakokonaisuuksiin. Luokittelussa dokumentit (tai muut tietoelementit) jaetaan joukkoihin, jotka käsittelevät jonkin kriteerin mukaan samaa aihetta. Tyypillisesti luokittelua sovelletaan sähköpostien, tekstimateriaalin tai tiedostojen lajitteluun hakemistohierarkiassa, rakenteisiin hakuihin ja aiheen tunnistamiseen [DPH98]. Luokittelu voi perustua etukäteen määriteltyihin luokkiin, jolloin luokkien erottavat tekijät on valittu opetusaineistoa käyttämällä. Järjestelmälle annetaan valmiit ohjeet, kuinka luokitella aineisto. Tätä kutsutaan ohjatuksi oppimiseksi. Ohjaamattomassa oppimisessa järjestelmälle syötetään esimerkkitiedostoja, joista se muodostaa jonkinlaisen mallin itselleen. Järjestelmä ei tiedä mihin luokkaan kukin dokumentti kuuluu, vaan sen tehtävä on jakaa dokumenttijoukko ryhmiin siten, että jokin annettu kriteeri optimoituu. Kun dokumentit esitetään moniulotteisessa vektoriavaruudessa, voi vektoriavaruudessa hahmottua jonkinlaisia vektorikeräymiä. Luokittelijan suunnittelu on näiden vektorien välisten rajojen määräämistä.

Ohjattu oppiminen

Luokittelua, jossa luokittelukriteerit on opittu ohjatusti, kutsutaan kategorisoinniksi. Tekstin kategorisoinnissa luonnollisen kielen teksti luokitellaan sisällön perusteella yhteen tai useampaan ennalta määrättyyn luokkaan [DPH98]. Kategorisoinnin suorittaa automaattinen luokittelija, joka on opetettu opetusjoukolla. Luokat ovat valmiiksi kiinnitettyjä ja opetusjoukon dokumentit on sijoitettu oikeisiin luokkiinsa manuaalisesti.

Yleinen tekniikka dokumenttien esittämiseen on esittää ne vektorina moniulotteisessa avaruudessa, jossa kukin termi vastaa yhtä akselia. Suurimpana ongelmana on termiavaruuden valtava koko. Tavallisen kokoinen tekstikokoelma sisältää helposti kymmeniä tai satoja tuhansia termejä [YP97]. Termiavaruutta on yleensä pienennettävä, kuitenkin vähentämättä kategorisoinnin tarkkuutta. Termiavaruuden pienentämistä kutsutaan piirreirrotukseksi ja se on kategorisoinnin ensimmäinen vaihe. Piirreirrotus voidaan joissain tapauksissa tehdä manuaalisesti, mutta yleensä sen automaattinen toteutus on toivottua. Piirreirrotuksessa opetusaineistosta valitaan piirreavaruus eli termit, joista kullekin luokalle muodostetaan ominaisvektori. Termistön on oltava tarpeeksi laaja erotellakseen luokat toisistaan, mutta tehokkuussyistä luonnollisesti optimoitu.

Termien valintaan on olemassa useita menetelmiä, muun muassa termin esiintymistiheys (Document Frequency Thresholding), tiedon kertymä (Information Gain), keskinäinen yhteys (Mutual Information), CHI-tilastot (CHI) ja termin vahvuus (Term Strength) [YP97]. Tutkielman esimerkkitapauksessa luvussa 6.1 Tytti: luokittelu ja suodatus on termistöön valittu 3000 sanaa tiedon kertymä menetelmällä. Käytettävän termistön lisäksi kerätään usein poistolista (stoplist) sisältäen sanat, joita luokittelussa ei huomioida niiden huonon erottelukyvyn takia (esimerkiksi konjunktiot, pronominit) [YP97]. Piirreirrotusmenetelmien hyvyyttä voidaan verrata kahdella mitalla, *saannilla* (recall) ja *tarkkuudella* (precision). Saanti kertoo, kuinka suuren osuuden kaikista oikeista luokista menetelmä onnistuu löytämään: *(löydetyt oikeat luokat / kaikki oikeat luokat)*. Tarkkuus tarkoittaa löydettyjen, oikeiden luokkien suhdetta kaikkiin löydettyihin luokkiin: *(löydetyt oikeat luokat / kaikki löydetyt luokat)*. Näiden suureiden avulla saadaan kunkin valintamenetelmän keskimääräinen tehokkuus testiaineistolla.

Opetusdokumenttien ominaisvektorien muodostama piirreavaruus jaetaan luokkia vastaaviin toisensa poissulkeviin osajoukkoihin. Luokittelijan tehtävä on siis määrittää rajat ennalta määrättyjen dokumenttien piirreavaruuksien välille. Luokittelijat muodostetaan automaattisesti ja ne perustuvat induktiivisiin oppimismetelmiin. Luokittelijan luontivaiheessa järjestelmälle syötetään esimerkkiaineistoa kustakin kategoriasta, jonka perusteella järjestelmä oppii luokittelumallin parametrit [DPH98]. Kun luokittelija on olemassa, on helppo määrätä kullekin uudelle dokumentin piirrevektorille oikea luokka. Valmis luokittelija saa syötteenä termiavaruusvektorina esitetyn dokumentin ja palauttaa tulosteena luokan, johon dokumentti kuuluu tai todennäköisyydet dokumentin kuulumisesta kuhunkin luokkaan.

Luokittimet voivat perustua tilastolliseen malliin, jolloin luokkien ominaisvektorien todelliset todennäköisyysjakaumat oletetaan tunnetuiksi. Myös luokkien esiintymistodennäköisyydet oletetaan tunnetuiksi. Vaikka näitä onkin yleensä epärealistista olettaa, on luokittimet koettu hyödyllisiksi. Todennäköisyyksille voidaan olettaa jokin parametrinen malli, usein moniulotteinen normaalijakauma ja johtaa luokitin tämän avulla. Jos luokittelija ei oleta tunnetuksi mitään luokkien todellisia tiheysfunktioita, perustuu oppiminen opetusjoukon näytteisiin. Kun uusi piirrevektori täytyy määrittää kuuluvaksi johonkin luokkaan, laskee luokitinalgoritmi vektorin etäisyyden kunkin opetusjoukon ominaisvektoriin ja näistä valitaan lähin.

Induktiiviseen (tilastolliseen) oppimisalgoritmiin perustuvia luokittelijoita ovat Find Similar, päätöspuut (Decision Trees), naiivi bayes -luokittelijat (Naïve Bayes), Bayes verkot (Bayes Nets) ja Support Vector Machines [DPH98]. Esimerkiksi Bayes-luokittelijoissa kaikki opetusesimerkit talletetaan ja piirteet erotetaan. Luokiteltavan tapauksen luokaksi valitaan se, jonka opetusaineiston joukossa esiintyy luokiteltavan tapauksen piirrevektori useammin kuin minkään muun luokan opetusaineiston joukossa. Muita luokittimia ovat lähimmän naapurin luokitin (Nearest Neighbor), k:n lähimmän naapurin luokitin (k-Nearest Neighbor), neuroverkkoluokittimet, monikerroksinen perseptroniverkko (multilayer perceptron MLP network), oppiva vektorikvantisointiverkko (learning vector quantizer LVQ network).

Ohjaamaton oppiminen

Ryvästämisessä (clustering) ei ole etukäteen määriteltyjä luokkia, vaan dokumenttijoukko ryhmitellään ja ryhmät muodostavat luokat. Ryvästäminen on tapausten (yleensä vektorimuodossa tai pisteenä moniulotteisessa avaruudessa esitettyjen dokumenttien) järjestämistä ryppäisiin samankaltaisuuden perusteella [JMF99]. Annettu kokoelma luokittelemattomia tapauksia tulee siis ryhmitellä mielekkäisiin, tarkoituksenmukaisiin ryppäisiin. Ryvästämisprosessi koostuu seuraavista vaiheista: tapausten esittäminen (sisältäen mahdollisesti piirreirrotuksen ja/tai piirrevalinnan), samankaltaisuusmitan määrittäminen ja ryvästäminen. Tapausten (tai mallin, kaavan) esittämisessä osa tiedoista voidaan määrittää etukäteen, esimerkiksi tulevien ryppäiden määrä, ryvästämisalgoritmin käyttämien piirteiden määrä ja tyyppi. Kuten aiemmin mainittiin, mallit esitetään usein moniulotteisina vektoreina, joissa jokainen ulottuvuus kuvaa yhtä piirrettä. Piirteet voivat olla määrällisiä (paino) tai laadullisia (väri). Piirrevalinnalla tunnistetaan tehokkaimmat termit koko termistöstä. Piirreirrotuksessa keskeisille piirteille tehdään transformaatio, jotta tuotetaan uusia, keskeisiä piirteitä. Samankaltaisuutta mitataan yleensä dokumenttiparien etäisyydellä vektoriavaruudessa. Ryvästämistapoja on useita: jako voidaan tehdä esimerkiksi hierarkkisiin ja heuristisiin menetelmiin [Sal89]. Lisäksi menetelmä saattaa olla luonteeltaan esimerkiksi kokoava (ensin kukin tapaus omana ryppäänään, joista samankaltaisimpia aletaan yhdistää keskenään) tai hajottava (lähtötilanteessa kaikki tapaukset kuuluvat samaan ryppääseen, josta aletaan erottamaan ryppäitä) ja kiinteä (hard, toisensa poissulkevat ryppäät) tai sumea (fuzzy, tapaus kuuluu osittain yhteen ryhmään, osittain toiseen) [JMF99]. Kaikki ryvästysalgoritmit tuottavat ryppäitä – siitä huolimatta sisältääkö data ryppäitä vai ei.

Hierarkkiset ryvästysmenetelmät perustuvat siihen, että tapausten samankaltaisuudet tiedetään. Samankaltaisuudet voidaan laskea esimerkiksi dokumenttien vektoriesityksistä. Hierarkkinen ryvästäminen muodostaa samat ryppäät riippumatta järjestyksestä, jossa dokumentit esitetään. Heuristisissa ryvästysmenetelmissä dokumenttien etukäteistiedoista ja ryppäiden vakaudesta luovutaan. Yksinkertaisimmillaan heuristisessa ryvästämisessä otetaan dokumentti kerrallaan käsittelyyn, sitä verrataan olemassa oleviin ryppäisiin ja jos se ei ole riittävän samanlainen minkään ryppään kanssa, se muodostaa oman uuden ryppään. Samankaltaisuuksien määrittelyyn käytetään ryppäiden keskiarvoja, joita päivitetään aina kun ryppääseen liitetään uusi dokumentti. Tällä menetelmällä ryppäistä tulee helposti epätasaisia, toisten ollessa valtavan isoja, toisten hyvin pieniä. Erikokoisuuden estämiseksi voidaan rajoittaa ryppään kokoa, ryppäiden määrää ja ryppäiden välistä päällekkäisyyttä (yhteisten elementtien) osuutta. Ryppään kokoa voi valvoa jakamalla rypäs kahtia, kun se saavuttaa maksimikoon. Toinen heuristinen ryvästysmenetelmä on etsiä dokumentteja, jotka sijaitsevat dokumenttiavaruudessa toisten dokumenttien ympäröiminä. Ympäröidyistä dokumenteista tehdään ryppään keskuksia ja kuhunkin ryppääseen kootaan sen läheisyydessä olevat dokumentit. Ryväskeskukset löytyvät tiheystestin avulla – aina kun mihinkään ryppääseen kuulumatonta dokumenttia ympäröi tarpeeksi tiheä määrä toisia dokumentteja, dokumenteista muodostetaan oma ryväs. Myös muita heuristisia ryvästysmenetelmiä on olemassa.

Kokoavassa (merging, agglomerative) ryvästyksessä kukin hahmovektori muodostaa alkutilanteessa oman ryhmänsä ja ryhmiä yhdistetään asteittain kooten samanlaisia tapauksia yhdistetyksi ryppääksi [Sal89]. Ensin siis kaksi samanlaisinta dokumenttia yhdistetään yhdeksi luokaksi. Kun suurempia ryppäitä yhdistetään, täytyy miettiä miten ryppäiden samankaltaisuus määritellään. Samankaltaisuus voidaan määritellä kahden eri ryppään kaikkein samankaltaisempien tapausten perusteella, kaikkein vähiten samankaltaisten tapausten perusteella tai ryhmän keskiarvojen samankaltaisuuden perusteella. Hajottavassa tai erottavassa (splitting, divisive) ryvästämisessä kaikki hahmovektorit ovat aluksi yhtenä ryhmänä, jolloin kokonaisuus muodostaa yhden ryväksen. Tätä jaetaan pienemmiksi palasiksi.

Ryvästettyihin tiedostoihin kohdistuvat haut mahdollistavat usein nopeita hakutoimintoja [Sal89]. Ryväshaut perustuvat yleensä ryppäiden keskuksiin (cluster centroid). Ryppään keskus on tyypillisesti keskiarvovektori, joka lasketaan yhden ryppään dokumenttien vektoreista. Ryväshaussa etsitään hakua lähin keskus ja kohdistetaan haku tarkemmin siihen ryppääseen. Ryväshaussa on kaksi eri strategiaa: ylhäältä alas tai alhaalta ylös kohdistuvat haut. Ylhäältä alas haussa hakua verrataan ensin korkeimman tason keskukseen ja siirrytään sitten tutkimaan parhaiten osuvaa ryvästä. Alhaalta ylös haussa vain alimman tason keskukset tallennetaan ja ylemmän tason ryväshierarkiaa ei oteta huomioon. Alimman tason keskuksia saattaa olla paljon, jolloin on hyödyllistä tehdä indeksointi keskustermeihin.

Dynaaminen ryhmittely suoritetaan jakamalla tiedostot aluksi täysin summittaisesti joihinkin ryhmiin, jonka jälkeen yksittäisiä hahmovektoreita aletaan siirtää ryhmästä toiseen sen perusteella, minkä ryhmän keskipistettä lähinnä ne ovat [HKt93].

4.3 Usean informaatiolähteen yhdistämistransformaatiot

Heterogeenisten tietolähteiden yhdistämisessä törmätään moniin ongelmiin. Tämän luvun ongelma ei tosin enää ole tiedon muodon tai rakenteisuuden erilaisuus – jos yhdistettävät tiedot eivät ole rakenteeltaan yhdenmukaisia, niille tehdään formaattitransformaatio, eikä ongelma kulkeudu yhdistämisvaiheeseen. Tietolähteistä on myöskin jo poimittu se tieto, joka käyttäjää kiinnostaa ja ainoana ongelmana on enää näiden eri tietolähteistä löydettyjen, kiinnostavien tietojen yhdistäminen. Tässä kappaleessa käsitellään tietojen integrointia, ensin siihen liittyviä ongelmia ja sen jälkeen yhdistämistoimintoja.

4.3.1 Ongelmia

Yhdistämisvaiheessa eri lähteistä voi löytyä yhteensopivuusongelmia. Lähteet esittävät tiedon usein omilla termeillään, jolloin tiedon vertailu vaikeutuu. Vaikka tietosisältö olisikin sama, ei vertailua voi perustaa suoraan sanasta sanaan vertailuun, vaan samankaltaisuuden tutkimiseen. Yksinkertaisia tapauksia ovat sellaiset terminologiaerot, kuten ATT, AT&T ja American Telephone and Telegraph, jotka voidaan kerätä taulukkoon. Lähteiden tarkkuustaso (jota voisi kutsua semanttiseksi granulariteetiksi) saattaa vaihdella (Helsingissä-Kalliossa). Tiedon epävarmuus tai epätarkkuus voivat tulla esiin näkökantaeroina (ainakin viisi – tasan viisi). Tiedot saattavat jopa olla ristiriitaisia.

Tietolähteet saattavat tarkentaa samaa tapahtumaa eri kannoilta eli on useita eitäydellisiä lähteitä, jotka pitää osata kytkeä toisiinsa. Toisaalta voi olla useampia samanlaisia tietoja, jotka kuitenkin kertovat eri tapahtumista. Lisäksi on käsiteltävä myös samaa mieltä olevat lähteet: otetaanko molemmat mukaan vai vain parempi ja miten paremmuus ratkaistaan. Ratkaisuun voi vaikuttaa mediaattorisäännöillä, kuten *"Lähde A on parempi lähde kuin lähde B"* tai *"uudempi tieto on parempaa"*. Kuitenkaan aina ei ole välineitä määritellä parempaa tietoa, jolloin käyttäjälle voidaan esittää molemmat lähteet [MR95, Wie99, FPNB99, LCN$^+$99].

Myös alkuperäisen lähteen esittäminen saattaa olla ongelmallista. Joskus sillä, mistä tieto on peräisin, ei ole mitää väliä, mutta yleensä tiedon alkuperän jäljitettävyys on itsestään selvä vaatimus. Alkuperäinen lähde voidaan esittää esimerkiksi linkittämällä tieto alkuperäiseen lähteeseen tai liittämällä alkuperäislähde tietoon metadatana.

4.3.2 Yhdistäminen

Perinteisten tietokantojen rakenne ja yhteydet ovat selkeitä. Tietokannat tarjoavat kuitenkin vain vähän tukea semanttisille yhteyksille.

Terminologian yhdistäminen voidaan taata kuvaustaulukoin (mapping table). Ariadnessa [KM98] yksi lähde valitaan ensisijaiseksi ja muut suhteutetaan siihen. Esimerkiksi jos toisessa lähteessä lukee Art's Deli, voidaan samaa paikkaa toisessa lähteessä kutsua nimellä Art's Delicatesse. Kuvaustaulukko määrittää tietojen vastaavuudet eri tietolähteissä.

Ontologia tarkoittaa tietorakennetta, joka selittää tietolähteen termien luonteen, piirteet ja suhteet [BGL+99, ZGG+99]. Jokaisesta lähteestä luodaan paikallinen ontologiakuvaus, joka kuvaa käsitteiden, sanojen ja sivujen suhteet. Ontologiaperustaisia työkaluja voidaan käyttää lähteiden yhdistämiseen. Eri ontologioiden väliset suhteet voidaan määritellä algebrallisesti [MWJ99]. Valittuihin ontologiaosuuksiin muodostetaan operaattorit, kuten *extract*, *union*, *intersection* ja *difference*. Aiemmin mainitut mediaattorit ja sovittimet ovat myös ratkaisuja yhdistämisongelmaan. Sovittimet ovat tietolähdekohtaisia välikappaleita, jotka tekevät kyselyitä lähteeseen ja mediaattorit tekevät varsinaisen tietojen yhdistämisen [GMHI+95, AK97]. Mediaattorissa määritellään miten toimitaan tässä kappaleessa esitettyjen tiedon yhdistämisongelmien kanssa. Esimerkiksi semanttisen granulariteetin erilaisuus voidaan esittää hierarkkisina moduleina mediaattorissa (postikoodi ja sen yhteys kaupunkeihin) [Wie99].

Käyttäjän iloksi on rakennettu automaattisia vertailupalveluita internetin kauppapaikkoihin. Vertailupalvelut yhdistävät monien lähteiden tietoja ja tarjoavat tiedot käyttäjälle vertailtaviksi. Suomenkielisessä palvelussa www.vertaa.fi vertaillaan eri tuotteita, hintoja ja tuotteiden tarjoajia keskenään. Palvelun kautta voi etsiä kirjoja tai CD-levyjä ja tuotteille haetaan hankintapaikat, myyntihinnat, postituskulut ja toimitusajat. Tiedot haetaan maailmanlaajuisesti kauppojen verkkosivuilta. Hakutuloksia voi järjestää tekijän, tuotteen nimen, numeron tai kokonaishinnan mukaan. Esimerkiksi kirjahaku tekijällä *"Fromm E"* palautti 18 eri kirjaa ja näistä valitulle kirjalle *"To have or to be"* löytyi kahdeksan ostopaikkaa.

Valmiit ohjelmistopaketitkin ovat alkaneet tarjoamaan enemmän eri tietolähteiden yhdistämistä ja ohjelmien välistä kommunikointia. Esimerkiksi Microsoftin uusin toimisto-ohjelmapaketti, Office XP tarjoaa entistä enemmän tietolähteiden yhdistämistä. Paketissa on tunnistuslogiikka, joka tunnistaa dokumentista kirjoitetun tekstin ja ehdottaa siihen liittyviä toimintoja käyttäjälle. Tietolähteinä voivat olla organisaation tietokanta, Internet-sivut tai jossain muussa ohjelmistossa oleva tieto. Jos käyttäjä esimerkiksi kirjoittaa Wordissa tietyn henkilön nimen, voisi ohjelma ehdottaa henkilön osoitteen hakemista tietokannasta tai sähköpostin lähettämistä [ITv01].

Varsinaiseen mediakonvergenssiin eli useamman eri kommunikointimedian yhdistämiseen on pyritty *Unified Messaging*-tuotteessa [5]. Unified messaging tarkoittaa yhtä

[5]Tutoriaali unified messagingista löytyy esimerkiksi IEC:ltä osoitteesta http://www.iec.org/tutorials/unified_mess/

käyttöliittymää ääni-, faksi- ja sähköpostiviestien lähettämiseen ja vastaanottamiseen. Käyttöliittymä voi olla tavallinen puhelin, matkapuhelin, PC tai PC internet-yhteydellä, jota kautta vastaanotetaan kaikenlaiset viestit: sähköpostit, tekstiviestit, puhelinvastaaja viestit ja faksit, eikä tarvitse erikseen tarkistaa kutakin viestivälinettä.

4.3.3 Tiedon vertailu

Usein tärkeäksi tiedon yhdistämisessä katsotaan tietojen vertailu ja erilaisuuksien esittäminen. Tiedonhaussa tehokas ja mahdollisimman tarkka vertailu tiedonhakuongelman ja tietovaraston kesken on avainasemassa. Käyttäjän esittämää tiedonhakuongelmaa verrataan tietovaraston dokumentteihin ja sieltä täytyy löytää ongelmaan kaikkein parhaiten vastaava dokumentti tai dokumentit. Kahden dokumentin sanotaan olevan samankaltaisia, jos niiden sanafrekvenssit korreloivat keskenään [GMGS96]. Vertailu voi perustua metatietoihin (dSCAM, [GMGS96]), joihin voidaan valita harvinaisia sanoja tai käyttää todennäköisyyksiä. Metatietoihin perustuva vertailu on kuitenkin hyvin hataraa.

Tarkempi vertailu saadaan aikaiseksi muodostamalla yhdistettävistä dokumenteista termivektorit. Saman dokumentin sisällä tapahtuvasta vertailusta termivektoreilla puhuttiin jo segmentoinnin yhteydessä ja eri tietolähteiden vertailu on periaatteessa samanlaista. Laskemista nopeuttamaan ja termivektoria yksinkertaistamaan voidaan vektoriin poimia vain merkitykselliset sanat. Näitä voivat olla esimerkiksi tietyn sanaluokan sanat. Sanan eri taivutusmuotojen esiintymistä voidaan myös poimia vain yksi muoto palauttamalla ne perusmuotoon. Yleinen termivektorien vertailuun käytetty mitta on kosinimitta.

Termivektoreja voidaan käyttää esimerkiksi uuden uutisen löytämiseen uutisvirrasta. Aihetta on tutkittu muun muassa *Topic Detection and Tracking* (TDT) -projektissa [ACD$^+$98]. Uutisvirran tarkkailu voidaan jakaa kahteen osaongelmaan: uuden tapahtuman etsimiseen ja tapahtumien seuraamiseen. Näissä kutakin uutta saapunutta uutista on verrattava olemassa olevaan uutistietokantaan ja päätettävä, onko uutinen mahdollisesti aivan uusi juttu vai kuuluuko se johonkin jo ilmestyneeseen uutiseen. Jos uutinen liittyy aimmin ilmestyneeseen uutiseen, on selvitettävä sisältääkö se jotain lisäinformaatiota kyseisestä uutisesta.

Helsingin yliopiston tietojenkäsittelytieteen laitoksen Tytti-projekti (Tietotyöläisen työasema), jota on esitelty esimerkkitapauksena luvussa 6.1, sisältää uutisvirran luokittelijan ja uusien tapahtumien etsimisen. Järjestelmä käyttää termivektoreita uutistapahtumien vertaamiseen keskenään.

MESSAGE: ID	TST–REU–001
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 3, 1996 11:30
PRIMSOURCE: SOURCE	
INCIDENT DATE	March 3, 1996
INCIDENT: LOCATION	Jerusalem
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	''killed: 18'' ''wounded: 10''
PERP: ORGANIZATION ID	

1. artikkeli

MESSAGE: ID	TST–REU–003
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996 14:20
PRIMSOURCE: SOURCE	
INCIDENT DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	''killed: at least 13'' ''wounded: more than 100''
PERP: ORGANIZATION ID	''Hamas''

3. artikkeli

MESSAGE: ID	TST–REU–002
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996 07:20
PRIMSOURCE: SOURCE	Israel Radio
INCIDENT DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	''killed: at least 10'' ''wounded: 30''
PERP: ORGANIZATION ID	

2. artikkeli

MESSAGE: ID	TST–REU–004
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996 14:30
PRIMSOURCE: SOURCE	
INCIDENT DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	''killed: at least 12'' ''wounded: 105''
PERP: ORGANIZATION ID	''Hamas''

4. artikkeli

> Reuters reported that 18 people were killed in a Jerusalem bombing Sunday. The next day, a bomb in Tel Aviv killed at least 10 people and wounded 30 according to Israel radio. Reuters reported that at least 12 . people were killed and 105 wounded. Later the same day, Reuters re–ported that hte radical Muslim group Hamas had claimed responsibility for the act.

Yhteenveto neljästä artikkelista

Kuva 10: Mallipohjiin kerätyt uutisartikkelit ja niistä automaattisesti generoitu tiivistelmä [Rad99].

Automaattinen yhteenvetojärjestelmä SUMMONS (SUMMarizing Online NewS articles) [MR95, Rad99] on suunniteltu useamman uutislähteen tiivistelmän tekemiseen. Se keskittyy etsimään uutisista asiat, joista useat uutislähteet ovat yksimielisiä, sekä ristiriitaisuudet. Järjestelmä sisältää aktiivisten uutisten tietokannan, historiatapahtumat sekä ontologiakuvaukset esimerkiksi maantieteellisille sijainneille. Syötedata kerätään mallipohjiin eristämissääntöjen avulla. Eristämisongelma on SUMMONS-projektissa jätetty muiden huoleksi ja projekti keskittyy vastaanotettujen faktojen jatkokäsittelyyn, tietojen vertailuun ja näiden perusteella tiivistelmän generointiin. Kuvassa 10 on esimerkki SUMMONS-järjestelmän tekemästä yhteenvedosta. Neljän toisiinsa liittyvän uutisartikkelin tiedot on poimittu mallipohjiin ja näistä mallipohjista on generoitu tiivistelmä. Järjestelmä vertaa mallipohjia ja selvittää muun muassa näkökulman muuttumisen, ristiriitaisuudet ja tiedon lisääntymisen ja näiden perusteella muodostetaan tiivistelmä.

4.3.4 Koostaminen

Tekstien koostamisella (document assembly) tarkoitetaan uuden dokumentin tietokoneavusteista muodostamista useita olemassaolevia tekstilähteitä hyväksi käyttäen. Koostamisprosessi sisältää relevanttien tekstikatkelmien etsimisen, katkelmien muokkaamisen ja järjestämisen sekä muokattujen katkelmien yhteenliittämisen uudeksi dokumentiksi. Käyttäjän tai ohjelman valinnat ohjaavat automaattisia tiedonhaku- ja koostamisvaiheita, joissa palaset poimitaan kokonaisuuksina ja liitetään toisiinsa. Koostaminen sopii esimerkiksi käyttöohjeisiin (laitteilla yhteiset ohjeet sekä versiokohtaiset ohjeet) tai räätälöityihin asiakaskirjeisiin. Kirjeen teksti voi olla erilainen riippuen asiakkaan iästä, sukupuolesta tai ammattitaustasta ja kirjeeseen liitetyt WWW-linkit voivat riippua esimerkiksi käyttäjän äidinkielestä [Leh01, Kno98, Pri99].

Dokumenttikomponenttien poimimista varten lähteet täytyy segmentoida sopivan kokoisiksi lähdetiedoiksi. Koostamiseen sopivat yleensä kokonaiset luvut, aliluvut tai asiakokonaisuudet. Segmentoidut dokumentit säilytetään tietovarastossa ja poimitaan sieltä aina tarvittaessa julkaistavaksi. Koostettavan tiedon on yleensä oltava rakenteellisesti homogeenista. Koostamisjärjestelmä voi sisältää jäsennys- ja kielenkäsittelyjärjestelmän, jolloin voidaan koostaa myös rakenteellisesti heterogeenista dataa.

Koostamisprosessi on interaktiivinen: käyttäjä tekee valintoja, joiden perusteella palaset valitaan. Valinnat voivat määräytyä myös käyttäjäprofiilin tai ohjelmatoimintojen perusteella.

Relevantit tekstisegmentit etsitään tietovarastosta, kootaan ja järjestetään automaattisesti. Käyttäjällä on yleensä mahdollisuus vielä muokata syntynyttä dokumenttia. Uuden dokumentin luominen koostamalla on uudelleenkäyttöä, joka säästää aikaa.

Koostamiseksi voidaan ajatella myös tietopalasten automatisoimatonta yhdistämistä, jolloin käyttäjä saa poimia käyttöönsä tietoa eri lähteistä. Tähän on kehitetty esimerkiksi VIKI, joka toimii käyttäjän työpöytänä, johon hän voi liittää linkkejä eri lähteisiin ja toisaalta liittää joukkoon omia tietopalasia ja rakentaa hierarkioita ryhmille [SML99].

5 Käyttäjän näkökulma

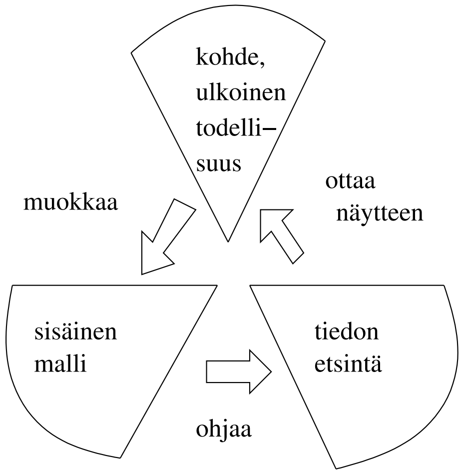
Tietoa on tähän asti tutkielmassa käsitelty lähinnä teknisenä elementtinä, on mietitty miten tietoa voidaan jakaa osiin, yhdistää eri osasia tai poimia tietoa tekstin seasta. Todellisuudessa tieto kuitenkin sisältää merkityksen, joka on tarkoitus välittää käyttäjälle. Tässä luvussa tarkastellaankin tietoa käyttäjän näkökulmasta, tiedon kognitiivisia aspekteja ja semanttista puolta. Yhtä lailla kuin käsiteltävät tietolähteet ja mediat ovat heterogeenisiä, myös järjestelmiä käyttävät käyttäjät ovat erilaisia vaihtelevin tarpein, silti ajatteluun ja havainnointiin liittyy myös yleisiä lainalaisuuksia.

Käyttäjää voidaan palvella paremmin ymmärtämällä ihmisen kognitiivista tiedon käsittelyä ja ottamalla se huomioon myös teknisessä toteutuksessa. Ajattelu toimii kognitiivisen taloudellisuuden periaatteella, toisin sanoen ihminen pyrkii minimoimaan ajatteluun käytetyn ajan ja vaivannäön [EK97]. Mitä enemmän käyttäjä vapautetaan ajattelun ponnistelusta ja annetaan käyttäjän keskittyä käsiteltävään työtehtävään, sitä enemmän hän sitä arvostaa. Käyttäjät tekevät usein mieluummin ylimääräisiä fyysisiä kokeiluja kuin mentaalista suunnittelua, tämä tulee hyvin esille esimerkiksi internetin käytössä: WWW-sivuja silmäillään, ei lueta ja yritetään mieluummin löytää etsittävä tieto yrityksen ja erehdyksen kautta seuraamalla linkkejä kuin lukemalla sivun sisältöä tarkasti.

Tutkielmassa käsitellyt tiedon transformaatiot vähentävät käyttäjän kognitiivista taakkaa. Transformaatioiden avulla käyttäjälle tarjotaan relevantti tieto automaattisesti muun tiedon seasta poimittuna. Transformaatiot helpottavat tiedon käsittelyä, auttavat tiedon jäsentämisessä ja vähentävät ihmisen muistin kuormitusta.

Sisäiset mallit

Ihmisen vuorovaikutuksellisen toiminnan perustana ovat sisäiset mallit (mental models). Ne auttavat ihmistä toimimaan ympäristön kanssa. Ne ovat mielen malleja ihmisestä itsestään, muista, ympäristöstä, asioista ja prosesseista, joiden kanssa ihminen on tekemisissä. Mallit ovat luonteeltaan kuvailevia, ne auttavat henkilöä ymmärtämään, toimimaan ja odottamaan asioita ja ne kehittyvät kokemuksen, harjoittelun tai ohjeiden kautta. Hyvä sisäinen malli helpottaa ymmärtämään ja ennustamaan toimintojemme vaikutuksia [SBCX97]. Kun käyttäjä on vuorovaikutuksessa tietojärjestelmän kanssa, hän päättelee kuinka järjestelmä toimii ja kehittää odotuksia, jotka vaikuttavat taas käyttäjän asioidessa järjestelmän kanssa seuraavan kerran. Yhden järjestelmän kanssa kehitetty malli siirretään helposti toiseen järjestelmään tai tehtävään. Mallit ovat dynaamisia ja niitä muutetaan tarpeen tullen ja sama kokemus voi saada aikaan erilaisen sisäisen mallin. Sisäiset mallit ohjaavat havaintotiedon etsintää ennakointien ja oletusten mukaisesti. Saadut havaintokokemukset puolestaan muokkaavat sisäisiä malleja, jotka jälleen ohjaavat havaintotiedon etsintää. Tämä kehämäinen havaintojen ja sisäisten mallien suhde on esitetty kuvassa 11.



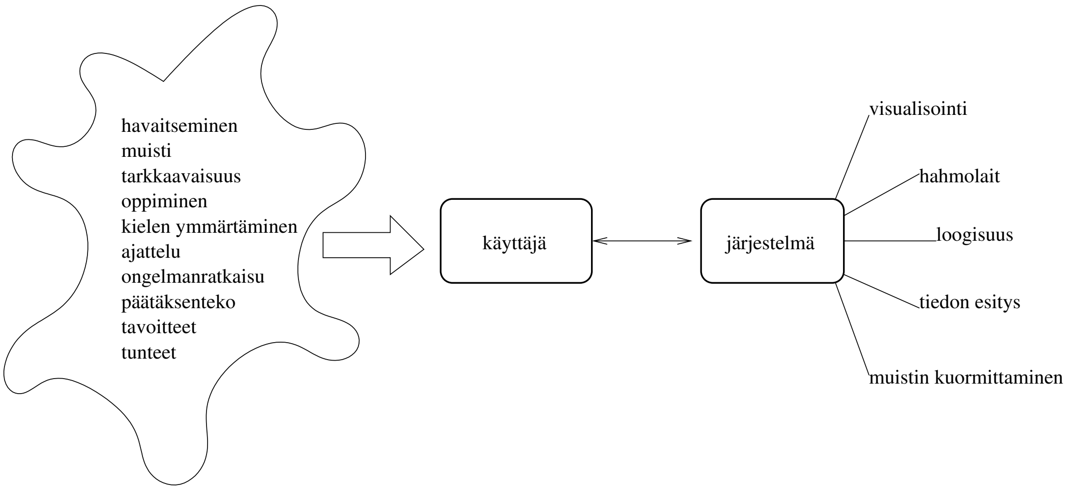
Kuva 11: Havaintokehä, jossa sisäiset mallit ohjaavat tiedon etsintää ja havainnot tiedosta jälleen muokkaavat sisäisiä malleja [Nei76].

Yleistävät, karkeat sisäiset mallit jaetaan usein noviiseihin ja ekspertteihin. Noviisin sisäinen malli kuvaa asiaan ensimmäisiä kertoja tutustuvan henkilön käsitystä asiasta ja ekspertin malli miltä asia näyttää harjoittelun tuloksena. Esimerkiksi autolla ajamista harjoittelevan sisäinen malli ajamisesta sisältää ohjeita vaihteiden vaihtamisesta ja kytkimen käyttöön, kun kokeneemman ajajan ajamiseen liittyvät ajatukset pyörivät yleensä reittivalinnassa. Toinen mahdollisuus on jakaa ne konseptuaalisiin (kuinka joku toimii) ja toiminnallisiin (miten käyttää jotakin; toimet, jotka on tehtävä tehtävän loppuunsaattamiseksi) malleihin. Koneista ja järjestelmistä puhuttaessa erotetaan joskus järjestelmä, järjestelmän konseptuaalinen malli sekä käyttäjän sisäinen malli järjestelmästä. Tällöin ensimmäinen tarkoittaa todellista, fyysistä tietokonejärjestelmää, konseptuaalinen malli tarkoittaa suunnittelijan kuvausta järjestelmän toiminnasta ja sisäinen malli sitä, miten käyttäjä ymmärtää järjestelmän (Norman 1983).

Sisäisten mallien hyödyntäminen järjestelmäsuunnittelussa olisi hyödyllistä, mutta niihin liittyy paljon avoimia kysymyksiä, kuten miten tiedetäään minkälainen sisäinen malli käyttäjällä on tai millaisen hän kehittää? Joka tapauksessa parantaaksemme ihminen–tietokone vuorovaikutuksen laatua, täytyy käyttäjää auttaa kehittämään ”oikea” konseptuaalinen malli niin nopeasti ja tarkasti kuin mahdollista. Positiivista vaikutusta tuo johdonmukaisesti toimiva järjestelmä, joka mahdollisesti vahvistaa olemassa olevia malleja ja vähentää kognitiivista taakkaa [THH95].

Kognitiivinen tiedonkäsittely

Käyttäjän näkökulmassa voidaan ottaa huomioon ihmisen kognitiivisia prosesseja laajemminkin. Tällöin ollaan kiinnostuneita siitä, mitä ihmismielessä tapahtuu. Ihmisajattelun voidaan ajatella toimivan kuten tietojärjestelmät, ihmisen kognitiivisen järjestelmän syötteenä on informaatio ja tulosteena toiminta tai muu informaatio. Toisaalta ajattelu myös ohjaa sitä, mitä havaitaan: omat mielenkiinnon kohteet erotetaan ärsykevirrasta automaattisesti. Kognitiivisia prosesseja ovat ymmärtäminen, muistaminen, järkeily, tarkkaavaisuus, havaitseminen, oppiminen, ongelman ratkaisu ja kielen ymmärtäminen. Lyhyesti sanottuna kaikki se, mikä liittyy siihen, miten tiedämme ja ymmärrämme. Kuvassa 12 on esitetty käyttäjän ja järjestelmän välistä interaktiota.



Kuva 12: Käyttäjän kokemukset vaikuttavat siihen, kuinka hän on yhteydessä järjestelmään.

Kognitiivinen tiedonkäsittely voidaan ottaa huomioon järjestelmäsuunnittelussa. Esimerkiksi ihmismuisti on hyvin rajallinen, työmuistin tietokapasiteetiksi on määritelty 5-9 yksikköä (yksikkö tosin voi olla vaihtelevan kokoinen). Muistin turhaa kuormitusta voi välttää esimerkiksi näyttämällä selkeästi valinnat, jotka käyttäjä on jo tehnyt. Havaitsemiseen liittyy lainalaisuuksia, joilla voidaan helpottaa hahmon erottamista taustasta. Tällaisia hahmolakeja ovat objektien läheisyys, samankaltaisuus, hyvä jatko, tuttu muoto ja sulkeutuneisuus. Nämä voidaan ottaa huomioon ainakin käyttöliittymäsuunnittelussa. Esimerkiksi hyvä jatko -lain mukaan kaksi risteävää viivaa mielletään mieluummin kahdeksi suoraksi viivaksi ristikkäin kuin kaksi 90 asteen kulmaa kulmikkain. Tuttu muoto -laki assosioi oikealle osoittavan kolmion tarkoittamaan *"play"*. Tarkkaavaisuutta on helppo ohjata korostamalla olennaisia asioita. Tarkkaavaisuuden suuntautuminen tapahtuu joko tahallisesti tai tahattomasti ja molempiin voidaan vaikuttaa käyttöliittymäsuunnittelulla. Tahallisesti ohjattua tarkkaavaisuutta auttaa hyvin järjestetty ja selkeästi esitetty käyttöliittymä. On vaikea olla reagoimatta äkilliseen ääneen, valon välähdykseen, muusta ympäristöstä poikkeavaan ärsekkeeseen tai nopeaan liikkeeseen, mutta tahattomaan tarkkaavaisuuteen ei käyttöliittymässä tarvitse vaikuttaa ihan näin radikaalein menetelmin. Järjestelmän kieli tulee olla ymmärrettävää, tekstin koherenttius on verrannollinen lukijan kykyyn ymmärtää ja muistaa tekstiä. Eri tasoiset käyttäjät voidaan ottaa huomioon kielenkäytössä, esimerkiksi aloittelijalle tekstin *"system time out"* tilalla voidaan käyttää selittävämpää tekstiä.

Tekniset elementit käyttäjän kannalta

Kognitiivisten aspektien lisäksi käyttäjän näkökulmassa tulee ottaa huomioon ergonomia, fyysinen ympäristö ja tekniset puitteet. Teknisiä elementtejä ovat järjestelmän luotettavuus ja turvallisuus, verkkoyhteyden ja koneen nopeuden ottaminen huomioon ja järjestelmän mukautuminen erilaisten käyttäjien mukaisesti. Tietojen turvallisuus saavutetaan salaamalla (encryptation), mutta käyttäjän näkökulmasta tärkeää on myös ilmaista turvallisuus käyttäjälle hyvin. Käyttäjän on ymmärrettävä ja luotettava, että yhteys on turvallinen. Yksityisyyteen voidaan turvallisuusaspektien lisäksi liittää toive mahdollisuudesta käyttää järjestelmää anonyymisti. Käyttäjien verkkoyhteyksien tehokkuus on otettava huomioon ja tarjottava sisältöä sen mukaisesti. Hitaalla modeemilla toimivalle käyttäjälle on tarjottava videokuvien ja korkearesoluutioisten kuvien tilalle vastaava tieto, mutta vähemmän verkkoa kuormittavassa muodossa.

Järjestelmän mukautuminen erilaisten käyttäjien mukaan on viime vuosina ollut hyvin suosittua. Mukautuminen voidaan toteuttaa personoinnilla, jolloin käyttäjä voi muuttaa sisältöä ja esitysmuotoa haluamakseen määrittämällä parametreja ja asetuksia toiveidensa mukaisesti. Personointia voidaan kutsua myös profiloinniksi, jolloin profiiliin määritellään toivotut ominaisuudet. Profilointi määräytyy usein myös käyttäjäryhmittäin. Järjestelmä voi myös adaptoitua automaattisesti käyttäjästä tekemiensä oletusten mukaisesti, ottamalla huomioon käyttäjän päämäärän, kiinnostuksen, kokemuksen ja kyvyt. Esimerkiksi ensikertalaisen käyttöliittymä voi olla yksinkertainen, riisuttu versio, kun taas kokeneemmalla käyttäjällä voi olla enemmän toimintoja tarjolla [FKS97, KS99, Fra99].

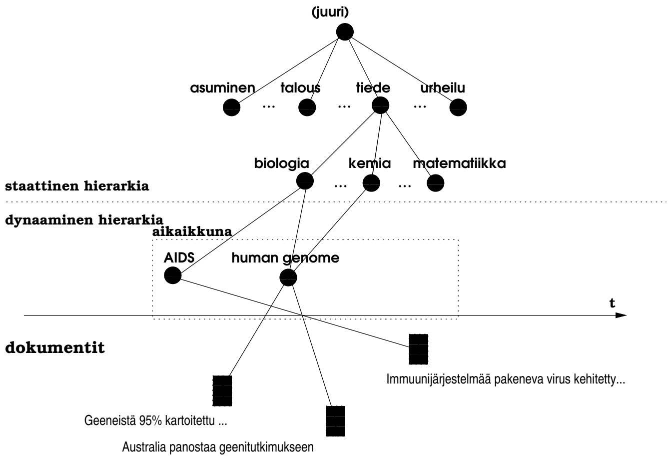
6 Esimerkkitapaukset

Tutkielmassa esitetyt sisällönhallintaan liittyvät muunnostoiminnot on löydetty esimerkkitapauksia analysoimalla. Tässä luvussa esitetään muutama esimerkkitapaus kokonaisuudessaan sekä erotetaan näistä löydetyt transformaatiot. Esimerkeissä näkyy selkeästi, että transformaatiot ovat harvoin ”puhtaita”, vain esimerkiksi puhtaasti tiedon eristämistä, vaan eri transformaatiot kietoutuvat toisiinsa ja tiedolle joudutaan yleensä tekemään useita eri transformaatioita. Esimerkeissä käytetään lyhenteitä:
FT = formaattitransformaatio
PT = tiedonpoimimistransformaatio
YT = yhdistämistransformaatio

6.1 Tytti: luokittelu ja suodatus

Helsingin yliopiston tietojenkäsittelytieteen laitoksen Tietotyöläisen työasema (Tytti) -projekti sisältää tekstinluokittelija- ja suodatuskomponentin [Mak00]. Tietotyöläisen käyttöön tulee heterogeenista materiaalia sekä tietovirtana ulkopuolelta että hänen itsensä tuottamana. Ulkoa tuleva tietovirta luokitellaan, päällekkäisyydet suodatetaan ja käyttäjälle näytetään häntä kiinnostavat uutiset. Osa luokista on staattisia, etukäteen luotuja kiinteitä luokkia, jotka perustuvat johonkin standardiluokitukseen tai aihepiirin ylläpitäjä on luonut ne. Järjestelmässä on myös dynaamisia luokkia, jotka syntyvät tarpeen mukaan, kun joku olemassa oleva luokka kasvaa liian suureksi. Kiinteät luokat määritellään oppimalla eli luokittelija opetetaan esimerkkidokumenttien avulla.

Opetukseen käytettävät dokumentit esikäsitellään muuntamalla sanat yhdenmukaisempaan muotoon sekä poistamalla liian yleiset, ei-kuvaavat sanat kuten ja, että, se jne. Sanojen yhdenmukaistaminen voidaan tehdä palauttamalla sanat perusmuotoon tai poistamalla sanasta taivutuspäätteet. Perusmuotoon palauttaminen on monimutkaista, koska usein on tarpeen tietää sanan sanaluokka ja ympäristö, esimerkiksi sana *"palaa"* voi olla palaa-verbin 3. muoto, palata-verbin imperatiivimuoto tai pala-substantiivin partitiivimuoto. Suomenkielen perusmuotoon palauttamista varten on olemassa ohjelma Fintwol, joka on Lingsoftin[6] kaupallinen tuote. Sanojen taivutuspäätteiden poistaminen on toteutukseltaan yksinkertaisempi, mutta se ei ole yhtä luotettava sanojen vertailussa. Tytti-projektissa taivutuspäätteiden poistamiseen käytetään Porterin algoritmiä, mutta myös perusmuotoon palauttamista tarkastellaan. Liian yleiset sanat hylätään ja niistä muodostetaan poistolista (stoplist), jonka sisältämät sanat hylätään kaikista luokiteltavista dokumenteista. Hyväksytyistä sanoista valitaan tietty määrä termejä (esimerkiksi 3000), joiden avulla muodostetaan kullekin dokumentille ominainen termivektori. Tytti-projektissa hyväksyttävät sanat valitaan information gain -menetelmällä. Luokan kuvaus muodostuu siihen kuuluvien dokumenttien termivektoreista. Luokilla voi olla aliluokkia, jotka käsitellään samoin kuin pääluokat. Esimerkki luokkahierarkiasta on esitetty kuvassa 13.



Kuva 13: Järjestelmään suunniteltu luokkahierarkia [Mak00].

Kun järjestelmä on opetettu, se luokittelee vastaanottamansa dokumentit tähän luokkahierarkiaan. Kun uusi dokumentti saapuu, se esikäsitellään kuten dokumentit opettamisvaiheessa eli palautetaan perusmuotoon tai poistetaan sanapäätteet sekä poistetaan sulkulistassa esiintyvät sanat. Dokumentille luodaan termivektori eli lasketaan kunkin termin esiintymiskerrat. Tätä termivektoria verrataan luokkien dokumenttien termivektoreihin ja dokumentti liitetään luokkaan, jonka kanssa sen termivektori on yh-

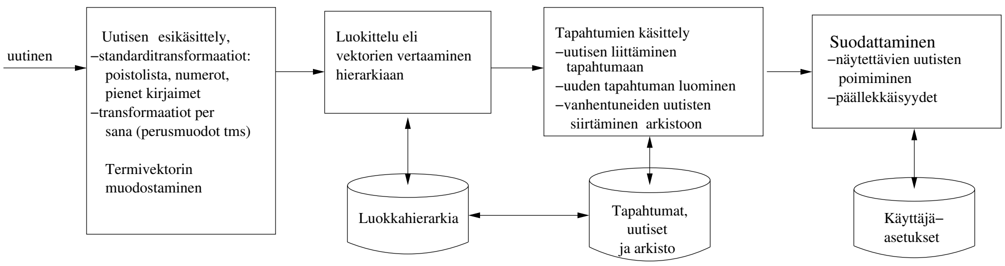
[6]http://www.lingsoft.fi/doc/fintwol/

teensopivin, esimerkiksi laskemalla vektorien välinen kulma, etäisyys tms. Luokitteluun käytetään Support Vector Machine -menetelmää [SSW$^+$98], joka hyödyntää vektorien välisiä sisätuloja.

Järjestelmään luodaan staattisten luokkien lisäksi dynaamisia luokkia tarpeen mukaan. Dynaamiset aliluokat ovat tapahtumia, jotka määritellään ajan, paikan ja erisnimien (henkilöiden, organisaation, tuotteiden) mukaan. Samasta tapahtumasta ilmestyy usein monia uutisia asian edetessä tai eri näkökulmilta, ja uutinen täytyy tunnistaa kuuluvaksi samaan tapahtumaan. Tapahtuman tunnistaminen tehdään vertaamalla uutisten aika- ja paikkatietoja sekä erisnimiä. Uutinen liitetään aina joko olemassa olevaan tapahtumaan tai sille luodaan uusi tapahtuma.

Järjestelmä suodattaa näytettäviä uutisia käyttäjän asetusten mukaisesti. Käyttäjä voi määritellä aihealueet (eli luokat), jotka häntä kiinnostavat ja muiden luokkien uutiset suodatetaan pois. Hän voi myös haluta vain uudet uutiset, jolloin tulee pitää kirjaa siitä mitkä uutiset käyttäjä on jo lukenut ja suodattaa nämä pois. Myös uutisten päällekkäisyydet voidaan suodattaa. Järjestelmällä on tietty ajanjakso, joka määrää kauanko uutinen säilyy ajankohtaisena. Vanhentunut uutinen siirtyy arkistoon.

Kuvassa 14 on esitetty järjestelmän toiminnot, kun järjestelmään saapuu uusi uutinen. Saapuva uutinen muunnetaan muotoon, jossa sen termejä voidaan vertailla muihin uutisiin. Saapuvalle uutiselle täytyy löytää luokka, johon se kuuluu. Tätä varten uutinen täytyy muuntaa riisuttuun muotoon. Uutista verrataan sen luokan muihin tapahtumiin ja liitetään kuuluvaksi johonkin olemassa olevaan tapahtumaan tai sille luodaan uusi tapahtuma. Käyttäjän asetuksista riippuen uutinen näytetään käyttäjälle tai suodatetaan pois käyttäjän näkymästä.



Kuva 14: Luokittelijan ja suodatusjärjestelmän toiminnot.

Järjestelmän transformaatiot:

(TP) Dokumentti riisutaan eli muunnetaan perusmuotoon tai poistetaan taivutuspäätteet ja poimitaan merkitykselliset termit. Rakenteisesta tekstistä täytyy poistaa tunnisteet (tags), myös osaelementtien valinta on mahdollista

(FT) Uutisen muuttaminen termivektoriksi.

(TP) Ajan, paikan ja henkilöiden poimiminen uutisesta.

(YT) Uutisen liittäminen johonkin tapahtumaan tai määrittely uudeksi tapahtumaksi.

(TP) Käyttöliittymän suodatus: minkä luokan uutiset näytetään, näytetäänkö päällekkäisyydet ja suodatetaanko jo nähdyt uutiset.

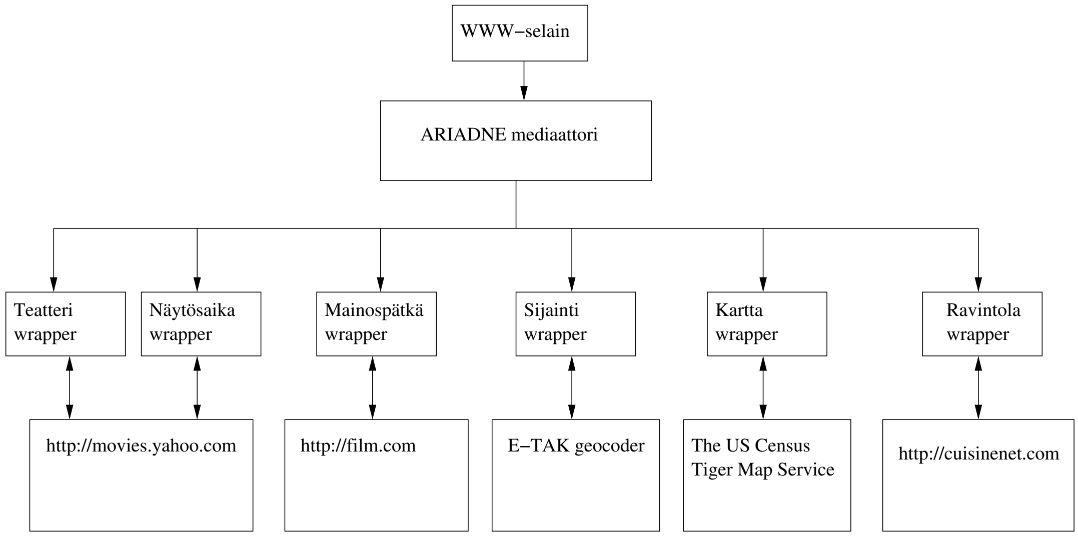
6.2 Ariadne, usean WWW-tietolähteen yhdistäminen

Ariadne on Etelä-Kalifornian yliopiston projekti WWW-perustaisen tiedon yhdistämiseen. Järjestelmä kerää ja yhdistää tietoa puolirakenteisilta WWW-sivuilta. Järjestelmän ylläpitäjä määrittelee tietolähteiden WWW-osoitteet ja rakentaa kuhunkin lähteeseen sovittimen. Useasta WWW-lähteestä rakennetaan yhdistetty näkymä, virtuaalisovellus, johon käyttäjä voi kohdistaa kyselyjä. Arkkitehtuuri perustuu mediaattoriteknologiaan eli "älykkääseen" keskitasoon, joka ohjaa kyselyt eri lähteisiin ja yhdistää niiden vastaukset.

Kuvassa 15 on esitetty TheaterLocin, Ariadnen komponenteista rakennetun sovelluksen arkkitehtuuri. Sivusto tarjoaa käyttäjälle tietoa ravintoloista ja elokuvateattereista eri kaupungeissa USA:ssa. Ensin käyttäjä valitsee kaupungin ja voi tämän jälkeen kohdistaa järjestelmään kyselyitä esimerkiksi elokuvateatterin lähettyvillä olevista ravintoloista (järjestelmä ei tue luonnollisen kielen käsittelyä, vaan kyselyt tehdään valitsemalla valmiista vaihtoehdoista tai kyselykielellä). Tieto sivustoon kerätään viidestä eri WWW-lähteestä.

Sovellus rakennetaan keräämällä kustakin lähteestä niiden sisältämät tiedot (attribuutit) ja rakentamalla sovittimet näiden tietojen poimimiseen. Eri lähteiden tiedot yhdistetään sovelluksen tietomalliksi, jolloin käyttäjä saa käytettäväkseen tiedon useasta WWW-lähteestä menemättä erikseen kuhunkin WWW-osoitteeseen.

Ariadnen komponentteja ovat sovelluksen aihepiirimallin muodostaminen, joka on ko-



Kuva 15: Ariadnen arkkitehtuuri yhdessä siihen perustuvassa sovelluksessa, TheaterLocissa.

koelma tietolähteiden kuvauksia, sovittimien rakentaminen kullekkin tietolähteelle ja kyselyn suunnittelija, joka optiomoi kyselyn.

Yhden lähteen tietosisällön mallintaminen (TP)

Sovelluksen kehittäjä tekee tietolähdekuvaukset poimimalla kustakin lähteestä sen tiedot tietolähdekuvaukseksi. Kukin tietolähde tarvitsee oman sovittimen, jonka sovelluksen kehittäjä rakentaa tietolähdekuvausta tehdessään. Sovitin luodaan demonstrointimenetelmällä, jossa käyttäjä näyttää esimerkkien avulla (leikkaamalla ja liimaamalla), mitkä tiedot sivuilta poimitaan. Tiedot nimetään ja poimitaan sivulta esimerkki vastaus. Etiketit muodostavat tietolähdekuvauksen eli määrittävät tiedon, jota sivulta voi eristää. Etiketit poimitaan upotettuun luetteloon (embedded catalog), joka on joko homogeeninen lista tai heterogeeninen kokoelma (tuple) koostuen numeroista, kirjaimista tai merkkijonoista, esimerkiksi *1, A, "test"*. Upotettu luettelo tarkoittaa, että lista voi koostua sisäkkäisistä luetteloista. Upotettu luettelo on siis rakenteistettu kuvaus sivulla olevasta tiedosta ja se voidaan kääntää XML:ksi. Sovittimen määrittämiseksi tarvitaan upotettu luettelo sekä eristämissäännöt jokaiseen kenttään ja iteraatiosäännöt kuhunkin listaan luettelossa.

(YT) Lähde koostuu useimmiten useammasta kuin yhdestä lähteestä. Tällöin tulee mallintaa myös yhteydet sivujen kesken. Yhteydet esitetään indeksisivulla, jolle täytyy tehdä myös oma sovitin.

(FT) Sovellus voisi olla täysin riippuvainen jonkun lähteensä hitaudesta ja tämä hidastaisi koko sovellusta jokaisessa kyselyssä. Tämä on estetty Ariadnessa datan välimuistiin tallennuksena. Tieto, jota ei päivitetä usein, materialisoidaan ja tallennetaan.

Koko sovelluksen tietomalli (YT)

(YT) Sivukohtaiset ja lähteen indeksointisovittimet yhdistetään aihepiirimalliksi. Kehittäjä määrittelee eri lähteiden sovittimien ja tietojen suhteet. Mediaattorin suunnittelija määrittelee aihepiirimallin, joka on yhdistetty sovellukseen liitetyistä tietolähteistä. Aihepiirimalli sisältää kuvauksen jokaisen tietolähteen sisällöstä, niiden välisistä suhteista ja kunkin lähteen sovittimesta. Aihepiirimalli on siis lähteiden attribuuttien kokoelma. Tässä vaiheessa ratkaistaan myös lähteiden väliset nimiristiriidat.
(YT) Nimiristiriidat on Ariadnessa ratkaistu ylläpitämällä erillistä tietolähdettä (kuvaustaulukkoa tai kuvausfunktiota), jonne kirjataan eri tietolähteiden toisistaan poikkeavat termit. Yksi tietolähde valitaan ensin primaaritietolähteeksi ja muiden eroavuudet suhteutetaan siihen. Kuvaustaulukon tai -funktion luomismenetelmä on puoliautomaattinen.

Kyselyn prosessointi

Sivujen mallintamisen ja yhdistämisen tarkoitus on tarjota käyttäjälle tietokantakyselyjä muistuttavia toimintoja. Käyttäjälle järjestelmä näkyy kuin kaikki tiedot olisivat tallennettuina yhdessä tietokannassa. Kyselyn prosessointiin liittyviä transformaatioita ovat: lähdekuvauksen käännetäminen off-line -tilassa sopivaan muotoon (FT), aksioomien luominen eli kaikkien mahdollisten kyselyissä esiintymien tietojen kombinaatioiden luominen johtamissääntöjä käyttäen (PT), käyttäjän syöttämien kyselyiden prosessointi, eli attribuuttien jäsennys, optimointi ja kyselyiden uudelleenkirjoittaminen tietolähteiden ymmärtämään muotoon (PT).

6.3 Tytti: koostaminen

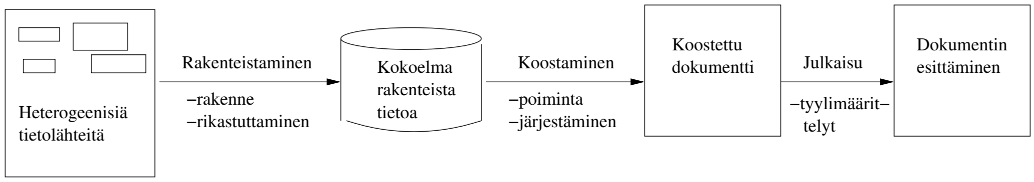
Osana Helsingin yliopiston tietojenkäsittelytieteen laitoksen Tytti-projektia (Tietotyöläinen työasema) on rakennettu koostamiskomponentti [Leh01]. Koostamisessa heterogeeniset tietolähteet rakenteistetaan ja talletetaan rakenteisen tiedon tietovarastoon elementteinä. Elementin koko voi vaihdella tapauskohtaisesti, tässä projektissa ne ovat tekstikappaleita tai kuvia. Koostamisvaiheessa tietokokoelmasta poimitaan ja järjestellään valittuja elementtejä. Elementtien väliin saatetaan lisätä vakiotekstejä tai käyttäjä voi itse tehdä lisäyksiä tai muunnella tekstejä. Tällöin saadaan tulokseksi koostettu dokumentti, joka valmiissa muodossaan julkaistaan tyylimäärittelyiden mukaisesti.

Koostamisen ensimmäisessä vaiheessa tietolähde rakenteistetaan. Tämä on tarpeen, jos lähdetiedostot ovat heterogeenisiä tai ne eivät ole rakenteellisesti järjestelmää vastaavia. Lähteinä voivat olla esimerkiksi Word- ja HTML-dokumentit, jolloin rakenteet ovat erilaisia. Projektissa lähteet muutetaan XML-muotoisiksi valmiilla tekstistä XML-muotoon kääntävällä ohjelmalla. Se analysoi dokumentin ja lisää siihen tarvittavat tunnisteet.

Lähdetieto segmentoidaan sopivan kokoisiksi elementeiksi ja sen rakenne muutetaan järjestelmän mukaiseksi. Rakenteistetut elementit talletetaan kokoelmaan. Jos tietoa päivitetään, täytyy muutokset tehdä ainoastaan tähän kokoelmaan, eikä kaikkiin erillisiin dokumentteihin, jossa tietoelementti on käytössä.

Rakenteistetusta tietokokoelmasta koostetaan tarvittavat dokumentit. Käyttäjä valitsee haluamansa dokumenttikokonaisuuden käyttöliittymästä. Poimittavat elementit määräytyvät joko järjestelmän pakottamien parametrien mukaan (jos halutaan käyttöohje puhelinmallille 3210, siirtyy tieto parametrina ja tietovarastosta poimitaan elementit tälle mallille) tai käyttäjän valintojen mukaisesti. Jos käyttöliittymän tarjoamat vaihtoehdot vaihtuvat usein, käyttöliittymän täytyy generoitua automaattisesti. Fragmenttien järjestetys määräytyy joko käyttöliittymän kautta käyttäjältä saatavien käskyjen mukaisesti tai valmiiden dokumenttimallien mukaisesti. Koostettu dokumentti näytetään käyttäjälle ennen sen julkaisua eli muotoilua tiettyyn formaattiin. Käyttäjä voi vielä lisätä, poistaa tai muokata dokumenttia.

Valmiin koostetun dokumentin julkaisu tapahtuu käyttäjän määrittelemien parametrien tai valmiiden dokumenttimallien mukaisesti. Koottu dokumentti saatetaan haluta julkaista usealla eri tavalla, esimerkiksi pdf-dokumenttina ja WWW-sivuna. Tällöin tyylimääritykset sivunumeroihin ja ylätunnisteisiin ovat erilaisia.



Kuva 16: Koostamisprosessi

Koostamisjärjestelmään liittyvät transformaatiot:

(FT) Tiedon rakenteistaminen

(PT) Tiedon segmentoiminen tietovarastoon

(YT) Tietopalasten koostaminen tietovarastosta

(FT) Kootun dokumentin julkaiseminen eri medioissa

6.4 Älykäs kysymys-vastaus -järjestelmä

Älykkäällä kysymys-vastaus -järjestelmällä tarkoitetaan automaattista kirjallisiin kysymyksiin vastaamista (sähköpostiviesti, WWW-sivun lomakevastaus tai tekstiviesti). Sitä voi soveltaa erityisesti asiakaspalveluun, jonne tulee samanlaisia kysymyksiä usein. Jos kysymykseen on jo kertaalleen vastattu, ei vastausta tarvitse kirjoittaa uudelleen.

Yksi kysely saattaa sisältää monia kysymyksiä, kommentteja tai huomautuksia, jolloin se on jaettava osiin. Segmentoinnilla asiakkaan viesti jaetaan loogisiin kysymysfragmentteihin, joista jokaisella on oma otsikko. Analysointiin käytetään sekä lingvistisiä että tilastollisia menetelmiä. Kysymysmuoto voi tietenkin olla hyvinkin erilainen, joten järjestelmän on analysoitava kysymys melko tarkasti. Erityisesti erisnimet (henkilöt, yritykset, tuotteet), aika ja paikkatiedototetaan huomioon.

Kysymysfragmentit yritetään luokitella eristetyn tiedon perusteella spesifeihin aihealueisiin, jotta vertailujoukko pienenee. Kategorisointia käytetään, jos luokkahierarkia tiedetään etukäteen. Tuntemattomaan aineiston luokitteluun käytetään ryvästämistä.

Kunkin kysymyksen ydintä verrataan olemassa olevaan kysymys-vastaustietokantaan. Samankaltaisuus tietokannassa olevan kysymyksen ja käsiteltävän kysymyksen välillä lasketaan jollakin samankaltaisuusmitalla. Samankaltaisuuden on ylitettävä tietty rajaarvo, jotta käsiteltävänä oleva kysymys voidaan hyväksyä samaksi kuin tietokannasta löytyvä kysymys-vastaus pari. Jos samankaltaisia kysymyksiä on useita, valitaan näistä paras tai esitetään asiantuntijalle useita vaihtoehtoja, joista hän valitsee parhaan. Jos kysymykseen ei löydy riittävän samankaltaista valmista kysymys-vastausparia, kysymys palautetaan asiantuntijalle, joka vastaa kysymykseen ja tallentaa kysymyksen vastauksineen tietokantaan.

Löydetyt ja hyväksytyt valmiit vastaukset koostetaan yhdeksi vastaukseksi. Siihen liitetään myös automaattiset vastauksen osat, kuten tervehdykset ja yhteystiedot. Valmis vastaus voidaan lähettää vielä valvojan kautta, jolloin hänellä on mahdollisuus tarkistaa vastaus, tehdä mahdollisia lisäyksiä, poistoja tai muutoksia.

Kysymys-vastausjärjestelmän transformaatiot:
(PT) Jos kysymys sisältää useampia eri aihetta käsitteleviä kysymyksiä, ne täytyy segmentoida
(PT) Tiedon eristäminen, merkittävän tiedon poimiminen kysymyksestä, jotta sitä voidaan verrata vastaustietokannassa oleviin valmiisiin kysymys-vastauspareihin
(PT) Kysymyksen luokittelu, jotta vertailtava kysymys-vastausjoukko pienenisi
(YT) Kysymyksen vertaaminen vastaustietokantaan: onko kyseiseen kysymykseen jo mahdollisesti vastattu
(YT) Kysymysten ja vastausten yhdistäminen ja automaattisten tervehdysten ja yhteystietojen lisääminen
(FT) Vastauksen muuntaminen asiakkaalle sopivaan muotoon

7 Yhteenveto

Tutkielmassa tutustuttiin sisällönhallintaprosessiin ja pyrittiin löytämään muunnoksia, joita tiedolle tehdään prosessin aikana. Sisällönhallintaprosessi voidaan yksinkertaistaa syöte-prosessointi-tuloste -ketjuksi, jolloin muunnokset tapahtuvat pääasiassa prosessointivaiheessa. Transformaatiot pyrittiin löytämään sisällönhallinnan esimerkkitapausten avulla, joista osa esiteltiin tutkielmassa.

Löydetyt transformaatiot kategorisoitiin formaatti-, poimimis- ja yhdistämistransformaatioiksi ja tämän jaon avulla monimutkaisetkin sisällönhallintaprosessit muuttuivat yksinkertaisiksi ja hallituiksi kokonaisuuksiksi.

Formaattitransformaatioilla tarkoitetaan tiedon esitysmuodon muuttamista siten, että tietosisältö säilyy ennellaan. Formaattitransformaatioita tarvitaan käsiteltäessä heterogeenisiä tietolähteitä, sillä yleensä tietolähteiden formaatti on yhdenmukaistettava. Formaattitransformaatioita tarvitaan myös esitysmuodosta toiseen siirryttäessä.

Tiedon poimimistransformaatiot ovat muunnoksia, jotka kohdistuvat yhteen tietolähteeseen. Poimimistransformaatioiden avulla tietolähdettä käsitellään siten, että tiedon jatkokäsittely helpottuu (kuten luokittelu) tai tiedosta erotetaan relevantein tieto (kuten eristäminen). Poimimistransformaatiotoimintoja ovat tiedon eristäminen, automaattinen yhteenveto, segmentointi, suodattaminen ja luokittelu. Toiminnot ovat keskenään osittain päällekkäisiä, esimerkiksi yhteenveto voidaan toteuttaa eristämällä.

Yhdistämistransformaatiot saavat syötteenä tietoa, joka on käynyt läpi tarvittavat formaatti- ja poimimistransformaatiot ja tarkoituksena on enää liittää useat eri lähteet yhteen. Yhdistämistransformaatiot käsittelevät eri tietolähteiden integrointiin liittyviä toimintoja. Yhdistämiseen liittyy tiedon vertailua ja koostamista sekä tapauskohtaisesti harkintaa, mitä tehdään lähteissä ilmenevien ristiriitaisten tai päällekkäisten tietojen kanssa.

Käyttäjän näkökulma -luvussa tarkasteltiin ajattelun yleisiä kognitiivisia piirteitä. Sisällönhallinnan tehtävä on helpottaa ihmisen muistin kuormitusta, tiedon järjestelyä ja käsittelyä, ja käyttäjän näkökulma on otettava huomioon ennen kaikkea käyttöliittymäsuunnittelussa.

Tutkielmassa analysoitiin eri transformaatioita ja jatkokehityksessä tämän teoriapohjan voisi työstää sisällönhallintaa helpottaviksi työkaluiksi. Tiedon keräämiseen liittyviä transformaatioita ovat erityisesti formaattitransformaatiot: lähteiden rakenteen yhdenmukaistaminen, metadatan poimiminen ja kielianalyysit. Työkalut tällaisiin muunnoksiin voisivat olla yleishyödyllisiä muiden transformaatioiden pohjana. Tiedon käsittelyyn liittyviä poimimistransformaatioita edustavia toimintoja löytyi paljon ja työkalut kaikkien näiden toimintojen automatisoimiseen olisivat hyödyllisiä. Useimmat toiminnot tarvitsevat toimintaansa tekstin sellaisessa muodossa, että sitä voi vertailla toisiin teksteihin. Tätä varten olisi hyvä olla mahdollisesti puoliautomaattinen tekstin muuntaminen vektorimuotoon. Automaattinen yhteenveto olisi toivottava lähes mihin tahansa lähteeseen erityisesti silloin, kun käyttäjä ei osaa määritellä informaatiotarvettaan ja hänen on päätettävä vasta dokumenttia lukiessaan kiinnostaako se häntä vai ei. Käyttäjä saisi tiivistelmän tiedosta ja voisi päättää haluaako tutustua koko lähteeseen. Eristäminen, luokittelu ja suodattaminen ovat välttämättömiä, koska tietoa on niin paljon: järjestelmän on autettava käyttäjää relevantin tiedon poimimisessa.Työkalut näille toiminnoille olisivat toivottuja. Työkalut voisivat toimia siten, että käyttäjä syöttää kiinnostuskohteensa ja järjestelmä eristää, luokittelee tai suodattaa tietoa tietolähteistä. Toisena vaihtoehtona käyttäjää kiinnostavat tiedot voitaisiin määritellä hänen käyttäjäprofiilissaan ja aina kun profiilia vastaava tietolähde ilmestyisi, tiedot tarjottaisiin käyttäjälle automaattisesti. Segmentointi-työkalun avulla teksti jaetaan automaattisesti loogisiin kokonaisuuksiin siten, että kukin segmentti käsittelee yhtä aihepiiriä.

Tiedon yhdistämistoimintoja tuetaan tällä hetkellä sisällönhallinnassa ehkä vähiten. Tiedon automaattiseen vertailuun tarvittaisiin työkalut. Järjestelmän tulisi pystyä tunnistamaan käsittelevätkö kaksi tietolähdettä samaa asiaa sekä esittämään samaa asiaa käsittelevien lähteiden eroavuudet.

Lähteet

ACD+98 James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study, final report. In *Proceedings of the Broadcast News Transcription and Understanding Workshop (Sponsored by DARPA)*, February 1998.

Ado95 Adobe Systems Incorporated. *TIFF (Tag Image File Format) 6.0 Specification*, 1995. http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf.

AI99 Douglas E. Appelt and David J. Israel. Introduction to information extraction technology - a tutorial prepared for IJCAI-99. Stockholm, August 1999. IJCAI. http://www.ai.sri.com/ appelt/ie-tutorial/IJCAI99.pdf.

AK97 Naveen Ashish and Craig A. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, December 1997. http://www.isi.edu/info-agents/papers/ashish97-coopis.pdf.

BBMS98 Maria Luisa Barja, Tore Bratvold, Jussi Myllymäki, and Gabriele Sonnenberger. Informia: a mediator for integrated access to heterogeneous information sources. In *Proceedings of Conference on Information and Knowledge Management (CIKM'98)*, pages 234–241, Zurich, Switzerland, 1998. ACM.

BC92 Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin. *Communications of the ACM*, 35(12):29–38, December 1992.

BGL+99 Chaitan Baru, Amarnath Gupta, Bertram Ludäscher, Richard Marciano, Yannis Papakonstantinou, Pavel Velikhov, and Vincent Chu. XML-Based Information Mediation with MIX. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 597–599, Philadelphia, Pennsylvania, USA, May 31 - June 3, 1999. ACM.

BN00 Branimir K. Boguraev and Mary S Neff. Discourse segmentation in aid of document summarization. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, HI, USA, 2000. IEEE.

Bun97 Peter Buneman. Semistructured data. In *Tutorial in Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 117–121. ACM, 1997.

Cho00 Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL'00)*, Seattle, USA, March 2000. ACL.

dCMCB99 Ana Maria de Carvalho Moura, Maria Luiza Machado Campos, and Cássia Maria Barreto. A metadata architecture to represent electronic documents on the web. In *Proceedings of the Third IEEE META-DATA Conference*, Bethesda, Maryland, USA, April 6-7, 1999. IEEE.

DPH98 Susan Dumais, John Platt, and David Heckerman. Inductive learning algorithms and representations for text categorization. In *Proceedings of Conference on Information and Knowledge Management (CIKM'98)*, pages 148–155, MD USA, 1998. ACM.

EK97 M. W. Eysenck and M. T. Keane. *Cognitive Psychology: A Students Handbook*. East Sussex: Psychology Press, 3rd edition, 1997.

FKS97 Josef Fink, Alfred Kobsa, and Jörg Schreck. Personalized hypermedia information provision through adaptive and adaptable system features: User modeling, privacy and security issues. In *Proceedings of the Fourth Sixth International Conference on User Modeling*, Chia Laguna, Sardinia, June 1997. User Modeling, Inc.

FPNB99 Jerry Fowler, Brad Perry, Marian H. Nodine, and Bruce Bargmeyer. Agent-Based Semantic Interoperability in InfoSleuth. *SIGMOD Record*, 28(1):60–67, 1999.

Fra99 Piero Fraternali. Tools and approaches for developing data-intensive web applications: A survey. *ACM Computing Surveys*, 31(3):227–263, September 1999.

GJ99 Oren Glickman and Rosie Jones. Examining machine learning for adaptable end-to-end information extraction systems. In *Workshop on Machine Learning for Information Extraction*, pages 7–14, Florida, July 1999. AAAI. http://www.isi.edu/ muslea/RISE/ML4IE/ml4ie.glickman&jones.ps.

GMGS96 Hector Garcia-Molina, Luis Gravano, and Narayanan Shivakumar. dSCAM: Finding Document Copies Across Multiple Databases. In *Proceedings Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, pages 68–79, Miami Beach, Florida, May 1996. IEEE Computer Society. http://www-db.stanford.edu/pub/papers/dscam.ps.

GMHI^{+}95 H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Integrating and accessing heterogeneous information sources in TSIMMIS. In *In Proceedings of the AAAI Symposium on Information Gathering*, Stanford,California, March 1995. AAAI. http://www.db.ucsd.edu/publications/tsimmis-abstract-aaai.pdf.

Gri97 Ralph Grishman. Information extraction: Techniques and challenges. In Maria Teresa Pazienza, editor, *Proceedings of the International Summer School on Information Extraction (SCIE-97)*, pages 10–27, Rome, Italy, 1997. Springer-Verlag.

GY00 Ralph Grishman and Roman Yangarber. Issues in corpus-trained information extraction. In *Proceedings of International Symposium on Spontaneous Speech: Toward the Realization of Spontaneous Speech Engineering*, Tokyo, Japan, February 2000.

Hea94 Marti A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 9–16, Las Cruces, New Mexico, USA, June 1994. ACL. http://www.sims.berkeley.edu/ hearst/papers/tiling-acl94/acl94.html.

Hea97 Marti A. Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, August 1997.

HGMC+97 J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the web. In *Proceedings of the Workshop on Management of Semistructured Data*, Tucson, Arizona, USA, May 1997. ACM/PODS/SIGMOD'97.

Hic87 James O. Hicks Jr. *Management Information Systems - A User Perspective*. West, 2nd edition, 1987.

HKt93 Eero Hyvönen, Ilkka Karanta, and Markku Syrjänen (toim.). *Tekoälyn ensyklopedia*. Gaudeamus, 1993.

HM00 Udo Hahn and Inderjeet Mani. The challenges of automatic summarization. *IEEE Computer*, pages 29–35, November 2000.

IET82 The Internet Engineering Task Force. *Standard for the Format of ARPA Internet Text Messages, RFC822*, 13 August, 1982. http://www.ietf.org/rfc/rfc0822.txt.

IET83 The Internet Engineering Task Force. *Standard for Interchange of USENET Messages, RFC 850*, June 1983. http://www.ietf.org/rfc/rfc0850.txt.

ISO86 ISO (International Organization for Standardization). *Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879*, October 1986.

ISO93 ISO (International Organization for Standardization). *Moving Pictures Expert Group (MPEG), ISO/IEC 11172*, 1993.

ISO96 ISO (International Organization for Standardization). *Document Style Semantics and Specification Language (DSSSL), ISO/IEC 10179*, August 1996.

ISO99 ISO (International Organization for Standardization). *Joint Photographic Experts Group (JPEG), ISO/IEC 10918-4*, 1999.

ITv01 Office xp yhdistää tietolähteitä. *ITviikko*, page 27, 15.3.2001.

JMF99 A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

KM98 Craig A. Knoblock and Steven Minton. The Ariadne approach to web-based information integration. *IEEE Intelligent Systems*, 13(5):17–20, September 1998.

KMA+98 Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. University of Southern California, 1998. http://www.isi.edu/info-agenst/papers/konblock98-ariadne.pdf.

Kno98 R. Knox. Publishing at light speed: The document in transition. Strategic analysis report, Gartner Advisory Research and Advisory Services, September 1998.

Koz93 Hideki Kozima. Text segmentation based on similarity between words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 286–288, Ohio, USA, 1993. ACL.

KPC95 Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of ACM Special Interest Group on Information Retrvieval (SIGIR'95) Seattle*, pages 68–73. ACM/SIGIR, 1995.

KS99 Sonja Kueppers and Michael Schilingno. Getting our act together: Human and technological factors in establishing an on-line knowledge base. In *Proceedings of The User Services Conference for University and College Computing Service Organizations*, pages 135–149, Denver, Colorado, USA, November 1999. ACM/SIGUCCS'99.

LCN+99 Thomas Lee, Melanie Chams, Robert Nado, Stuart Madnick, and Michael Siegel. Information integration with attribution support for corporate profiles. In *Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM)*, pages 423–429, Kansas City, Missouri, USA, November 2-6, 1999. ACM.

Leh01 Miro Lehtonen. Semi-automatic document assembly with structured source data. Pro gradu -tutkielma, Tietojenkäsittelytieteen laitos, Helsingin yliopisto, May 2001.

Mak00 Juha Makkonen. Automatisoitu tekstin luokittelu ja suodatus. Tyttiprojektin suunnitteludokumentti, Tietojenkäsittelytieteen laitos, Helsingin yliopisto, September 2000.

Man98 Christopher Manning. Rethinking text segmentation models: An information extraction case study. Technical report SULTRY-98-07-01, University of Sydney, Australia, July 1998.

Meg00 Megginson Technologies Ltd. *SAX 2.0: The Simple API for XML*, 5 May 2000. http://www.megginson.com/SAX/.

MR95 Kathleen McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th annual International Conference on Research and development in information retrieval*, pages 74–82, Seattle, WA USA, July 1995. ACM Special Interest Group on Information Retrieval (SIGIR).

Mus99 Ion Muslea. Extraction patterns for information extraction tasks: a survey. In *The AAAI-99 Workshop on Machine Learning for Information Extraction*, California, USA, 1999. American Association for Artificial Intelligence (AAAI).

MWJ99 Prasenjit Mitra, Gio Wiederhold, and Jan Jannink. Semi-automatic integration of knowledge sources. In *Proceedings of the Second International Conference on Information Fusion (Fusion '99)*, Sunnyvale, California, USA, July 6-8, 1999. International Society of Information Fusion (ISIF).

Nei76 U. Neisser. *Cognition and reality*. San Francisco: Freeman, 1976.

OS95 Adrian O'Riordan and Humphrey Sorensen. An intelligent agent for high-precision text filtering. In *Proceedingsof the 1995 International Conference on Information and Knowledge management (CIKM)*, pages 205–211, Baltimore, MD, USA, 1995. ACM.

Paz00 Michael J. Pazzani. Representation of electronic mail filtering profiles: a user study. In *Intelligent User Interfaces*, pages 202–206, New Orleans, Louisiana, USA, 2000. ACM.

Pri99 Michael Priestley. Dynamically assembled documentation. In *Proceedings on the 17th Annual International Conference on Computer Documentation*, pages 53–57, New Orleans, Louisiana, USA, September 1999. ACM.

RA95 Daniela Rus and James Allan. Does navigation require more than one compass? In *Proceedings of AAAI, fall '95 symposium, AI Applications in Knowledge Navigation and Retrieval*, Massachusetts Institute of Technology (MIT) Campus, Cambridge, Massachusetts, November 1995. AAAI (the American Association for Artificial Intelligence).

Rad99 Dragomir R. Radev. Generating natural language summaries from multiple on-line sources. Technical Report CUCS-005-97, New York, New York, USA, 1999.

Rey99 Jeffrey C. Reynar. Statistical models for topic segmentation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 357–364, College Park, Maryland, USA, June 1999. ACL.

RFC93 RFC (Request for Comments). *MIME (Multi-Purpose Internet Mail Extensions), RFC 1521 and RFC 1522*, September 1993. http://www.oac.uci.edu/indiv/ehood/MIME/1521/rfc1521ToC.html, http://www.oac.uci.edu/indiv/ehood/ MIME/1522/rfc1522ToC.html.

RSA97 Korin Richmond, Andrew Smith, and Einat Amitay. Detecting subject boundaries within text: A language independent statistical approach. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 47–54. Association for Computational Linguistics, Somerset, New Jersey, 1997.

Sal89 Gerard Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

SBCX97 Pamela A. Savage, Nicholas J. Belkin, Colleen Cool, and Hong Xie. An investigation of mental models and information seeking behavior in a novel task. In *Poster presented at SIGIR '97, The 20th Annual ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 116–122, Philadelphia, Pennsylvania, USA, July 1997. ACM/SIGIR.

SML99 Frank M. Shipman III, Catherine C. Marshall, and Mark LeMere. Beyond location: Hypertext workspaces and non-linear views. In *Proceedings of Hypertext 99*, pages 121–129. ACM, 1999.

SSBM96 Gerard Salton, Amit Singhal, Chris Buckley, and Mandar Mitra. Automatic text decomposition using text segments and text themes. In *Proceedings of the Hypertext '96 Conference*, pages 53–65, Washington D.C., USA, 1996. ACM.

SSW+98 C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support vector machine reference manual. Technical Report CSD-TR-98-03, Royal Holloway, University of London, Egham, Englanti, 1998.

THH95 Manfred Thuring, Jörg Hannemann, and Jörg M Haake. Hypermedia and cognition: Designing for comprehension. *Communications of the ACM*, 38(8):57–66, August 1995.

W3C96 W3C. *PNG (Portable Network Graphics) Specification 1.0 W3C Recommendation*, 01 October, 1996. http://www.w3.org/TR/REC-smil/TR/REC-png.html/.

W3C98 W3C. *Cascading Style Sheets level 2, CSS2 Specification, W3C Recommendation*, 12 May 1998. http://www.w3.org/TR/REC-CSS2/.

W3C99a W3C. *Cascading Style Sheets, level 1, W3C Recommendation*, 17 December 1996, revised 11 January 1999. http://www.w3.org/TR/REC-CSS1.

W3C99b W3C. *Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation*, 22 February 1999. http://www.w3.org/TR/REC-rdf-syntax/.

W3C99c W3C. *HTML 4.01 Specification, W3C Recommendation*, 24 December 1999. http://www.w3.org/TR/html401/.

W3C00a W3C. *Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation*, 21 November 2000. http://www.w3.org/TR/xsl.

W3C00b W3C. *XHTML 1.0 The Extensible HyperText Markup Language, A Reformulation of HTML 4 in XML 1.0, W3C Recommendation*, 26 January 2000. http://www.w3.org/TR/xhtml1/.

W3C00c W3C. *Resource Description Framework (RDF), Schema Specification 1.0 W3C Candidate Recommendation*, 27 March 2000. http://www.w3.org/ TR/ rdf-schema/.

W3C00d W3C. *Extensible Markup Language (XML) 1.0, W3C Recommendation*, 2nd edition, 6 October 2000. http://www.w3.org/XML/.

W3C01a W3C. *Document Object Model (DOM) Level 3, Content Models and Load and Save Specification Version 1.0, W3C Working Draft*, 09 February 2001. http://www.w3.org/TR/DOM-Level-3-CMLS/.

W3C01b W3C. *XML Schema Part 0: Primer W3C Recommendation, 2 May 2001*, 2 May 2001. http://www.w3.org/TR/xmlschema-o/.

W3C01c W3C. *Introduction to CSS3, W3C Working Draft*, 6 April 2001. http://www.w3.org/TR/css3-roadmap/.

Wie99 Gio Wiederhold. Mediation to deal with heterogeneous data sources. In Vckovski, Brassel, and Schek, editors, *Proceedings of the Interoperating Geographic Information Systems Conference (Interop'99)*, number 1580 in Lecture Notes in Computer Science, pages 1–16, Berlin, March 1999. Springer.

You91 Gilbert Youmans. A new tool for discourse analysis: the vocabulary-management profile. *Language*, 67(4):763–789, 1991.

YP97 Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of Fourteenth International Conference on Machine Learning (ICML'97)*, pages 412–420. Morgan Kaufmann, 1997.

ZGG+99 Xiaolan Zhu, Susan Gauch, Lutz Gerhard, Nicholas Kral, and Alexander Pretschner. Ontology-based web site mapping for information exploration. In *Proceedings of the 1999 ACM International Conf. on Information and Knowledge Management (CIKM)*, pages 188–194, Kansas City, Missouri, USA, November 2-6, 1999.