Jari-Pekka Heini

Organisaation tiedonhaku – tarkastelussa avoimen lähdekoodin ratkaisut

Tietotekniikan
pro gradu -tutkielma
9. elokuuta 2010

JYVÄSKYLÄN YLIOPISTO
TIETOTEKNIIKAN LAITOS

Jyväskylä

Tekijä: Jari-Pekka Heini
Yhteystiedot: jajuhein@gmail.com
Työn nimi: Organisaation tiedonhaku – tarkastelussa avoimen lähdekoodin ratkaisut
Title in English: Enterprise search – an insight to open source solutions
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 91
Tiivistelmä: Tässä tutkielmassa tarkastellaan, mitä tutkimuskohteita organisaation tiedonhaku pitää sisällään. Tarkastelun pohjaksi perehdytään tiedonhaun teoriaan. Organisaation tiedonhakujärjestelmän yleistä arkkitehtuuria hahmotellaan ja omisteisia organisaation tiedonhaun ratkaisuja esitellään. Näiden pohjalta muodostetaan luokitteleva viitekehys, jonka puitteissa arvioidaan, miten Apache Solr toteuttaa organisaation tiedonhaulle tyypilliset piirteet. Solrin käyttöönotto organisaatiossa vaatii paljon räätälöintityötä, mutta Solrin edellytykset räätälöintien tekemiselle ovat hyvät.
English abstract: This thesis examines research topics related to enterprise search. The theory of information retrieval is presented as a background to the discussion. A general architecture for an enterprise search system is sketched and proprietary enterprise search solutions are presented. Based on this information, a classification framework is developed for analysing, how Apache Solr fulfills the features typical to an enterprise search system. Adoption of Solr in enterprises requires many customisations, but the customisation ability of Solr is good.
Avainsanat: Organisaation tiedonhaku, tiedonhaku, Apache Solr, avoin lähdekoodi
Keywords: Enterprise search, information retrieval, Apache Solr, open source

Esipuhe

> ”Kaikki, mitä sanot, on ikävystyttävää ja käsittämätöntä, mutta tämä yksin ei tarkoita, että se olisi myös totta.”
>
> *Franz Kafka, novellista ”Erään taistelun kuvaus”, kirjoittajan vapaa suomennos*

Organisaation tiedonhaku oli aihe, joka tuli tämän tästä vastaan siirryttyäni – ennen valmistumistani, kuten joskus käy – opintojeni parista työelämään. Web-hakukoneet kertoivat minulle sadoin hakutuloksin, minkälaisen kuvauksen markkinavoimat aiheesta esittäisivät. Mutta mitä organisaation tiedonhaku tieteellisen tutkimuksen mielessä on? Kokonaiskuvan antavan vastauksen etsiminen on ollut suuri haaste; kirjoittaminen välttäen ikävystymistä (sekä omaa että muiden) ja ymmärrettävästi, mutta silti akateemisella tarkkuudella ja kaikkien faktojen mukaisesti, vielä suurempi haaste.

Tahdon kiittää KTT Anne Honkarantaa johdatuksesta aiheeseen ja professori Tommi Kärkkäistä siitä, että hän vielä kerran jaksoi tarttua puhelimeen ja kannustaa minua saattamaan taisteluni päätökseen. Kiitos kuuluu myös FM Miika Nurmiselle asiantuntevasta ja ajatuksia herättävästä ohjauksesta.

Tämän taistelun kuvaus ei olisi päätöksessään ennen kiitosta minua opinnoissani koko reilun neljännesvuosisadan pituisen matkan tukeneille vanhemmilleni ja sisaruksilleni: siis kiitos! Lopuksi vielä suuri ja aivan erityinen kiitos Veronikalle, joka tämän tutkielman eteen on joutunut osoittamaan – olen tästä täysin vakuuttunut – mitä suurinta kärsivällisyyttä.

Helsingissä, heinäkuussa 2010
Jari-Pekka Heini

Sisältö

1 Johdanto

Informaation etsiminen World Wide Webin valtavasta tietomäärästä on tuttu haaste akateemisessa maailmassa. Jopa suositulla Google-hakukoneella on akateemiset juuret [12]. Kun haun kohteena onkin organisaatio ja organisaation kaikki tietolähteet – ei ainoastaan organisaation intranetiin rakennettu portaali – on ongelmaa lähestyttävä toisesta näkökulmasta. Internet ja intranetit jakavat samoja ominaispiirteitä, mutta niihin kohdistuvalla tiedonhaulla on erilaiset lähtökohdat. Ja kuten todettua, tietoa on usein tarve etsiä myös hyperlinkitettyjen dokumenttien muodostaman aineiston ulkopuolelta: ulkoisista tietokannoista, sähköpostiarkistoista, levyjaoista, ja niin edelleen. Tällöin Web- tai työpöytähakukoneet eivät yksin riitä ratkaisemaan ongelmaa, vaan vähintäänkin niiden toiminnallisuutta pitää laajentaa kattamaan kaikki informaation etsijän kiinnostuksen kohteet. Digitaalisena järjestelmänä hakukone voi löytää vain digitaalisessa muodossa olevaa aineistoa, joka on lisäksi jollakin tavoin sen saatavilla. Tämä on tiedon löytymisen perusedellytys. Itse hakuoperaatio, halutun tiedon löytäminen suuresta massasta tietoa, vaikuttaa pintapuolisestikin tarkasteltuna haastavalta tehtävältä, sillä organisaatioiden sovellus- ja järjestelmäratkaisut saattavat olla monimutkaisia.

Tiedonhaku (engl. information retrieval, IR) tutkii – lyhyesti ottaen – tiedon tallennusta, lajittelua ja hakua. Tiedonhaun tutkimuksella on pitkä historia 1950-luvulta nykypäivään. Organisaation tiedonhaun tutkimus on yksi uusimmista tiedonhaun tutkimusalueista, ja se voidaan nähdä jatkumona perinteisen tiedonhaun ja Web-hakukoneiden tutkimukselle. Organisaation tiedonhaku tarkoittaa tiedonhaun soveltamista organisaatioiden piirissä olevaan digitaaliseen tietoon.

Tässä tutkielmassa kuvataan organisaation tiedonhaun taustaa ja teoriaa. Taustaksi organisaation tiedonhaulle on esitetty tiedonhaun perusteoriaa niiltä osin kuin se on aiheen käsittelyn kannalta tarpeellista. Tutkielmassa pyritään selvittämään, mitä omisteisen ja avoimen lähdekoodin ratkaisuja organisaation tiedonhakuun on olemassa. Näistä ratkaisuista lähempään tarkasteluun on valittu Apache Solr. Tarkastelua varten on kehitetty teoreettinen viitekehys, jonka puitteissa Apache Solr -sovellusta tutkitaan. Tutkimus suoritetaan kirjallisuuskatsauksena sekä analysoimalla avoimen ohjelmiston ratkaisua käytännössä.

Luku 1 on tutkielman johdanto. Luvussa 2 perehdytään tiedonhakuun ja sen tutkimukseen yleisesti. Luvussa käydään läpi tiedonhaun tämän tutkielman kannalta oleel-

linen teoria. Luvussa 3 selvitetään, mitä organisaation tiedonhaku pitää sisällään käyttäen pohjana tiedonhaun taustateoriaa. Luvussa 4 selvitetään, minkälaisia suljetun ja avoimen lähdekoodin ratkaisuja organisaation tiedonhakuun on olemassa. Luvussa 5 kehitetään viitekehys näiden ratkaisujen arviointiin. Luvussa 6 sovelletaan kehitettyä viitekehystä Apache Solr -sovellukseen. Lopuksi esitetään yhteenveto luvussa 7.

2 Tiedonhaku

Organisaation tiedonhaun tutkimus pohjautuu vahvasti tiedonhaun käsitteistöön. Tiedonhaku (engl. information retrieval eli IR) on jo 1950-luvulta lähtien tutkinut dokumenttien hakua homogeenisistä tekstikokoelmista, kuten lehtiarkistoista, tieteellisistä tiivistelmistä ja CD-ROM-hakuteoksista. World Wide Web (tästä eteenpäin lyhyemmin Web) on tuonut 1990-luvulta lähtien omat vaatimuksensa tiedonhaun tutkimukselle, ja viimeaikaisena kehitysaskeleena tiedonhaun ala on joutunut huomioimaan myös organisaatioiden vaatimukset tiedonhaulle [5]. Tässä luvussa luodaan pohja luvun 3 käsitteistölle perehtymällä tiedonhaun tutkimusalueeseen.

2.1 Tiedonhaun tutkimus

Termiä tiedonhaku on alettu käyttää jo vuoden 1952 tienoilla, ja sen perusteet on luotu jo vuonna 1945 Vannevar Bushin toimesta [15]. Bush puhui tuolloin ”laitteesta, johon voidaan tallentaa kaikki kirjallinen ja suullinen aineisto, ja josta tuo aineisto sitten on nopeasti saatavilla” [25].

Järvelinin [33] mukaan tiedonhaun tutkimus ”kehittää käsitteitä, menetelmiä ja järjestelmiä, joiden avulla kaikki tieto, olipa se missä tahansa muodossa ja missä tahansa paikassa, saadaan vaivattomasti kenen tahansa sitä tarvitsevan ulottuville mahdollisimman hyödyllisessä ja helposti omaksuttavassa muodossa”. Kuten Järvelinin määritelmästä käy ilmi, ei tiedonhaun tutkimuksen ulkopuolelle ole rajattu mitään tiettyä alaa, mediaa tai sisältöä, vaan se pyrkii yhtälailla palvelemaan kaikkia osapuolia, jotka sen tuotoksista voivat hyötyä. Alkuvaiheessa tutkimus on keskittynyt tukemaan kirjastojen ja muiden painettua tietoa arkistoivien laitosten työtä. Se on tuolloin keskittynyt nykyistä kapea-alaisempaan ongelmakenttään: tekstipohjaisen tiedon etsimiseen ihmislähtöisesti. Tietokoneiden yleistymisen myötä tiedonhakua on ryhdytty soveltamaan elektronisessa ympäristössä, ja se on saanut paljon uusia tutkimuksen kohteita, kuten järjestelmäarkkitehtuurit, datan visualisointi, käyttöliittymät ja erilaiset formaalit ja luonnolliset kielet [5]. Viimeisimpänä kehitysvaiheena Internet ja erityisesti Web ovat tuoneet valtayleisön ulottuville laajan määrän elektronista aineistoa, jota pitää pystyä hakemaan. Itse asiassa Web sisältää niin suuren määrän inhimillisesti kiinnostavaa tietoa yhtenäisen käyttöliittymän takana, että sen voidaan nähdä dominoivan tiedonhaun tutkimusta [25].

2.2 Tiedonhaun taustakäsitteet

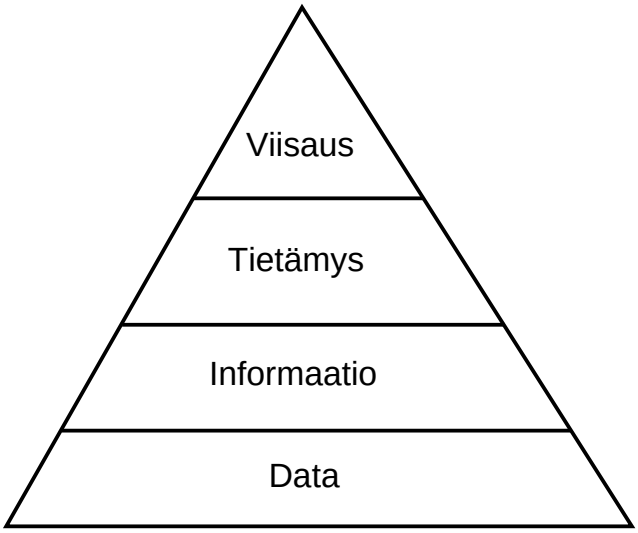
Seuraavissa alaluvuissa käydään läpi tiedonhaun taustakäsitteet.

2.2.1 Tieto, informaatio ja data

Järvelin [33] korostaa sanan tieto ongelmallisuutta suomen kielessä. Hänen mukaansa pitäisi oikeastaan puhua informaationhausta, mutta sana tieto on oikeutettu, sillä johdonmukaisuuden vuoksi tulisi puhua myös datakoneista, datakannoista ja niin edelleen. *Datan* erottaminen tiedosta ja informaatiosta on ainakin teknisessä mielessä suhteellisen helppoa, ja sitä voisi pitää tieteellisenä yleistietona: data, esimerkiksi sarja tietokantaan tallennettuja bittejä, muuttuu informaatioksi, kun sille annetaan tulkinta, esimerkiksi sovitaan, että tietty bittijono tarkoittaa päiväystä. Datasta tulee *informaatiota*, kun ihminen tulkitsee datan tai datalle annetaan jokin tulkinta. Data on siis eräänlaista ”potentiaalista informaatiota”. Tieto taas rinnastetaan keskustelussa usein dataan, vaikka sillä voidaankin joskus tarkoittaa nimenomaan informaatiota. Termin tiedonhaku kohdalla tiedolla tarkoitetaan nimenomaan informaatiota [33]. Tässä tutkielmassa englanninkielinen termi ”information retrieval” suomennetaan tiedonhaku, ja sanaa tieto käytetään muutoinkin viittaamaan informaatioon, jotta saadaan säilytettyä yhteneväisyys lähdeaineiston ja alan kielessä vakiintuneen käytännön kanssa.

Nürnberger *et al.* [39] ovat tehneet yhteenvetoa dataan ja informaation liittyvien käsitteiden osalta. Heidän mukaansa informaation yläpuolella on *tietämyksen* (engl. knowledge) käsite. Tietämys on suhteilla toisiinsa liittyvää informaatiota. Täten tietämys voi ottaa kantaa kysymykseen ”miten”. Tähän voitaisiin edelleen liittää harvemmin käytetty käsite *ymmärrys*, joka syntyy, kun esimerkiksi tilastollisin menetelmin otetaan kantaa kysymykseen ”miksi”. Kaiken huipulla on *viisauden* (engl. wisdom) käsite, jota on vaikea tarkasti määritellä, mutta jonka Nürnberger *et al.* tiivistävät olevan kokonaisuus erikoistaidoista, joita tarvitaan jonkin ongelman ratkaisemiseen. Inhimillisessä kontekstissa tämä liittyy elämän hallitsemiseen ja – ymmärtämiseen. Tämän tutkielman kannalta on olennaista ymmärtää näiden käsitteiden keskinäinen suhde, jota on havainnollistettu kuvassa 2.1.

Kyse on siis eräänlaisesta jalostusketjusta, jossa datasta tulee informaatiota, joka luo pohjan tietämykselle ja ymmärrykselle (tämä välimuoto on jätetty pois kuvasta 2.1). Syvimmillään tämä ymmärrys voi jalostua viisaudeksi. Tietokoneen tai filosofisesti ottaen koneen osa tästä ketjusta on alkupäässä, datan ja informaation välimaastossa. Ehkä tietämyksenkin määritelmän mukaan tietokoneiden voisi vielä helposti olettaa sisältävän tietämystäkin. Mutta ymmärrys ja viisaus ovat vielä enemmän inhimillisiä kuin koneellisia ominaisuuksia. Tiedonhakujärjestelmän tavoitteena voi olla informaa-



Kuva 2.1: Data jalostuu informaatiosta tietämykseen ja lopulta viisaudeksi [39].

tion ja miksei tietämyksenkin välittyminen ihmiselle niin, että edellytykset ymmärrykselle ja lopulta viisaudelle syntyvät.

2.2.2 Haku ja selaaminen

Termillä haku (engl. search) tarkoitetaan usein avainsanojen avulla tehtävää hakua, tiedon etsimistä. Esimerkiksi suosituimpien Web-hakukoneiden joukkoon[1] kuuluvat Google[2] ja Bing[3] mainostavat hakevansa tietoa Webistä (engl. "searching information from Web"). Mukherjee ja Mao [37] määrittelevät termin merkityksen laajemmaksi, kattamaan monimutkaisemmat hakuominaisuudet ja navigaation – siis erilaiset käyttöliittymät – sekä informaation löytämisen. Haku voi siis tarkoittaa – avainsanahaun ja erilaisten hakukäyttöliittymien lisäksi – ihmisen suorittamaa aktiviteettia: tiedon konkreettista, ihmislähtöistä etsimistä.

Voidaan puhua myös tiedon selaamisesta (engl. browsing). Monet Web-hakukoneet, kuten Yahoo![4], mahdollistavat Web-sivuille osoittavien linkkien selaamisen ja siirtymisen sivulle linkkiä seuraamalla. Toinen reaalimaailman esimerkki tiedon selaamisesta on kirjan etsiminen kirjaston hyllystä. Kummassakin tapauksessa tieto on tavalla tai toisella tuotu tietoa hakevan eteen, jonka jälkeen varsinaista hakua tiedon löytämiseksi tapahtuu enää tiedon hakijan itsensä taholta.

[1] http://www.seoconsultants.com/search-engines
[2] http://www.google.fi
[3] http://www.bing.com
[4] http://www.yahoo.com

2.2.3 Tiedonhaku ja datahaku

Baeza-Yates ja Ribeiro-Neto [5] johdattavat lukijansa ensi alkuun *datahakuun* (engl. data retrieval), joka lienee tiedonhakua helpommin lähestyttävä käsite. Datahaulla tarkoitetaan datajoukon hakemista siten että datajoukko toteuttaa kaikki täsmällisesti esimerkiksi relaatioalgebran tai sääntölausekkeen avulla määritellyt hakuehdot. Elävän elämän esimerkkinä datahausta he esittävät perinteisen relationaalisen tietokantajärjestelmän, jonka dataa voidaan hakea esimerkiksi SQL-kielen keinoin.

Tiedonhaku Baeza-Yatesin ja Ribeiro-Neton mukaan puolestaan keskittyy datan hakemisen sijaan tiedon, tarkemmin ilmaistuna informaation, hakemiseen. Tällöin haku ei palauta täsmällistä joukkoa dataa, jossa kaikki hakutulokset täsmäävät matemaattisen yksikäsitteisesti hakuehtoihin, vaan joukon tuloksia, joiden relevanttius on lopulta yksin hakutuloksia selaavan ihmisen pääteltävissä. Useallakaan ”väärällä” hakutuloksella ei ole merkitystä, jos haluttu informaatio on helposti löydettävissä hakutulosten joukosta. Tässä yhteydessä Järvelinin [33] korostama datan ja informaation ero tulee selvästi esille.

Jotta saataisiin haettua juuri haluttu tieto, tulee tiedonhaun ottaa kantaa myös tiedon tallennukseen ja lajitteluun. Tiedon hakeminenhan on vain osa kokonaisuutta: se vaatii, että tieto on valmista haettavaksi. Tämän tarpeen tiedonhaun tutkimus pyrkii täyttämään kehittämällä tallennukseen ja lajitteluun liittyviä algoritmeja, tietorakenteita ja menetelmiä, joiden avulla tiedon hakemisesta tulee mahdollisimman tehokasta.

2.2.4 Digitaalinen dokumentti ja dokumenttikokoelma

Tiedonhaun yhteydessä puhutaan usein selkeyden vuoksi dokumenttien hausta abstraktimman ilmaisun ”tiedon haku” sijasta. Dokumentti ansaitseekin tarkemman määrittelyn.

Salminen [42] määrittelee *dokumentin* sen ominaisuuksien kautta:

- Dokumentti on tarkoitettu ihmisen vastaanotettavaksi, informaatioksi tietystä aiheesta.
- Dokumentilla on sisältö ja yksi tai useampi ulkoinen esitysmuoto.
- Dokumentin sisältö koostuu osista, osat koostuvat symboleista, ja osat on järjestetty ihmiselle helposti ymmärrettäviksi.
- Dokumentti on tallennettu medialle.
- Dokumentti voidaan tunnistaa ja käsitellä kokonaisuutena.

Digitaaliseen dokumenttiin, jolle dokumentti sähköisessä maailmassa usein toimii synonyymina, liittyvät lisäksi Salmisen määritelmässä vielä seuraavat ominaisuudet:

- Sisältö on tallennettu digitaaliselle medialle.
- Dokumentti on yhdistetty sellaiseen laitteistoon ja ohjelmistoon, joka pystyy tunnistamaan digitaalisesta sisällöstä tunnisteet ja osien rakenteen ja tuottamaan ulkoisen esitysmuodon ihmisten ymmärrettäväksi.

Edellä mainittujen ominaisuusvaatimusten perusteella esimerkiksi tekstinkäsittelyohjelmalla tuotettu elektroninen tiedosto on helppo ymmärtää dokumentiksi, koska sillä on kaikki Salmisen mainitsemat ominaisuudet. Dokumentin käsite auttaa ymmärtämään tiedonhaun teoriaa, koska sen avulla tiedonhaun talletusmenetelmät, algoritmit ja tietorakenteet saavat konkreettisen esitystavan koko tutkimusalueen ytimessä olevalle haun päämäärälle, tiedolle. Salmisen määritelmä sisältää tärkeän lauseen: dokumentti on ihmiselle kohdistettua informaatiota tietystä aiheesta. Dokumentti siis kapseloi tiedon ihmiselle – tai koneelle – helpommin ymmärrettäväksi ja käsiteltäväksi kokonaisuudeksi.

Usein tiedonhaun teoriassa mainitaan termi *dokumenttikokoelma*. Tiedonhaun tutkimuksen kannalta dokumenttikokoelma-termin käyttö on yksinkertainen tapa viitata kaikkiin tiedonhaun kannalta kiinnostaviin, tiedonhakujärjestelmän piirissä oleviin dokumentteihin.

2.2.5 Metatieto

Metatieto tai metadata on ”tietoa tiedosta”. Sille ei löydy yksikäsitteistä määritelmää, sillä termiä käytetään eri tieteenaloilla erilaisissa merkityksissä. Metatiedon voisi kuitenkin määritellä, Gilliland-Swetlandin [24] mukaan, seuraavasti: ”kokonaissumma kaikesta, mitä voidaan sanoa mistä tahansa informaatio-objektista millä tahansa aggregaatiotasolla”. Informaatio-objekti voi olla mikä tahansa kokonaisuus, joka on ihmisen tai järjestelmän manipuloitavissa. Se voi siten olla vaikkapa dokumentti.

Metatiedon käyttö on yksi tapa mahdollistaa multimediasisällön hakeminen tekstuaalisen aineiston hakuun kehitetyillä menetelmillä, mutta metatieto on oleellinen osa myös tekstipohjaisia dokumentteja. Esimerkiksi dokumentin otsikko, sijainti ja koko voidaan nähdä metatietoina. Vaikka esimerkiksi otsikko voi hyvinkin kuulua varsinaisesti dokumenttiin itseensä, eikä se siten ole dokumenttiin liitettyä, ”ylimääräistä” tietoa, ei metatiedon edellä esitetty määritelmä silti estä kutsumasta otsikkoakin metatiedoksi.

2.2.6 Rakenteinen ja rakenteeton tieto

Rakenteiseksi tiedoksi tai tarkemmin *rakenteiseksi dataksi* (engl. structured data) kutsutaan dataa, joka pohjautuu johonkin skeemaan [14]. Koska skeema määrittelee datalle ohjelmallisesti tulkittavissa olevan rakenteen, voidaan rakenteellisen datan pääsääntöisesti ajatella olevan helpommin ohjelmallisesti käsiteltävissä. *Rakenteeton data* (unstructured data) puolestaan on raakadataa, esimerkiksi kuva- tai äänitiedosto tai puhdasta tekstiä [14]. Rakenteisuuden määritelmän selkeä heikkous on se, että se nojaa pitkälti tarkastelukulmaan. Laajassa kontekstissa esimerkiksi kuvatiedosto voidaan nähdä hyvinkin täsmällisesti rakenteellisena datana, mikäli ollaan kiinnostuttu tiedoston esittämästä kuvasta kokonaisuutena.

Esimerkkeinä rakenteisesta datasta esitetään usein tietokannan tieto tai XML-dokumentit, mutta ei ole mitenkään mahdotonta, että tietokanta tai XML-sovellus olisi määritelty niin, että osa tietokannan kentistä tai XML-entiteeteistä sisältäisi rakenteetonta dataa (esimerkiksi SQL:n Binary Blob -tietotyyppi tai XML-entiteetti, joka sisältää base64-koodattua äänidataa) [20]. Tällaista dataa, jolla ei ole, jälleen näkökulmasta riippuen, täysin rakenteista tai rakenteetonta muotoa, kutsutaan *puolirakenteiseksi dataksi* (engl. semistructured data) [1].

Rakenteisuuden ja rakenteettomuuden käsitteitä voidaan, melko abstraktilla tasolla tosin, soveltaa datasta eteenpäin tietoon, informaatioon ja dokumentteihin saakka. Periaatteessa rakenteisuutta pitäisi mitata prosentuaalisena suureena, sillä sen määritelmän moniselitteisyydestä johtuen jonkin kokonaisuuden kutsuminen poissulkevasti joko rakenteiseksi tai rakenteettomaksi saattaa johtaa liiaksi harhaan. Joka tapauksessa rakenteisen datan hyvinä puolina erityisesti tiedonhaun kannalta voidaan pitää sen tuomaa lisäinformaatiota automaattiseen käsittelyyn. Rakenteinen tieto vaatii vähemmän ihmisen tulkintaa, ja täten sen käyttöarvoa tiedonhakujärjestelmälle voidaan pitää suurempana. Tekstipohjainen tiedonhaku ei välttämättä merkittävästi hyödy rakenteellisuudesta, mutta rakenteisen tiedon käyttö mahdollistaa tiedon kontekstin ja semantiikan koneellisen ymmärtämisen: tietämyksen määritelmä, suhteilla toisiinsa liittyvä informaatio, oikeastaan vaatii jonkinlaista semanttista rakennetta.

2.2.7 Hyperteksti ja Web-dokumentit

Hyperteksti nähdään ennen kaikkea Webille ominaisena ilmiönä, mutta se on käsitteenä Webiä vanhempaa perua: Ted Nelson lanseerasi termin jo vuonna 1965 [38]. Viimeistään Webin aikakaudella hyperteksti on saavuttanut laajempaa kiinnostusta. Hypertekstin sovellutuksia ovat lähes kaikki käytetyimmät Webin dokumenttityypit, ensimmäisistä

SGML-johdannaisista HTML-versioista uudempiin XML-sovelluksiin, kuten XHTML, saakka.

Web-dokumentteja kutsutaan *sivuiksi*, ja niihin voidaan yksikäsitteisesti viitata erilaisilla mekanismeilla, joista URL (Unified Resource Locator) lienee tunnetuin. Toisiinsa linkittävien ja usein samassa osoiteavaruudessa sijaitsevien Web-sivujen loogisten kokonaisuuksien muodostamia ryhmiä kutsutaan *sivustoiksi*. Viittausta sivusta toiseen kutsutaan *hyperlinkiksi* tai *linkiksi* (engl. link tai anchor), ja Web-sivujen linkkien perusteella muodostamaa verkkoa tai graafia *hyperavaruudeksi*. Linkit, jotka on tarkoitettu Webin käyttäjän seurattaviksi, on usein varustettu linkin kohdetta kuvaavalla tekstillä. Tätä sanotaan *linkkitekstiksi* (engl. anchor text).

Toisiinsa viittaavat dokumentit tuovat omat erityispiirteensä tiedonhakuun. Dokumenttiviittausten käsite ei ole rajoittunut vain Web-maailmaan – esimerkiksi tieteelliset lähdeviittaukset ovat toinen esimerkki hypertekstistä – mutta Web on tuonut ongelman laajemman yleisön tietoisuuteen.

2.3 Tiedonhaun perusprosessi

Kuka tai mikä tietoa hakee? Tähän tiedonhaun tutkimus [23, 5, 25] vastaa yksiselitteisesti: *käyttäjä*, siis ihminen. Oikeastaan jo sana informaatio edellyttää ihmistä osana tiedonhaun prosessia: data muuttuu informaatioksi vasta inhimillisen tulkinnan kautta; tietokoneelle data on aina dataa, se vain muuttaa muotoaan. Tiedonhaun tuloksena saatu informaatio on harvoin täysin irrallista ympäröivästä maailmasta. Tällainen informaation suhde toiseen informaatioon, esimerkiksi dokumentin sisältämän informaation suhde toiseen dokumenttiin, voi siis jalostua kuvan 2.1 mukaisesti tietämykseksi ja mahdollisesti jopa viisaudeksi.

Baeza-Yates ja Ribeiro-Neto [5] korostavat käyttäjän tärkeyttä tiedonhaussa. Koko tiedonhaun prosessi tähtää käyttäjän *tiedontarpeen* tyydyttämiseen. Käyttäjää palvelee *tiedonhakujärjestelmä*, siis kone, joten käyttäjän täytyy esittää tiedon tarpeensa järjestelmän ymmärtämällä kielellä. Ihanteellisessa tapauksessa käyttäjä käyttäisi jotakin luonnollista kieltä, mutta tietokonejärjestelmän kannalta rajatumpi kielioppi on tarpeen. Käyttäjän tällaisella formaalilla kielellä antamaa syötettä kutsutaan *hakulauseeksi* tai *kyselyksi*. Mikäli tämä hakulause jaetaan osiin, esimerkiksi yksittäisiin luonnollisen kielen sanoihin, kutsutaan osia *hakutermeiksi* tai hakusanoiksi.

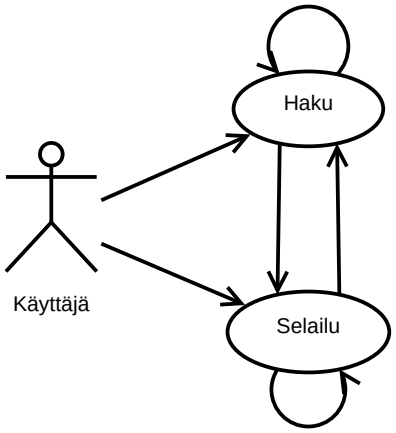
On syytä huomata, ettei käyttäjä välttämättä onnistu löytämään haluamaansa tietoa ensimmäisellä yrittämällä. Tämä tekee tiedonhaun prosessista iteratiivisen. Toisaalta käyttäjä voi halutessaan muokata hakulausettaan myöhemmillä iteraatioilla, mah-

dollisesti ymmärrettyään itsekin selkeämmin tiedontarpeensa, jolloin prosessi saa myös inkrementaalisen ulottuvuuden. [5]

Toisen aspektin tiedon hakemisen prosessiin tuo tiedon selailu. Käyttäjä voi hakulauseen antamisen sijasta myös etsiä tietoa valmiista valikoimasta esimerkiksi navigaationaalisen käyttöliittymän avulla. Tällöin hakulausetta ei tarvita, vaan tietoa tarjoavan järjestelmän tulee vain kyetä esittämään tieto järkevällä tavalla käyttöliittymässä ja tarjoamaan käyttäjälle pääsy valitsemaansa tietoon. Ei ole periaatteellista syytä myöskään estää käyttäjää siirtymästä selailusta hakuun ja takaisin. [5]

Jos katsotaan asiaa vielä käyttäjäpsykologian kannalta, ei käyttäjälle usein ole merkitystä sillä, miten haun tulokset on löydetty, kunhan hän saa mahdollisimman relevantit dokumentit mahdollisimman tehokkaasti. Löydettyjen dokumenttien esittäminen käyttöliittymässä on usein vain jäävuoren huippu kaikesta siitä toiminnallisuudesta, joka on jouduttu rakentamaan, jotta käyttäjä saa haluamansa. [5]

Tiedonhaun prosessia käyttäjän näkökulmasta on havainnollistettu lähdettä [5] mukaillen kuvassa 2.2.



Kuva 2.2: Käyttäjälähtöinen tiedonhaun perusprosessi.

Seuraavissa alaluvuissa perehdytään tarkemmin tiedonhaun perusteoriaan.

2.3.1 Dokumenttien klusterointi

Dokumenttien klusteroinnilla tarkoitetaan samankaltaisten tai toisiinsa liittyvien dokumenttien ryhmittelemistä luokiksi [5]. Hypoteesi on, että tämä edesauttaa haluttujen dokumenttien löytymistä, koska yksittäisten dokumenttien sijasta voidaankin hakutuloksina palauttaa klustereita ja kohdistaa jatkohaku niiden sisältämiin dokumentteihin [18]. Yksinkertaisimmillaan luokittelu voi tarkoittaa samankaltaisten dokumenttien

manuaalista merkitsemistä, monimutkaisimmillaan voidaan käyttää tilastollisia tai todennäköisyyslaskentaan perustuvia menetelmiä luokittelun aikaansaamiseksi [25].

Tiedonhaun perusprosessissa, josta kuva 2.3 antaa yleiskäsityksen, klusterointi voi olla mukana kahdella periaatteellisella tavalla. Klusterointiin kehitettyä algoritmiä voidaan käyttää haussa niin, että hakulauseketta evaluoidaan yksittäisten dokumenttien sijasta dokumenttiklustereita vasten. Näin haku voidaan kohdistaa vain tiettyyn klusteriin tai klustereihin, joiden sisältämien dokumenttien joukkoon voidaan käyttää normaaleja hakumenetelmiä. Cutting et al. huomauttivat jo 1992, ettei tämä tapa tutkimuksen valossa välttämättä tuo erityistä hyötyä hakuun. Toinen tapa on hyödyntää dokumenttiklustereita selailupohjaisessa käyttöliittymässä. Tämän vaihtoehdon mukaan tiedonhakujärjestelmän käyttöliittymä tarjoaisi alussa näkymän dokumenttikokoelmasta klusterointialgoritmin perusteella löytyviin klustereihin. Tällainen näkymä voisi yksinkertaisesti sisältää jonkinlaisen kuvauksen, kuten otsikon jokaisesta klusterista esimerkiksi listana. Käyttäjä voisi porautua syvemmälle valitsemaansa klusteriin tai klustereihin, ja jokaisella porautumisella muodostettaisiin uusi lista valittujen klusterien sisältä löydetyistä klustereista. Klusterien koon pienentyessä tarpeeksi käyttöliittymä voisi esittää klusterien sijasta dokumentteja, jotka täten muodostaisivat hakutulosten joukon. [18]

2.3.2 Semanttinen haku

Yksi hakulauseeseen perustuvan, luonnollisen kielen sanoja käyttävän haun heikkouksista on se, että eri dokumenteissa samoja sanoja voidaan käyttää eri merkityksissä tai jopa useammissa merkityksissä. Luonnollisessa kielessä sanoihin liittyy niiden kirjoitusasun lisäksi merkitystä eli semantiikkaa. Perinteisesssä sanahaussa esimerkiksi hakulause ”eurooppalainen matkapuhelinpuhelinvalmistaja” ei löytäisi dokumenttia, jossa lukee ”suomalainen Nokia”, vaikka dokumentti saattaisi tyydyttää hakijan tiedontarpeen. Dokumentin löytämiseksi tiedonhakujärjestelmän tulisi osata yhdistää suomalainen eurooppalaiseen ja Nokia matkapuhelinvalmistajaan. Seuraavat määritelmät ja niiden pohjalta tehdyt huomiot perustuvat Gökerin ja Daviesin toimittamaan tuoreeseen tiedonhaun tutkimusta yhteenvetävään teokseen [25].

Eräs formaali tapa esittää tällaisia informaation suhteita, tietämystä, on ontologia, joka on nelikko:

$$O = \langle C, R, I, A \rangle$$

Tässä C on luokkien (engl. classes) eli tarkasteltavien konseptien joukko (kansalaisuudet, yritykset, tuotteet, jne.); R on yhteyksien (engl. relations) joukko luokkien ins-

tanssien kesken (”yrityksellä on kansallisuus”); I on instanssien (engl. instances) joukko (”Nokia on suomalainen”); A on aksioomien (engl. axioms) joukko (esim. ”jos tuotteen hinta on yli 100 euroa, toimitus on ilmainen”).

Ontologia mahdollistaa yleisen tai alakohtaisen tietämyksen mallintamisen ja käsittelyn automaattisesti. Göker *et al.* jakavat ontologiat skeema-, aihe-, ja leksikaalisiin ontologioihin, jotka eroavat paitsi luonteeltaan myös kyvyltään esittää semantiikkaa. Skeema-ontologiat määrittelevät olioperustaisen ajattelun tavoin luokkia attribuutteineen, ja näiden luokkien instanssien eli olioiden yhteyksiä toisiinsa. Aihe-ontologiat ovat taksonomioita, jotka määrittelevät aihekohtaisia, puumaisia hierarkioita. Leksikaaliset ontologiat, kuten asiasanastot, mallintavat luonnollisten kielten sanojen leksikaalisia yhteyksiä toisiinsa.

Yksi tapa lisätä semantiikkaa dokumentteihin on manuaalisesti tai automatisoidusti määritellä niille metatietoja liittyen dokumentissä käytettyihin nimettyihin entiteetteihin, kuten ihmisiin, organisaatioihin tai paikannimiin, ja näiden entiteettien välisiin suhteisiin. Tällaisia tunnisteita kutsutaan *annotaatioiksi* tai *nimikkeiksi* (engl. annotation) ja tällaisen annotaatiokokoelman sisältämä tietämys voidaan liittää hakuprosessiin esimerkiksi erityisen kyselykielen avulla.

Semantiikka voidaan huomioida tiedonhaun prosesseissa pääasiassa kahdella tavalla. Kokotekstihaku voidaan ottaa lähtökohdaksi, ja semanttista hakua voidaan käyttää tukemaan kokotekstihaun toimintaa: mikäli hakulauseen termeihin on yhdistettävissä semantiikkaa esimerkiksi johonkin ontologiaan perustuvan tietorakenteen avulla, voidaan sitä käyttää lisätulosten etsimiseen. Toinen vaihtoehto on, että yritetään ontologiaa apuna käyttäen kiinnittää huomiota siihen, mitä käyttäjä hakulauseellaan varsinaisesti hakee, ja tulkitaan haku alusta alkaen ontologian keinoin.

Semantiikan tutkimus on keskittynyt pääasiassa Web-tiedonhaun alueelle: semantiikkaa sisältävästä ”uudesta” Webistä käytetään nimitystä semanttinen Web. Web-dokumenttien, esimerkiksi HTML-sivujen sisällön automaattinen prosessointi on perinteisesti ollut rajoittunutta, sillä niiden semantiikka aukeaa vain ihmiselle. Vaikka yhä useammat Web-dokumentit ovat XML-pohjaisten sovellusten myötä entistä puhtaammin rakenteisia, sisältävät ne harvoin metatietoa sisältämänsä datan merkitykseen liittyen. Web-dokumenttien semantiikan lisäämiseksi on kehitetty XML-pohjaisia standardeja, tärkeimpinä ehkä RDF ja OWL, jotka perustuvat ontologioiden teoriaan ja mahdollistavat automaattisen prosessoinnin kaipaaman semantiikan esittämisen. Tutkimus on tällä saralla ollut vahvaa ehkä siksi, että semanttisen haun (joka vaatii semantiikkaa piirissään olevilta dokumenteilta) on arvioitu nousevan merkittäväksi sovellukseksi Webin piirissä. Konkreettisia esimerkkejä semanttisesta hausta on kuitenkin edelleen olemassa varsin vähän. Tähän löytynee monia syitä, mutta yleisesti voidaan

todeta, etteivät semanttisen Webin standardit ehkä ole nousseet suuren yleisön tietoisuuteen siinä määrin, että Web-dokumentit todella sisältäisivät riittävästi semantiikkaa järkevän semanttisen haun, tai muidenkaan semanttisen Webin sovellusten, mahdollistamiseksi. Pelkkä semantiikan määrittely ei vielä riitä, tarvitaan myös työkaluja ja sovelluksia, jotta semantiikkaa voidaan hyödyntää.

2.4 Tiedonhakujärjestelmä

Tiedonhakujärjestelmän tavoite, saattaa relevantti tieto käyttäjän saataville, toteutuu Gökerin [25] mukaan pääasiassa seuraavien kolmen abstraktion avulla:

1. Dokumenttien sisällön esitys.
2. Käyttäjän tiedontarpeen esitys.
3. Kahden edeltävän esityksen vertaaminen keskenään.

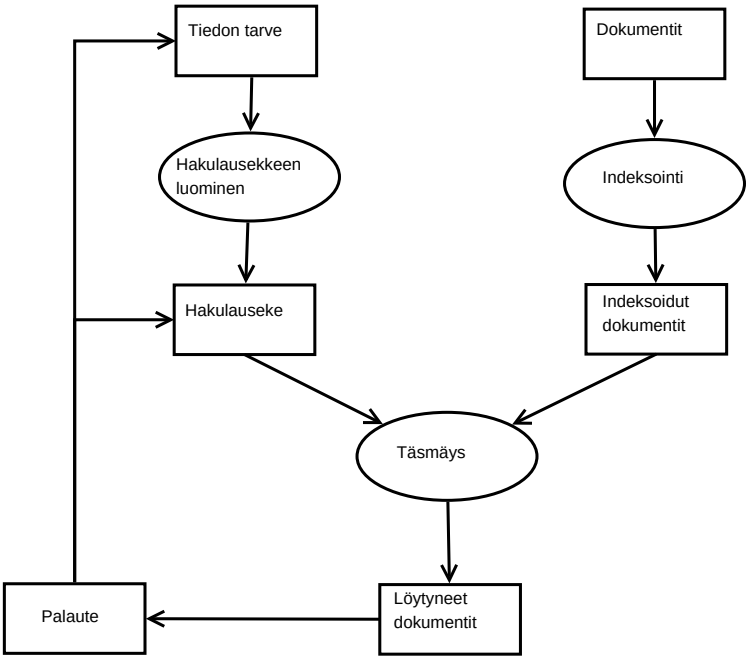
Ensinnäkin tiedonhakujärjestelmän on oltava tietoinen dokumenteista, joita sen piiriin kuuluu. Dokumenttien kokoamista tiedonhakujärjestelmän piiriin kutsutaan *indeksoinniksi* ja indeksoinnin avulla luotua tiedon, siis dokumenttien *loogisen esityksen*, kokoelmaa *(dokumentti-)indeksiksi*. Tiedonhakujärjestelmä voi tässä yhteydessä tallettaa dokumentin omaan haltuunsa, mutta tämä ei ole välttämätöntä. Usein on tehokkaampaa tallettaa indeksoinnin yhteydessä vain joitakin dokumentin metatietoja, kuten otsikko, tiivistelmä ja sijainti. Joka tapauksessa indeksoinnin tarkoitus on tallettaa dokumentista jotain sellaista dataa – kuten dokumentissa esiintyvät sanat, *(indeksi)termit* – joiden avulla alkuperäinen dokumentti voidaan jälkeenpäin identifioida. Tätä voi verrata esimerkiksi kirjassa olevaan hakemistoon, jonka sanat sisältävät viitteen, sivunumeron, jonka avulla sanan sisältämä sivu löydetään. Indeksi on siis dokumenttien pohjalta jollakin algoritmillä koottu tietorakenne.

Toiseksi, tietoa etsivän käyttäjän on kyettävä esittämään tiedontarpeensa formaalilla kielellä, jota tiedonhakujärjestelmä ymmärtää. Tämä prosessi voi olla iteratiivinen, koska käyttäjän tiedontarve on subjektiivinen käsite, eikä käyttäjä välttämättä ensimmäisellä yrittämällä ole täysin selvillä siitä, mitä tietoa hän järjestelmästä haluaa hakea. Tiedonhakujärjestelmän formaali näkemys eheästä hakulauseesta ja käyttäjän subjektiivinen näkemys kaipaamastaan tiedosta kohtaavat siinä vaiheessa, kun käyttäjä on tyytyväinen saamaansa tietoon, toisin sanoen saa tiedonhakujärjestelmän kautta haltuunsa haluamansa, *relevantit* dokumentit. Täydellinen tiedonhakujärjestelmä palauttaisi automaattisesti relevantit dokumentit, mutta tämä ei ole mahdollista, koska

tiedon relevanttius on määritelmänsä mukaisesti viime kädessä käyttäjän subjektiivisen käsityksen varassa.

Kolmanneksi, jotta tiedonhakujärjestelmä osaisi hakea relevantin dokumentin käyttäjälle, tulee sen osata yhdistää käyttäjän hakulause indeksiin, ja löytää hakulausetta vastaavat dokumentit. Tällaisia dokumentteja voi löytyä yksi tai useampia, tai ei yhtäkään. Mikäli dokumentteja löytyy useampi, esittää tiedonhakujärjestelmä dokumentit usein sellaisessa järjestyksessä, että niiden relevanttius – jokin dokumentti voi olla relevantimpi kuin toinen – tulee käyttäjälle selväksi, eikä tämän tarvitse selata löytyneitä dokumentteja enempää. Edellä mainitun mahdollistamiseksi ohjelmallisesti, tulee hakulauseella ja dokumenttien indeksillä olla jokin täsmällisesti määritetty yhteys. Tällaista yhteyttä kutsutaan *tiedonhakumalliksi*.

Kuva 2.3 hahmottaa edellä mainittuja tiedonhakujärjestelmän osa-alueita. Kuva esittää tiedon tarpeen pohjalta syntyvän hakulauseen ja dokumenttien indeksoinnin yhteyden, täsmäyksen. Löytyneiden dokumenttien joukko ja käyttäjän toimintaa tukevat hakuominaisuudet antavat palautetta paremman hakulausekkeen luomiselle, tai aiheuttavat muutoksia tiedon tarpeessa.



Kuva 2.3: Tiedonhakujärjestelmän osa-alueet [25].

2.5 Esimerkki tiedonhakujärjestelmästä: Web-hakukone

Googlen alkuperäisten kehittäjien tittelin saaneet Brin ja Page *et al.* johdattivat jo vuonna 1998 lukijansa siihen, miten Web tietyiltä osin on tuonut uudenlaisia vaatimuksia perinteiselle tiedonhaun tutkimukselle [12, 40]. Webille on ominaista käyttöliittymän sidonnaisuus dokumenttikokoelmaan. Web-tiedonhakujärjestelmien eli Web-hakukoneiden käyttöliittymien voidaan katsoa olevan rakennettu HTML-sivuista tai muista hypertekstidokumenteista. Web-hakukoneen käyttöliittymä olisi siis määritelmän mukaan itsekin haun piirissä oleva indeksoitu dokumentti. Flash ja vastaavat sovellukset tai esimerkiksi PDF-tiedostot[5] eivät ole varsinaista hypertekstiä, eivätkä varsinaisesti edes helposti käsiteltäviä tekstuaalisia dokumentteja, ja juuri siksi ne indeksoituvat niin heikosti mukaan Web-haun piiriin. Hypertekstin rakenne asettaa indeksoinnille erityisvaatimuksia, ja indeksoinnin tehokas toteuttaminen onkin merkittävä tekijä Web-hakukoneiden tehokkuuden kannalta. Web-käyttöliittymä puolestaan asettaa raamit Webissä käytettäville kyselykielille. Web-hakukoneen käytettävyyden kannalta helpointa on rakentaa kyselykieli yksinkertaisia HTML-lomakkeita hyväksikäyttäen, jolloin monimutkaisimmat kyselykielen rakenteet ovat erilaisia avainsanoja vapaatekstikentässä. Näin on laita esimerkiksi Google-hakukoneen tapauksessa. Hakulausekkeen ja indeksoitujen dokumenttien täsmäys tuo löytyneet dokumentit käyttäjän tiedoksi yleensä HTML-sivuna, johon dokumentit on listattu. Monet hakukoneet tarjoavat yksinkertaisen sanahaun lisäksi muita, hakua helpottavia käyttöliittymäominaisuuksia. Tällaiset hakuominaisuudet voivat suoraviivaista hakutulosten listaa paremmin esimerkiksi antaa palautetta siitä, miten haku kannattaisi muotoilla. Esimerkiksi hakulausekkeen oikoluku, läheisesti hakutermeihin liittyvät vaihtoehtoiset termit tai löytyneiden dokumenttien määrä voivat antaa vihjeitä hakulauseen parantamiseen tai jopa täsmentää käyttäjän käsitystä tiedontarpeestaan.

2.6 Tiedonhakumallit

Tiedonhakumallit luovat perustan yhdistää relevantit dokumentit käyttäjän antamaan hakulauseeseen. Tiedonhakumallin tavoitetta on mahdotonta täydellisesti saavuttaa, koska relevanttius on subjektiivinen käsite, mutta tiedonhakumallin avulla voidaan päästä riittävän hyvään aproksimaatioon todellisuudesta.

Seuraavissa alaluvuissa esitellään klassiset tiedonhakumallit – Boolen malli, vektorimalli ja todennäköisyysmalli – jotka perustuvat dokumentti-indeksissä esiintyvien

[5]Tosin nykyisin esimerkiksi tekstiä sisältäviä PDF-tiedostoja voidaan varsin tehokkaasti muuntaa HTML-muotoon, jolloin ne ovat myös hakukoneiden käytettävissä.

indeksitermien ja hakulausekkeen sanojen yhdistämiseen. Näistä malleista Boolen malli eroaa olennaisesti muista: se on yksi ensimmäisistä kehitetyistä hakumalleista ja yksinkertaisin. Boolen malli ei mahdollista hakutulosten lajittelua relevanssin mukaan; vektori- ja todennäköisyysmalleissa tämä on puolestaan otettu mallin kehittelyn lähtökohdaksi. Tämän lisäksi esitellään linkkipohjaisia hakumalleja, jotka on kehitetty erityisesti hypertekstidokumenttien keskinäisen relevanssin pisteytykseen.

2.6.1 Boolen malli

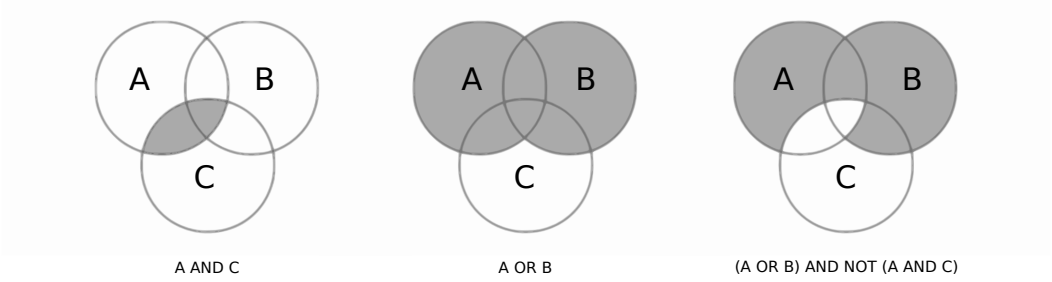
Ensimmäinen kolmesta klassisesta tiedonhakumallista on Boolen malli. Mallin idea on yksinkertainen ja helposti omaksuttavissa, ei vähiten siksi, että se perustuu yksinkertaiseen Boolen algebraan ja joukko-oppiin. Toisaalta se on myös kritisoiduin malli. Yksinkertaisuudessaan Boolen mallin mukaan hakutermeillä on binääriset painoarvot. Esimerkiksi hakutermi ”opiskelu” täsmää kaikkiin niihin dokumentteihin, jotka on indeksoitu termillä ”opiskelu”. Näitä painoarvoja yhdistelemällä hakuehdon mukaisesti loogisilla lausekkeilla, saadaan pääteltyä, onko dokumentti relevantti vai epärelevantti haun kannalta. Boolen malli määrittelee kolme tällaista loogista operaattoria:

AND Binäärinen operaattori, joka vaatii, että kummatkin sen operandeina olevista hakutermeistä ovat myös dokumentin indeksitermejä, jotta dokumentti tulee mukaan hakutuloksiin.

OR Binäärinen operaattori, joka vaatii, että jompi kumpi tai kummatkin sen operandeista ovat myös indeksitermejä.

NOT Unaarinen operaattori, joka vaatii, ettei sen operandi ole indeksitermi.

Näiden operaattoreiden toimintaa on havainnollistettu Vennin diagrammien avulla kuvassa 2.4.



Kuva 2.4: Boolen mallin hakutermejä Vennin diagrammein esitettynä.

Boolen mallin selkeä vahvuus on, että sitä käytettäessä on ilman tarkempaa analyysiä heti selvää, miksi dokumentti täsmää hakulauseeseen. Käyttäjälläkin on tällöin

edellytykset hahmottaa hakulauseen ja indeksin yhteys. Hakutuloksena saatujen dokumenttien joukkoa on myös helppo pienentää tai kasvattaa ottamalla hakulauseesta hakutermejä pois tai yhdistämällä siihen lisää hakutermejä loogisin operaattorein.

Boolen malli ei kuitenkaan kykene järjestämään löytyneiden dokumenttien joukkoa mitenkään. Dokumentti joko täsmää hakuehtoihin tai sitten ei, joten haetuilla dokumenteilla ei ole minkäänlaista keskinäistä arvojärjestystä. Tässä mielessä Boolen malli muistuttaa jopa enemmän datanhakumallia kuin tiedonhakumallia. Lisäksi mallin binäärisyys saattaa johtaa epätoivottuihin lopputuloksiin käyttäjän kannalta. Esimerkiksi hakulause

```
kalastus AND kala AND järvi
```

ei täsmäisi dokumenttiin, joka on indeksoitu vain termeillä ”kalastus” ja ”kala”, koska hakutermi ”järvi” puuttuu. Tällainen dokumentti kuitenkin todennäköisesti tyydyttäisi käyttäjän tiedontarpeen, eikä käyttäjän voi aina olettaa osaavan muotoilla hakulausettakaan niin, ettei se sulje haluttuja tuloksia pois. [5, 25]

2.6.2 Vektorimalli

Vektorimallissa haku- ja indeksitermien yhteys esitetään vektorina, jolloin mallin avulla on mahdollista esittää tilanne, jossa dokumentti täyttää hakuehdot vain osittain. Vektorimallin perusteella saatu tulosjoukko on mahdollista lajitella samankaltaisuuden mukaan. [5]

Hans Peter Luhn esitti ensimmäisenä, että dokumenttien haun lähtökohdaksi voitaisiin ottaa esimerkkidokumentti, joka on mahdollisimman samanlainen haettujen dokumenttien kanssa. Luhn määritteli tämän samankaltaisuuden seuraavasti:

> Mitä enemmän kaksi esitystä vastaavat toisiaan elementtiensä (indeksitermien, kirj. huom.) ja niiden esiintymisen suhteen, sitä todennäköisemmin nämä kaksi esitystä sisältävät samankaltaista informaatiota [35].

Kahden dokumentin vastaavuuden laskemiseksi pitäisi siis laskea yhteisten haku- ja indeksitermien määrä. Tätä varten oletetaan, että dokumentin indeksissä oleva looginen esitys on vektori $\vec{d} = (d_1, d_2, \ldots, d_m)$, missä kukin komponentti $d_k \mid k \in [1, m]$ on yhdistetty indeksitermiin [25]. Lisäksi oletetaan, että hakulause on vastaava vektori $\vec{q} = (q_1, q_2, \ldots, q_m)$, missä kukin komponentti on yhdistetty samoihin indeksitermeihin. Oletetaan vielä, että komponentit ovat binäärisiä, ts. $\forall k : d_k, q_k \in \{0, 1\}$. Nyt yhteisten termien määrälle saadaan mitta vektorien sisätulosta:

$$\text{score}(\vec{d}, \vec{q}) = \sum_{k=1}^{m} d_k q_k$$

Yleisempi ja käyttökelpoisempi esitys saataisiin muuttamalla vektorikomponentit d_k ja q_k luonnollisiksi luvuiksi tai reaaliluvuiksi. Tällöin kukin komponentti kuvaisi vastaavan indeksitermin painoarvoa suuremmalla skaalalla, ja painoarvoja voisi säätää tarpeen mukaan binäärisiä painoarvoja joustavammin.

Vektorimallin eri puolia painottavia sovelluksia on useita, mutta varsinaiseksi vektorimalliksi kutsutaan usein Saltonin *et al.* [43] kehittämää mallia, jossa vektoreille d_k ja q_k annetaan oma kummallekin oma dimensionsa. Lisäksi vektorien sisätulo normalisoidaan vektorien pituuden suhteen:

$$\text{score}(\vec{d}, \vec{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}} \tag{2.1}$$

Kaava mittaa vektoreiden $\vec{d}$ ja $\vec{q}$ välisen kulman kosinia, joten vektoreiden samankaltaisuutta voidaan visualisoida kolmiulotteisessa avaruudessa.

Vektorikomponenttien määrittäminen on vektorimallin suurimpia heikkouksia. Jotta vektorimalli kuvaisi todellisuutta riittävällä tarkkuudella, tulee komponenttien arvoille valita sopivat arvot, ts. *painottaa* komponentit (engl. term weighting) sopivilla painoilla (engl. weights) [25]. Saltonin ja Yangin [44] mukaan tämä ei ole lainkaan triviaali ongelma. Monet modernit painotusalgoritmit perustuvat heidän kehittämäänsä kaavaan

$$d_k = q_k = tf(k, d) \cdot \log \frac{N}{df(k)}, \tag{2.2}$$

missä $tf(k, d)$ on termin k esiintymien lukumäärä dokumentissa d, $df(k)$ on termin k sisältämien dokumenttien lukumäärä ja on N dokumenttien kokonaismäärä indeksissä. Näitä painoja nimitetään usein *tf.idf*-painoiksi, missä *tf* viittaa kaavan 2.2 termin esiintymien lukumäärään tf ja *idf* on käänteisesti suhteessa termin sisältämien dokumenttien määrään df.

2.6.3 Todennäköisyysmalli

Vektorimallin keskeinen heikkous on vektorikomponenttien määrittäminen. Todennäköisyysmalli keskittyy olennaisesti komponenttien painotuksen formalisointiin. Mallissa jokaiselle indeksitermille T asetetaan todennäköisyys $P(T \mid D)$, jolla se liittyy dokumenttiin D. Toisin sanoen $P(T \mid D)$ on todennäköisyys sille, että käyttäjä käyttää hakulausetta T dokumentin D hakemiseen.

Bayesin lauseen mukaan voidaan laskea $P(D \mid T)$ eli todennäköisyys sille, että dokumentti D on relevantti, jos sitä on haettu hakulauseella T:

$$P(D \mid T) = \frac{P(T \mid D)P(D)}{P(T)}$$

[25]

Todennäköisyysmallin perusajatus on, että kutakin hakulausetta kohti löytyy yksiselitteinen, ideaali joukko dokumentteja, jotka ovat relevantteja suhteessa kyseiseen hakulauseeseen. Tällaisen ideaalin joukon löytäminen olisi helppoa, jos tiedettäisiin, millä indeksitermeillä sellainen tulisi määritellä. Indeksitermeillä, jotka esiintyvät relevanteissa dokumenteissa jollakin annetulla hakulauseella, tulisi lähtökohtaisesti olla suurempi todennäköisyys. Koska tällaista indeksitermien riippuvuutta ei etukäteen voida tietää, lähdetään todennäköisyysmallin sovelluksissa arvauksesta, jota sitten yritetään esimerkiksi käyttäjäperustaisesti parantaa. Näin ollen lopullisen arvion voidaan olettaa lähestyvän ideaalia dokumenttien joukkoa riittävällä tarkkuudella. Tällainen alkuarvion antaminen ei välttämättä ole triviaalia, ja se kuuluukin todennäköisyysmallin suurimpiin heikkouksiin.

2.6.4 Linkkipohjaiset hakumallit

Tieteellisten artikkelien lähdeviitteet ovat olleet tutkimuksen kohteena jo vuosikymmeniä. Hypertekstuaaliset viitteet, linkit, voivat hyödyntää lähdeviitteiden teoriaa, ja linkkeihin perustuvia hakumalleja on kehitetty useampia. Niistä teoreettisesti tärkeimpiä lienevät HITS- ja PageRank-algoritmit, joista PageRank esitellään tässä tarkemmin.

PageRank on muodostunut lähes synonyymiksi Web-sivujen keskinäiselle vertailulle [25]. Se laskee Web-sivulle pistemäärän perustuen sivuun osoittaviin linkkeihin ja niiden pistemääriin. Mitä suurempi pistemäärä sivuun osoittavilla linkeillä on, sitä enemmän ne lisäävät sivun omaa pistemäärää [12].

Göker *et al.* [25] formalisoivat sivun d PageRank-pistemäärän $P(D = d)$ seuraavasti:

$$P(D = d) = (1 - \lambda)\frac{1}{\text{sivujen lkm}} + \lambda \sum_{i \mid i \text{ viittaa } d\text{:hen}} P(D = i)P(D = d \mid D = i)$$

Intuitiivisesti tämä tarkoittaa satunnaisen Webiä selaavan käyttäjän, surffaajan, todennäköisyyttä vierailla sivulla d. Jos oletetaan, että satunnainen käyttäjä aloittaa selaamisen sivulta i, on satunnaisen linkin seuraamisen todennäköisyys tällöin $P(D = d \mid D = i)$. Algoritmi toimii siis rekursiivisesti summaamalla kaikkien sivuun d viittaavien sivujen i PageRank-pistemäärän $P(D = i)$ ja todennäköisyyden

$P(D = d \mid D = i)$ tulot. Algoritmin tasapainotukseen on käytetty vakiota λ: Koska kaikkiin sivuihin ei välttämättä osoita yksikään linkki, on linkin seuraamiselle määritelty todennäköisyys λ. Todennäköisyydellä $(1 - \lambda)$ mitään linkkiä ei seurata, vaan siirrytään satunnaiselle sivulle (intuitiivisesti tämä tarkoittaisi esimerkiksi osoitteen kirjoittamista Web-selaimen osoiteriville).

On huomionarvoista, ettei PageRank-algoritmi ota kantaa hakulauseeseen. Se onkin niin sanottu staattinen pisteytysalgoritmi tai -funktio (engl. static ranking function), joka lasketaan kullekin sivulle indeksoinnin yhteydessä. Laskenta tulee toki tehdä aika ajoin uudelleen, mikäli sivut päivittyvät. Varsinaisen tiedonhakumallin työn tekeekin usein jokin perinteinen tiedonhakumalli, kuten Boolen malli. Esimerkiksi Boolen mallia käytettäessä voitaisiin hakea hakulausetta vastaavat dokumentit Boolen mallin mukaisesti, ja PageRank toimisi vain saadun tulosjoukon dokumenttien keskinäisenä pisteyttäjänä. Vektori- ja todennäköisyysmallien kanssa PageRank-algoritmin antama pistemäärä voi toimia dokumentin *a priori* -todennäköisyytenä, johon vektori- tai todennäköisyysmallin todennäköisyys yhdistetään.

PageRank on kehitetty alunperin Google-hakukonetta varten [12], mutta sittemmin Google on toki lisännyt ison liudan muitakin ominaisuuksia määrittämään Web-sivujen keskinäistä paremmuutta suhteessa hakulauseeseen.[6] Tämä on luonnollista, sillä Webin luonne on jatkuvasti muuttunut, ja PageRank-algoritmiä on opittu käyttämään hyväksi, kuten Googlea edeltävien Web-hakukoneiden algoritmeja ennen sitä.

2.7 Indeksointi

Hakulauseen etsimisen esimerkiksi tekstidokumentista ei tarvitse aina olla monimutkaista. Esimerkiksi Unix-käyttöjärjestelmän grep-komento osaa etsiä parametrina annettua sanaa (hakutermiä) sille parametrina annetuista tiedostoista (dokumenteista):

```
$ grep ''luennot'' kurssikuvaus*.txt
```

Miten etsiminen tässä yhteydessä oikeastaan tapahtuu? Komento grep yksinkertaisesti käy läpi eli *skannaa* (engl. scan) tekstitiedostot alusta loppuun etsien parametrina annettua sanaa tekstin joukosta. Sana määritellään tässä yhteydessä joukoksi peräkkäisiä merkkejä. Tällainen lähestymistapa voi toimia hyvin pieniin ja usein muuttuviin dokumentteihin, mutta se käy nopeasti liian hitaaksi, jos dokumenttien koko ja lukumäärä kasvaa.

Toinen tapa lähestyä ongelmaa on rakentaa dokumenttien hakuun liittyvä tietorakenne, indeksi. Dokumenttien läpikotaisen skannaamisen sijasta tätä tietorakennetta

[6] http://www.google.com/corporate/tech.html

vastaan rakennettu algoritmi voi tällöin suorituskykyisemmin osoittaa, mistä dokumentista hakutermi löytyy.

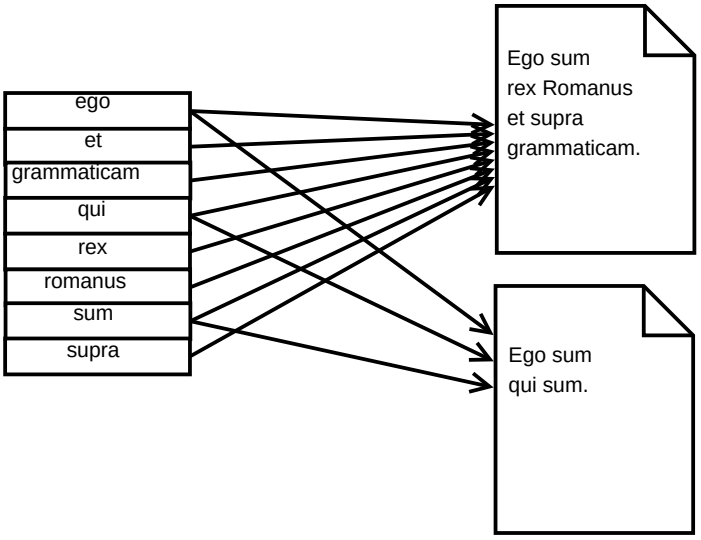
Seuraavissa alaluvuissa esitellään tärkeimmän indeksitietorakenteen, käänteistiedoston, rakentamista ja hakualgoritmiä, joka käyttää hyväkseen tätä tietorakennetta. Mainittakoon, ettei käänteistiedosto suinkaan ole ainoa indeksitietorakenne – mainittavia esimerkkejä ovat suffiksitaulukot (engl. suffix arrays) ja nimikirjoitustiedostot (engl. signature files) – mutta se on tällä hetkellä käytetyin ja tärkein [5].

2.7.1 Käänteistiedosto

Käänteistiedosto (engl. inverted file, inverted index) on luonnollisen kielen sanan käsitteeseen perustuva mekanismi tekstidokumenttien kokoelman indeksointiin. Oletetaan, että kuhunkin dokumenttikokoelman dokumenttiin liittyy lista avainsanoja eli indeksitermejä. Näillä indeksitermeillä voi tarvittaessa olla myös painoarvot. Toisin päin esitettynä kukin indeksitermi osoittaa dokumentteihin, joissa kyseinen termi esiintyy. Käänteistiedosto on nyt näiden indeksitermien lajiteltu lista. [23]

Käänteistiedoston periaatetta on korkealla tasolla havainnollistettu kuvassa 2.5. Kuvan oikeassa laidassa on esitetty kaksi yksinkertaista tiedostoa, jotka sisältävät latinankielistä tekstiä. Perinteisen latinalaisten kirjainten aakkoston mukaan järjestettynä dokumenteista saadaan indeksitermeiksi kuvan vasemmassa laidassa näkyvä lista. Kuvan esimerkki on varsin yksinkertainen, mutta antaa kuitenkin viitteitä indeksoitavien dokumenttien esikäsittelystä, joka usein on tärkeää, jotta käänteistiedosto ei kasva liian suureksi ja pysyy käyttökelpoisena. Kuvassa ei ole otettu kantaa indeksitermien painoarvoihin tai muihin ominaisuuksiin (esimerkiksi indeksitermien esiintymien lukumäärä), joita indeksitermeihin on mahdollista liittää.

Kuten kuvasta huomataan, isot alkukirjaimet on sanoista poistettu. Siis ”romanus” riittäisi hakutermiksi, jos haluttaisiin etsiä dokumentteja, jotka sisältävät sanan ”Romanus”. Myös erikoismerkit kuten pisteet on poistettu. Esimerkin dokumenttien kieli on latina, mutta se voisi yhtä hyvin olla jokin toinen kieli, tai kieliä voisi olla useampikin. Ainakin latinan kielen sana ”et” on sellainen, joka esiintyy hyvin useasti latinankielisessä tekstissä ja ainoastaan konjuktion merkityksessä. Täten hakua ajatellen sanaa ”et” ei välttämättä kannattaisi ottaa mukaan indeksiin, sillä on intuitiivisesti selvää, ettei se hakuterminäkään olisi kovin merkityksellinen. Edellä mainittuja dokumenttien esikäsittelyyn liittyviä toimenpiteitä on kuvattu tarkemmin luvussa 2.8.



Kuva 2.5: Käänteistiedoston periaate.

2.7.2 Haku käänteistiedostosta

Käänteistiedostosta etsivä hakualgoritmi kulkee seuraavien vaiheiden mukaisesti:

1. Hakulauseen termejä etsitään indeksitermien listasta
2. Indeksitermien kanssa yhteen osuneista hakutermeistä (osumista) muodostetaan lista
3. Osumia pitää vielä prosessoida mikäli on etsitty fraasia tai käytetty esimerkiksi Boolen operaattoreita

Mikäli hakulause koostuu yksittäisistä termeistä, riittää siis tuottaa lista osumista eli linkeistä dokumentteihin, joista hakutermi löytyy. Tietojärjestelmästä riippuen voi tietysti olla, että hakutermi ei ole yksiselitteinen, vaan voi täsmätä useampaan sanaan. Hakutermi voi olla esimerkiksi säännöllinen lauseke (engl. regular expression), jolloin se täsmäisi useampaan sanaan, ja kaikkiin näihin sanoihin liittyvät dokumentit tulee tällöin ottaa huomioon osumien listaa muodostettaessa. Fraasien tai monimutkaisempien kyselyjen tekeminen monimutkaistaa hakua jonkin verran, mutta on yhtäkaikki mahdollista käänteistiedostoa käyttäen.

Huomattakoon, että käänteistiedoston toteutus voi olla vaihdella tilanteen mukaan. Esimerkiksi indeksitermien lista voidaan pitää erillisessä tiedostossaan, jolloin se mahtuu helpommin muistiin ja on täten prosessoitavissa nopeammin. Baeza-Yates [5] huomauttaa, että indeksitermien aakkostamisesta luonnollisen kielen mukaiseen aakkosjärjestykseen ei kuitenkaan kannata luopua, sillä se mahdollistaa binäärihaun indeksistä ja tällöin algoritmin aikavaativuus on luokkaa $O(\log n)$.

Käänteistiedoston käyttö nopeuttaa hakua verrattuna dokumenttien skannaamiseen monta kertaluokkaa. Toki käänteistiedoston koko (luokkaa 10 – 100 prosenttia indeksoitavan tekstin koosta) ja käänteistiedoston jatkuvan ylläpidon tarve dokumenttien muuttuessa tekevät kuvasta vähemmän mustavalkoisen.

Hakutoiminnallisuus, hakulauseen täsmäys indeksiin ja siihen liittyvät operaatiot, voidaan nähdä kuvan 2.3 mukaisesti vastineena dokumenttien indeksoinnille. Hakutoiminnallisuuden toteuttavaa osaa tiedonhakujärjestelmästä voidaan kutsua esimerkiksi *hakumoottoriksi* (engl. search engine).

2.7.3 Indeksinhallinta

Indeksoinnilla viitataan toisinaan indeksi-tietorakenteen luomiseen; toisinaan sitä käytetään synonyyminä dokumenttien kokoamiselle indeksiä varten eli *indeksinhallinnalle*. Oleellinen osa indeksin ylläpitoa on lisätä uusia dokumentteja indeksin piiriin. Ihannetapauksessa indeksin piirissä olisivat kaikki dokumenttikokoelman dokumentit, mutta tämä ei tietenkään ole itsestäänselvyys.

Indeksinhallinnan idea on yksinkertainen, mutta toteutus riippuu paljolti tiedonhakujärjestelmästä ja siihen liittyvän dokumenttikokoelman laajuudesta ja monimuotoisuudesta. Täten indeksinhallinnan toteutus saattaa olla hyvinkin monimutkainen. Tämä saattaa olla syy sille, miksi alan oppikirjat [5, 25, 33, 23] eivät käsittele indeksinhallintaa kovin selkeästi omana kokonaisuutenaan.

Indeksinhallinnan tehtävät suoritetaan pääsääntöisesti kahdella eri tavalla: joko *noutamalla* tietoa (engl. pull) tai *vastaanottamalla* (engl. push). Tiedon hakeminen tarkoittaa, että tiedonhakujärjestelmä on vastuussa indeksoitavien dokumenttien löytämisestä. Järjestelmäarkkitehtuurin kannalta katsottuna tämä tarkoittaa sitä, että indeksinhallintaa ohjaava *indeksointimoottori* joutuu itse keräämään indeksoitavat dokumentit, toisin sanoen se joutuu tätä varten pyytämään muilta sen piirissä olevilta järjestelmiltä tai sovelluksilta tietoja niiden hallussa olevista dokumenteista. Tiedon vastaanottaminen taas merkitsee arkkitehtuurisesti päinvastaista: tiedonhakujärjestelmään liittyvät sovellukset ja järjestelmät rakennetaan niin, että ne osaavat itse syöttää dokumentteja tiedonhakujärjestelmälle indeksoitaviksi.

Indeksinhallinnan ja indeksointimoottorin katsotaan tässä tutkielmassa sisältävän myös indeksitietorakenteen ja sen muodostamiseen liittyvän toiminnallisuuden.

2.7.4 Indeksihallinta Webissä

Hypertekstin luonteesta johtuen Web-sivujen indeksinhallinnalla on omat erityispiirteensä. Se toimii toisaalta myös hyvänä käytännön esimerkkinä indeksinhallinnan toteutuksesta.

Lähtökohtana on, että Web-hakukoneen indeksointimoottori[7] pystyy käymään valtaosan Web-sivuista läpi, jos se lähtee liikkeelle jostakin paljon (korkealaatuisia) linkkejä sisältävästä sivusta, ja siirtyy sivun sisältämiä linkkejä pitkin rekursiivisesti käsittelemään aina uusia sivuja. Ensimmäisen sivun linkit lisätään käsittelyjonoon, ja niiden osoittamat sivut noudetaan (tässä yhteydessä englanninkielinen termi pull on suomenkielistä paljon kuvaavampi) HTTP-pyynnöillä ensimmäisen sivun tapaan. Tässä yhteydessä indeksoitaviin dokumentteihin voidaan soveltaa staattisia tiedonhakumalleja, kuten PageRank-algoritmia.

Käsittelyjonossa olevia sivuja voidaan jakaa eri palvelimille, ja Webin valtavan sivumäärän huomioon ottaen tämä on oikeastaan välttämätöntä. Kaikkia sivuja ei tietenkään saavuteta: ensinnäkään kaikille sivuille ei viittaa yksikään linkki; toisaalta aloitussivun pohjalta lähtevä linkkiketju ei välttämättä yllä kaikille sivuille; kolmanneksi aika saattaa tulla esteeksi, mikäli jotkut sivut ovat pitkän linkkipolun päässä. Hawkingin [30] mukaan Web-hakukoneiden indeksoiman datan määrä oli vuonna 2006 suuruusluokaltaan 400 teratavua, ja valistunut arvaus on, ettei tämä datamäärä ainakaan ole pienentynyt.

Tiedonhakujärjestelmä ei ole ainoana osapuolena rasituksen alla, vaan indeksointimoottorin toiminta asettaa paineita myös Web-sivuille. Indeksointimoottorin sivupyyntö aiheuttaa Web-palvelimelle rasitusta siinä missä ihmiskäyttäjänkin sivupyyntö. Tästä syystä indeksointimoottori ei voi kaiken aikaa päivittää indeksiä, vaan sivupyyntöjä pitää hillitä ajatellen myös Web-palvelimia.

2.8 Dokumenttien esikäsittely

Kaikkien dokumenttikokoelmassa olevien dokumenttien sisältämien sanojen indeksointia indeksiin kutsutaan *kokoteksti-indeksoinniksi* (engl. full-text indexing) ja näin muodostunutta indeksiä *kokoteksti-indeksiksi* (engl. full-text index). Tällaisessa tapauksessa indeksiin kuitenkin tulee mukaan haun kannalta merkityksettömiä sanoja (englannin kielen artikkelit ”a” ja ”the” ovat hyviä esimerkkejä). Nämä haun kannalta merkityksettömät sanat lisäävät tiedonhakuprosessin vaikeutta: ne ensinnäkin hidastavat hakual-

[7] Webin yhteydessä indeksointimoottoria nimitetään usein englannin kielessä termeillä (search) robot, spider tai crawler. Suomen kielessä kuvaava termi voisi olla Web-robotti tai hakurobotti.

goritmeja, koska indeksin koko kasvaa, ja toiseksi helposti tuovat hakutuloksiin epärelevantteja dokumentteja. Mikäli tällaisia sanoja kuitenkin karsitaan indeksoitavien termien joukosta, voi tämä johtaa epätoivottuihin tuloksiin. Esimerkiksi jokin dokumentti saattaa sisältää lauseen ”the house of the lord”. Kyseinen lause sisältää monta ilmentymää the- ja of-sanoista, jotka usein jätetään indeksoimatta. Täten käyttäjä, joka muistaa kyseisen lauseen esiintyvän dokumentissä, ihmettelee, miksi dokumenttia ei löydykään ensimmäisten hakutulosten joukosta. [5]

Vaihtoehtona on tietenkin indeksoida dokumenttien kaikki sanat välittämättä semanttisesti vähämerkityksisten sanojen tuomasta ”taustahälystä”. Näin tehdään Baeza-Yatesin [5] mukaan monessa Web-hakukoneessa, koska se helpottaa hakuprosessia ja tekee hausta intuitiivisemman peruskäyttäjälle. Usein näiden kahden vaihtoehdon väliltä on kuitenkin löydettävissä jokin kompromissiratkaisu.

Seuraavissa aliluvuissa esitellään tärkeimmät dokumenttien esikäsittelyyn liittyvät vaiheet. Epärelevanttien sanojen, ns. sulkusanojen poiston lisäksi esitellään kaikkia muita operaatioita yleensä edeltävä leksikaalinen analyysi, sekä sanavartaloiden erottaminen, indeksitermien valinta ja synonyymisanastot.

2.8.1 Leksikaalinen analyysi

Leksikaalisessa analyysissa (engl. lexical analysis) dokumenttien teksti (sarja peräkkäisiä merkkejä) muutetaan sarjaksi peräkkäisiä sanoja. Jokainen näin muodostettu sana on luonnollisesti kandidaatti indeksitermiksi.

Sanojen erottelu tekstistä on enemmän kuin välilyönnein eroteltujen merkkiryhmien erottamista. Erityisesti erilaiset välimerkit täytyy ottaa huomioon, mikä usein tarkoittaa niiden poistamista, sillä niistä ei ole erityistä hyötyä indeksoinnissa. Myös numerot jätetään usein indeksoinnin ulkopuolelle. Väliviivojen ja muiden sanoja yhdistävien merkkien kanssa tulee olla tarkkana: usein riippuu täysin tapauksesta, milloin esimerkiksi F-18 (paremmin Hornet-nimellä tunnetun hävittäjän tyyppikoodi) tulee indeksoida erillisenä sanana. Isot alkukirjaimet eivät yleensä ole tärkeitä, joten ne muutetaan pieniksi kirjaimiksi. Tässäkin voi olla tapauskohtaisia eroja, esimerkkinä vaikkapa C-ohjelmointikielen komennot, joita kirjoittaessa isot ja pienet kirjaimet ovat merkityksellisiä. [5]

2.8.2 Sulkusanojen poisto

Sulkusanat (engl. stopwords) ovat sanoja, jotka nähdään indeksoinnin kannalta liian yleisiksi, jolloin ne soveltuvat huonosti indeksi- ja hakutermeiksi. Baeza-Yatesin mukaan [5] oikeastaan kaikki sanat, joita esiintyy yli 80 prosentissa dokumenttikokoel-

man dokumenteista, ovat liian yleisiä indeksoitaviksi. Tällaisessa tapauksessahan hakulausetta vastaavien dokumenttien joukko muodostuisi helposti liian suureksi, toisin sanoen hakutulosten joukossa olisi todennäköisesti paljon epärelevantteja dokumentteja. Kaikkien sulkusanojen ei tarvitse olla artikkeleita tai muita semanttisesti vähämerkityksisiä sanoja – jotka toki useimmiten ovat hyviä kandidaatteja sulkusanoiksi – vaan sulkusanoiksi voidaan määritellä myös muutoin haun kannalta epäoleellisia sanoja, esimerkiksi tiettyjä verbejä tai adjektiiveja.

Sulkusanojen poisto paitsi helpottaa relevanttien – ja vain relevanttien – sanojen indeksointia myös pienentää indeksin kokoa oleellisesti, Baeza-Yatesin [5] mukaan tyypillisesti jopa 40 prosenttia tai enemmän. Kuten aikaisemmassa esimerkissä on mainittu, sulkusanojen käyttö voi myös aiheuttaa haittavaikutuksia. Formaalisti ilmaistuna sulkusanat voivat pienentää saantia. Esimerkiksi hakulauseen ”to be or not to be” kaikki sanat saattaisivat monessa tiedonhakujärjestelmässä joutua sulkusanojen listalle, jolloin käyttäjällä olisi hyvin vähäiset mahdollisuudet löytää tällaisella hakulauseella dokumentteja.

2.8.3 Sanavartaloiden erottaminen

Indeksitermien leksikaalisen analyysin lisäksi luonnollisten kielten sanojen kohdalla analyysin täytyy mennä vielä pidemmälle, jotta indeksoinnista saadaan täysi hyöty irti. Luonnollisissa kielissä käytetään erilaisia etu- ja takaliitteitä, joita liitetään sanavartaloon muuttamaan esimerkiksi sanan sijamuotoa. *Sanavartaloiden erottaminen* (engl. stemming) ja käyttäminen indeksitermeinä erilaisten taivutusmuotojen sijasta saattaisi siis edesauttaa haun tehokkuutta. Ainakin indeksin koon pieneneminen on tämän menetelmän ilmeinen etu.

Baeza-Yatesin [5] mukaan ei ole aivan selvää näyttöä siitä, saadaanko sanavartaloiden erottamisella varsinaista hyötyä hakua ajatellen vai ei. Hänen mukaansa tämä on syynä esimerkiksi siihen, etteivät monet Web-hakukoneet tue sanojen taivutusmuotoja. Tästä suomalaisittain erityisen hyvänä esimerkkinä on ollut Google, joka ei ole tukenut suomalaisten taivutusmuotojen käyttöä haussa. Pieni testi Googlella osoittaa, että tämä tilanne on muuttumassa, mutta historiallisessa mielessä taivutustuen puuttuminen on merkittävä seikka.

Sanavartaloiden erottamiseen liittyvät algoritmit liittyvät vahvasti luonnollisten kielten tutkimukseen. On helppo nähdä, että luonnollisten kielten usein suuretkin eroavaisuudet tekevät ongelmasta varsin haastavan. Esimerkiksi Web-hakukoneiden kansainvälistyessä ei englannin esimerkiksi suomeen verrattuna varsin yksinkertaisten taivutusmuotojen tukeminen enää riitä.

2.8.4 Indeksitermien valinta

Kokoteksti-indeksoinnissa kaikkia tekstissä esiintyviä sanoja käytetään indeksitermeinä, pois lukien tietenkin sulkusanat. Esimerkiksi kirjastotieteissä indeksitermit on voitu perinteisesti valita käsin, mutta myös automatiikkaa voidaan soveltaa *indeksitermien valintaan*. Substantiiveillä on usein suuri semanttinen merkitys, joten niiden poimiminen muiden sanojen joukosta saattaisi osoittautua hyödylliseksi. Tätä voidaan vielä yleistää poimimalla indeksitermeiksi substantiiviryhmiä yksittäisten substantiivien sijasta. Täten ei sivuutettaisi esimerkiksi englanninkielessä yleistä tapaa esittää yhdyssanat kahden erillisen sanan sanaliittoina, esimerkiksi yhdyssanassa ”computer science”. [5]

2.8.5 Asiasanastot

Asiasanastojen (engl. thesauri, yks. thesaurus) voidaan nähdä vievän indeksitermien valinnan periaatetta vielä askeleen pidemmälle. Asiasanastot koostuvat listasta kyseiselle ongelmakentälle keskeisiä luonnollisen kielen sanoja, termejä. Jokaista tämän listan sanaa kohti on lisäksi lista ko. sanan synonyymeistä, tai yleisimmin ko. sanaan jollakin tavalla liittyvistä sanoista tai lauseista. Asiasanasto ei siis välttämättä listaa pelkästään sanojen synonyymejä. [5]

Foskettin [22] mukaan asiasanastojen tehtävä on standardin sanaston luominen indeksointia ja hakua varten. Asiasanasto on eräs ontologioiden teorian sovellus, ja muiden ontologioiden tavoin se lähentää tietokoneen ja ihmisen käyttämää kieltä keskenään: indeksointiautomatiikka ja käyttäjä voivat puhua asioista samoilla termeillä. Asiasanastoissa voidaan myös luokitella sanoja hierarkioittain, jolloin ne voivat toimia myös apuna muokatessa käyttäjän hakulausetta yleisemmäksi tai tarkemmaksi.

Asiasanasto sopii parhaiten ongelmakenttään, jossa käsiteltävän tekstin sisältöä on tavalla tai toisella rajoitettu. Esimerkiksi lääketieteellisellä alalla käytettävä sanasto synonyymeineen voidaan rajata melko tarkkaankin. Toisaalta esimerkiksi Webin tapauksessa yhtenäisen sanaston löytäminen on liki mahdoton tehtävä. Toisaalta navigaatioperusteisten hakukoneiden tapauksessa taustalla voidaan nähdä olevan jonkinlainen yhteinen sanasto, ja tässä mielessä asiasanastojen idea soveltuu Webiinkin. [5]

2.9 Tiedonhakujärjestelmän evaluoinnista

Tiedonhakujärjestelmän tehokkuutta, so. kykyä löytää hakuehtoja vastaavia relevantteja dokumentteja, tulee voida evaluoida. Minkä tahansa järjestelmän evaluointiin kuu-

luu järjestelmän toiminnallisuuksien arviointi, so. vastaako järjestelmä niitä määrityksiä, joiden perusteella se rakennettiin. Tiedonhakujärjestelmälle tärkeimpiä laadullisia ominaisuuksia ovat vasteaika ja järjestelmän toimiakseen vaatima tila. Tiedonhakujärjestelmän kykyä noutaa mahdollisimman relevantteja dokumentteja on erityisen oleellista tarkastella. Tätä kutsutaan hakutehokkuuden evaluoinniksi (engl. retrieval performance evaluation). Hakutehokkuuden evaluointi tehdään yleensä jotain testattavien dokumenttien kokoelmaa, *testikokoelmaa* (engl. test reference collection) vastaan. Testikokoelma sisältää dokumenttien lisäksi esimerkinomaisia hakutehtäviä kokoelmaa vastaan, sekä asiantuntijoiden jokaista hakutehtävää vastaan valitsemat "oikeat vastaukset". Yksi merkittävimmistä testikokoelmista on TREC[8]. [5]

Hakutehtävä, jonka avulla hakutehokkuutta evaluoidaan, voi olla hakulause tai esimerkiksi selailupohjaisen käyttöliittymän istunto. Hakuympäristö (käyttöliittymä, testikokoelman dokumentit, ym.) vaikuttaa osaltaan siihen, miten evaluoinnin tuloksiin tulisi suhtautua. Hakutehokkuuden evaluoinnissa tarkastellaan *saantia,* joka on vastauksena saatujen relevanttien dokumenttien suhde kaikkiin relevantteihin dokumentteihin, ja *tarkkuutta,* joka on vastauksena saatujen relevanttien dokumenttien suhde kaikkiin vastauksena saatuihin dokumentteihin. Olkoon R on relevanttien dokumenttien joukko ja $|R|$ näiden dokumenttien lukumäärä. Olkoon A vastaukseksi saatujen dokumenttien joukko jollakin hakustrategialla käyttäen hakulausetta I, ja olkoon $|A|$ tämän joukon dokumenttien lukumäärä. Olkoon lisäksi Ra joukkojen R ja A leikkaus ja $|Ra|$ tämän joukon dokumenttien lukumäärä. Täten

$$\text{saanti} = \frac{|Ra|}{|R|}$$

ja

$$\text{tarkkuus} = \frac{|Ra|}{|A|}$$

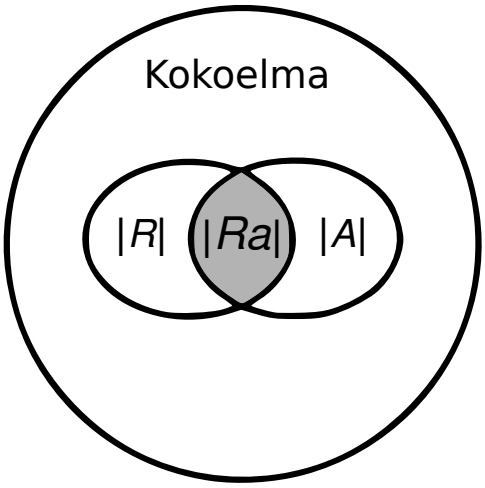
Kaavoja on havainnollistettu kuvassa 2.6. [5]

2.10 Muita tiedonhaun osa-alueita

Tiedonhaku tutkii myös muita osa-alueita, jotka esitellään tässä vain lyhyesti.

Kyselykielten tutkimus pyrkii formalisoimaan erilaisia hakulauseen syöttämiseen tarkoitettuja menetelmiä. Datahaun kohdalla SQL on usein ensimmäiseksi vastaan tuleva esimerkki kyselykielestä. SQL kuten muutkin datan kyselykielet pyrkivät kuitenkin hakemaan eksaktin joukon hakutuloksia, kun taas tiedonhaussa

[8] http://trec.nist.gov/



Kuva 2.6: Saanti ja tarkkuus [5].

on usein käytännöllisempää etsiä mahdollisimman hyvin hakulauseeseen liittyvät hakutulokset [5]. Tiedonhaun kyselykielten kannalta XML-pohjaiset sovellukset kuten XQuery ja XPath ovat saaneet kiinnostusta osakseen, koska XML on luonteva tapa rakenteisen tiedon mallintamiseen, ja sen avulla on mahdollista määritellä myös metatietoja ja semantiikkaa [36].

Luonnollisten kielten tutkimus on tiiviissä yhteydessä erityisesti tekstihakuun, onhan suurin osa elektronisessakin muodossa olevasta tekstistä jonkin luonnollisen kielen mukaista. Taivutusten huomioonottaminen, yhdyssanat, eri kielten sekoittaminen samassa dokumentissa ja merkistökoodaukset ovat muutamia esimerkkejä tiedonhakua kiinnostavista tutkimuksen kohteista. [5]

Tekstin pakkaus on merkityksellistä indeksoinnin tehostamisen kannalta. Mitä tiiviimpään kokoon indeksi saadaan tallennettua, sitä tehokkaammin se on esimerkiksi ladattavissa tietokoneen muistiin ja siten käsiteltävissä. Nykyaikaisten tietokoneiden suorituskyky on jo sitä luokkaa, ettei tekstin pakkaamiseen käytettävien algoritmien suoritusaika välttämättä muodostu esteeksi. [5]

Tekstin tiivistys Tiedon pakkaamista voidaan tutkia myös korkeammalla abstraktiotasolla: esimerkiksi klusterointimenetelmät hyötyvät tehokkaasta tavasta tiivistää dokumenttien sisältöä vaikkapa termilistaksi tai yleisesti ”luettavaksi” lyhennelmäksi. [45]

Multimediahaku Multimedialla tai multimediadokumenteilla tarkoitetaan tässä yhteydessä ääntä, kuvaa tai muuta ei-tekstuaalista informaatiota sisältäviä dokumentteja. Kuten todettua, haku tällaisista dokumenteista onnistuu myös teksti-

haun menetelmin, mikäli haku kohdistetaan dokumentteihin liitettyyn metadataan. Aina tällaista metadataa ei kuitenkaan ole käytettävissä (esimerkiksi sitä ei yksinkertaisesti ole määritelty) tai saatetaan haluta porautua syvemmälle itse multimediadokumenttien sisältöön. Multimediahaun tutkimus yrittää haastata tähän haasteeseen, joka on yhä merkityksellisempi multimediasisällön lisääntyessä. [25]

Työpöytähaku Vuonna 2004 The New York Times lehdessä [41] esiteltiin Googlen uutta aluevaltausta: Googlen Web-haku oli tuotteistettu työpöytäkoneelle asennettavaksi ja koneen tiedostoja hakevaksi sovellukseksi. Työpöytähaun kohteena on työpöytäkoneen sisältämä tieto. Se joutuu siis ottamaan kantaa Web-dokumentteja heterogeeniseempään dokumenttien joukkoon. Erityisesti tässä suhteessa työpöytähaku on lähellä seuraavassa luvussa esiteltävää organisaation tiedonhakua.

3 Organisaation tiedonhaku

Luvussa perehdytään saatavilla oleviin lähteisiin sekä luvussa 2 esiteltyyn teoriaan perustuen organisaation tiedonhaun tutkimusongelmiin. Luvun lopussa pyritään hahmottamaan, miltä organisaation tiedonhakujärjestelmän arkkitehtuuri voisi näyttää. Tämä tehdään keskittyen yksinomaan organisaation tiedonhaun tutkimuksen esiin tuomiin näkökohtiin, ja vastapainoksi organisaation tiedonhaun kaupallista puolta valotetaan luvussa 4 perehtymällä kaupallisiin organisaation tiedonhaun ratkaisuihin.

3.1 Organisaation tiedonhaun tutkimus

Organisaation tiedonhaku (engl. enterprise search) ei terminä ole vielä kovin vakiintunut, vaan puhutaan tiedonhausta yritysten Webistä tai intranetistä [21], tai tiedonhausta yrityksissä (engl. "information retrieval in business") [26, 10, 11]. Yhtenä harvoista Hawking [29] tuntuu vakiinnuttaneen termin organisaation tiedonhaku omaan tutkimustyöhönsä. Organisaation tiedonhausta puhuvat myös Mukherjee ja Mao [37], mutta ainakin kyseenomaisessa tutkimuksessaan siteeraavat pääasiassa Hawkingia. Organisaation tiedonhaun ala on jatkuvassa muutoksessa ja sitä ohjaavat osaltaan myös markkinavoimat, joten yhtenäisen sanaston ja kieliopin löytäminen on tutkimuksellisesti tärkeä alkuaskel. Hawking [29] toteaa, että Web-hakuun liittyvää kirjallisuutta on olemassa enemmän kuin organisaation tiedonhakuun liittyvää. Organisaation tiedonhakua käsitteleviä kovakantisia opuksia odotellessa on tärkeää saada muodostettua perusteoria aiheesta tutkimustyön kautta.

Termi organisaation tiedonhaku on kirjoittajan käännös englannin kielen sanaliitosta "enterprise search". Kuten todettua, organisaation tiedonhaku on niin tuore termi, ettei sille löydy yksiselitteistä määritelmää alan kirjallisuudesta saati vakiintunutta käännöstä suomen kielelle. Termin kuvaamaa käsitettä voidaan kuitenkin lähteä määrittelemään perehtymällä ensin organisaation tiedonhaun taustakäsitteisiin. Tullaan huomaamaan, että termin käännös englannista suomeen ei ehkä ole sanatarkka, mutta merkitykseltään lähellä alkuperäistä.

3.2 Organisaation tiedonhaun taustakäsitteet

Luvussa 2 havainnoitiin suomen kielen sanojen tieto ja data yhteyttä toisiinsa, vaikka ne eivät teknisesti tarkoitakaan aivan samaa. Niitä kuitenkin pidetään usein arkisessa ja akateemisessakin kielenkäytössä itsestäänselvyyksinä. Organisaation tiedonhaku sisältää jo ääneen lausuttaessa kaksi huomiota herättävää sanaa, joita niitäkään ei ole syytä ohittaa. Nämä sanat ja muut taustakäsitteet on syytä ensi alkuun ymmärtää. Tiedonhaku on määritelty luvussa 2, mutta organisaatio vaatii oman esittelynsä. Myös muut seuraavissa alaluvuissa määritellyt termit tulevat usein vastaan organisaation tiedonhakuun liittyvässä kirjallisuudessa. Seuraavissa alaluvuissa määritellään taustakäsitteiden esittelyn lopuksi itse organisaation tiedonhaun käsite.

3.2.1 Organisaatio

Mukherjee ja Mao [37] määrittelevät englannin sanan enterprise tarkoittavan mitä tahansa kollaboratiivista *organisaatiota*, johon liittyy omisteista informaatiota. Tällainen organisaatio voi olla kaupallinen, akateeminen, valtionhallinnollinen tai voittoa tavoittelematon. Organisaatiota ei siis ole syytä rajoittaa tarkoittamaan ainoastaan kaupallista, voittoa tavoittelevaa liikeyritystä, kuten englannin sana enterprise usein tulkitaan. Sana enterprise on tässä tutkielmassa käännetty sanaksi organisaatio, ja sillä on Mukherjeen ja Maon kuvaama laajempi merkitys.

3.2.2 Intranet

Intranet on varsin yleinen termi, mutta sille on vaikeaa löytää täsmällistä määritelmää. Tämän tutkielman kannalta riittää kuitenkin seuraava Lambin ja Davidsonin [34] vertaus Internetiin: intranetit ovat kehittyneet 1990-luvun puolesta välistä lähtien alun perin yritysten toimesta; Internetin perustekniikoita on käytetty pystyttämään lokaaleja ja globaaleja verkkoja organisaatioiden sisällä.

Periaatteessa intranet voi tarkoittaa mitä tahansa muutakin Internet-protokollaa käyttävää yksityistä verkkoa, mutta usein se rinnastetaan jonkin organisaation Internetiin liitettyyn sisäverkkoon. Oleellista on, että verkon käyttö on rajoitettu vain tietylle käyttäjäryhmälle, esimerkiksi tietyn yrityksen käyttäjille. [52]

3.2.3 Extranet

Jos intranetille on vaikeaa löytää määritelmää, on *extranet* vielä vaikeampi tapaus. Watson [52] antaa extranetille määritelmän tukeutuen intranetin määritelmään: Kun

intranetiin annetaan pääsy jollekin organisaation ulkopuoliselle käyttäjälle tai käyttäjäryhmälle, kutsutaan tällä tavoin muodostuvaa verkkoa silloin usein extranetiksi. Tämän tutkielman kannalta tämä intranetin ja extranetin hienoinen, joskin verkkoteknisesti ja tietoturvan kannalta merkittävä, ero ei ole huomionarvoinen. Useimmat alan tutkimukset keskittyvät puhumaan yksinomaan intraneteistä, ja tätä tapaa noudatetaan pääsääntöisesti tässäkin tutkielmassa.

3.2.4 Portaali

Portaalit ovat alun perin kehittyneet hakukoneiden ympärille. 1990-luvun puolivälissä Internetin hakukoneisiin alettiin liittää mahdollisuuksia selata kategorioittain lajiteltuja linkkihakemistoja perinteisen sanahaun lisäksi. Näiden oheen ryhdyttiin liittämään erilaisia lisätoimintoja, kuten virtuaaliyhteisöjä tai keskustelupalstoja. Hakukonesivuston ulkonäön muokkaaminen persoonalliseen makuun sopivaksi tuli mahdolliseksi, ja pääsy esimerkiksi kaupalliseen aineistoon tehtiin mahdolliseksi [19]. Esimerkkejä nykyaikaisista Web-portaaleista ovat Yahoo![1] ja MSN[2].

On tietenkin vaikeasti selvitettävissä, missä vaiheessa portaalien kehityskaarta portaalien luontoisia sivustoja tarkalleen ottaen alettiin kutsua portaaleiksi. Voidaan sanoa, että perinteisen Web-sivuston pääsivuineen ja portaalin välinen raja on melko häilyvä. Yritysmaailma kuitenkin omaksui portaalin käsitteen ja portaaleita ryhdyttiin rakentamaan intranet-ympäristöihin [19]. Tällaisia portaaleita nimitetään yleensä yritysportaaleiksi tai organisaation omasta näkökulmasta usein intranet-portaaleiksi tai yksinkertaisesti, joskin jo varsin harhaanjohtavasti, intraneteiksi (engl. business portal, corporate portal, corporate information portal, enterprise information portal eli EIP tai intranet portal). Liike-elämän kaupallisten intressien vaikutus portaalien kehityksessä näyttäisi olevan varsin ominaista portaalien kehitykselle. Kädenvääntöä on tietenkin käyty siitä, mikä nimitys on oikea, tai mikä tuote milloinkin täyttää portaalin määritelmän.

3.2.5 Organisaation tiedonhaku

Hawking [29] määrittelee *organisaation tiedonhaun* sisältävän

- minkä tahansa organisaation, jolla on tekstuaalista aineistoa elektronisessa muodossa,
- organisaation julkiseen Web-sivustoon kohdistuvaa hakua,

[1] http://www.yahoo.com
[2] http://fi.msn.com

- organisaation sisäisiin Web-sivustoihin (tai yleisemmin intranetiin) kohdistuvia hakuja,
- muun elektronisen tekstuaalisen aineiston hakua (sähköpostit, tietokanta-aineisto, levyjaoilla säilytettävät dokumentit, ja niin edelleen).

Huomattakoon, että Hawkingin määritelmän mukaan haettavan tiedon luonne on tekstuaalinen. Multimedian – esimerkiksi video- tai ääniaineiston etsiminen muun kuin siihen liitetyn tekstuaalisen metatiedon avulla on siis hänen määritelmänsä ulkopuolella. Multimediaan liitetty metadata voi kuitenkin toimia haussa varsinaisen tekstuaalisen sisällön korvikkeena, jolloin myös multimedia saadaan mukaan haun piiriin.

3.3 Tiedonhaun merkitys organisaatioille

Organisaatiot pyrkivät liittämään hallussaan olevan tiedon yhteen useimmiten portaalin välityksellä, jotta työntekijät pääsevät tehokkaimmin hyötymään siitä. Usein organisaatioiden ongelma on, että tieto on hajallaan pitkin organisaatiota, eikä sitä näin ollen kyetä riittävän tehokkaasti hyödyntämään. Organisaation tiedonhaku on yksi luonnollinen elementti tiedosta perille pääsemiseen, aivan kuten Web-hakukoneet ovat olleet Webin tietotulvan kannalta. Hawkingin [29] mukaan organisaation tiedonhaun suurin haaste ei kuitenkaan ole Webin kaltainen tietotulva, vaan tiedon hajautuminen eri puolille organisaatiota. Organisaatiosta riippuen tiedon määräkin voi toki olla varteenotettava haaste. Organisaatioiden kaikki tieto ei vielä 2000-luvullakaan ole digitaalisessa muodossa. Ainoastaan organisaation hallussa olevalla digitaalisella tiedolla on edellytys olla haun piirissä, joten tiedon saattaminen digitaaliseen muotoon on ensimmäinen askel. Toisaalta kokotekstihakua kehittyneemmät hakumenetelmät kuten semanttisen haun sovellutukset usein edellyttävät rakenteellisuutta, ja rakenteellisen tiedon osuus organisaatioiden kaikesta digitaalisesta tiedosta on edelleen heikoissa kantimissa [37].

Organisaation tiedonhakua, kuten muitakin organisaation digitaaliseen tietoon kohdistuvia prosesseja, voidaan tukea huolehtimalla tiedon eheydestä. Esimerkiksi erilaiset raportointi- ja päätöksentekojärjestelmät hyötyvät eniten siististä, organisoidusta ja jäsennellystä tiedosta. Organisaatioiden mahdollisesti valtavien tietomäärien manuaalinen eheyttäminen tai rakenteellisuuden lisääminen on työlästä, joten apuun tarvitaan automaattisia menetelmiä. Esimerkiksi tiedon eristämistä (engl. Information Extraction) tai tekstitiedonlouhintamenetelmiä (engl. Text Mining) voidaan yrittää hyödyntää tiedon rakenteellisuuden lisäämiseksi. Tiedon eristäminen, rakenteellisen informaation seulominen rakenteettomasta datasta, on sidoksissa tiedon laatuun yleensä, erityisesti, jos operaatiota halutaan automatisoida. Tätä lähellä oleva tekstitiedonlou-

hinta käyttää hyödykseen esimerkiksi luonnollisten kielten rakenteita, tilastollisia tai rakenteellisen analyysin tietoja etsiäkseen informaatiota tekstistä. Vastaavalla tavalla tekstitiedonlouhinnan prosessit helpottuvat, kun prosessoitavana oleva tieto on mahdollisimman laadukasta. [37]

Organisaation tiedonhaun ratkaisujen kehittäminen tehokkaammiksi tarjoaisi organisaatioille suurta taloudellista hyötyä. Tästä organisaation tiedonhaun tutkimus näyttäisi olevan yksimielinen [29, 37, 46]. Taloudelliset perusteet löytyvät, kun havainnoidaan, miten dokumenttien etsimiseen käytetään valtavasti työaikaa yrityksissä. Gordonin [26] yhteenvedosta käy ilmi, että yritysten henkilöstö käyttää jopa 25 prosenttia päivittäisestä työajastaan dokumenttien jakeluun, täyttämiseen ja etsimiseen. Jos Gordon pääsi tällaisiin lukuihin vuonna 1997, kuvailee Blair [10] tilannetta vähintäänkin yhtä vakavaksi. Hän siteeraa Interleafin tutkimusta, jonka mukaan yrityksen johto käyttää 40 prosenttia työajastaan pelkästään dokumenttien käsittelyyn. Blair viittaa myös Gartnerin tutkimukseen, jonka mukaan jopa 90 prosenttia yrityksen informaatiosta olisi tallennettuna dokumentteihin. Gartner ei tässä yhteydessä erottele, onko kyse digitaalisista dokumenteista, mikä saattaa selittää, miksi tulokseksi saatu prosenttiluku kuulostaa suurelta. Tiedonhaun ratkaisemiseen organisaatioissa lienee kuitenkin syytä suhtautua vakavasti, vaikka digitaalisia dokumentteja koskeva osuus tästä olisi puoltakin pienempi.

Organisaation tiedonhaun markkinoiden koko kertoo jotain alan tärkeydestä ainakin liike-elämälle. Gartnerin tuoreemman markkinatutkimuksen[3] mukaan organisaation tiedonhaun markkinat kasvavat 1,2 miljardiin dollariin liikevaihdossa mitattuna vuoteen 2010 mennessä. Myös Gartnerin visiota ja ketteryyttä kuvaava nelikenttä syyskuulta 2009 [3] organisaation tiedonhaun toimijoista on samoilla linjoilla: vuoden 2013 ennuste markkinoiden koolle on jo 1,9 miljardia dollaria.

3.4 Organisaation tiedonhaun suhde Web-tiedonhakuun

Organisaation tiedonhakua koskevassa tutkimuksessa Web-tiedonhaku otetaan usein vertailukohdaksi tutkimukselle (muiden muassa [29, 37, 21]). Web-tiedonhaku lienee tärkein tiedonhaun alaan viime aikoina vaikuttanut suuntaus, ja tästä syystä Mukherjeen ja Maon [37] huomio siitä, että organisaation käyttäjät odottavat organisaation tiedonhaulta Web-haun kaltaisia ominaisuuksia ja vastaavaa tehokkuutta, on looginen.

Organisaation intranet-portaali, osa organisaation intranetia, on organisaation julkisia Web-sivuja lukuun ottamatta organisaation eniten Webiä muistuttava kokonai-

[3] http://www.gartner.com/it/page.jsp?id=596407

suus. Intranet-portaalien ansiota lienee, että organisaation tiedonhakua lähdetään usein tarkastelemaan Web-haun näkökulmasta. Intranet-portaalin ja intranetin piiriin kuuluvaa hakua oli Hawkingin [29] mukaan tutkittu jo vuoteen 2004 mennessä varsin laajalti. Tutkimusten johtopäätös kautta linjan on, että Internet ja intranet näyttävät päällisin puolin samankaltaisilta, mutta tarkasteltaessa niitä tiedonhaun näkökulmasta, löydetään oleellisia eroavaisuuksia. Fagin *et al.* [21] tiivistävät nämä eroavaisuudet neljäksi "aksioomaksi".

Aksiooma1 Intranet-dokumentit on usein luotu ennemminkin informaation esille tuomiseksi kuin minkään erityisen ryhmän kiinnostuksen herättämiseksi tai säilyttämiseksi. Esimerkiksi hakulause "Unohdin Windows-salasanani" johtaisi laajaan joukkoon vastauksia Internetissä, kun taas intranetissä relevantteja vastauksia saattaisi olla vain yksi.

Aksiooma2 Useille kyselyille on ominaista, että niihin on olemassa hyvin pieni joukko oikeita vastauksia (usein vieläpä uniikkeja), ja näillä uniikeilla vastauksilla ei yleensä ole mitään erityisiä piirteitä, jotka erottaisivat ne muita vastauksista.

Aksiooma3 Roskaposti ei käytännössä ole intranetien ongelma. Intranetien käyttäjäkunta ja sisältö on hyvin kontrolloitua, jolloin roskaposti ei pääse leviämään intraneteissä käytännössä lainkaan.

Aksiooma4 Laajat osuudet intraneteistä eivät ole hakukoneystävällisiä.

Naiivi ratkaisuehdotus organisaation tiedonhaun ongelmalle olisi rinnastaa organisaation kaikki digitaaliset dokumentit sen intranet-portaalissa sijaitseviin Web-dokumentteihin, mikä tarkoittaisi dokumenttien muuntamista hypertekstiksi. Tämän jälkeen dokumentit linkitettäisiin toisiinsa niin, että ne voidaan indeksoida Web-indeksinhallinnan keinoin esimerkiksi intranet-portaalin pääsivulta käsin. Näin Web-maailman ratkaisuja voitaisiin hyödyntää suoraan organisaation Webiin. Kaikkien dokumenttien muuttaminen hypertekstiksi muodostuisi kuitenkin helposti ylitsepääsemättömän työlääksi operaatioksi. Myös Faginin *et al.* aksioomat todistavat tätä ajatusta vastaan: Web-hakuun suunnitellut tiedonhakumallit eivät ilman muutoksia toimi tehokkaasti, sillä Intranetit eivät yksinkertaisesti ole riittävän samankaltaisia Internetin kanssa, jotta niissä sijaitsevien dokumenttien yhtäläisyydestä voitaisiin tehdä pitkälle meneviä olettamuksia.

Indeksinhallinta voi toki hyötyä Web-hakukoneiden indeksointimoottorien periaatteista, mutta aksiooma 4 viittaa organisaatioiden kompleksisiin tietoavaruuksiin: tieto on usein hajallaan heterogeenisissä tietolähteissä. Tämä vaatii erityistä huomiota indeksointimoottoreiden toteutukselta. Siinä missä Web-robotti voi indeksoida tietoa hyperdokumenttien avaruudessa pääosin yhden pseudoalgoritmin mukaisesti, ei se pääsisi

intranet-portaalista aloittaessaan (linkitettyjä) Web-dokumentteja pidemmälle. Vaikka intranet-portaalien tarkoitus onkin koota organisaation tietoa mahdollisimman kattavasti yhteen [2], ei silti ole lainkaan selvää, että kaikkeen organisaation tietoon on viitattu portaalissa. Kaikki organisaation intranet-dokumentit eivät välttämättä ole edes hypertekstidokumentteja (ne voivat olla esimerkiksi PDF- tai Microsoft Office -dokumentteja), eivätkä ne siten myöskään ole hyperlinkein yhdistettävissä esimerkiksi yrityksen kotisivuihin tai intranet-portaaliin.

Hakukäyttöliittymä ja tapa hakea tietoa ovat kuitenkin Web-tiedonhaussa ja organisaation tiedonhaussa lähellä toisiaan. Käyttäjän näkökulmasta intranet-portaalin ja Webin osalta ei ole välttämätöntä tehdä eroa. Käyttäjä voi hyvin olettaa, että intranet-portaalin hakukenttä toimii ja esittää hakutuloksia samalla tavalla kuin jonkin suositun Web-hakukoneen käyttöliittymä. Broder [13] on määritellyt taksonomian erilaisille Webin hakutavoille, joita käyttöliittymän tulee tukea. Hänen mukaansa perinteinen tiedonhaun tutkimus olettaa käyttäjän toimivan, koska tällä on yksinkertaisesti tarve löytää informaatiota. Web-haun ongelma on monisyisempi. Broder esittää kolme erilaista hakutyyppiä, joita Web-haussa esiintyy:

Navigationaalinen. Käyttäjä pyrkii saamaan selville tietyn Web-sivun osoitteen.

Informationaalinen. Käyttäjän tavoite on löytää informaatiota yhdeltä tai useammalta Web-sivulta.

Transaktionaalinen. Käyttäjä haluaa löytää Web-sivun, jolla hän voi suorittaa jonkin transaktion (esimerkiksi verkkokauppa).

Hawkingin [29] mukaan tämä Broderin taksonomia tulee kokonaisuudessaan edustetuksi organisaation tiedonhaussa. Hänen mukaansa myös Broderin esittämät näkemykset ”kolmannen sukupolven” käyttöliittymäominaisuuksista koskevat organisaation tiedonhakua. Organisaation tiedonhaun kannalta tärkeitä käyttöliittymäominaisuuksia ovat muun muassa hakulauseen oikoluku (engl. query spell checking), ennakoiva tekstinsyöttö (engl. autocompletion), hakutulosten esikatselu (engl. search results previewing) ja moninäkymähaku (engl. faceted search). Näistä erityisesti moninäkymähaku vaatii käyttöliittymän ulkopuolisia ominaisuuksia toimiakseen, koska siinä haettavat dokumentit tulee ensin luokitella metatiedoin siten että niitä voidaan käyttöliittymässä selata tehdyn luokittelun perusteella [51].

3.5 Organisaation tiedonhaun tyypilliset piirteet

Tiedonhaun perusteet – indeksointi, tiedonhakumallit, indeksinhallinta, käyttöliittymät, ja niin edelleen – ovat keskeisessä asemassa missä tahansa tiedonhakujärjestelmässä. Organisaation tiedonhaun tutkimukselle on luonnollista, että sen pääpaino on näiden tiedonhaun perusteiden tutkimisen sijaan organisaation tiedonhaun ominaisten piirteiden kartoituksessa ja tutkimuksessa. Organisaation tiedonhaun tyypilliset piirteet muovautuvat pääsääntöisesti haun käyttäjäkunnan – organisaatioiden – asettamien vaatimusten perusteella. Mukherjee ja Mao [37] luokittelevat piirteet seuraavasti:

3.5.1 Tietolähteiden ja -formaattien laaja kirjo

Tietolähteet (engl. repository) mielletään organisaation tiedonhaun tutkimuksessa usein tekniikaltaan, toteutustavaltaan tai sijainniltaan erilaisina informaation, tiedon varastoina, joiden sisältämä tieto, dokumentit, on saatava tiedonhakujärjestelmän piiriin. Järjestelmäarkkitehtuurin kannalta ne voivat olla erillisiä taustajärjestelmiä suhteessa varsinaiseen tiedonhakujärjestelmään. Tässä yhteydessä tietolähdettä ei kuitenkaan ole tarve kiinnittää tarkoittamaan ainoastaan jotain taustajärjestelmää, vaan yhtä hyvin tietolähteellä voidaan viitata johonkin abstraktimpaan tiedon kokonaisuuteen, kuten organisaation sähköpostiarkistoihin. Tärkeintä on joustaa määritelmässä niin, että yhdessä tietolähteiden voidaan nähdä muodostavan organisaation tiedonhaun dokumenttikokoelman. Mukherjee ja Mao luettelevat muutamia tyypillisimpiä tietolähteitä:

- Erilaiset yrityssovellukset kuten Microsoft Exchance, Lotus Notes tai Documentum.
- Web-sivut, joko intranetissä tai Internetissä.
- Tiedostojärjestelmät ja levyjaot.

Listaan on helposti löydettävissä uusiakin kandidaatteja:

- intranetit ja extranetit kokonaisuudessaan
- blogit
- wikit
- erilaiset uutissyötteet (engl. news feeds)
- sähköpostiarkistot
- postituslista-arkistot

- sisällönhallintajärjestelmät (engl. CMS eli Content Management Systems)
- tietokannat
- vanhemmat ohjelmat, jotka säilyttävät tietoa omissa datasiiloissaan
- ja niin edelleen.

Toisaalta *dokumenttiformaatteja*, dokumenttikokoelmaan kuuluvien dokumenttien erilaisia tyyppejä, voi olla lukuisia:

- HTML sekä XML eri sovelluksineen
- erilaiset yrityssovellusten tuottamat dokumentit kuten Microsoft Wordin DOC-formaatti
- pelkkä teksti (plain text)
- RTF
- PDF
- erilaiset omisteiset tiedostoformaatit
- binääritiedostot
- ja niin edelleen.

Lisäksi dokumenttien kieli voi vaihdella: teknisesti samassa dokumentissa voidaan käyttää useampaakin kieltä. Viimeistään englantia eksoottisempien kielten kohdalla myös dokumentin merkistökoodaus astuu kuvaan. On erittäin paljon oletettu, että kaikki organisaation dokumentit noudattaisivat vain yhtä merkistökoodaustapaa, kuten UTF-8:aa. Todellisuudessa merkistökoodauksia voi olla käytössä useita; on helppo kuvitella sellainenkin tilanne, ettei kaikista käytetyistä merkistökoodauksista olla organisaatiossa edes tietoisia.

Indeksoinnin ongelma ei organisaatiossa periaatteiltaan eroa muistakaan tiedonhaun aloista. Tietolähteiden ja formaattien laaja kirjo vaikuttaa kuitenkin erityisesti indeksinhallintaan ja dokumenttien esikäsittelyyn, koska kaikki tietolähteet ja niiden sisältämät dokumentit tulee saada indeksin piiriin. Ongelma voidaan ratkaista esimerkiksi niin, että indeksinhallinnan toteutuksia indekseineen ja hakualgoritmeineen on useampia, parhaimmassa tapauksessa yksi kutakin oleellisesti erilaista tietolähdettä kohden. Tietolähteisiin liittyvien rajapintojen lisäksi tietolähteiden sisältö, dokumentit voivat olla luonteeltaan erilaisia. Esimerkiksi sähköpostiarkiston dokumentit eroavat

organisaation intranet-portaalissa ylläpidetyistä dokumenteista. Indeksoinninhallinta joutuu organisaatioympäristössä toteuttamaan sekä nouto- että vastaanotto-tyyppistä (engl. push ja pull) dokumenttien keräämistä indeksin piiriin. Esimerkiksi intranet-portaalin Web-dokumentit voidaan noutaa HTTP-pyynnöin aivan kuin Webissäkin, mutta esimerkiksi sähköpostiarkiston tapauksessa saattaa olla helpompaa, että sähköpostia hallinnoiva järjestelmä jollakin mekanismilla aika ajoin työntää uusia sähköpostiviestien eriä indeksointimoottorille, joka vastaanottaa ja indeksoi ne.

Erilaisten tietolähteiden ja formaattien dokumentaation puute, epästandardin mukainen toteutus, tai jopa korkeat lisenssimaksut tuovat omat haasteensa. On selvää, että paraskin organisaation tiedonhakujärjestelmä voi indeksoida vain osan organisaation heterogeenisesti jakautuneesta tietomassasta.

3.5.2 Tietoturvan huomioiminen

Tiedonhakujärjestelmän käyttäjän oikeudet nähdä tietty dokumentti ovat tärkeässä asemassa organisaatioissa. Joissakin organisaatioissa saatetaan haluta rajoittaa jopa käyttäjän oikeuksia nähdä dokumenttia hakutuloksissa. Tällaisessa tapauksessa jo dokumentin olemassaolosta tietäminen rikkoisi organisaation tietoturvasäädöksiä.

Tiedonhakujärjestelmä voi esimerkiksi tukea Kerberos-autentikointia, jolloin järjestelmän käyttöliittymä voi välittää käyttäjätiedot Kerberosta tukeville tietolähteille ja saada vastauksena vain käyttäjän oikeuksia vastaavat dokumentit. Ongelmana on, että erilaisia käyttäjäntunnistus-, oikeuksienhallinta ja kertakirjausmenetelmiä on olemassa lukuisia – esimerkkeinä mainittakoon Kerberoksen lisäksi LDAP, RADIUS ja SSL. Taloudellisessa mielessä haastavuutta lisää se, etteivät kaikki näistä menetelmistä ole minkään avoimen standardin mukaisia, ja joissakin tapauksissa niiden käyttö edellyttää lisenssimaksujen suorittamista. Ei siis riitä, että tietoturva otetaan huomioon vain organisaation tiedonhakujärjestelmässä, vaan taustalla olevien tietolähteiden tietoturvaan on otettava myös kantaa. Ovathan ne määritelmän mukaan olennainen osa tiedonhakujärjestelmää. Tämä on, kuten Mukherjee ja Mao [37] huomauttavat, erittäin haastava tehtävä, koska aina näistä taustajärjestelmistä ei voida tehdä pitkälle meneviä oletuksia. Kaikkien niistä ylläpito ei välttämättä täysin ole organisaation itsensä hallussa, vaikka niin toki voisi olettaa.

3.5.3 Rakenteinen ja rakenteeton tieto

Organisaation tiedonhaun kannalta sekä rakenteinen, rakenteeton että puolirakenteinen tieto tulee ottaa huomioon. Tieto pitää voida myös esittää käyttäjälle järkevässä muodossa, riippumatta sen luonteesta. Mukherjee ja Mao [37] korostavat, ettei raken-

teisen ja rakenteettoman tiedon erottelu aina ole selvä. Rakenteeton tieto voi sisältää metatietoa, joka tekee siitä puolirakenteista. Tietokannan Binary Blob -kentät tai Varchar-arvot voidaan ymmärtää rakenteettomana tietona, vaikka tietokannan periaate itsessään on tallettaa nimenomaan rakenteista tietoa.

Mukherjee ja Mao väittävät, että juuri rakenteinen informaatio on organisaatioille kaikkein arvokkainta. Ongelma on heidän mukaansa kuitenkin se, että suurin osa organisaatioiden informaatiosta on talletettu rakenteettomaan muotoon ja toisaalta osa rakenteellisen tiedon metatiedoistakin on haun kannalta hyödytöntä: oletusarvoja tai suoranaisesti harhaanjohtavaa ja virheellistä tietoa. Mukherjee ja Mao esittävät, että organisaatiot voivat lisätä rakenteettoman tiedon arvoa tekemällä siitä rakenteista informaatiota. Rakenteen ja metatiedon lisääminen dokumentteihin voi heidän mukaansa tapahtua ideaalisessa tapauksessa tiedon eristämisen ja tiedonlouhinnan menetelmin, ja he odottavat edistystä laadukkaiden automaattisten järjestelmien kehityksessä tässä suhteessa. Prosessista on kuitenkin syytä tehdä pääosin automaattinen; manuaalinen muokkaaminen ei skaalautuisi suuria tietomääriä vastaan. On syytä huomata, että rakenteisuudella on monta tasoa: otsikoiden ja muiden yksinkertaisten metatietojen huomioiminen ei vielä vaadi paljoa työtä, mutta esimerkiksi entiteettien saati semantiikan merkitseminen on huomattavasti työläämpää. Halevy *et al.* [27] kritisoivat Mukherjeen ja Maon ajatusta vastaan. Heidän mukaansa rakenteisuuden lisääminen valtaviin tietomääriin ei ensinnäkään tule ilmaiseksi, vaan vaatii paljon työtä. Web-dokumenttien valtavan määrän vuoksi ajatus on ymmärrettävä, mutta yhtä hyvin voidaan epäillä, onko organisaationkaan mahdollista saada kaikkea tietoaan rakenteiseen muotoon siinä määrin, että siitä olisi erityistä hyötyä haulle. Halevy *et al.* painottavatkin, että tutkimuksen pitäisi keskittyä rakenteettoman tiedon analysointiin, koska näin saavutettavia hyötyjä on turhaan vähätelty rakenteisuuden rinnalla.

Usein dokumenttien sisältä on mahdollista tunnistaa kokonaisuuksia, kuten henkilöiden nimiä tai yritysten osoitteita. Näitä kokonaisuuksia, *entiteettejä* (engl. entity) voivat muodostaa esimerkiksi erilaiset listat, kappaleet tai muut dokumentista erottuvat kokonaisuudet. Rakenteiset Web-dokumentit ovat hyvä esimerkki dokumenteista, joista entiteettejä on helppo erottaa. Aina Webissä olevaa tietoa ei edes ole kytketty varsinaisen dokumentin sisään, vaan sitä liikutellaan esimerkiksi uutissyötteiden tai jonkin sovellustason protokollan avulla paikasta toiseen. Entiteettejä on joskus luontevaa koostaa myös perinteisten dokumenttien ulkopuolisista tietolähteistä, esimerkiksi tietokannoista, joiden sisällä tieto ei välttämättä ole dokumentin kaltaisissa, helposti käsiteltävissä olevissa kokonaisuuksissa. Organisaation tiedonhaun kannalta tärkeä *entiteettihaun* osa-alue on ihmisten haku (engl. people search), joka keskittyy varsinaisten

dokumenttien haun sijasta entiteettien, tässä tapauksessa ihmisten, hakuun esimerkiksi henkilön osaamisen perusteella. [7]

3.5.4 Pisteysmekanismi

Perinteiset tiedonhakumallit ja niihin liittyvät algoritmit, jotka pisteyttävät dokumentteja relevanttiuden mukaiseen järjestykseen, eivät yksittäin käytettyinä ole kovin tehokkaita organisaatioympäristössä. Pelkästään esimerkiksi Web-hakua varten suunnitellulla käänteistiedostoperustaisella indeksillä ja linkkipohjaiseen pisteytykseen perustuvalla PageRank-algoritmillä ei päästäisi kovin pitkälle, kun vastaan tulee esimerkiksi organisaation sähköpostiarkisto. Riippuen luonteeltaan erilaisten tietolähteiden määrästä, voi olla järkevää käyttää useampia algoritmeja, ja niitä pitää osata luovasti yhdistellä, jotta ne kykenisivät pisteyttämään eri tietolähteiden dokumentteja järkevään järjestykseen.

Mikäli tarkastellaan hakua vain intranet-portaalin tasolla, Hawkingin *et al.* [31] mukaan Web-tiedonhaun hakumallit, jotka pisteyttävät hypertekstidokumentteja niihin viittaavien ulkoisten linkkien perusteella, eivät ole kovin hyödyllisiä organisaation tiedonhaussa. Tämän kertovat myös luvun 3.4 aksioomat. Hawkingin *et al.* mukaan tämä johtuu juuri organisaatioiden intranetien luonteen eroista suhteessa Internetiin: Organisaation tärkeimmät, jotakin asiakohtaista tietoa sisältävät sivut, on yleensä linkitetty riittävän hyvin, eikä ylimääräisten linkkien lisääminen muuta lopputulosta. Lisäksi päätason sivulle ja muutamille muille avainsivuille on erittäin paljon linkkejä, mikä tekee linkkirakenteesta epätasapainoisen. Myös ulkoisten linkkien vanhentuminen suhteessa linkitettyyn kohteeseen saattaa muodostua ongelmaksi intranetissä, jossa linkkien kokonaismäärä ei välttämättä kompensoi tällaista tilannetta.

Pisteytysmekanismien ei tarvitse aina perustua kokosanahakuun, vaan erilaisia klusterointimenetelmiä tai metatietoihin perustuvia hakumenetelmiä voidaan käyttää tarpeen mukaan. Metatiedon ja semantiikan hyödyntäminen haussa on yksi organisaation tiedonhaun tutkimushaaroista. Solskinnsbakk ja Gulla [50] kutsuvat perinteistä hakulauseeseen perustuvaa kokosanahakua *syntaksiseksi hauksi*. Jos haussa käytetään erilaisia tiedon rakenteita tukevaa indeksiä, voidaan hakua kutsua Mukherjeen ja Maon [37] mukaan *parametrisoiduksi hauksi* (engl. parametric search). Tällainen haku on Mukherjeen ja Maon mukaan lähellä tietokantahakua, jossa haettua datajoukkoa voidaan rajoittaa (tai laventaa) määrittelemällä tietokannan taulun kentille arvorajoitteita. Organisaation tiedonhaun tapauksessa parametroitua hakua voidaan havainnollistaa esimerkiksi hakulauseella

```
''kurssikuvaus format:pdf''
```

Esimerkissä haettujen dokumenttien joukkoa rajoitetaan dokumentin tyypin (PDF) perusteella. Tämän mahdollistamiseksi kyselykielessä on käytetettävä syntaksia, jolla parametri saadaan välitettyä. Esimerkissä syntaksi vaatii, että parametrina toimivan metatiedon nimen jälkeen kirjoitetaan kaksoispiste, jolloin ”format:pdf” luetaan haun parametriksi eikä hakutermiksi kuten ”kurssikuvaus”. Parametroitu haku on luonnollinen tapa toteuttaa erilaisia rakenteisen haun muotoja, esimerkiksi entiteettihaku, joka sekin on esimerkki hausta, jossa on mukana syntaksista hakua enemmän semantiikkaa.

3.5.5 Federoitu haku

Federoitu haku tarkoittaa, että erilaisten hakukoneiden hakutuloksia yhdistetään yhtenäisen käyttöliittymän avulla. Tietolähteillä itsellään voi olla hakujärjestelmän kaltaisia ominaisuuksia; määritelmää venyttäen jopa varsinainen hakukone voitaisiin laskea tietolähteeksi. Federoidun haun haaste on yhtenäistää useista erilaisista hauista koostuvan tiedon esitystapa. Koska eri hakukoneiden pisteytysmekanismit todennäköisesti ovat erilaisia, on suuri haaste esittää kaikki tulokset lopulta yhtenäisessä muodossa ja määritellä niiden välinen relevanssi tasa-arvoiseksi riippumatta dokumentin tietolähteestä.

Federoinnin kannalta riittää eri hakujärjestelmien integrointi niin, että käyttöliittymässä kyetään esittämään eri indekseihin kootut tiedot yhtenäisessä muodossa. Käyttäjää ajatellen kyse on haun helpottamisesta: käyttäjän ei tarvitse huolehtia taustalla olevan ympäristön kompleksisuudesta, vaan sama käyttöliittymä hakee tietoa kaikista järjestelmistä.

3.5.6 Sisällöntuotanto

Jo Page *et al.* [40] toivat esiin Internetin demokraattisen puolen: kuka tahansa voi tuottaa Web-sivuja edullisesti ja yksinkertaisin välinein. Tämä huomio on kiistatta yhä keskeinen Internetin luonnetta määräävä tekijä, ja Web-hakukoneiden kehittäjät ovat joutuneet kamppailemaan paitsi epäkiinnostavan tai hyödyttymän sisällön myös suoranaisen roskapostin kanssa. Luvun 3.4 aksioomat osoittavat, että ainakin organisaation intranet-portaalien osalta asia on lähes päinvastainen. Intranet-portaalien sisällöntuotanto on itse asiassa varsin byrokraattista; intranet-portaali on organisaation äänenkannattaja, joten sisällöntuotanto uskotaan harvojen ja valittujen käsiin. Usein intranet-portaalin sisältämän tiedon tarkkuuteen ja paikkansapitävyyteen kiinnitetään aivan erityistä huomiota, ja se voi olla jopa organisaation laaduntarkkailun alaisena. Vahvimmillaan portaaleissa saatetaan ylläpitää esimerkiksi jonkin laatujärjestelmän

mukaisia toimintaohjeita tai muuta konsistenttia ja tarkasti tiettyyn asiaan kohdistettua tietoa.

Wikit voitaisiin nostaa tietolähteiden listalta omaksi sisältötyypikseen. Hyvä Webmaailman esimerkki wikistä on suosittu Wikipedia[4], mutta wikejä on pikku hiljaa otettu käyttöön myös organisaatioissa. Esimerkiksi Jyväskylän yliopiston Informaatioteknologian tiedekunnalla on oma wikinsä[5]. Organisaatioidenkin käytössä olevien wikien[6] luonteeseen kuuluu, että niiden sisältöä voidaan sisällönhallintajärjestelmää vapaammin muokata – itse asiassa wikien sisältö on yleensä oletuksena kaikkien muokattavissa. Wikeissä voidaan varsinaisen sisällön lisäksi esittää aiheeseen liittyvää keskustelua ja tämä on omiaan edistämään mielipiteenvaihtoa, joka organisaation kannalta edullisimmassa tapauksessa tiivistyy konsensukseksi.

Muiden tietolähteiden sisältö ja täten myös sisällöntuotanto on kirjavaa. Esimerkiksi sisällönhallintajärjestelmästä voidaan saada paljon hyödyllistä ja eheää tietoa, kunhan organisaatiossa pidetään huoli sen laadusta. Esimerkiksi sähköpostiviestien sisältöön on hankala vaikuttaa, joten sähköpostiarkistot pitää usein ottaa haun piiriin sellaisenaan. Jotkin järjestelmät saattavat jopa tuottaa automaattista sisältöä, jonka sisällyttäminen hakuun ei välttämättä aina ole järkevää. Ylipäätään organisaation on loogista pyrkiä laadukkaaseen sisällöntuotantoon, mutta tämä on jo pelkästään tietolähteiden paljouden saati taloudellisten ja aikataulullisten haasteiden vuoksi lähtökohtaisesti haastavaa.

3.5.7 Käyttäjät ja roolit

Organisaation tiedonhaun käyttäjien voidaan ajatella toimivan erilaisissa rooleissa paitsi sisältöä luodessaan myös hakua suorittaessaan. Rooleja voivat muodostaa esimerkiksi organisaation erilaiset osastot, tiimit tai intressiryhmät. Extranetit muuttavat organisaation tiedonhaun asetelmaa erityisesti käyttäjäkunnan suhteen. Extranetien käyttäjiä voidaan tässä yhteydessä nähdä erityisenä intranetin käyttäjäryhmänä tai -roolina. Oletusarvoisesti käyttäjät hakevat tietoa roolinsa mukaisesti, joten käyttäjän roolin tunnistaminen ja hyödyntäminen on tärkeä osa organisaation tiedonhaun tutkimusta.

Käyttäjän profiilitietojen ja tiedontarpeen taustalla olevan tehtävän, haun kontekstin, huomioimista haussa voidaan nimittää vaikkapa Hawkingin [32] mukaan *profiloinniksi* (engl. user and task profiling). Hawkingin mukaan [29] profilointi on yksi tärkeim-

[4]http://wikipedia.org

[5]https://trac.cc.jyu.fi/projects/it/

[6]Wikipedia, paitsi esimerkki wikistä, osaa myös kertoa niin wikeistä (http://en.wikipedia.org/wiki/Wiki) kuin organisaatiotasoisistakin wikeistä (http://en.wikipedia.org/wiki/Enterprise_wiki) tarkemmin.

mistä organisaation tiedonhaun tutkimuskohteista. Profiloinnin idea käy ilmi esimerkistä: Jos kirjastossa vieraillessaan kysyy jonkin kirjan olinpaikkaa kirjastonhoitajalta, voi kirjastonhoitaja käyttää kysyjän taustoja kirjan löytämiseen. Opinnäytetyötä tekevä opiskelija kysyy – ehkä tietämättään – kurssikirjaa tai jotakin käsikirjaston teosta. Satunnainen ohikulkija voi osata kertoa olevansa kiinnostunut Picassosta, muttei tiedä etsiäkö taiteilijaa taiteiden, elämäkertakirjallisuuden vai historian luokasta. Vastaavalla tavalla konteksti voi toimia apuna missä tahansa tiedonhaun piiriin kuuluvassa ongelmassa. Erityisen hyödyllistä kontekstin huomioiminen on organisaation tiedonhaussa, sillä käyttäjät toimivat usein – ainakin ideaalitapauksessa – täsmällisissä rooleissa. Lisäksi käyttäjiltä usein vaaditaan autentikoitumista organisaation verkkoon, työpöytäkoneelle, tai intranet-portaaliin. Mikäli autentikaatiotietoa voidaan hyödyntää hakujärjestelmässä, edellytykset käyttäjän profiilin tutkimiseksi ja huomioimiseksi haussa ovat olemassa. Hawking [32] toteaa omista, CSIRO ICT -keskuksen sähköpostiarkiston lokeihin liittyvistä tutkimustuloksistaan, että noin kolmannes kyselyistä on ainoastaan yhden sanan mittaisia. Toisaalta Hawkingin mukaan aikaisemmin on Web-käyttäjien osalta päädytty havaintoon, että noin 70 prosenttia käyttäjistä tyypillisesti käyttää vain yhtä hakusanaa. Jos organisaatioiden käyttäjät toimivat samansuuntaisesti, ei tämä tarjoa erityisen hyvää lähtökohtaa (ensimmäisellä yrityksellä) onnistuneelle haulle, kontekstin mukaan saamisesta puhumattakaan. Hawking toteaa, että usein käyttäjät käyttävät eri hakukoneita erilaisten tehtävien suorittamiseen: sähköpostihakua sähköpostiarkistosta etsimiseen, työpöytähakua jonkin omalla koneella sijaitsevan dokumentin etsimiseen ja niin edelleen. Haaste on toteuttaa yksi yhtenäinen käyttöliittymä, jonka takaa kaikkien edellä mainittujen hakutehtävien suorittaminen on mahdollista. Haun tulisi osata huomioida käyttäjän profiili esimerkiksi käyttäjän selainistunnon (jos haun käyttöliittymä on selainpohjainen) tai käyttöjärjestelmässä käytetyn autentikoinnin perusteella. Käyttäjän profiilia vaikeampi haaste on selvittää käyttäjän hakutehtävä. Joissain tapauksissa sen tietää vain käyttäjä itse, jolloin joudutaan turvautumaan parametroituun hakuun, jonka avulla käyttäjän tulee itse määritellä, minkä tyyppistä tehtävää hän on suorittamassa.

3.6 Organisaation tiedonhakujärjestelmistä

Seuraavissa alaluvuissa luodaan katsaus organisaation tiedonhakujärjestelmien yleiseen arkkitehtuuriin ja muihin ulottuvuuksiin pelkästään tässä luvussa esitellyn tutkimuksen valossa.

3.6.1 Järjestelmien tutkimuksesta

Organisaation tiedonhaun tutkimuksissa on huomionarvoista, että ne harvoin perustuvat minkään olemassa olevan järjestelmän analysointiin, vaan rakentavat teoriaa nimenomaan aikaisemman tiedonhaun tutkimuksen kautta. Poikkeuksiakin löytyy, esimerkiksi FASTia ja Lucenea [16] sekä Verityä [2] on analysoitu. On ymmärrettävää, että ainakin vailla liike-elämän yhteyksiä olevien tutkijoiden on vaikeaa päästä analysoimaan usein liikesalaisuuksina pidettyjen organisaation tiedonhakujärjestelmien arkkitehtuuria ja toimintaa. Tilanne on analoginen Web-tiedonhakuun, joka alkoi esimerkiksi Googlen osalta avoimesti akateemisena tutkimuksena; nykypäivänä Googlen ratkaisut ovat tarkasti varjeltu liikesalaisuus.

Organisaation tiedonhakujärjestelmät ovat luonteeltaan monimutkaisia erityisesti tietolähteiden ja käyttäjäkunnan kirjavuuden vuoksi, joten tällaisen järjestelmän tekeminen pelkästään tutkimustarkoituksiin on varsin haastavaa. Selvästi helpompi lähestymistapa on evaluoida jotain osakokonaisuutta, kuten tiedonhakumallia. On selvää, ettei tietolähteiden tai käyttäjäkunnan kirjo voi tällaisessa testiasetelmassa vastata todellisen organisaation – tai ainakaan kaikkien mahdollisten organisaatioiden – tilannetta. CERC-testikokoelmaa[7] [6] voidaan käyttää organisaation tiedonhakujärjestelmän tai sen osien evaluointiin. Kokoelma perustuu CSIRO-organisaatiolta (Commonwealth Scientific and Industrial Research Organisation) kerättyyn dokumenttikokoelmaan. Kokoelman avulla on viime vuosina testattu erityisesti dokumentti- ja entiteettihakua [48].

3.6.2 Järjestelmän yleinen arkkitehtuuri

Organisaation tiedonhakujärjestelmän arkkitehtuuria on hahmoteltu kuvassa 3.1. Arkkitehtuuri pyrkii kuvaamaan järjestelmän keskenään yhteydessä olevia peruselementtejä, jotka tässä yhteydessä tarkoittavat järjestelmän ominaisuuksien kokonaisuuksia. Elementtejä on kuvattu laatikoina, ja elementtien välisiä suhteita sekä keskinäistä kommunikaatiota, ohjelmistoarkkitehtuurien kielellä rajapintoja, on kuvattu nuolilla. Elementit voivat myös kuulua osaksi toista elementtiä. Indeksiä ja tietolähteitä kuvaavien elementtien esitystapaa on tyylitelty perustuen yleisesti käytettyihin dokumentin ja tietolähteen hahmoihin. Arkkitehtuurin lähtökohdiksi on otettu kuva 2.2, joka korostaa tiedonhaun perusprosessin käyttäjälähtöisyyttä, sekä kuva 2.3, joka määrittää vaiheet, joiden perusteella tiedonhakujärjestelmä löytää käyttäjän haluaman tiedon. Arkkitehtuurin organisaation tiedonhallintajärjestelmään liittyvät osuudet perustuvat luvussa 3 tehtyihin havaintoihin organisaation tiedonhaulle tyypillisistä piirteistä. Kirjoittajan oma näkemys ei voi olla vaikuttamatta siihen, mitä asioita arkkitehtuurissa

[7] http://es.csiro.au/cerc/

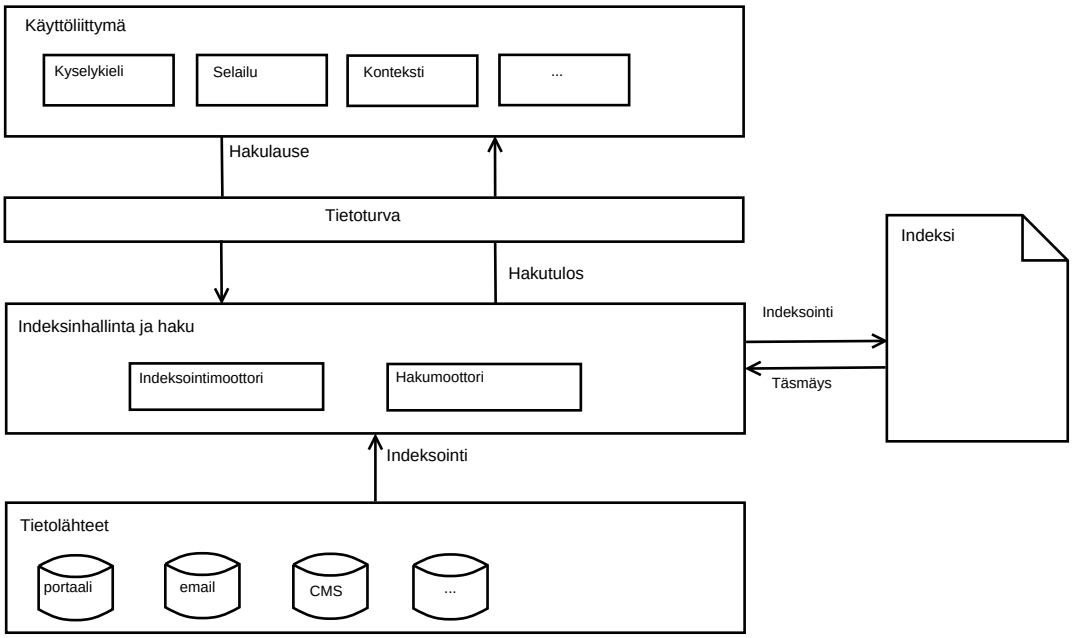
on korostettu, mutta tarkoitus on kuitenkin ollut pyrkiä tasapainoiseen arkkitehtuurin kuvaukseen pelkästään tässä luvussa esitetyn tutkimustiedon valossa.

Käyttöliittymä on nostettu kuvioon ylimmäiseksi. Tiedonhaun perusprosessi on käyttäjälähtöinen: käyttäjä hakee tietoa järjestelmästä ja hakutulokset esitetään haun vastauksessa. Haku tapahtuu pääasiassa jonkin kyselykielen mukaisen hakulauseen syöttämisellä tai selailupohjaisen, esimerkiksi moninäkymähakua noudattavan käyttöliittymän kautta. Mitä paremmin käyttöliittymä antaa palautetta haun onnistumisesta, sitä paremmin käyttäjä pystyy tekemään uusia kyselyitä järjestelmään ja ymmärtämään tiedon tarvettaan paremmin. Tämän ympärille liittyy erinäisiä, organisaation tiedonhaulle ominaisia piirteitä, kuten hakutulosten federointi, käyttäjän profiilin ja hakutehtävän huomioiminen ja entiteettihaku.

Indeksinhallinnan ja haun toiminnot on tietoisesti sijoitettu kuvan keskelle. Niiden voidaan katsoa olevan peruselementejä kaikissa tiedonhakujärjestelmissä. Indeksointi- ja hakumoottorit on kumpainenkin esitetty selvyyden vuoksi yhtenä laatikkona, vaikkakin niiden toteutus voi hyvin sisältää useita erilaisia tapoja indeksoida ja hakea tietoa. Indeksejä voi olla yksi tai todennäköisemmin useampi, riippuen indeksinhallinnan ja hakuoperaatioiden vaatimuksista, jotka puolestaan pohjautuvat organisaatioiden tietolähteiden vaatimuksiin. Klusterointia ja hakuun liittyviä lisäominaisuuksia ei ole kuvassa selkeyden vuoksi esitetty, mutta niitä ei ole tarpeen myöskään sulkea pois.

Tietoturva on esitetty kerroksena käyttöliittymän ja indeksi- sekä hakutoimintojen välillä. Tällä on tahdottu korostaa sitä, että tietoturvaan on otettava kantaa viimeistään siinä vaiheessa, kun hakulausetta vastaavia hakutuloksia ollaan esittämässä käyttäjälle. Todellisuudessa toteutus voi olla monimutkaisempikin. Organisaation tiedonhaussa eri rooleissa toimivien käyttäjien merkitys korostuu paitsi käyttöliittymässä myös tietoturvan kannalta.

Kaiken pohjaksi on sijoitettu tietolähteet, joiden piirissä olevat dokumentit indeksoidaan haun piiriin. Jokaisen eri tyyppisen tietolähteen voidaan katsoa asettavan omia, perustavanlaatuisia vaatimuksiaan järjestelmälle. Mikäli tietolähteinä olisi esimerkiksi vain hypertekstidokumentteja sisältävä intranet-portaali, voisi järjestelmälle riittää yksi indeksi, yksi tehokasta tiedonhakumallia noudattava indeksointi- ja hakumoottori, sekä Web-haun omainen käyttöliittymä. Tällaisen asetelman voidaan olettaa muodostavan minimaalisen organisaation tiedonhakujärjestelmän; todellisuus lienee usein paljon monimutkaisempi.



Kuva 3.1: Organisaation tiedonhakujärjestelmien yleinen arkkitehtuuri.

3.6.3 Järjestelmän toteuttaminen tuotteena tai palveluna

Organisaation tiedon jakautuminen useisiin heterogeenisiin tietolähteisiin vaikeuttaa organisaation tiedonhakujärjestelmän tuotteistamista. Web-hakukoneita tarjotaan (usein ilmaisena) palveluna Internet-verkon yli [37]. Organisaatioiden informaatio on kuitenkin levällään keskenään erilaisissa tietolähteissä, joiden lukumäärä ja luonne riippuu tarkasteltavasta organisaatiosta, joten Webin kaltaista yhtä ja yhtenäistä dokumenttikokoelmaa ei ole löydettävissä. Mikäli organisaation tiedonhakujärjestelmä tuotteistetaan ja sitä tarjotaan sovelluksena organisaatioille, joutuu se luonnollisesti ottamaan kantaa kaikkien erilaisten organisaatioiden tarpeisiin. Tällaisia tarpeita ovat muun muassa laitteistovaatimukset, käyttöjärjestelmät, käytettävissä oleva kaistanleveys, palomuurit, erilaiset tietolähteet, tietoturvamallit, dokumenttiformaatit, käyttäjät, rajapinnat ja maantieteellinen etäisyys [37]. Hawkingin [29] mukaan juuri tämä seikka on esteenä sille, että Web-hakukoneiden tarjoama teho saavutettaisiin organisaation tiedonhakujärjestelmien osalta samalla vaivalla. Organisaatioiden on lisäksi usein vaikea olla kyllin joustavia uusien ominaisuuksien tai muutosten toteuttamisessa. Tämä on ristiriidassa organisaatioiden joustavuudelle, tietoturvalle ja räätälöidyille ratkaisuille asettamien kovien vaatimusten kanssa [37].

4 Organisaation tiedonhaun ratkaisuja

Seuraavissa alaluvuissa esitellään ensin avoimen ja omisteisen sovelluksen erot. Tämän jälkeen tarkastellaan sekä suljettuun että avoimeen lähdekoodiin perustuvia organisaation tiedonhaun ratkaisuja.

4.1 Avoin ja omisteinen sovellus

Free Software Foundation[1], lyhyemmin FSF, lienee ensimmäisenä esittänyt ajatuksen, miksi tietokoneohjelmistojen tulisi olla avoimia. Tämä ajatus sisältää paljon enemmän kuin lähdekoodin avoimuuden. FSF puhuukin avointen ohjelmien sijaan *vapaaohjelmista* (engl. Free Software). Vapaaohjelma on FSF:n mukaan sellainen, jonka lisensointi antaa käyttäjälle

- vapauden suorittaa (käyttää) ohjelmaa mihin tahansa tarkoitukseen,
- vapauden tutkia ohjelman sisäistä toimintaa ja muokata sen toiminnallisuutta (tämä edellyttää, että lähdekoodi on saatavilla),
- vapautta julkaista ja levittää kopioita ohjelmasta, ja
- vapautta julkaista ja levittää myös omia muokattuja versioitaan ohjelmasta (jälleen lähdekoodin on oltava saatavilla).

Vapaa ohjelma ei siis ole pelkästään avoin: myös sen levittäminen muokattuna tai sellaisenaan on mahdollista. On huomionarvoista, ettei vapaan ohjelman määritelmä puutu millään lailla ohjelman myymiseen tai muun taloudellisen hyödyn hankkimiseen. Määritelmä kyllä helposti johtaa siihen, että vapaa ohjelma on usein käytännössä ilmainen, koska ohjelmasta maksaneetkin saavat oikeuden lisenssiehtojen mukaisesti levittää sitä eteenpäin.

Toinen merkittävä ohjelmistojen avoimuutta ajava organisaatio on Open Source Initiative (OSI)[2]. OSIn näkemys *avoimesta ohjelmasta* eroaa FSF:n vapaaohjelman määritelmästä oikeastaan vain filosofisilta perusteluiltaan siitä, miksi ohjelman tulisi olla avoin. OSI:n näkemys on usein tulkittu myönteisemmäksi länsimaiden kaupalliselle

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

kulttuurille, joten OSI:a käytetään usein referenssinä sille, onko jokin ohjelmistolisenssi avoin vai ei. OSI luokittelee avoimiksi ohjelmistolisensseiksi muun muassa seuraavat avointen ohjelmistojen maailmassa suositut lisenssit: GPL, LGPL, BSD ja Apache Licence.

Tässä tutkielmassa vapaiden ja avointen ohjelmien kesken ei tehdä eroa, vaan puhutaan yksinkertaisesti avoimista ohjelmistoista. FSF:n määritelmä vapaille ohjelmille on kuitenkin tämän avoimuuden mittari, ja sovelluksia, jotka eivät täytä tätä määritelmää, kutsutaan tässä tutkielmassa omisteisiksi. Tämä ei ole aivan tarkka määrittely, sillä kiistelyä voidaan käydä siitä, minkälaisen lisenssin tarkalleen ottaen avointen ohjelmistojen yhteisö sallisi avointen ohjelmistolisenssien joukkoon.

Avoimen lähdekoodin hyödyllisyys on kiistanalainen asia. Yee *et al.* [53] ovat lyhyesti tiivistäneet avointen tiedonhaun ratkaisujen hyödyllisyyttä tutkimuksen kuten myös yksityishenkilöiden tai organisaatioiden kannalta. Heidän mukaansa ratkaisun toiminnallisuuden läpinäkyvyys on oleellisin saavutettava hyöty. Tämä mahdollistaa ratkaisun julkisen tarkastelun ja kehittämisen. Monet avoimen lähdekoodin kriitikot muistuttavat, että avoimen lähdekoodin maailmasta löytyy paljon keskeneräisiä ja ideointivaiheessa olevia projekteja. Heidän mukaansa avointen ratkaisujen taso ei yleensä yllä omisteisten tasolle. On selvää, että avoimilla ohjelmistoilla on pienempi julkaisukynnys verrattuna kaupallisiin suljetun lähdekoodin ohjelmistoihin. Asialla on positiivinenkin puolensa: avoimen lähdekoodin sovellus saattaa saada lisää innostuneita kehittäjiä jo varhaisessa vaiheessa, ainakin mikäli idea vetää puoleensa. Omisteisella, usein lähtökohdiltaan kaupallisella sovelluksella todistustaakka laadun ja ominaisuuksien suhteen on usein alkuperäisillä kehittäjillä itsellään. Avointen ratkaisujen kyvykkyyden osoittaminen jää, jos ei median ja markkina-analyytikkojen, niin viime kädessä akateemisen vertailun varaan. Tämän tutkielman kannalta avoimen lähdekoodin edut ovat kuitenkin ilmeiset: tähän tutkielmaan valittu avoin sovellus on ilmaiseksi ladattavissa, siihen on olemassa paljon ilmaista dokumentaatiota eikä sen asennuskaan vaadi käyttöjärjestelmän lisäksi ylimääräisiä investointeja.

4.2 Omisteiset ratkaisut

Seuraavissa alaluvuissa esitellään organisaation tiedonhaun alan suurimpien toimittajien omisteisia ratkaisuja.

4.2.1 Alan suurimmat toimittajat

Schmidt [46] on käynyt läpi organisaation tiedonhaun kaupallisia toimijoita Bennettin vuoden 2007 analyysin [8] pohjalta ja toteaa, että Autonomy[3] ja Fast[4] ovat alan suurimmat toimittajat. Tilanne on muuttunut Bennettin vuoden 2009 arvion [9] perusteella: Microsoft osti Fastin toukokuussa 2008[5], mikä on nostanut Microsoftin suurimpien nimien joukkoon.

Gartnerin [3] mukaan Microsoft oli vuonna 2009 ”merkittävä toimittaja kaiken kokoisiin organisaation tiedonhaun ratkaisuihin”. Itse asiassa Gartner on arvioinut Microsoftin johtavien toimittajien luokan kärkeen. Samassa luokassa korkeimmilla sijoilla ovat myös Autonomy ja ZyLab. Haastajien luokassa korkeimmilla sijoilla ovat Google ja Oracle, joista Google on sijoittunut hieman paremmin. Yksinkertaisempien ratkaisujen osalta Gartner toteaa Googlen ja Microsoftin dominoivan alaa.

Bennett [9] on jakanut suurimmat toimittajat kahteen tasoon (engl. Tier 1 ja Tier 2), ja maininnut täydellisyyden vuoksi myös muita kaupallisia toimijoita, edullisia tai ilmaisia tuotteita sekä avoimen lähdekoodin haastajia. Bennettin listalla tasolla yksi (suurimmat toimijat) ovat Autonomy, Endeca, Exalead, Google sekä Microsoft. ZyLabia Bennett ei sen sijaan ole nostanut ensimmäiselle tasolle, vaan maininnut sen olevan kooltaan muita olennaisesti pienempi.

Tarkempaan käsittelyyn ovat edellä olevan perusteella valikoituneet Autonomy, Google ja Microsoft. Näiden toimijoiden edistyksellisyydestä edellä siteeratut markkina-analyytikot tuntuvat olevan yhtä mieltä. Tämän mittakaavan markkinoilla voisi olettaa, että edistyksellinen liiketoiminta vaatii edistyksellisiä ratkaisuja niin laadullisten kuin toiminnallistenkin ominaisuuksien suhteen. Tämän tutkielman kannalta on oleellista, että valittujen toimijoiden ratkaisut antavat mahdollisimman kokonaisvaltaisen kuvan organisaation tiedonhaun ratkaisujen nykytilasta. Markkinoiden evoluutio omalta osaltaan toimii todistusvoimana sille, miltä organisaation tiedonhaun tulisi tällä hetkellä pitää sisällään, jotta sen käyttäjät – organisaatiot – siitä olisivat valmiit maksamaan. Näiden ratkaisujen ominaisuudet, arkkitehtuuri ja käyttökohteet kertovat mikä organisaation tiedonhaun kannalta on oleellista, mikä on tarpeettomana jäänyt historiaan, ja arvioitavissa olisi sekin, mikä on mahdollisesti tulevaisuuden suunta.

[3] http://www.autonomy.com
[4] http://www.fastsearch.com
[5] http://www.fastsearch.com/l3a.aspx?m=1105

4.2.2 Autonomyn, Googlen ja Microsoftin ratkaisut

Autonomyn tuoterepertuaari[6] pohjautuu Intelligent Data Operating Layer (IDOL) -palvelimeen. Sen pääasiallinen tarkoitus on tiedon indeksointi. IDOL-palvelin tarjoaa tätä tietoa muiden Autonomyn tuotteiden jatkojalostettavaksi ja käytettäväksi. Autonomyn tuotteet tarjoavat organisaation tiedonhaun kannalta paitsi perinteistä Boolen mallin mukaista hakua, myös edistyneempää automatiikkaa haun tehostamiseksi: esimerkiksi hakutulosten visualisointia, ennustavaa hakulausekkeen syöttöä ja automaattista hakutulosten analysointia tuetaan. Autonomy kertoo myös tukevansa hakukontekstia, parametroituja hakuja, entiteettihakua sekä hakuagentteja, jotka voivat toteuttaa hakutehtäviä käyttäjän puolesta. Tietoturvan osalta Autonomy tukee ainakin LDAP-, SSL- ja NTLM-autentikointitapoja. Käyttäjän oikeudet suhteessa eri dokumentteihin on mahdollista määritellä. Tuettuja tietolähteitä on yli 400, mutta Autonomy on laskenut tähän listaan mukaan myös dokumenttiformaatit.

Google jakaa ratkaisunsa kahteen luokkaan. Ratkaisujen ideana on palvelinräkkiin asennettava, tuotteistettu hakupalvelin, joka hyödyntää Googlen kehittämää hakuteknikkaa[7]. Pienille yrityksille tarjotaan Google Mini Search Appliance -ratkaisua, joka yltää Googlen mukaan enintään 300 tuhannen asiakirjan indeksointiin. Keskisuurille ja suurille yrityksille tarjottavassa Google Search Appliance -ratkaisussa[8] indeksoitavien dokumenttien määrä on 30 miljoonaa. Googlen mukaan järjestelmää voidaan skaalata yksinkertaisesti asentamalla uusi Google Appliance -palvelin osaksi olemassa olevaa asennusta. Käyttöjärjestelmistä tai laitteistoräätälöinneistä ei tarvitse välittää. Useiden Google Search Appliance -palvelinten skaalaamisella voidaan Googlen mukaan saavuttaa useiden miljardien asiakirjojen kattavuus. Google Search Appliancen käyttöliittymä muistuttaa Googlen Web-hakua, ja hakuparadigma on hakulausekeperustaisen haun kaltainen. Erilaisia hakua tukevia ominaisuuksia on lukuisia, esimerkiksi kontekstin huomioiminen ja entiteettihaku ovat mahdollisia. Tietolähteinä voidaan käyttää muun muassa Documentumia, FileNetiä, SharePointia, LiveLinkiä, Lotus Dominoa ja useita tietokantajärjestelmiä. Tuettuja tiedostotyyppejä on 220, Google mainitsee esimerkkeinä seuraavat: HTML, Microsoft Office, PDF, PostScript, WordPerfect ja Lotus. Tietoturvan osalta Google Search Appliance tukee olemassa olevia käyttäjäoikeuksien tunnistamistapoja, kuten LDAP, NTLM-todennus, PKI-järjestelmän X.509-varmenteet, Kerberos ja Windows-autentikointi (Windows Integrated Authentication).

[6] http://www.autonomy.com/content/Products/products.en.html
[7] http://www.google.com/enterprise
[8] http://www.google.com/enterprise/gsa/features.html

Microsoftin viimeisimpiä organisaation tiedonhaun ratkaisuja ovat Search Server Express 2010 ja SharePoint 2010.[9] Microsoft tarjoaa edelleen myös FAST ESP -tuotetta sellaisenaan[10], tai osana SharePoint-portaalia[11]. FAST ESP, ja siihen liitttyvä FAST Search Server 2010 for SharePoint on näistä ratkaisuista kehittynein[12], ja voi indeksoida yli miljardin dokumentin kokoelmaa. Se tarjoaa perinteisen sanahaun lisäksi entiteettihakua yli 40 erilaisen entiteetin perusteella. Myös käyttäjän profilointia tuetaan. Tuettuja autentikointitapoja ei ole erikseen mainittu, mutta erilaisten tietolähteiden ja integraatiomahdollisuuksien lista on kattava. Tuettuja dokumenttiformaatteja on yli 370 kappaletta.

Selkeä yhteinen nimittäjä omisteisille ratkaisuille on indeksin teoreettinen laajuus. Jokainen esitellyistä ratkaisuista kykenee indeksoimaan valmistajan ilmoituksen mukaan vähintään kymmeniä miljoonia dokumentteja. Toisaalta omisteiset ratkaisut tukevat laajalti erilaisia tietolähteitä ja dokumenttiformaatteja. Esimerkiksi tuettuja dokumenttiformaatteja on jokaisessa ratkaisussa satoja. Tietoturva on jokaisessa tarkastelluista ratkaisuista huomioitu olennaisena osana järjestelmää, ja monia olemassa olevia autentikointitapoja tuetaan suoraan. Perinteisen Web-haun kaltaisen sanahaun lisäksi kaikki ratkaisut tarjoavat kontekstiperustaista hakua ja erilaisia entiteettihaun muotoja.

4.3 Avoimet ratkaisut

Tämän tutkielman tutkimuskohteeksi valittiin Apache Solr, joka on varteenotettavin avoimen lähdekoodin ratkaisu organisaation tiedonhakuun.

4.3.1 Avointen ratkaisujen kartoitusta

Tässä tutkielmassa avointen ratkaisujen etsimiseen on käytetty lähteinä freshmeat.net-hakupalvelua[13], Googlea[14] sekä englanninkielistä Wikipediaa[15]. Aihetta käsittelevät

[9] http://www.microsoft.com/enterprisesearch/searchserverexpress/en/us/compare.aspx

[10] http://www.microsoft.com/enterprisesearch/en/us/Fast.aspx

[11] FAST-liitännäinen SharePointiin: http://sharepoint.microsoft.com/en-us/product/capabilities/search/Pages/Fast-Search.aspx

[12] http://www.microsoft.com/downloads/details.aspx?familyid=920E8C04-E8A6-4079-8B17-F3FB070FDF0D&displaylang=en

[13] http://freshmeat.net

[14] http://www.google.fi

[15] http://en.wikipedia.org

blogit[16] ovat antaneet osviittaa ratkaisuista, joihin kannattaa perehtyä tarkemmin. Päällimmäinen havainto on, ettei vakavasti otettavia avoimia ratkaisuja Apachen Lucenen ja Solrin lisäksi ole ainoatakaan.

Erilaisiin tarkoituksiin toteutettuja hakukoneita, hakukoneoptimoijia, metadatatyökaluja ja tarkastelukulmaltaan hyvin yleisiä tuotteita on avoimen lähdekoodin maailmassa olemassa paljon. Esimerkki tällaisesta sovelluksesta on Nutch[17], joka toteuttaa vain Web-hakukoneen eikä ota kantaa organisaation tiedonhaun vaatimuksiin. Silmiinpistävää on, että monet ratkaisut perustuvat pääosin johonkin toiseen avoimen lähdekoodin ratkaisuun, eivätkä siten itsessään sisällä mitään uutta tai merkittävää. Usein tällaiset ratkaisut yksinkertaisesti sisältävät jonkin toisen avoimen sovelluksen käärittynä helposti myytävään pakettiin[18]. Monet yritykset lisäksi kauppaavat konsultointia, käyttötukea tai kehittäjiä johonkin avoimeen ratkaisuun, eivätkä siten tarjoa mitään uutta tuotetta markkinoille. Asiakasreferenssiensä perusteella hyvä valtavirran esimerkki tästä on Lucid Imagination[19]. Vastaan tulee aina silloin tällöin sovelluksia, joiden lähdekoodia tuntuu olevan vaikea tai jopa mahdoton löytää[20]. Täten ne eivät muodollisesti täytä avoimen lähdekoodin sovellukselle asetettuja vaatimuksia, eivätkä muutoinkaan ole kovin hedelmällisiä tutkittavia lähdeaineiston puuttuessa. Ratkaisut, joita ei enää aktiivisesti kehitetä (tämä näkyy usein siinä muodossa, ettei ratkaisulla ole omia kotisivuja), ovat luonnostaan riittämättömiä lähemmän tarkastelun kohteiksi. Tällaisia tapauksia ovat muun muassa NVBase[21] ja Locust[22].

4.3.2 Apache Solr

Apache Solr (tästä eteenpäin lyhyemmin Solr) perustuu Java-pohjaiseen Lucene-hakukirjastoon, jota käytetään varsinaiseen tekstipohjaisen tiedon indeksointiin ja hakuun. Tämän päälle Solrissa on rakennettu Web-palvelin (Apache Tomcat tai Jetty) -pohjainen REST-tyyppinen HTTP/XML- ja JSON-ohjelmointirajapinta. Solr on siis Lucenen palvelintoteutus ja Solrin ohjelmointirajapinnat perustuvat vahvasti Javaan. Solr on lähtöisin CNET Networksin vuonna 2004 aloittamasta "Solar" projektista, jonka tuotokset luovutettiin vuoden 2006 alussa Apache Foundationille[23]. Apache lisensoi

[16] http://www.enterprisesearchblog.com, http://searchdoneright.com, http://searchtools.livejournal.com

[17] http://lucene.apache.org/nutch

[18] http://www.jumpernetworks.com, http://www.kneobase.com

[19] http://www.lucidimagination.com

[20] http://www.simplexo.com, http://www.youramigo.com

[21] http://sourceforge.net/projects/nvbase

[22] http://sourceforge.net/projects/locust

[23] http://wiki.apache.org/solr/FAQ#Where_did_Solr_come_from.3F

lähdekoodin avoimen lähdekoodin vaatimukset täyttävällä Apache-lisenssillä. Projektin versionhallinnan tietojen perusteella nykyinen kehitys näyttää olevan aktiivista[24] ja Solria voidaan asiakasreferenssiensä[25] perusteella pitää varsin stabiilina. [49]

4.3.3 Lucene-kirjasto

Lucene on Javalla toteutettu avoimen lähdekoodin tiedonhakukirjasto. Lucenea pitäisi oikeastaan kutsua nimellä Lucene Java, sillä Lucene on nimenä myös Apache Software Foundationin perustamalle projektille, joka sisältää useita muitakin tiedonhakuaiheisia osaprojekteja. Apachen mukaan Lucene Java on edellä mainittujen osaprojektien lippulaiva, joten viittaaminen Lucene Javaan nimellä Lucene lienee oikeutettua. [17]

Mainittakoon, että Lucene-projekti sisältää myös .NET-, Python ja C-toteutukset Lucene Java -kirjastosta.[26] Nämä kaikki perustuvat kuitenkin alkuperäiseen Lucene Java -kirjastoon, joten niiden tarkastelu erikseen ei tuottane olennaisesti uutta informaatiota. Python-toteutus ei oikeastaan ole edes käännös toiselle ohjelmointikielelle, vaan se yksinkertaisesti kapseloi Java-toteutuksen Python-luokiksi. [17]

4.3.4 Solrin valinnasta

On merkillepantavaa, että useimmat avoimen lähdekoodin ratkaisut perustuvat Apachen Java-pohjaiseen Lucene-hakukirjastoon. Solr kuuluu samaan Apachen projektiperheeseen, joten sillä on tässä suhteessa etuoikeutettu asema muiden Luceneen perustuvien projektien rinnalla. Varteenotettavaa avointa ratkaisua, johon Solria voisi verrata, ei oikeastaan edes löydy. Vaikuttaisi siltä, että myös kaupallisessa mielessä Solr on kykenevin haastaja edellä mainituille omisteisille ratkaisuille. Solrin alkuperäisenä kehittäjänä CNET Networks on ehkä jäävi arvioimaan Solrin kilpailukykyä [4], mutta myös omisteisia toimijoita kartoittanut Bennett [9] lukee Apache Lucenen ja Solrin tunnetuimmiksi avoimen lähdekoodin ratkaisuiksi organisaation tiedonhakuun. Bennett tosin huomauttaa, ettei yhtäläisyysmerkkejä Solrin ja suurimpien omisteisten ratkaisujen välille voida aivan vielä piirtää: vaikka avoimet ratkaisut ovat varsin kehittyneitä ottaen huomioon alan koko ja omisteisiin ratkaisuihin tehdyt investoinnit, on ne usein suunniteltu ja paketoitu pikemminkin kehittäjien näkökulmaa ajatelleen. Kaupalliset toimijat kuten Lucid Imagination[27], jotka tarjoavat konsultointipalveluita

[24] http://svn.apache.org/viewvc/lucene/dev/

[25] Lucid Imagination ylläpitää listaa Solr:n avulla toteutetuista hakuratkaisuista osoitteessa http://www.lucidimagination.com/developer/Community/Application-Showcase-Wiki.

[26] http://lucene.apache.org/lucene.net/, http://lucene.apache.org/pylucene/, http://lucene.apache.org/lucy/

[27] http://www.lucidimagination.com

näille avoimille ratkaisuille, ovat Bennettin mukaan muuttamassa tätä tilannetta parempaan suuntaan: niiden intresseissä on tuotteistaa ratkaisut ja tarjota muun muassa käyttö- ja ylläpitotukea.

Akateeminen kiinnostus Apachen hakuratkaisuja kohtaan on vielä vähäistä, joskin Lucene ja Solr näyttäisivät johtavan kilpailua muihin avoimiin ratkaisuihin tässäkin suhteessa. Eräs organisaation tiedonhakuun läheisesti liittyvä tutkimus on tehty Chernovin *et al.* toimesta jo vuonna 2006 [16]. Tutkimuksessa integroidaan Microsoftille sittemmin siirtynyt FAST-hakujärjestelmä ja Lucene-hakukirjasto tarkoituksena toteuttaa federoitu hakuympäristö. Lucenesta ja Solrista on olemassa myös painettua kirjallisuutta: Hatcher *et al.* ovat kirjoittaneet varsin perusteellisen hakuteoksen Lucenesta [28]; Smiley ja Pugh vastaavan Solrista [47]. Smileyn ja Pughin kirja ottaa lisäksi esimerkein kantaa Solrin soveltamiseen organisaatioympäristössä, joten se ei ole pelkästään tuotedokumentaation kaltainen hakuteos, vaan osaltaan todistaa, että Solr on kypsä todelliseen käyttöön.

Solrin valintaa tukee myös se, että Solr perustuu Javaan, joka on akateemisessa maailmassa paljon käytetty ohjelmointikieli ja virtuaalikone. Erityisesti Solr on avoimen lähdekoodin sovellus, joten sen tutkiminen, tarvittaessa yksityiskohtia myöten, on paljon helpompaa kuin jonkin omisteisen ratkaisun. Solrin tapauksessa esimerkiksi käytetyt tiedonhakumallien algoritmit eivät ole salaisuus, vaan ne voi tarvittaessa etsiä lähdekoodista ja niitä voi – jatkotutkimuksia ajatellen – jopa muokata, puhumattakaan, että muokattuja versioita on avoimen lisenssin nojalla mahdollista edelleen levittää tai jopa myydä.

5 Viitekehys organisaation tiedonhakujärjestelmän arviointiin

Tässä luvussa kehitetään viitekehys organisaatioiden tiedonhakujärjestelmän arvioimiseksi jäsennellysti. Viitekehys on suhteessa akateemiseen tutkimukseen ja olemassa oleviin organisaation tiedonhakujärjestelmiin. Organisaation tiedonhakujärjestelmää pystytään arvioimaan suhteessa omisteisiin ratkaisuihin tämän viitekehyksen puitteissa vain epäsuorasti, mutta viitekehys antaa kuvaa siitä, miten varteenotettava järjestelmä on organisaation tiedonhakujärjestelmänä.

5.1 Viitekehykseen kohdistuvat tarpeet

Viitekehyksiä, joiden perusteella organisaation tiedonhakujärjestelmiä voitaisiin analysoida, ei olemassa olevasta alan tutkimuksesta ole löydettävissä. Tarve tällaiselle viitekehykselle on kuitenkin ilmeinen: ilman viitekehystä aiheen ympärillä suurimmat kaupalliset toimittajat olisivat aina askeleen edellä muita ratkaisuineen, koska aiheeseen perehtymätön asiakas lukisi määräämiksi organisaation tiedonhaun ominaisuuksiksi sellaisetkin ominaisuudet, joita ilman voitaisiin hyvin selvitä perushakuongelmien suhteen. Viitekehyksen tulisi siis aivan päällimmäisenä vastata kysymykseen, missä kulkee raja, jolloin tiedonhakujärjestelmää voidaan oikeutetusti kutsua organisaation tiedonhakujärjestelmäksi. Toisaalta organisaation tiedonhakujärjestelmää on voitava verrata olemassa oleviin ratkaisuihin: Onko tarkasteltava järjestelmä muiden vastaavien kaltainen, ja millä tasolla sen eri ominaisuudet ovat verrattuna muihin järjestelmiin?

5.2 Viitekehyksen kuvaustapa

Tietojärjestelmät voidaan asiakas–toimittaja-näkökulmasta ymmärtää perinteisen vaatimusmäärittelyn termein, toisin sanoen aluksi voidaan tutkia, mitä teknisiä ja laadullisia ominaisuuksia järjestelmältä odotetaan ja sitten toteuttaa nämä ominaisuudet kattava järjestelmä. Tällainen toiminnallisuuden analysointi on oleellinen vaihe myös organisaation tiedonhakujärjestelmän arvioinnissa. Organisaation tiedonhakujärjestelmän voidaan luvun 3.6.2 mukaisesti odottaa noudattavan tietynlaista perusarkkiteh-

Luokka	Arvioitavat ominaisuudet
Ydintoiminnot	tiedonhakumallit, parametroitu haku
Liittymät	tietolähteet, dokumenttiformaatit
Käyttäjät	käyttöliittymäominaisuudet, kontekstin huomioiminen
Tietoturva	dokumenttien luottamuksellisuus, tuetut autentikointitavat

Taulukko 5.1: Viitekehyksen luokat.

tuuria. Sen voidaan olettaa toteuttavan tämän arkkitehtuurin kuvaamat elementit niiden sisältämine ominaisuuksineen.

5.3 Viitekehyksen määrittely

Viitekehyksen lähtökohtana on luvussa 3 esitetty organisaation tiedonhaun aiempi tutkimus: organisaation tiedonhaun tutkimuksen erilaiset ulottuvuudet ja niiden pohjalta hahmoteltu organisaation tiedonhakujärjestelmien perusarkkitehtuuri. Tämän teorian vastapainoksi aikaisemmin tässä luvussa on esitelty omisteisia organisaation tiedonhakujärjestelmiä. Ne kertovat, mitkä ominaisuudet itse organisaatioissa on koettu tarpeellisimmiksi. Kovin formaalia viitekehystä näistä lähtökohdista on mahdotonta rakentaa. Koska organisaation tiedonhaku käsittää monia erilaisia, osin hyvinkin laajoja kokonaisuuksia, tulisi formalisoidusta viitekehyksestä väkisinkin maailmoja syleilevä eli se on ajatuksenkin tasolla mahdoton. Perusarkkitehtuurin elementit rajaavat kuitenkin kukin sisäänsä joukon selkeästi hahmotettavia ominaisuuksia, ja arkkitehtuurin tavoite toisaalta on esittää koko tiedonhaun ongelmakentän ominaisimmat kysymykset järjestelmän elementeiksi koottuina. Tämä antaa aiheen ryhmitellä nämä ominaisuudet abstraktiotasoltaan hieman elementtejä korkeammiksi luokiksi. Pidättäytyminen pelkästään teknisisissä ominaisuuksissa jättäisi järjestelmän laadulliset ominaisuudet huomiotta. Tällaisia laadullisia ominaisuuksia on löydettävissä kaikista organisaation tiedonhaun osa-alueista, joten ne on syytä ottaa mukaan tarkasteluun. Teknisten ja laadullisten ominaisuuksien voidaan nähdä tukevan toisiaan, joten kokonaiskuvan muodostumisen kannalta ei ole välttämätöntä tehdä eroa näiden kahden välillä.

Taulukossa 5.1 on esitetty viitekehyksen muodostava luokittelu. Taulukossa on luokkien lisäksi mainittu kunkin luokan sisältämät ominaisuudet. Luokkien valinnat ja tarkempi sisältö on perusteltu seuraavissa alaluvuissa.

5.3.1 Ydintoiminnot

Ydintoiminnoiksi on luettu hakumoottori toimintoineen. Tiedonhaun kirjallisuudessa ja tutkimuksessa indeksitietorakenteet ja tiedonhakumallit algoritmeineen ovat kautta linjan perusta muulle teorialle. Siksi niiden voidaan katsoa kuuluvan ydintoimintojeen luokkaan, vaikka ne voitaisiin lukea osaksi indeksinhallintaakin. Indeksoinnin ja haun tehokkuus on tiedonhakujärjestelmän tehoon olennaisesti vaikuttava tekijä, joten suorituskykyä ja sen edellytyksiä on tässä yhteydessä syytä tarkastella. Pelkkä kokotekstihaku ei useinkaan kata kaikkia organisaation tiedonhaun tarpeita, joten myös parametroitua hakua ja sen sovellutuksia, kuten semanttista hakua, on syytä tarkastella.

Omisteisten ratkaisujen toimittajat eivät usein ilmoita, miten ratkaisujen indeksointialgoritmit on toteutettu tai mitä tiedonhakumalleja on käytetty. Näitä seikkoja saatetaan käsitellä usein jopa yrityssalaisuuksina. Tämä kuitenkin osaltaan todistaa niiden tärkeyden puolesta. Indeksointi ja tiedonhakumallit ovat toisaalta niin olennaisia tiedonhakujärjestelmän osa-alueita, ettei niiden esitteleminen tuotedokumentaatiossa toisi mitään oleellista tietoa tuotteen sellaisista ominaisuuksista, jotka erottavat ne kilpailijoista. Korkeintaan voidaan korostaa esimerkiksi indeksoinnin tai tiedonhakumallien algoritmien tehokkuutta, mutta tällaiset lauseet jäävät helposti vaille tarkempia perusteluja, mikäli ratkaisun toteutusta ei ole mahdollista tarkastella.

Ydintoimintojen osalta on syytä kiinnittää erityistä huomiota seuraaviin kysymyksiin:

- Minkälainen tiedonhakumalli on käytössä? Onko käytetty esimerkiksi vektorimalliin perustuvaa mallia, vai yhdistelty useampia malleja?
- Onko parametroitu haku tuettu? Entä mitkä edellytykset ovat entiteettihaulle tai muun kaltaiselle semanttiselle haulle?

5.3.2 Liittymät

Liittymät pitää sisällään indeksinhallinnan ja siihen osakokonaisuuksina kuuluvat tietolähteiden ja dokumenttityyppien käsitteet. Indeksinhallintaa ei ole käsitelty ydintoimintojen ohessa, jonne sen myös luontevasti voisi sijoittaa, koska on katsottu, että indeksinhallinta liittyy läheisesti tietolähteiden ja dokumenttiformaattien huomioimiseen. Indeksinhallintaan liittyy myös federoidun haun käsite, joten federoitu haku on nostettava tarkasteluun mukaan. Liittymän käsite kuvaa organisaation tiedonhaun yleisasetelmaa: tiedonhakujärjestelmä koostuu lähtökohtaisesti useista eri tietolähteistä, joiden piiristä tietoa haetaan. Indeksinhallinta liittää nämä tietolähteet haun piiriin.

Mikäli tietolähteet ovat hakuominaisuuksia sisältäviä osajärjestelmiä, voidaan federoitua hakua käyttää tulosten liittämiseksi haun piiriin.

Organisaation tiedonhakujärjestelmien toimittajien tavoitteena on luonnollisesti rakentaa tiedonhakujärjestelmän liittymät mahdollisimman kattaviksi. Mitä laajemmin organisaation dokumentit ovat haun piirissä, sitä laajempaa hyötyä tiedonhakujärjestelmän voidaan katsoa tuottavan. Tämä on usein suurten kaupallisten toimittajien myyntivaltti, sillä niiden on helppo tukea omia tietolähteitään ja formaattejaan, ja toisaalta ne kykenevät investoimaan suuria summia asiakasorganisaation tai kolmansien osapuolten tietolähteiden integroimiseksi haun piiriin. Onkin selvää, että omisteisten ratkaisujen toimittajat herkästi mainitsevat, kuinka monta sataa tai tuhatta tietolähdettä tai dokumenttityyppiä heidän ratkaisunsa tukee.

Liittymien osalta on syytä kiinnittää erityistä huomiota seuraaviin kysymyksiin:

- Minkälaisia tietolähteitä tuetaan? Tuetaanko omisteisten ratkaisujen tapaan tietokantajärjestelmiä, ja mitkä edellytykset on liittää erilaisia taustajärjestelmiä haun piiriin? Onnistuuko federoitu haku?

- Minkälaisia dokumenttiformaatteja tuetaan? Miten hyvin tuetaan omisteisten ratkaisujen näkyvimmin mainostamia Web-, Microsoft Office- ja PDF-formaatteja?

5.3.3 Käyttäjät

Käyttöliittymän tarkastelu vastaa kysymykseen, minkälainen käyttäjäkokemus järjestelmässä on. Kuten luvussa 2 nähtiin, tiedonhaun perustehtävä on tyydyttää käyttäjän tarpeet, ja niiden tyydyttämiseen käyttöliittymän ominaisuudet vaikuttavat keskeisesti. Yksinkertaistaen voidaan tehdä hypoteesi, että tarpeellinen määrä järkevästi toteutettuja ja tehokkaan haun kannalta tarpeellisia teknisiä käyttöliittymäominaisuuksia vaikuttaa positiivisesti käytettävyyteen. Täten varsinaista käytettävyyden teoriaa tai käyttäjäpsykologiaa ei tarkastella, vaan keskitytään käyttöliittymän teknisten ominaisuuksien tutkimiseen. Käyttäjän profilointi ja hakukonteksti ovat tärkeitä organisaation tiedonhakujärjestelmän ominaisuuksia, joten ne on syytä ottaa tarkasteluun mukaan.

Käyttäjän näkökulma tulee usein ensimmäisenä vastaan, kun tarkastellaan omisteisten ratkaisujen tuotedokumentaatiota tai markkinointimateriaalia. Organisaatiot hankkivat organisaation tiedonhakujärjestelmän ensisijaisesti työntekijöitänsä varten. On selvää, että tiedonhakujärjestelmän tulee tukea työntekijöiden usein kirjavaa teknistä osaamista ja vastata heidän erityistarpeisiinsa. Näiden tarpeiden tyydyttämiseksi, erottuakseen kilpailijoista, omisteiset ratkaisut pyrkivät tarjoamaan käyttäjille kehittyneitä käyttöliittymäominaisuuksia.

Käyttäjien osalta on syytä kiinnittää erityistä huomiota seuraaviin kysymyksiin:

- Millaisia käyttöliittymäominaisuuksia tuetaan (esimerkiksi hakutulosten tiivistys, hakulauseen oikoluku, moninäkymähaku)?
- Tuetaanko käyttäjän profilointia ja hakukontekstia?

5.3.4 Tietoturva

Tietoturva on oleellinen osa kaikkia organisaation tiedonhaun ratkaisuja, ellei jopa kaikkia organisaatioissa käytettäviä järjestelmiä, joten se on nostettu omaksi ulottuvuudekseen. Monet tietoturvaominaisuudet heijastuvat nimenomaan käyttöliittymään hakutuloksia rajoittavina tekijöinä. Tietoturvaa ei kuitenkaan pidä nähdä päälle liimattuna rajoittavana ominaisuutena, vaan se on olennainen osa hakuprosessia. Tietoturvaa voisi tarkastella myös liittymät-ulottuvuuden näkökulmasta: organisaation tiedonhakujärjestelmän tulee ottaa huomioon erilaisten tietolähteiden ja taustajärjestelmien tukemat tavat tunnistaa käyttäjä ja kohdentaa dokumenttien käyttöoikeus tietyille käyttäjille.

Tietoturva voidaan nähdä myös osana liittymät-luokkaa. Mikäli liittymäpintoja erilaisiin organisaatioissa käytettäviin tietoturvamekanismeihin ei ole toteutettu valmiiksi, joudutaan niihin todennäköisesti ottamaan kantaa ennen ratkaisun käyttöönottoa. Tässä mielessä ratkaisujen toimittajien intresseissä on tukea mahdollisimman monia käyttäjäntunnistus-, oikeuksienhallinta ja kertakirjausmenetelmiä, jotta eri tietolähteistä noudettujen dokumenttien tietoturvaan voidaan helposti ottaa kantaa.

Tietoturvan osalta on syytä kiinnittää erityistä huomiota seuraaviin kysymyksiin:

- Miten dokumenttien luottamuksellisuus voidaan määrittää?
- Mitä autentikointitapoja tuetaan? Esimerkkeinä omisteisten ratkaisujen kattavasti tukemat LDAP-, SSL- ja NTLM-autentikointitavat.

6 Havainnot Solr-järjestelmästä

Tässä luvussa esitellään Solr-järjestelmä yleisesti sekä tutkimuksen keskeiset havainnot suhteessa luvussa 5 määriteltyyn viitekehykseen.

6.1 Tutkimuksen alkuasetelma

Tarkasteluun valittiin Ubuntu 10.04 LTS ("Lucid Lynx") -käyttöjärjestelmästä valmiiksi paketoidut Solr 1.4.0 ja Lucene 2.9.2. Asennus tapahtui komennolla

```
apt-get install solr-tomcat
```

eli ajoalustana toimivaksi palvelimeksi valittiin Tomcat Jetty-Web-palvelimen sijaan.[1] Solriin on esimerkeissä viitattu URL:llä

```
http://localhost:8080
```

Solr 1.4 on julkaistu 10. marraskuuta 2009, ja se on viimeisin vakaa Solrin versio.[2] Lucenen viimeisin versio on 3.0.1, mutta se on käytännössä Javan versiolle 5 tehty käännös Lucenen versiosta 2.9.2, joka on toteutettu Java 1.4:llä.[3]

Pääasiallisenä lähteenä tutkimuksessa käytettiin Solrin ja Lucenen kotisivuja[4]. Projektien kotisivuilla sijaitsevaa Javadoc-dokumentaatiota käytettiin apuna lähdekoodia[5] tutkittaessa. Muut dokumentaation lähteet on seuraavissa alaluvuissa erikseen mainittu.

6.2 Ydintoiminnot

Solrin käsitys dokumentista perustuu Luceneen, jossa dokumentti on indeksoinnin ja haun atominen osa, kuten tiedonhaun periaatteisiin kuuluu. Dokumentti (Lucenen

[1] Solrin dokumentaatio ei ota kantaa, mikä palvelin on paras Solr-Java-servletin ajamiseen: http://wiki.apache.org/solr/FAQ

[2] http://mirror.eunet.fi/apache/lucene/solr/

[3] http://lucene.apache.org/java/docs/

[4] Solr: http://lucene.apache.org/solr/, ja Lucene: http://lucene.apache.org

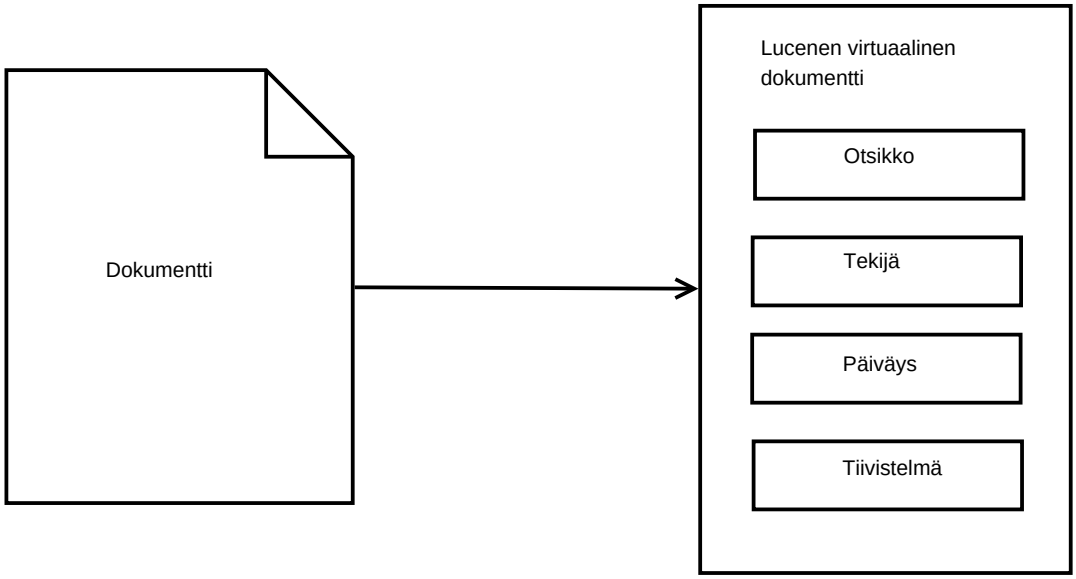
[5] Solrin lähdekoodi: http://mirror.eunet.fi/apache/lucene/solr/1.4.0/, Lucenen lähdekoodi: http://archive.apache.org/dist/lucene/java/lucene-2.9.2-src.tar.gz

Document-Java-luokka) on Lucenen termein vain kokoelma *kenttiä* (engl. fields), joissa on dokumentin varsinainen, haun kannalta kiinnostava tekstuaalinen sisältö. Kentille voidaan määritellä asetuksia, jotka ohjaavat, miten Lucene tarkalleen ottaen indeksoi kyseisen kentän. Lucene siis käsittää dokumentin kahdella tavalla. Tietolähteissä sijaitsevista, varsinaisista dokumenteista tai muusta tiedosta koostetaan tekstiä kentiksi, jotka kapseloidaan Lucenen ymmärtämäksi *"virtuaaliseksi dokumentiksi"*. Nämä virtuaaliset dokumentit toimivat indeksoinnin perustana. Koska varsinainen indeksoitava tieto koostetaan kentiksi, ei indeksoitavan tiedon tarvitse olla dokumentin muodossa, vaan yhtä hyvin esimerkiksi tietokannan kentät voidaan indeksoida. Indeksoitavan tiedon tulee kuitenkin olla tekstipohjaista, sillä Lucene käyttää vain Javan java.lang.string- ja java.io.Reader-luokkia kaikkien operaatioidensa taustalla.

Lucenen virtuaalinen dokumentti voi periaatteessa sisältää vapaasti millaisia kenttiä tahansa. Kentät ovat haun kannalta aina merkkijonotyyppisiä, mutta esimerkiksi lajittelun mahdollistamiseksi halutut kentät voidaan määritellä päivämäärä- tai kokonaislukutyyppisiksi. Lucenen virtuaalidokumentin ja kenttien käsitettä voidaan pitää varsinaisena skeemana vain lainausmerkeissä. Toisaalta esimerkiksi Solrissa käytetään termiä skeema, mikä oikeuttaa sen käytön. Solrissa schema.xml-konfigurointitiedosto määrittelee kaikki mahdolliset kentät ja kenttien ominaisuudet, joita millä tahansa virtuaalisella dokumentilla voi olla, mutta edelleen virtuaaliset dokumentit voivat koostua vapaasti schema.xml:stä löytyvistä kentistä. Tärkeä Lucenen skeeman ominaisuus on sen yksitasoisuus: kaikki kentät ovat keskenään samanarvoisia. Tämä voi tietysti olla rajoite, jos halutaan indeksoida tietoa, jolla on monimutkaisempi rakenne. Virtuaalinen dokumentti mahdollistaa helposti erilaisten entiteettien esittämisen indeksissä. Jos halutaan esimerkiksi koostaa tietolähteistä ihmisiä esittäviä dokumentteja, voidaan virtuaalisen dokumentin kentiksi valita kentät nimi, ikä, titteli, ja niin edelleen.

Virtuaalidokumenttien kentistä valitaan indeksitermit leksikaalisen analyysin avulla. Lucenen leksikaalisessa analyysissa käsitellään siis kenttiä dokumenttien sijasta. Tämänkin seikan takia kenttien valintaan on syytä kiinnittää erityistä huomiota. Lucenen indeksoinnille luvataan 20 megatavun minuuttivauhtia Pentium M 1.5 GHz -koneella. Kekomuistia vaaditaan vain megatavu verran. Lucenen indeksin kooksi luvataan 20 – 30 prosenttia indeksoitavan tekstin koosta.

Lucenen virtuaalisen dokumentin käsitettä on havainnollistettu kuvassa 6.1, jossa virtuaaliseen dokumenttiin on koostettu tietolähteen dokumentistä kentät otsikko, tekijä, päiväys ja tiivistelmä.



Kuva 6.1: Lucenen ”virtuaalinen dokumentti”.

6.2.1 Tiedonhakumallit

Lucene pisteyttää haettuja dokumentteja Boolen mallin ja vektorimallin yhdistelmällä.[6] Boolen malli määrittelee kentät, jotka täsmäävät hakuehtoihin, ja nämä kentät pisteytetään vektorimallin avulla. Pisteytyksen tulokset muutetaan dokumenttien keskinäisiksi pisteiksi ja dokumentit palautetaan hakutuloksissa.

Käytetty vektorimalli on aiemmin esitellyn kaavan 2.1 mukainen. Lucene sisältää kuitenkin lisäominaisuuksia perinteiseen vektorimalliin nähden. Mielenkiintoinen ja tärkein lisäominaisuus on haluttujen dokumenttien tai kenttien *korostus* (engl. boosting) hakutuloksissa, joka tarkoittaa, että indeksoinnin tai haun yhteydessä voidaan määritellä, että tietyllä dokumentilla tai kentällä on voimakkaampi painoarvo pisteytystä laskettaessa. Dokumenteillä on oletuksena korostus 1.0, jota kasvattamalla tai pienentämällä dokumentin painoarvoa hakutuloksissa voidaan muuttaa. Yksinkertaistettuna dokumentin kenttien pistemääristä saadaan dokumentin pistemäärä summaamalla kenttien pistemäärät yhteen, ottaen samalla korostukset huomioon. Lucenen dokumentaation mukaan sopivan painoarvon löytäminen vaatii yleensä kokeilua, ja useimmat hakusovellukset pärjäävätkin ilman korostuksia. Kun dokumenttien korostusta muutetaan, tarkoittaa tämä kaikkien dokumentin kenttien korostusten muuttamista samaan arvoon. Kuten todettua, myös yksittäisten kenttien korostuksia voidaan muuttaa, mikä tarkoittaa intuitiivisesti, että esimerkiksi sähköpostiviestin Subject-kentälle voitaisiin antaa suurempi painoarvo kuin Body-kentälle.

[6] http://lucene.apache.org/java/2_9_2/api/core/org/apache/lucene/search/Similarity.html

Lucenen tiedonhakumallia on mahdollista muokata kirjoittamalla oma toteutus tiedonhakumalliin liittyvistä Java-luokista (liikkeelle kannattaa lähteä Similarity-luokasta[7]). Kyselyä edustavilla luokilla on myös suuri vaikutus siihen, mitä tuloksia haku palauttaa. Niiden osalta ei heti ole tarvetta lähteä kirjoittamaan omaa toteutusta (Query-, Weight- ja Scorer-luokat ovat tällöin hyviä lähtökohtia), vaan Query-luokan erilaisia aliluokkia voidaan käyttää tarpeen mukaan.[8]

6.2.2 Parametroitu haku

Kuvassa 6.1 esitetty Lucenen virtuaalidokumentti voidaan esittää XML-tagein seuraavasti:

```
<doc>
  <field name=''id''>3007WFP</field>
  <field name=''name''>Dell Widescreen UltraSharp 3007WFP</field>
  <field name=''manu''>Dell, Inc.</field>
  <field name=''cat''>electronics</field>
  <field name=''cat''>monitor</field>
  <field name=''features''>30-inch TFT active matrix LCD</field>
  <field name=''includes''>USB cable</field>
  <field name=''weight''>401.6</field>
  <field name=''price''>2199</field>
  <field name=''popularity''>6</field>
  <field name=''inStock''>true</field>
</doc>
```

Kuten aikaisemin todettiin, Lucenen virtuaalidokumenttien malli kykenee helposti esittämään erilaisia entiteettejä, tässä esimerkissä elektroniikkatuotetta. Parametroitu haku onnistuu suoraan: minkä tahansa dokumentin kentän nimi voidaan antaa hakuehdoksi, jolloin hakusanaa verrataan vain kyseisen kentän arvoon. Esimerkiksi dokumenttiformaatti voitaisiin tuoda virtuaalidokumentin kentäksi, jolloin sitä olisi mahdollista käyttää parametroidun haun parametrina.

Kuten nähdään, parametroidun haun toteutus ja erilaisten entiteettien hakeminen on Lucenessa yksinkertaista. Paljon vaikeampi ongelma on entiteettien tunnistaminen ja kokoaminen tietolähteistä.

[7] http://lucene.apache.org/java/2_9_2/api/core/org/apache/lucene/search/package-summary.html#changingSimilarity

[8] http://lucene.apache.org/java/2_9_2/api/core/org/apache/lucene/search/package-summary.html#query

6.3 Liittymät

Solr kommunikoi ulospäin pääasiassa HTTP-protokollan avulla. Solrissa on toteutettu HTTP-rajapintoja, Solrin termein tapahtumankäsittelijöitä (engl. request handlers), joiden läpi voidaan toteuttaa kyselyn suorittamisen lisäksi indeksinhallinnan operaatiot. Tapahtumankäsittelijät ovat tekniseltä toteutukseltaan Web-palveluita (engl. Web Service), joita ajetaan Solrin Web-palvelimen alaisuudessa. Erilaiset integrointirajapinnat, kuten SolrJ ja Solr Flare, hyödyntävät näitä HTTP-rajapintoja. SolrJ-rajapinnan avulla voidaan toteuttaa HTTP-integraation lisäksi myös integraatio ilman verkkokerrosta tai sovellusten välistä kommunikaatiota. Tämä on tarpeellista esimerkiksi sulautetuille järjestelmille, joiden suorituskykyvaatimuksia HTTP-integraatio ei välttämättä täytä. SolrJ- ja Solr Flare -rajapinnat ohitetaan tässä ja keskitytään tapahtumankäsittelijöiden tutkimiseen. Huomionarvoista on, että kaikki liittymätavat vaativat konfigurointia tai koodausta.

HTTP-rajapinnan läpi ei suorituskykysyistä ole välttämätöntä siirtää indeksoitavaa dataa, vaan siihen voidaan kohdistaa ainoastaan Solria ohjaavat komennot. Itse datan Solr voi hakea rajapintakomennossa annetun viitteen (URL tai tiedostoviite) takaa, esimerkiksi Solrin asennuspalvelimen kiintolevyltä tai tietokannasta.

Solrin indeksinhallinnan perusrajapinta on Update-tapahtumankäsittelijä, jonka osoite on

```
http://localhost:8080/solr/update
```

Tapahtumankäsittelijä ottaa vastaan muodollisesti XML:ää mukailevia komentoja. Palvelua vastaan voidaan kohdistaa taulukon 6.1 mukaisia komentoja, jotka pohjautuvat Lucenen indeksointimalliin.

Komentojen syntaksi on varsin yksinkertainen. Esimerkiksi add-komento näyttää seuraavalta:

```
<add>
  <doc>
    <field name=''employeeId''>05991</field>
    <field name=''office''>Bridgewater</field>
    <field name=''skills''>Perl</field>
    <field name=''skills''>Java</field>
  </doc>
  [<doc> ... </doc>[<doc> ... </doc>]]
</add>
```

Komento	Kuvaus
add/update	Lisää tai päivittää XML-pohjaisen dokumentin Lucene-indeksiin. Dokumentille annetaan yksilöllinen tunniste. Dokumentti voidaan antaa osana komentoa tai viitteenä (URL tai tiedostonimi). Dokumentti on Lucenen virtuaalidokumentti, ja sen kenttien tulee pohjautua Solriin konfiguroituun skeemaan (schema.xml).
delete	Poistaa dokumentin yksilöllisen tunnisteen tai kyselyn perusteella.
commit	Toteuttaa (engl. commit) indeksiin tehdyt muutokset. Vasta tämän komennon jälkeen lisäykset tai poistot astuvat voimaan. Lucenessa ei kuitenkaan ole transaktion käsitettä, eli kahden samanaikaisen lisäyksen tai poiston toimenpiteet voivat mennä ristiin keskenään, ellei tilannetta huomioida.
rollback	Toteuttamattomat muutokset voidaan perua rollback-komennolla. Edellisen rivin huomio transaktioiden puutteesta pätee tässäkin.
optimize	Lucenen indeksi on toteutettu sisäisesti segmentteinä, ja uudet lisäykset muodostavat automaattisesti uuden segmentin. Yhden commit-komennon aikana tehdyt muutokset muodostavat automaattisesti uuden segmentin. Useampi kuin yksi segmentti tarkoittaa, että indeksi on epäoptimaalisessa tilassa, joten optimize-komento auttaa korjaamaan tilanteen. Lucene tekee optimointia myös automaattisesti, mutta ei kuitenkaan takaa, että indeksi olisi aina optimaalisessa tilassa.

Taulukko 6.1: Solrin indeksinhallinnan komennot.

Hakasuluissa olevat doc-tagit esittävät, että dokumentteja saadaan yhden komennon avulla indeksoitua tarvittaessa useampia.

CSV-dokumentteja voidaan indeksoida lähettämällä ne CSV-tapahtumankäsittelijälle:

```
http://localhost:8080/solr/update/csv
```

Datan indeksointi tietokannasta JDBC-adapterin avulla tai XML-dokumentista HTTP:n yli voidaan tehdä DataImportHandler (DIH) -nimisen tapahtumankäsittelijän avulla. DIH:n periaate on, että sille määritellään XML-konfiguraatio, jonka mukaan tietolähteestä koostetaan indeksoitavat kentät. Solrin version 1.4 myötä on DIH:iin lisätty tuki sähköpostien indeksoinnille IMAP-protokollan yli[9], sekä Apache Tika -kirjastoa hyödyntävä TikaEntityProcessor[10], jonka avulla voidaan indeksoida tekstiä erilaisista dokumenttiformaateista. DIH-tapahtumankäsittelijän osoite on

```
http://localhost:8080/solr/db/dataimport
```

ExtractingRequestHandler eli Solr Cell tarjoaa vaihtoehtoisen ja suositeltavan keinon Apache Tikan hyödyntämiseksi. Tapahtumankäsittelijä sijaitsee osoitteessa

```
http://localhost:8080/solr/update/extract
```

Solr Cell -tapahtumankäsittelijä toimii vastaanottamalla Tikalta XHTML-tietovirran, joka syötetään Solrissa sijaitsevalle SAX-käsittelijälle. SAX-tapahtumien perusteella luodaan indeksoitavat kentät. Oletuksena kaikki data viedään yhteen kenttään, mutta tämä on määriteltävissä omassa konfiguraatiotiedostossaan. Solr Cell ottaa lisäksi vastaan Update-tapahtumankäsittelijän tapaan commit- ja rollback-komentoja, joilla muutoksia voidaan paremmin hallita.

Solrista löytyy vielä AnalysisRequestHandler-tapahtumankäsittelijä, jonka avulla voidaan simuloida Update-tapahtumankäsittelijän toimintaa. AnalysisRequestHandler ei tee muutoksia indeksiin, vaan palauttaa ainoastaan tiedon siitä, miten sille lähetetyt kentät olisi indeksoitu, toisin sanoen mitä indeksitermejä niiden perusteella olisi muodostunut.

Lähempänä indeksointimoottoria on Lucene Connectors Framework (LCF) -sovellus[11]. LCF ei ole osa Solria eikä sitä ole vielä virallisesti hyväksytty Apache-projektiksikaan, mutta se tukee jo lukuisia tietolähteitä, muun muassa tiedostojärjestelmiä, JDBC-tietokantayhteyksiä, Webiä ja erilaisia kolmannen osapuolen sovelluksia kuten SharePointia. Tärkeä huomio LCF.n yhteydessä on, että omisteisia ratkaisuja integroitaessa

[9] http://wiki.apache.org/solr/MailEntityProcessor
[10] http://wiki.apache.org/solr/TikaEntityProcessor
[11] http://incubator.apache.org/connectors/

joudutaan LCF:n toteutukseen väistämättä sisällyttämään omisteista koodia kirjastojen tai muun materiaalin muodossa. Itse LCF on kuitenkin avoimen lähdekoodin projekti. LCF:n avulla on mahdollista määritellä ja ajastaa ajoja, jotka määräajoin hakevat päivittyneitä dokumentteja tietolähteistä ja automaattisesti indeksoivat ne tuettuihin indekseihin, esimerkiksi Solriin Solr Cellin avulla. On huomionarvoista, ettei LCF:kään toteuta varsinaista federoitua hakua – koska se hakee dokumentteja eikä hakutuloksia – joten mikäli federoidulle haulle on tarvetta, on se rakennettava Solriin erikseen.

6.3.1 Tietolähteet

LCF:n avulla voidaan käyttää seuraavia tietolähteitä:

- Unix- ja Windows-tiedostojärjestelmät
- Windows-levyjaot
- JDBC
- RSS (Atom, RSS 2.0 ym.)
- Web (HTML 1.0, 1.1, 2.0, Atom, RSS 2.0, ym.)
- FileNet P8
- Documentum
- LiveLink
- Patriarch
- Meridio
- SharePoint 2003 ja 2007

Solrin tapahtumankäsittelijöiden avulla voidaan indeksoida tietoa periaatteessa mistä tahansa tietolähteestä, mutta tapahtumankäsittelijät eivät ota kantaa indeksointimoottorin operaatioihin. Tieto on siis ensin saatettava tapahtumankäsittelijöiden tietoon.[12] Tiedon ollessa levyjärjestelmällä tämä voidaan tehdä esimerkiksi yksinkertaisella skriptillä, joka ajastetaan ajettavaksi tarpeen mukaan[13], esimerkiksi viikoittain; muissa tapauksissa tällainen lähestymistapa ei välttämättä riitä. LCF on vielä kehitysvaiheessa

[12]Tätä kysymystä käsitellään yllättävän vähän esimerkiksi Smileyn ja Pughin kirjassa [47]. Tiedon lisääminen indeksiin on harvoin kertaluonteinen operaatio, vaan indeksiä tulisi päivittää tiedon muuttuessa.

[13]Tällainen esimerkkiskripti ”start.jar” löytyy Solrin lähdekoodista.

oleva projekti, mutta se näyttää lupaavalta tässä suhteessa, eikä toisaalta ole sidonnainen edes Solriin, vaan sitä voidaan käyttää periaatteessa muidenkin hakukoneiden yhteydessä.

HTTP-rajapintojen avulla Solr on helposti integroitavissa muiden järjestelmien kanssa, mutta Java, JavaScript- (JSON), Ruby- ja PHP-kielille löytyy lisäksi erityisiä ohjelmointirajapintoja, joilla uusia tietolähteitä voidaan lisätä haun piiriin.

6.3.2 Dokumenttiformaatit

Apache Tika tukee seuraavia dokumenttiformaatteja:

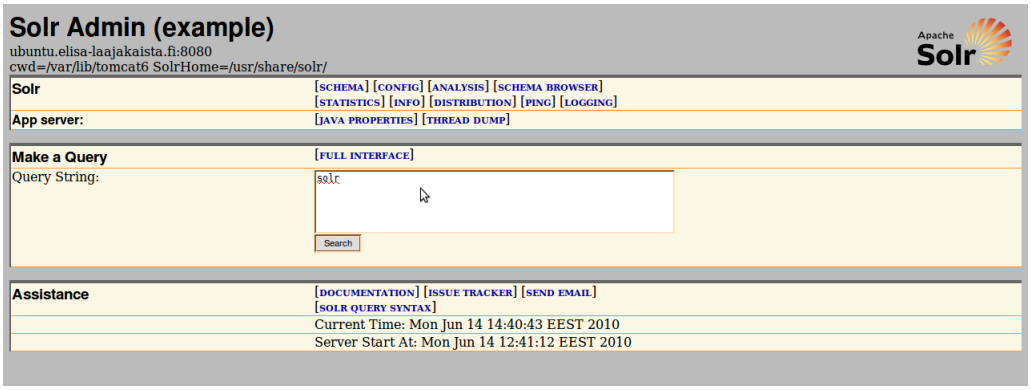
- HTML (TagSoup-kirjaston avulla; tukee lähes kaikkia HTML-muotoja),
- XML,
- Microsoft Office (sekä uusi OOXML-formaatti että vanhemmat formaatit alkaen Microsoft Officen versiosta 97),
- OpenDocument (ODF) sekä aikaisempi OpenOffice 1.0 -formaatti,
- Portable Document Format (PDF),
- Electronic Publication Format (EPUB),
- RTF,
- Erilaisten pakkausformaattien sisällä oleva tieto: tar, zip, bzip2, cpio, ar, gz, jar,
- Eri merkistöillä koodatut tekstitiedostot,[14]
- Audio- ja midi-tiedostot, mukaan lukien MP3-formaatti,
- javax.imageio-kirjaston mukaiset kuvatiedostot ja EXIF-tiedostojen parserointi JPEG-kuvista,
- Flash-videoformaatti (FLV),
- Javan class-tiedostot (sekä JAR-paketit),
- Unixin mbox-formaatti.

[14]Tekstitiedostojen tukeminen ei välttämättä ole triviaalia, koska merkistökoodaus on muunnettava UTF-8:aan, joka on ainoa Lucenen tukema merkistökoodaus.

Solrin tapahtumankäsittelijät tukevat näiden lisäksi vielä CSV-formaattia. XML:ää voi Tikan lisäksi indeksoida Update- ja DIH-tapahtumankäsittelijöiden avulla. Näissäkin tapauksissa oletuksena on, että indeksoitava tieto on UTF-8-merkistökoodauksen mukaista.

6.4 Käyttäjät

Solr tarjoaa oletuksena Web-käyttöliittymän[15], jonka kautta hakuja voidaan tehdä. Käyttöliittymä on esimerkinomainen ja suunniteltu järjestelmänvalvojan näkökulmasta varsin tekniseksi, eikä siten ole välttämättä helposti lähestyttävä loppukäyttäjälle. Kuva 6.2 on kuvakaappaus käyttöliittymän perusnäkymästä ja kuvasta 6.3 nähdään, miltä edistynyt näkymä näyttää.



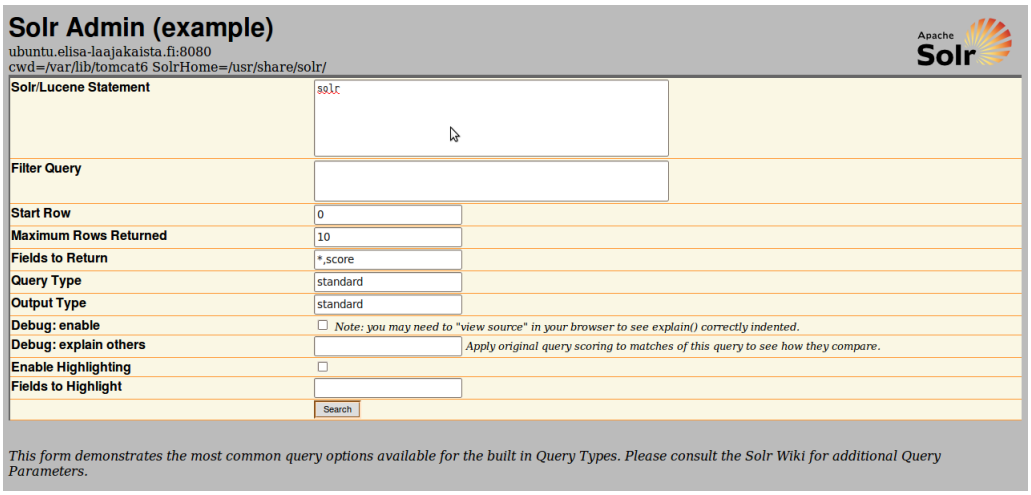
Kuva 6.2: Solrin esimerkkikäyttöliittymän perusnäkymä.

Perusnäkymä koostaa erilaisia Solr-palvelimen järjestelmänvalvojalle suunnattuja palveluja yhdelle sivulle. Sivun kautta pääsee tarkastelemaan Solrin XML-pohjaisia schema- ja config-pääkonfiguraatiotiedostoja, logeja, statistiikkaa, ja muuttamaan erinäisiä palvelimen parametreja. Kehittäjiä varten sivulle on koottu linkkejä esimerkiksi Solrin dokumentaatioon ja ohjelmistovirheiden jäljityssovellukseen. Myös kyselyn tekeminen on mahdollista, mutta paremmin kyselyjen kokeileminen onnistuu edistyneestä näkymästä, joka on käytännössä yksinkertainen Web-lomake ja esittelee tavallisimpia Solrille lähetettäviä hakuparametreja.

Solr tukee erilaisia kyselykieliä liitännäisarkkitehtuurin kautta, mutta oletuksena käytetty kyselykieli on laajennos Lucene-kirjaston kyselykielestä[16]. Hakulauseet ja

[15]Solrin käyttöliittymä pyörii palvelimella, jolle Solr on asennettu: http://localhost:8080/solr/admin/

[16]http://wiki.apache.org/solr/SolrQuerySyntax



Kuva 6.3: Solrin esimerkkikäyttöliittymän edistynyt näkymä.

haun vastaukset kulkevat indeksinhallinnan komentojen tavoin HTTP-pyyntöjen kautta Solrille ja takaisin, joten käyttöliittymä voidaan rakentaa tämän perustan päälle. Esimerkiksi seuraava kysely määrittelee hakutermeiksi (parametri q) ”organisaatio” ja ”tiedonhaku”:

```
http://localhost:8080/solr/select?indent=on&version=2.2&
q=organisaatio%20tiedonhaku&q.op=AND&start=0&rows=10&
qt=standard&wt=standard
```

Muut parametrit ohjaavat Solrin toimintaa. Esimerkiksi q.op määrittelee, yhdistetäänkö hakutermit AND- vai OR-operaattoreilla toisiinsa. Hakutulosten sivutukseen voidaan vaikuttaa start ja rows-parametreilla. Parametri qt määrittelee käytettävän *hakukäsittelijän* (engl. Request Handler), joka on vastuussa hakulauseen tulkitsemisesta ja vaikuttaa siten myös hakulauseen syntaksiin ja pakollisiin parametreihin. Solr palauttaa kyselyn vastauksen oletuksena XML-muodossa (wt-parametri), mutta muitakin muotoja tuetaan (esimerkkeinä JSON sekä eval-funktiolle annettava Python-, PHP- tai Ruby-koodi). Vastaus voi näyttää esimerkiksi seuraavalta:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<response>
  <responseHeader>
    <status>0</status>
    <QTime>2</QTime>
  </responseHeader>
  <result numFound=”1” start=”0”>
```

```
    <doc>
      <str name=''id''>221</str>
      <str name=''name''>Organisaation tiedonhaku</str>
    </doc>
  </result>
</response>
```

Vastaus on varsin luettava: dokumentin kentät palautetaan doc-tagien sisällä; elementti responseHeader kapseloi yleistietoa haun tuloksista, ja vastaavaa tietoa saadaan result-elementin attribuuteista. Mahdolliset lisätoiminnallisuudet, kuten hakutulosten korostus tai moninäkymähaku toimivat samalla mekanismilla: niiden parametrit annetaan HTTP-kyselyn välityksellä ja niihin liittyvä hakuvastaus lähetetään osana vastausviestiä.

6.4.1 Käyttöliittymäominaisuudet

Solr tukee fraasien, jokerimerkkihakujen (engl. wildcard queries), erilaisten arvovälien ja Levensteinin etäisyysalgoritmiin perustuvan samankaltaisuushaun lisäksi seuraavia käyttöliittymäominaisuuksia:

Moninäkymähaku perustuu kenttiin, päivämääriin tai kyselyyn (Solr laskee kyselyyn täsmäävien tulosten lukumääriä). Näkymät muodostetaan skeeman kenttien perusteella, ja hierarkkinen porautuminen on mahdollista.

Hakutulosten korostaminen tarkoittaa, että hakutuloksissa näytetään hakutermeihin täsmäävää tekstiä dokumenteista.

Kyselyn muokkaaminen (engl. Query Elevation) on hyödyllistä, mikäli halutaan määritellä, että tietyt dokumentit näkyvät tietyillä hakusanoilla aina ylimpänä hakutuloksissa. Vastaavasti voidaan määritellä, että tietyllä hakusanalla ei jotakin dokumenttia näytetä hakutuloksissa, vaikka haku muuten täsmäisi.

Hakutulosten oikoluku tarkoittaa, että Solr voi esittää hakutuloksissa oikolukuehdotuksia perustuen sille määriteltyyn sanastoon (esim. englannin kielen sanoja listaava tekstitiedosto).

More like this on klusteroinnin sovellus hakutuloksia vastaavien – joskaan ei suoraan hakulauseeseen täsmäävien – dokumenttien ehdottamiseen.

Samankaltaisten hakutulosten ryhmittely on tuttu esimerkiksi Google-hakukoneesta. Mikäli hakutuloksilla (virtuaalisilla dokumenteilla) on samanlaisia kenttiä (esimerkiksi, jos indeksoitaisiin musiikkikappaleita), voidaan hakutulokset ryhmitellä valitun kentän (esimerkiksi kappaleen esittäjä) mukaan.

6.4.2 Kontekstiperustainen haku

Solr ei tue käyttäjien profilointia tai hakukontekstin huomioimista millään tavalla. Solria ajavan Web-palvelimen keinoin on toki mahdollista rakentaa toiminnallisuus käyttäjän profiilitietojen tutkimiseen. Profiilitiedot voidaan kuljettaa kyselyn mukana Solrille, jos tätä varten määritellään oma hakukäsittelijä, joka osaa ottaa parametrit huomioon ja välittää ne eteenpäin. Toinen tapa on tehdä räätälöity toteutus Solrin Java-luokista. Joka tapauksessa tarvitaan runsaasti räätälöintiä, jotta käyttäjäprofiiliin saati hakukontekstiin voidaan ottaa mitään kantaa.

6.5 Tietoturva

Oletuksena pääsy Solr-järjestelmää ajavaan Web-palvelimeen on avoin kaikille osapuolille, jotka saavat yhteyden siihen. Myöskään indeksinhallinnan operaatioiden käyttöoikeuksia ei ole oletuksena rajoitettu. Koska Solr on Java WAR-pakettina toteutettu Java Servlet, Solrin pääsynhallintaa voidaan hallinnoida Web-palvelimen (Tomcat tai Jetty) pääsynhallinnan keinoin tai rajoittamalla pääsyä esimerkiksi palomuurin avulla.

Palvelimen pääsynhallinta on joka tapauksessa hoidettava ennen puuttumista hakutulosten pääsynhallintaan, koska vapaa pääsy palvelimelle ja Solrin adminointioikeudet voivat avata käyttäjälle pääsyn dokumentteihin, joita hän ei tietoturvapolitiikan mukaan saisi nähdä.

6.5.1 Dokumenttien luottamuksellisuus

Solr ei ota kantaa dokumenttien luottamuksellisuuteen, vaan suosittelee käytettäväksi LCF:ää, joka tarjoaa indeksointimoottorin operaatioiden lisäksi mahdollisuuden selvittää käyttäjän oikeudet indeksoituun tiedostoon. Prosessi on seuraavan kaltainen:

1. LCF:ää on käytettävä dokumenttien keräämiseen ja lähettämiseen Solrin indeksoitavaksi

2. Haettaessa LCF:n autorisointipalvelulta on kysyttävä hakua suorittavan käyttäjän oikeudet

3. Hakutulokset on filtteröitävä tai hakulausetta on muokattava ennen hakuoperaatiota, jotta käyttäjän oikeudet voidaan täsmätä löytyneiden hakutulosten oikeuksiin

Kohdassa 1 LCF lähettää indeksoitavien dokumenttien mukana tiedon käyttäjätunnuksista, joilla on oikeus nähdä dokumentti. Kohdassa 2 haetaan hakua suorittavan käyttäjän käyttäjätunnustiedot, ja kohdassa 3 yhdistetään hakua suorittavan käyttäjän oikeudet hakutulosten oikeuksiin, jotta nähdään, onko käyttäjällä oikeutta nähdä dokumentti.

LCF:n käyttöönotto vaatii paljon manuaalista työtä. Esimerkiksi Solrin dokumentaatiossa sitä on käsitelty vain yhden rivin verran; Smileyn ja Pughin kirjassa [47] ei lainkaan. Tärkein tietolähde onkin siis LCF:n kotisivu[17], joka sekin vaikuttaa keskeneräiseltä.

Smiley ja Pugh [47] esittävät dokumenttien luottamuksellisuuden määritettäväksi moninäkymähaun avulla. Solrin moninäkymähaku mahdollistaa dokumenttien kategorisoimisen kategorioihin, jotka näkyvät vain tietyille käyttäjille.

Solrin wiki ehdottaa LCF:n lisäksi hakukäsittelijä-mekanismin käyttöä dokumenttien luottamuksellisuuden määrittämiseen. Tämä vaatii omaa toteutusta SolrDispatchFilter-Java-luokasta, tai vähintään uusia hakuparametreja, joissa käyttäjätieto välitetään.

6.5.2 Tuetut autentikointitavat

Koska Solr ei ota kantaa tietolähteiden tietoturvaan, on autentikointitapojen tuki LCF:n varassa. LCF ei varsinaisesti tue autentikointia eli käyttäjän tunnistamista lainkaan. Sen sijaan LCF keskittyy autorisointiin eli käyttäjän käyttöoikeuksien määrittämiseen. Autentikointi on siis hallittava jollakin muulla menetelmällä. LCF tukee Microsoft AD:n kaltaista tapaa hallinnoida käyttäjätietoja, ja autorisointia tuetaan suhteessa kaikkiin tuettuihin tietolähteisiin. Erityisesti tuettujen kolmansien osapuolten sovellusten lista on tässä suhteessa kiinnostava:

- FileNet P8,
- Documentum,
- LiveLink,
- Patriarch,

[17] http://incubator.apache.org/connectors/

- Meridio,
- SharePoint 2003 ja 2007.

Kaikki edellä mainitut sovellukset ovat kaupallisia sisällönhallintajärjestelmiä, joskin esimerkiksi SharePointissa on paljon muitakin ominaisuuksia.

6.6 Johtopäätökset

Päällimmäinen havainto Solrista on, että se on pikemmin ohjelmisto- tai palvelinalusta kuin varsinainen omisteisten ratkaisujen kaltainen järjestelmä käyttöliittymineen ja valmiine elementteineen. Toisaalta Solria ei tällaisena mainostetakaan, ja räätälöinti organisaation käyttöympäristöä vastaavaksi voidaan joka tapauksessa tehdä erilaisia avoimia standardeja ja ohjelmointirajapintoja apuna käyttäen. Vastaavasti Solr on kehitetty pääosin sovelluskehittäjien mieltymysten mukaiseksi; kaupallinen näkökulma puuttuu. Tämäkin on ymmärrettävää, sillä avoimena projektina Solrin ei ole tarkoituskaan olla kaupallinen tuote: tehtävä on jätetty kolmansille osapuolille.

Taulukkoon 6.2 on tiivistetysti koottu Solrin viitekehyksen mukaiset ominaisuudet ja niitä on verrattu luvussa 4 esiteltyjen omisteisten ratkaisujen ominaisuuksiin. Erityisen huomattava puute on kunnollisten liittymien puuttuminen. LCF on lupaava projekti mutta vielä kehitysvaiheessa, eikä siitä ole saatavilla kattavaa dokumentaatiota. Solrin ja Lucenen tapa indeksoida dokumentteja on eheä ja toimiva, mutta tapahtumankäsittelijät eivät ota kantaa indeksointimoottorin toimintoihin, jotka ovat ratkaiseva tekijä mikäli Solria halutaan käyttää autenttisessa ympäristössä. Toinen huomattava puute on tietoturvan huomioiminen. Solr ei oikeastaan huomioi dokumenttien luottamuksellisuuden ongelmaa millään tavalla, vaan vastuu jää tästäkin LCF:lle. Erilaisia standardin mukaisia keinoja (HTTP, XML, Web-palvelut, ohjelmointirajapinnat) on sekä liittymien että tietoturvan puutteiden korjaamiseksi useita, mutta niiden käyttöönotto vaatii huomattavaa räätälöintityötä. Tuettuja dokumenttiformaatteja Solrissa on kuitenkin kohtuullinen määrä. Lucenen indeksointimallin avulla dokumenteista voidaan eristää entiteettejä tai niihin voidaan kohdistaa parametroitua hakua varsin mutkattomasti. Myöskään käyttöliittymäominaisuuksissa Solr ei kalpene omisteisten ratkaisujen rinnalla.

Solrin taustalla oleellisena tekijänä vaikuttava Lucene-kirjasto on varsin huolellisesti rakennettu, ja sillä on edellytykset hyvään suorituskykyyn (jota tosin tämän viitekehyksen puitteissa ei ole mahdollista arvioida). Solr osoittaa, että Lucene-kirjastoa on helppo käyttää erilaisissa yhteyksissä, mutta sen käyttäminen tehokkaasti vaatii paljon räätälöintityötä. Toisaalta Solrin avoin luonne ja kehittäjiä varten suunnattu

laaja dokumentaatio tarjoavat hyvät mahdollisuudet räätälöintien tekemiseen. Organisaation tiedonhaun kohdalla laajennettavuus ja räätälöintimahdollisuudet nousevat korkeaan arvoon, sillä organisaatioiden tietolähteet, dokumenttiformaatit ja oikeastaan koko tiedonhakua koskeva kenttä ovat luonteeltaan heterogeenisiä. Erilaisilla ohjelmointikielillä (päällimmäisinä Java ja JavaScript) sekä standardeilla (HTTP, XML, JSON, Web-palvelut) voidaan Solrin päälle rakentaa lähes millainen (Web-)käyttöliittymä tahansa, ja toki itse sovellustakin on mahdollista muokata. Voidaan kysyä, missä määrin omisteiset ratkaisut tarjoavat tällaisia räätälöintimahdollisuuksia. Ainakaan ne eivät mainosta niitä samaan tapaan kuin varsinaisia hakuominaisuuksia.

Solrin hyödyntämä Lucenen virtuaalidokumenttien malli tarjoaa hyvän alustan erilaisille parametroidun haun sovelluksille. Esimerkiksi semanttisen haun esteenä on usein semantiikan kerääminen hakua varten. Tämä työ ei omisteistenkaan sovellusten tapauksessa kuulu tuotteen hintaan. Myös käytettävien tietolähteiden, tietoturvaratkaisujen ja käyttöoikeuksien osalta työtä on yhtä lailla omisteisissa ja avoimissa ratkaisuissa.

Viitekehyksen luokka	Solr	Omisteiset ratkaisut
Ydintoiminnot: tiedonhakumallit ja parametroitu haku	Tiedonhakumallit muokattavissa tarpeen mukaan. Parametroitu haku onnistuu, kunhan tarvittava tieto saadaan indeksoitua.	Tiedonhakumalleihin vaikea ottaa kantaa. Parametroitu haku kuten Solrin tapauksessa. Esim. Autonomy sisältää monimutkaisempia ominaisuuksia tiedon kokoamiseen ja automaattiseen analysointiin.
Liittymät: tietolähteet ja dokumenttiformaatit	LCF:n avulla voidaan käyttää moninaisia tietolähteitä. Dokumenttiformaatteja kymmeniä erilaisia, mukana myös olennaiset Web-, PDF- ja Microsoft Office -formaatit.	Lukuisia tietolähteitä tuetaan. Dokumenttiformaatteja satoja erilaisia.
Käyttäjät: käyttöliittymäominaisuudet ja kontekstin huomioiminen	Lukuisia käyttöliittymäominaisuuksia, mm. moninäkymähaku. Käyttäjän profilointia ei kuitenkaan erikseen tueta.	Käyttäjän profilointia tuetaan. Käyttöliittymäominaisuudet jonkin verran Solria edellä.
Tietoturva: dokumenttien luottamuksellisuus ja tuetut autentikointitavat	Vain autorisointi LCF:n kautta tuettu; ei autentikointia.	Lukuisia autentikointi- ja autorisointitapoja tuetaan.

Taulukko 6.2: Solr suhteessa omisteisiin ratkaisuihin.

7 Yhteenveto

Tässä tutkielmassa pyrittiin avaamaan organisaation tiedonhaun käsitettä. Aihe on melko uusi, joten organisaation tiedonhaun kartoitukselle haettiin pohjaa aikaisemmalta, vakiintuneelta tiedonhaun tutkimukselta. Organisaation tiedonhakuun liittyviä tutkimusongelmia ja organisaation tiedonhaun tyypillisimpiä piirteitä tarkasteltiin. Kävi ilmi, että nuoresta iästään huolimatta organisaation tiedonhaun perusongelmia on tutkittu jo laajalti. Samaten yritykset tarjoavat hakuongelman ratkaisemiseksi organisaatioille jo moninaisia ratkaisuja. Erityisesti uusimpien tutkimuskohteiden kuten kontekstiperustaisen haun ja entiteettien tunnistamisen osalta tutkimus on vielä nuorta, ja organisaation tiedonhaun tutkimuksen ja ratkaisujen voi odottaa lähitulevaisuudessa radikaalisti kehittyvän.

Organisaatio on keskeisessä roolissa organisaation tiedonhakujärjestelmien määrittelijänä, joten organisaatioiden intresseistä osin riippuu, mitkä ominaisuuksista osoittautuvat tärkeimmiksi tulevaisuudessa. Organisaatioiden toivomukset uusista ominaisuuksista varmasti osaltaan heijastuvat myös akateemiseen tutkimukseen, jolle kuitenkin on onneksi voida määritellä ja tutkia asioita myös atomisella tasolla, ottamatta kantaa siihen, mitä ylätason termi organisaation tiedonhaku tarkalleen ottaen kullakin ajanhetkellä ja kunkin ratkaisutoimittajan tai asiakkaan mielestä pitää sisällään.

Tutkielmassa kehitettiin viitekehys, joka luokittelee organisaation tiedonhakujärjestelmältä vaadittavat ominaisuudet, ja siten auttaa arvioimaan, kuinka hyvin organisaation tiedonhakujärjestelmä kattaa organisaation tiedonhaun perustarpeet. Omisteisia organisaation tiedonhakujärjestelmiä kartoitettiin viitekehyksen taustaksi, ja avointa Apache Solr -sovellusta arvioitiin viitekehystä vasten. Olennainen osa Solr-sovellusta on Apachen Lucene-projekti erilaisine osaprojekteineen. Tämä otettiin tutkielmassa huomioon. Viitekehyksessä pyrittiin ottamaan huomioon koko organisaation tiedonhaun ongelmakenttä, joten sen avulla ei toisaalta voida pureutua kovin syvälle esimerkiksi tiedonhakumallien tai jonkin muun mielenkiinnon kohteen tutkimiseen.

Solrin todettiin täyttävän organisaation tiedonhaun perustarpeet viitekehyksen puitteissa kohtalaisesti. Erityisesti tietoturvan ja liittymäratkaisujen osalta tehtävää on kuitenkin vielä paljon. Solria voisi kuitenkin harkita käytettäväksi kaupallisten vaihtoehtojen rinnalla hyödyntäen esimerkiksi federoitua hakua – se voisi ainakin yksinkertaisimmissa tapauksissa olla jopa vaihtoehto kaupallisten ratkaisujen tilalle. Toisaalta reaalimaailman testausta, esimerkiksi hakutehokkuuden evaluointia, ei sovellukselle tä-

män tutkielman puitteissa ollut mahdollista tehdä, eikä Solrin soveltuvuudesta todelliseen käyttöympäristöön vielä tämän tutkimuksen perusteella voida tehdä pitkälle meneviä johtopäätöksiä. Tutkielmassa kehitetty viitekehyskään ei tukisi tämän kaltaista tarkastelua. Mielenkiintoista olisikin nähdä Solr käytössä ”omassa elementissään” jonkin organisaation aktiivisessa käytössä. Mikäli Solr tällöin osoittaa täyttävänsä kaikki organisaation sille asettamat vaatimukset, voitaneen sen nähdä kokonaisvaltaisesti lunastaneen lupauksensa. Akateemisessa mielessä Solr on joka tapauksessa osoittautunut hyödylliseksi ja kiinnostavaksi tutkimuskohteeksi. Avoimen luonteensa takia siihen perehtyminen on kaiken lisäksi ollut mutkatonta.

8 Lähteet

[1] S. Abiteboul. Querying semi-structured data. *Database Theory-ICDT'97*, ss. 1–18, 1997.

[2] M. Abrol, N. Latarche, U. Mahadevan, J. Mao, R. Mukherjee, P. Raghavan, M. Tourn, J. Wang ja G. Zhang. Navigating large-scale semi-structured data in business portals. Kirjassa *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, ss. 663–666, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[3] W. Andrews. Magic quadrant for information access technology. *Gartner*, 2009. `http://www.gartner.com/technology/media-products/reprints/microsoft/vol7/article2/article2.html`.

[4] M. Asay. Open-source lucene threatens microsoft, google enterprise search. *CNET News*, 2009. `http://news.cnet.com/8301-13505_3-10288143-16.html`.

[5] R. Baeza-Yates ja B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[6] P. Bailey, N. Craswell, I. Soboroff ja A. P. de Vries. The CSIRO enterprise search test collection. Kirjassa *ACM SIGIR Forum*, osa 41, ss. 42–45. ACM, 2007.

[7] K. Balog. People search in the enterprise. Kirjassa *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, s. 916. ACM, 2007.

[8] M. Bennett. Enterprise search engine vendors in 2007. *New Idea Engineering*, 2007. `http://www.ideaeng.com/tabId/98/itemId/128/Enterprise-Search-Engine-Vendors-in-2007.aspx`.

[9] M. Bennett. 2009 overview of the enterprise search market. *New Idea Engineering*, 2009. `http://www.ideaeng.com/tabId/98/itemId/181/Overview-of-the-Enterprise-Search-Market-2009.aspx`.

[10] D. C. Blair. The challenge of commercial document retrieval, part I: major issues, and a framework based on search exhaustivity, determinacy of representation and

document collection size. *Information Processing & Management*, 38(2):273–291, 2002.

[11] D. C. Blair. The challenge of commercial document retrieval, part II: a strategy for document searching based on identifiable document partitions. *Information Processing & Management*, 38(2):293–304, 2002.

[12] S. Brin ja L. Page. The anatomy of a large-scale hypertextual web search engine. Kirjassa *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, ss. 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[13] A. Broder. A taxonomy of web search. Kirjassa *ACM Sigir Forum*, osa 36, ss. 3–10. ACM, 2002.

[14] P. Buneman. Semistructured data. Kirjassa *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, s. 121. ACM, 1997.

[15] V. Bush. As we may think. *Reading digital culture*, s. 9, 2001.

[16] S. Chernov, B. Fehling, C. Kohlschütter, W. Nejdl, D. Pieper ja F. Summann. Enabling Federated Search with Heterogeneous Search Engines–Combining FAST Data Search and Lucene. *Federated Search Project Report*, 2006.

[17] D. Cutting et al. Apache lucene -dokumentaatio. `http://lucene.apache.org`.

[18] D. R. Cutting, D. R. Karger, J. O. Pedersen ja J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. Kirjassa *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, ss. 318–329. ACM, 1992.

[19] C. Dias. Corporate portals: a literature review of a new concept in Information Management. *International Journal of Information Management*, 21(4):269–287, 2001.

[20] R. Elmasri ja S. Navathe. *Fundamentals of Database Systems (2nd ed.)*. The Benjamin/Cummings Publishing Company, Inc., Redwood, 1994.

[21] R. Fagin, R. Kumar, K. S. McCurley, J. Novak, D. Sivakumar, J. A. Tomlin ja D. P. Williamson. Searching the Workplace Web. Kirjassa *Proceedings of the 12th international conference on World Wide Web*, s. 375. ACM, 2003.

[22] D. J. Foskett. Thesaurus. Kirjassa *Readings in Information Retrieval*, ss. 111–134. Morgan Kaufmann Publishers Inc., 1997.

[23] W. B. Frakes ja R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., New Jersey, 1992.

[24] A. J. Gilliland-Swetland. Setting the stage. *Introduction to Metadata: Pathways to Digital Information*, 1998.

[25] A. Göker ja J. Davies, toim. *Information Retrieval - Searching in the 21st Century*. John Wiley & Sons, Ltd., United Kingdom, 2009.

[26] M. D. Gordon. It's 10 am Do you know where your documents are? The nature and scope of information retrieval problems in business. *Information Processing & Management*, 33(1):107–122, 1997.

[27] A. Halevy, P. Norvig ja F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[28] E. Hatcher, O. Gospodnetic ja M. McCandless. *Lucene in Action, 2nd Edition*. Manning Publications Co., 2009.

[29] D. Hawking. Challenges in enterprise search. Kirjassa *Proceedings of the 15th Australasian database conference-Volume 27*, ss. 15–24. Australian Computer Society, Inc., 2004.

[30] D. Hawking. How things work: Web search engines (part 1). *IEEE Computer*, ss. 86–88, June 2006. `http://es.csiro.au/pubs/hawking_howthingswork1.pdf`.

[31] D. Hawking, F. Crimmins, N. Craswell ja T. Upstill. How valuable is external link evidence when searching enterprise Webs? Kirjassa *Proceedings of the 15th Australasian database conference-Volume 27*, ss. 77–84. Australian Computer Society, Inc., 2004.

[32] D. Hawking, C. Paris, R. Wilkinson ja M. Wu. Context in enterprise search and delivery. Kirjassa *Proc. IRiX Workshop, ACM SIGIR*. Citeseer, 2005.

[33] K. Järvelin. *Tekstitiedon haku tietokannoista*. Gummeruksen kirjapaino Oy, Jyväskylä, 1995.

[34] R. Lamb ja E. Davidson. Understanding intranets in the context of end-user computing. *SIGMIS Database*, 36(1):64–85, 2005.

[35] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[36] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft ja J. Allan. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology*, 53(6):415–437, 2002.

[37] R. Mukherjee ja J. Mao. Enterprise search: Tough stuff. *Queue*, 2(2):36–46, 2004.

[38] T. Nelson. The Hypertext. Kirjassa *Proceedings of the World Documentation Federation Conference*, 1965.

[39] A. Nürnberger, R. Seising ja C. Wenzel. On the fuzzy interrelationships of data, information, knowledge and wisdom. Kirjassa *Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American DKE Group, Otto-von-Guericke-Univ. Magdeburg, Magdeburg, Germany*, osa 11, ss. 1–6, 2009.

[40] L. Page, S. Brin, M. Rotwani ja T. Winograd. The pagerank citation ranking: Bringing order to the web. Tekninen raportti, Stanford Digital Library Technologies Project, 1998.

[41] D. Pogue. Google Takes on Your Desktop. *New York Times*, 2004.

[42] A. Salminen. *Methodology for document analysis*, sarjan *Enclyclopedia of Library and Information Science* osa 67 (supplement 30), ss. 299–320. Marcel Dekker, Inc., New York, 2000.

[43] G. Salton, A. Wong ja C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):620, 1975.

[44] G. Salton ja C. S. Yang. On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372, 1973.

[45] J. D. Schlesinger, J. M. Conroy, M. E. Okurowski ja D. P. O'Leary. Machine and human performance for single and multidocument summarization. *IEEE Intelligent Systems*, 18(1):46–54, 2003.

[46] K. U. Schmidt, D. Oberle ja K. Deissner. Taking Enterprise Search to the Next Level. *Relation*, 10(1.65):6134, 2009.

[47] D. Smiley ja E. Pugh. *Solr 1.4 Enterprise Search Server*. Packt Publishing, 2009.

[48] I. Soboroff, K. Balog, P. Thomas, N. Craswell, P. Bailey ja A. P. Vries. Overview of the TREC 2008 enterprise track. Kirjassa *Proceedings of the 17th Text Retrieval Conference. Retrieved January*, osa 11, s. 2010, 2009.

[49] Apache Solr. Apache solr -dokumentaatio. `http://lucene.apache.org`.

[50] G. Solskinnsbakk ja J. Gulla. Ontological profiles in enterprise search. *Knowledge Engineering: Practice and Patterns*, ss. 302–317, 2008.

[51] O. Suominen. Käyttäjäkeskeinen moninäkymähaku semanttisessa portaalissa. Pro gradu, Helsingin yliopisto, Tietojenkäsittelytieteiden laitos, 2008. `http://www.seco.tkk.fi/publications/2008/suominen-gradu-2008.pdf`.

[52] I. Watson. Internet, intranet, extranet: managing the information bazaar. Kirjassa *Aslib Proceedings*, osa 51, ss. 109–114. ASLIB, 1999.

[53] W. G. Yee, M. Beigbeder ja W. Buntine. SIGIR06 workshop report: Open Source Information Retrieval systems (OSIR06). Kirjassa *ACM SIGIR Forum*, osa 40, s. 65. ACM, 2006.