Indeksointi ja haku rakenteisista dokumenteista

Miika Nurminen

17. marraskuuta 2004

minurmin@cc.jyu.fi

Tiivistelmä

Harjoitustyössä määritellään puolirakenteinen tieto ja käsitellään rakenteisia dokumentteja esimerkkinä puolirakenteisesta tiedosta. Lisäksi käydään läpi indeksirakenteita ja hakumalleja XML-dokumenteille. Käsittelyssä otetaan huomioon tekstisisältö, linkit ja dokumentin rakenne.

1 Johdanto

Harjoitustyössä keskitytään rakenteisten dokumenttien indeksointiin ja tiedonhakuun. Rakenteiset dokumentit ovat esimerkki puolirakenteisesta datasta, joka on joustava tietomalli tietokannoissa olevan rakenteisen tiedon ja rakenteettoman datan välimaastossa. Merkittävin tapa rakenteisten dokumenttien esittämiseen on XML-kieli. Tekstisisällön lisäksi dokumentteihin voidaan merkitä hierarkkisia elementtejä, attribuuttitietoa tai linkkejä. Lisäksi ulkoista metatietoa voidaan käyttää. XML mahdollistaa monipuoliset indeksointirakenteet ja hakumallit, jotka vaihtelevat yksinkertaisista taulukoista ja polkulistoista puurakenteisiin.

Työ jakautuu seuraaviin osiin: luku 2 esittelee tekstitiedonhaun, jonka käsitteitä muut luvut hyödyntävät. Luvussa 3 käydään läpi rakenteisten dokumenttien ominaisuuksia yleisesti. Luvussa 4 perehdytään rakenteisen tekstin analysointitekniikoihin tekstitiedonhaun näkökulmasta. Luku 5 on yhteenveto.

2 Tiedonhaun peruskäsitteet

Tiedonhaku (*Information Retrieval*, *IR*) on tietojenkäsittelytieteen osa-alue, joka tutkii relevanttien dokumenttien hakua dokumenttikokoelmasta. Dokumenttien haun tarkoitus on tyydyttää käyttäjän tietotarpeet [3, sivu 444]. Nämä ilmaistaan ideaalisesti luonnollisella kielellä. Hakusanat voivat kuulua kontrolloituun sanastoon. Korostettaessa haettavien dokumenttien tyyppiä voidaan puhua myös tekstitiedonhausta, multi- tai hypermediatiedonhausta tai rakenteisesta tiedonhausta.

Jatkossa käytetään kuitenkin yleensä lyhyttä nimitystä. Tiedonhakuprosessi on iteratiivinen ja interaktiivinen. Käyttäjä arvioi hakutuloksena saatua dokumenttikokoelmaa ja tarvittaessa tarkentaa kyselyään, kunnes hakutulos täyttää tietotarpeet (tai, jos soveltuvia dokumentteja ei löydy, käyttäjä luovuttaa) [13]. Tiedonhaku voidaan tulkita myös loogisena päättelynä, jossa tarkoituksena on löytää ne dokumentit, joista käyttäjän kysely seuraa [30].

Tiedonhaun määritelmässä oleva tieto viittaa *informaatioon*, ei *dataan*. Data on käsitteellisesti informaatiota matalammalla tasolla viitaten esim. säännöllisillä lausekkeilla tai relaatioalgebralla haettavaan tietoon. Käyttäjän täytyy tietää täsmällisesti haettavan tiedon muoto (esim. tietokannan skeema) ja kyselykielen syntaksi (esim. SQL). Haettavan datan relevanssia ei arvioida, vaan hakutulokset ovat syntaktisella tasolla käyttäjän kyselyyn sopivat tietueet. Vastakohtana ”datahaulle” tiedonhaun tarkoituksena on palauttaa käyttäjälle tietoa käyttäjän haluamista *aiheista*. Tiedonhakujärjestelmän on tulkittava käyttäjän tekemä kysely, verrattava sitä dokumenteista erotettuun tietoon (indeksitermit) ja esitettävä hakutulokset käyttäjälle järjestettynä arvioidun relevanssin mukaan. Tiedonhakujärjestelmän päätavoite on palauttaa mahdollisimman paljon käyttäjän kyselyyn sopivia relevantteja dokumentteja jättäen pois epärelevantit [3, sivut 1-3]. Käytännössä kysely koostuu hakusanoista, jotka voivat olla osa jotain kontrolloitua sanastoa.

Relevanssin käsitteen tarkka määrittely on hankalaa, koska se riippuu käyttäjän yksilöllisistä tietotarpeista ja tiedonhakujärjestelmän kyselykieli asettaa rajoituksia näiden tarpeiden muotoilulle. Käytännössä tiedonhakujärjestelmiä arvioitaessa on tapauskohtaisesti arvioitava, mitkä dokumentit ovat tietyn haun suhteen relevantteja ja mitkä eivät. Tämän jälkeen tutkitaan, kuinka hyvin järjestelmä pystyy löytämään ko. dokumentit. Formaalisti tiedonhakujärjestelmän suorituskykyä arvoidaan tarkkuuden (*precision*) ja saannin (*recall*) avulla. Tarkkuus on palautettujen relevanttien dokumenttien osuus kaikista palautetuista, saanti on palautettujen relevanttien dokumenttien osuus kaikista relevanteista dokumenteista. Formaalisti tämä voidaan määritellä seuraavasti: olk. A relevanttien dokumenttien joukko ja B palautettujen dokumenttien joukko. Tällöin tarkkuudelle saadaan määritelmä $P = \frac{|A \cap B|}{|B|}$, ja saannille $R = \frac{|A \cap B|}{|A|}$ [60, sivut 113-115]. Tiedonhakujärjestelmän suorituskyky on tyypillisesti kompromissi haun ja saannin suhteen. Jos järjestelmällä on korkea tarkkuus, saanti on matalampi ja päinvastoin.

Tiedonhakujärjestelmän toiminnallisuuden perustana on tiedonhakumalli (*information retrieval model*), joka määrittää perusoletukset ja ennusteet dokumenttien relevanssille. Tärkeimpiä tiedonhakumalleja (esim. vektorimalli, probabilistiset mallit) käsitellään tekstianalyysia käsittelevässä luvussa 4 yhdessä tekstin samanlaisuusmittojen kanssa. Formaalisti määriteltynä tiedonhakumalli on nelikko $(\mathbf{D}, \mathbf{Q}, \mathcal{F}, R(q_i, d_j))$, missä [3, sivut 19-34]

- $\mathbf{D}$ on loogisten näkymien joukko dokumenttikokoelmaan. Loogisia näkymiä voivat olla dokumentin sisältötiedot, looginen rakenne, ulkoasutiedot tai metatiedot. [30]
- $\mathbf{Q}$ on loogisten näkymien joukko käyttäjän tietotarpeille. Käytännössä nämä esitetään hakukyselyinä.
- $\mathcal{F}$ on kehysrakenne dokumenttiesitysten, kyselyjen ja niiden suhteiden mallinnukseen. Kehysrakenne vaihtelee suuresti tiedonhakumallista riippuen.
- $R(q_i, d_j)$ on täsmäytysfunktio (*matching function*) (formaalisti $R : \mathbf{D} \times \mathbf{Q} \rightarrow \mathbf{R}$), joka liittää jokaiseen kysely- ja dokumenttiesitysparin reaaliluvun, jota sanotaan relevanssiksi. Täsmäytys määrittää dokumenteille järjestyksen yksittäisten kyselyjen suhteen.

3 Rakenteiset dokumentit

Luvussa määritellään puolirakenteinen tieto ja arvioidaan rakenteisuuden merkitystä tiedonhaun kannalta. Lisäksi käsitellään rakenteisia dokumentteja ja erityisesti XML-kieltä esimerkkinä puolirakenteisesta tiedosta.

3.1 Rakenteisuuden merkitys

Tiedonhaun kannalta rakenteisen tiedon erityisongelma on oikean tiedon esitystavan valinta [37, sivu 470]. Rakenteisten dokumenttien osalta esitystavat vaihtelevat esimerkiksi sanalistasta monimutkaisiin puuindekseihin. Esitystapa vaikuttaa suoraan tiedonhaussa sovellettavaan täsmäytykseen.

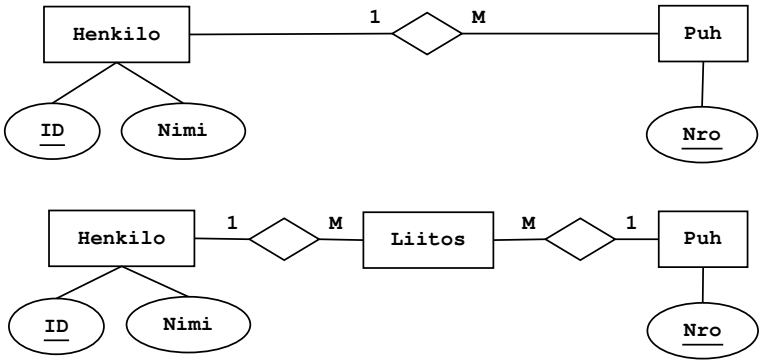
Luvussa pohditaan rakenteisuuden asteita ja ilmenemistä eri tietojoukoissa. Rakenteisuuden seurauksia käsitellään lähinnä tiedonhaun kannalta.

3.1.1 Puolirakenteinen tieto ja rakenteiset dokumentit

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

Teoreettinen malli rakenteisen tiedon mallintamiseen on E. Coddin 1970 kehittämä relaatiomalli, jossa tietokohteet esitetään määrämuotoisina tauluina eli relaatioina. Kohteen ominaisuudet ovat kenttiä ja ilmentymät tietueita. Relaatiomallin erityisvaatimuksena on, että kenttien arvojen pitäisi olla jakamattomia [25, sivut 195-199]. Esimerkiksi tilanteessa, jossa henkilöllä voi olla monta puhelinnumeroa, puhelinnumeroita ei voi määrittää henkilön kentäksi, vaan tarvitaan erillinen *puhelinnumerot*-aputaulu, jossa jokainen puhelinnumero on erillinen tietue ja viittaa edelleen johonkin henkilöön. Jos monella henkilöllä voi olla vielä sama puhelinnumero, tilanne monimutkaistuu entisestään. Tämän *monesta-moneen* -suhteiden hallinnan hankaluuden takia relaatiomallia pidetään joustamattomana tietomallina. Kuvassa 1 on ER-kaavio puhelinnumerotietokannasta kummassakin tilanteessa.

Rakenteisen ja rakenteettoman tiedon rinnalle on viimeisen vuosikymmenen aikana noussut puolirakenteinen tieto (*semistructured data*). Buneman [10] määrittelee puolirakenteisen datan olevan tietoa, joka hyödyntää jotain yleistä rakennemallia (tyypillisesti puu tai graafi), mutta tiedolla ei ole skeemaa — ainakaan tietokantojen vaatimalla tarkkuudella. Lisäksi osa tiedoista (esim. ”kenttien” nimet), jotka yleensä on kuvattu skeemassa, kuvataan puolirakenteisessa datassa itsessään. Tästä johtuen puolirakenteista dataa kutsutaan joskus itsekuvaavaksi, vaikka se ei sitä



Kuva 1: Puhelinnumeroesimerkki.

semanttisessa mielessä ole. Buneman samaistaa puolirakenteisen ja rakenteettoman datan, mutta tämän kirjoittajan mielestä niillä on selvä ero: puolirakenteisella datalla on yleinen rakennemalli, joka puuttuu rakenteettomalta datalta. Wang [61] lisää myös heterogeenisista lähteistä yhdistetyn tiedon puolirakenteisen datan joukkoon.

Tärkein esimerkki puolirakenteisesta tiedosta ovat rakenteiset dokumentit. Nimitys on yleistynyt erityisesti SGML- ja XML-dokumenttien myötä, mutta yhtä lailla sähköpostit, uutisryhmäarkistot, ohjelmakoodit ja LaTeX-dokumentit [61] ovat rakenteisia. Myös WWW:tä voidaan kokonaisuutena pitää valtavana puolirakenteisena tietokantana. Bunemanin mielestä kaikki rakenteiset dokumentit ovat puolirakenteista tietoa. Perusteluna tähän on, että rakenteisiin dokumentteihin käytetyt skeemat (SGML-dokumenteissa DTD, XML-dokumenteissa myös XML Schema) ovat olennaisesti relaatiomallia joustavampia. Relaatiomallilla sisäkkäisten tietoalkioiden ja erityisesti *monesta moneen* -suhteiden esittämien on työlästä, puolirakenteisella datalla — skeemalla tai ilman — tämä on helpompaa. Puhelinnumeroesimerkki voitaisiin esittää yhtenä XML-dokumenttina helposti:

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

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

```
<henkilot>                      <puhelinnumerot>
  <henkilo id="h1">               <nro type="tyo" id="p1">435-345345</nro>
```

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

Esityskyvyltään rakenteisissa dokumenteissa ei sinänsä ole mitään uutta tietokantoihin verrattuna. Samaa tyyppiä olevat dokumentit voitaisiin tulkita relaatiotietona siten, että sisäkkäiset elementit ja dokumentit ovat tauluja, muut elementit ja attribuutit ovat kenttiä, linkit suhteita ja tietueet muodostuisivat dokumenttien sisällöstä. Tämä ei kuitenkaan ole luonnollinen tapa rakenteisten dokumenttien kuvaamiseen, koska dokumentin teksti hajaantuu moneen eri tauluun ja esim. järjestyksen esittäminen vaatii ylimääräisiä apukenttiä. Tekstianalyysi monimutkaistuisi oleellisesti, joten relaatiotiedonlouhintaan perustuvia analyysimenetelmiä ei käsitellä tässä tarkemmin. Dokumenttikokoelmasta koostetun indeksin esittäminen tietokannassa on eri asia itse dokumenttien säilyttämiseen nähden ja monet tiedonhakujärjestelmät käyttävätkin tietokantaa indeksin tallentamiseen. Harjoitustyössä esitellyt indeksointimenetelmät eivät pääsääntöisesti ota kantaa siihen, miten indeksi on tallennettu.

Koska XML-dokumenttien rakennemalli on puu, sanoja *rakenne* ja *hierarkia* käytetään usein synonyymeina. Hyperteksti-termin keksijä Ted Nelson on kritisoinut voimakkaasti tätä käsitystä vedoten siihen, ettei hierarkialla voida esittää luontevasti mielivaltaisia informaatiorakenteita kuten rinnakkaisuutta, kaksisuuntaisia linkkejä tai tietyn olion sijaintia eri paikoissa samanaikaisesti [47]. Hierarkkinen dokumenttimalli on relaatiotietokantaa joustavampi, mutta ei tarpeeksi yleinen. Harjoitustyössä rakenteisuuden ymmärretään olevan hierarkiaa laajempi käsite, mutta aiheen käsittely rajataan yksinkertaisuuden vuoksi XML-dokumentteihin, jotka voivat sisältää yksisuuntaisia linkkejä ja joilla voi olla ulkopuolista metadataa.

3.1.2 Rakenteiset dokumentit tiedonhaussa

Tiedonhaussa dokumentteja on pidetty tyypillisesti jakamattomina ja rakenteettomina, mistä johtuen perinteisiä tiedonhakumalleja täytyy laajentaa rakenteisia dokumentteja silmälläpitäen. Hypermediamallit ovat käyttökelpoisia rakenteisten dokumenttien käsittelyyn. Tällöin rakenteisen dokumentin osaa pidetään hypertekstijärjestelmän solmuna, joka on linkittynyt muihin saman dokumentin osiin ja mahdollisesti muiden dokumenttien solmuihin. Hypertekstijärjestelmät ovat aiemmin keskittyneet lähinnä dokumenttikokoelman selaukseen linkkien avulla, mutta rakenteesta saadaan myös arvokasta lisätietoa, jota voidaan hyödyntää tiedonhaussa [32]. Yhtä lailla rakenteesta saadaan lisätietoa dokumenttien klusterointiin ja tietämyksen muodostamiseen yleisesti. Chiaramella [13] on analysoinut rakenteisuuden vaikutusta perinteisissä tiedonhakujärjestelmissä, käytännön esimerkkinä voidaan pitää WWW:tä ja pelkkiin avainsanoihin perustuvaa hakuko-

netta.

- **Kyselyt**. Hakukoneet eivät osaa erotella web-sivuilla olevaa oleellista tietoa ”kohinasta” (mainokset ym.). Lisäksi web-sivu saattaa jakautua moneen loogiseen osioon, joiden aiheet vaihtelevat. Laajemmassa mittakaavassa hakukoneet eivät osaa erottaa, mitkä sivut kuuluvat asiayhteydeltään samaan sivustoon — kaikki linkit ovat samanarvoisia.
- **Selaus**. Jos hypertekstijärjestelmässä on olemassa loogista rakennetta, tämän rakenteen pitäisi tulla käyttäjälle selväksi. WWW-linkit eivät kuvaa tätä rakennetta riittävällä tarkkuudella. Linkeistä ei yleisesti voi päätellä, millaista tietoa sen ”takana” on, viekö linkki toiseen paikaan nykyisellä sivustolla vai siirrytäänkö kokonaan toiseen asiakokonaisuuteen. Tämä ja linkkien yksisuuntaisuus johtavat siihen, että kokemattomat käyttäjät voivat tuntea ”eksyvänsä hyperavaruuteen”.
- **Hakutulosten järjestys**. Puhtaaseen tekstiin keskittyvät tiedonhaun tekniikat ovat tehottomia rakenteisia dokumentteja haettaessa. Web-hakukoneiden tuhansia dokumentteja sisältävät hakutuloslistat ovat tästä tunnettu esimerkki. Laajassa dokumentissa oleva relevantin osion painoarvo on vähäinen, jos dokumentti indeksoidaan jakamattomana ja rakenneosista riippumatta (esim. otsikot) sanoille annetaan sama paino. Hakutulosten suuren määrän lisäksi myös niiden järjestys saattaa olla siis vääristynyt.
- **Hakutulosten esittäminen**. Hakukoneiden tulisi viitata tulosdokumenttien relevantiksi arvioituun osaan eikä koko dokumenttiin. Tämä vähentäisi käyttäjän työmäärää hakutulosten arvioinnissa.

Rakenteisten dokumenttien ja puhtaaseen tekstiin erikoistuneiden hakukoneiden epäyhtenäisyys yhdistettynä WWW:n valtavaan kokoon ja siihen, että käyttäjien kyselyt koostuvat tyypillisesti vain muutamasta sanasta takaavat sen, että hakutuloslistat pysyvät jatkossakin epäkäytännöllisinä. Myönnettäköön, että uudemmat hakukoneet ovat korjanneet tilannetta jonkin verran. Esim. linkkitiedon (PageRank-algoritmi, katso luku 4.3.2) käyttö Googlessa tai hakutulosten esittäminen klustereina Vivísimo-yhtiön Clusty[1]-metahakukoneessa mahdollistavat täysin käyttökelpoisen haun, vaikka niiden kyselykielet eivät dokumenttien rakennetta tuekaan. Tästä huolimatta käyttäjän täytyy tietää melko tarkasti aiheeseen liittyvät oikeat avainsanat.

Rakenteisia dokumentteja varten on kehitetty runsaasti erilaisia kyselykieliä ja indeksirakenteita. Tyypilliset lähestymistavat ovat SQL:n tyyppisen kielen laajentaminen tukemaan puolirakenteista tietomallia ja kokotekstihakua, tai määrittelemällä kieli suoraan puolirakenteisen tietomallin pohjalta [10]. Kyselyt voivat sisältää rajoitteita dokumentin sisällön, rakenteen tai linkkitiedon suhteen [32]. Jos käytettävissä on metatietoa, sitä pitäisi pystyä myös hyödyntämään. Haun tuloksena saatavat tietoyksiköt (*information units*, *dokumenttikomponentit*) voivat olla kokonaisia dokumentteja, dokumenttifragmentteja tai eri dokumenteista saatu dokumenttifragmenttien yhdistelmä [44] (viimeistä vaihtoehtoa voidaan verrata tilanteeseen, jossa haetaan kenttiä usean taulun liitoksesta relaatiotietokannassa). Dokumenttifragmenttien haku on mielekästä, jos käyttäjä tietää etukäteen, millaista tietoa halutaan hakea (esim. ”Hae kaikkien dokumenttien otsikot ja ensimmäiset tekstikappaleet”). Laajat dokumentit voivat sisältää tietoa useista aiheista (esim. monesta osasta koostuva kirja voi olla yksi looginen dokumentti), jolloin hyvän hakualgoritmin pitäisi pystyä myös päättelemään dokumentin relevantit osat ja näyttämään ne käyttäjälle Chiaramellan huomioiden mukaisesti. Erityisen hankalaksi tilanne menee, jos haettava dokumenttikokoelma on

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

heterogeeninen sisältäen laajuudeltaan ja tyypiltään erilaisia dokumentteja — kuten WWW.

3.2 XML:n erityispiirteet

XML (Extensible Markup Language) on W3C:n kehittämä metakieli rakenteisten dokumenttien esittämiseen. XML on kehitetty Charles Goldfarbin kehittämän ja ISO:n standardoiman[2] SGML-kielen (Standard Generalized Markup Language) pohjalta siten, että XML on oleellisesti SGML:n yksinkertaistettu osajoukko. XML:n alkuperäiset suunnittelutavoitteet olivat seuraavat: [8]

1. XML on suoraviivaisesti käytettävissä Internetissä.
2. XML tukee laajaa sovellusjoukkoa.
3. XML on yhteensopiva SGML:n kanssa.
4. On helppoa kirjoittaa XML:ää käsitteleviä ohjelmia.
5. XML:n valinnaisten ominaisuuksien määrä pidetään mahdollisimman vähäisenä.
6. XML-dokumenttien tulisi olla ihmisen luettavissa ja suhteellisen selkeitä.
7. XML:n määritys valmistellaan nopeasti.
8. XML:n määritys on formaali ja tiivis.
9. XML-dokumenttien luonti on helppoa.
10. Tiiviys ei ole merkittävää XML-merkkauksessa.

XML-kielen voidaan sanoa ylittäneen tavoitteensa. Kieli on epäilemättä kaikista rakenteisista dokumenttiformaateista laajimmalle levinnyt. Syy tähän lienee se, ettei XML-määritys ole sitoutunut mihinkään tiettyyn sovellusalueeseen, vaan tarjoaa puolirakenteisen puumaisen rakennemallin, jota joustavuudessaan voidaan soveltaa lähes minkä tahansa tiedon kuvaamiseen. Lisäksi W3C on määritellyt joukon liitännäiskieliä, jotka laajentavat XML-kielen sovellettavuutta entisestään (esim. XML-linkit, XSL-muunnokset). Uusien liitännäiskielten myötä XML-kieliperheestä on tosin tullut monimutkaisuudessa täysin SGML-standardiin verrattava kokonaisuus. Erilaisten XML-pohjaisten kielten välillä voidaan tehdä karkea jako data- tai dokumenttikeskeisiin kieliin [7, sivut 3-13].

Datakeskeiset kielet on tarkoitettu sovellusten väliseen kommunikointiin, tietorakenteiden ja esim. relaatiotietokannoissa olevan tiedon esittämiseen. Esimerkkejä datakeskeisistä XML-kielistä ovat W3C:n kehittämä, web-sovelluspalvelujen perustana oleva SOAP[3] (*Simple Object Access Protocol*), graafien kuvauskieli GXL[4] (*Graph eXchange Language*) ja yritysten sähköiseen tiedonvaihtoon kehitetty ebXML[5]. Datakeskeisten formaattien tarkempi käsittely sivuutetaan, koska ne ovat muodoltaan lähempänä tietokantoja tai viestiprotokollakehyksiä kuin tekstidokumentteja. Siten tekstitiedonhaun soveltaminen niihin ei ole mielekästä.

Dokumenttikeskeiset kielet on tarkoitettu perinteiselle SGML:n sovellusalueelle, elektroniseen julkaisemiseen ja yleensäkin rakenteisten *dokumenttien* käsittelyyn. SGML:n ja sittemmin XML:n suuri lupaus on ollut monikanavajulkaisun mahdollistaminen: sisältö kirjoitetaan kertaalleen rakenteiseen dokumenttiin, josta tyylitiedostojen avulla muunnetaan automaattisesti esitykset eri formaatteihin, kuten paperijulkaisuun (PDF), WWW-ympäristöön tai esim. CD-ROM:ille

[2] ISO 8879
[3] http://www.w3.org/TR/soap12-part0/
[4] http://www.gupro.de/GXL/
[5] http://www.ebxml.org/

(käytännössä prosessi tuskin on näin suoraviivainen). Liitännäiskielten avulla dokumenttien sisälle tai niiden välille voidaan määritellä linkkejä ja dokumentteihin voidaan liittää metatietorakenteita. Dokumenttikeskeisiä kieliä ovat HTML-kielen XML-versio XHTML[6], romaanitekstien merkkaamiseen suunniteltu TEI[7] (*Text Encoding Initiative*) ja teknisessä dokumentoinnissa käytetty DocBook[8].

3.2.1 Dokumentin rakenne ja sisältö

XML-dokumentit koostuvat sisäkkäisistä ja peräkkäisistä elementeistä muodostaen puolirakenteiselle tiedolle tyypillisen puumaisen rakenteen, jossa jokainen elementti on yksi solmu. Solmu voi sisältää alisolmujen lisäksi tekstiä tai attribuutteja. Elementit muodostavat dokumentin loogisen rakenteen, attribuuteilla voidaan merkitä yksittäisiin elementteihin liittyvää metatietoa. Lisäksi dokumentilla on entiteeteistä (*entity*) muodostuva fyysinen rakenne. Entiteetit voivat olla mitä tahansa tietoyksiköitä, kuten yksittäisiä merkkejä, dokumenttifragmentteja, viittauksia binääridataan tai toisiin dokumentteihin. Esimerkiksi elementtitageja varten varattuihin < - ja > -merkkeihin voidaan viitata `<` ja `>` -entiteeteillä. Entiteetit mahdollistavat alkeellisen dokumentin osien uudelleenkäytön. [7, sivut 3-13]

XML-dokumentteja voidaan käyttää ilman erillistä skeemaa viittaamalla suoraan dokumentissa esiintyviin elementteihin. Skeemattoman XML-dokumentin täytyy olla syntaktisesti oikea (esim. vain yksi juurielementti, jokaisen elementin alkutagia vastaa lopputagi, attribuutit on erotettu lainausmerkeillä, tagit eivät mene ”ristiin”) ja samojen sääntöjen täytyy olla voimassa dokumentissa viitatuilla entiteeteillä. Tällöin dokumentin sanotaan olevan hyvinmuodostettu (*well-formed*) [8]. Hyvinmuodostetut dokumentit täyttävät täysin Bunemanin puolirakenteisen datan määritelmän. Täsmällisen käsittelyn kannalta on kuitenkin toivottavampaa, että dokumenteilla on jokin skeema ja jatkossa näin oletetaan, ellei toisin ole mainittu. XML-kieli tarjoaa tähän kaksi vaihtoehtoista mekanismia: DTD ja XML Schema.

DTD (*Document Type Definition*) on mekanismi formaaliin rakennesääntöjen määrittelyyn. DTD määrittelee käytössä olevat elementit ja rajoitteet niiden käyttöön suhteessa toisiinsa. Lisäksi DTD:ssä määritellään elementteihin liittyvät attribuutit. Attribuuteille voidaan antaa yksinkertaisia rajoitteita: keskeisimpiä ovat omat luetellut tyypit, vapaa teksti tai viite toisaalla dokumentissa olevaan tunnisteeseen [7, sivut 47-69]. DTD-määritykset ovat aiemmin olleet käytössä SGML-kielessä, mutta kieltä on jonkin verran yksinkertaistettu XML:aa varten (esim. elementin lopputagi on aina pakollinen). Dokumenttia, joka on hyvinmuodostettu ja joka lisäksi noudattaa rakennesääntöjään, kutsutaan validiksi [8]. Luvun 3.1 puhelinnumeroesimerkin dokumenttityyppi voitaisiin määritellä DTD:llä seuraavasti (oletetaan, että tiedot ovat yhdessä tiedostossa):

```
<!-- elementit                                                    -->
<!ELEMENT puhelinluettelo (henkilot,puhelinnumerot)>
<!ELEMENT henkilot        (henkilo+)> <!-- +: 1 tai useampi elementti -->
<!ELEMENT puhelinnumerot  (nro+)>
<!ELEMENT henkilo         (nimi,puh+)>
```

[6] http://www.w3.org/MarkUp/
[7] http://www.tei-c.org/
[8] http://docbook.org/

```
<!ELEMENT puh              EMPTY>     <!-- ei alielementtejä          -->
<!ELEMENT nimi            (#PCDATA)>  <!-- #PCDATA: jäsennetty teksti -->
<!ELEMENT nro             (#PCDATA)>

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

Uudempi tapa rakennesääntöjen määrittelyyn on XML Schema -kieli, jossa säännöt voidaan määritellä XML-kielisellä syntaksilla. XML Schema soveltuu erityisesti datakeskeisiin dokumentteihin, koska siinä voidaan määritellä omia datatyyppejä ja dokumentin sisällölle voidaan määrätä rajoitteita huomattavasti DTD:tä tarkemmalla tasolla [27]. Esimerkiksi puhelinnumeroesimerkin `<puh>` -elementin `xref` -viitekenttä voi DTD-määrityksen mukaan viitata joko henkilöön tai puhelinnumeroon, vaikka viittaukset tulisi sallia vain puhelinnumeroon. Lisäksi DTD:llä ei voi määritellä puhelinnumerolle tarkkoja syntaksirajoitteita. XML Schemalla nämä rajoitteet ovat mahdollisia, joten suositus soveltuu datakeskeisille dokumenteille DTD:tä paremmin. Tekstidokumenttien kuvaamiseen DTD tarjoaa kuitenkin riittävän tarkkuuden. Toistaiseksi DTD-määritykset ovat myös huomattavasti XML Schema -määrityksiä yleisempia. Dokumentin rakennetta analysoimalla voidaan verrata eri dokumenttien rakenteellista samanlaisuutta. Rakennepuun eri osiin voidaan viitata esim. XPath-kielellä [14], jossa dokumentin elementtejä vastaavat tiedostojärjestelmän polkuja muistuttavat polkulausekkeet.

3.2.2 Hyperlinkit

Ehkä tärkein syy WWW:n suosiolle on mahdollisuus linkittää dokumentteja toisiinsa. Linkit ovat yleistetty ja yksinkertaistettu vastine tieteellisten artikkelien kirjallisuusviitteille. HTML:ssä ja eri XML-pohjaisissa kielissä on erilaisia tapoja viittauksen ilmaisuun. Dokumentin sisällä voidaan käyttää ID- ja IDREF-tyyppisiä attribuutteja, dokumenttien välillä tarvitaan yhteisesti sovittu nimeämiskäytäntö, joka WWW-ympäristössä on URI. Keskitetyissä hypertekstijärjestelmissä linkit ovat yleensä kaksisuuntaisia, mikä takaa niiden yhtenäisyyden vaikka jokin dokumentti siirretään tai poistetaan. Tämä yhtenäisyyden vaatimus jätettiin tarkoituksella pois WWW:stä, mikä on sallinut dokumenttien ja linkkien määrän rajoittamattoman kasvamisen ja hajautetun arkkitehtuurin [4].

W3C:n suosittelema tapa linkkien ilmaisemiseen XML-dokumenteista on Xlink-kieli [22]. Jos linkin kohteena on myös XML-dokumentti, siihen voidaan osoittaa käyttämällä XPointer-kielistä osoitusta URI:n osana [21]. XLink on WWW:n hyperlinkkien yleistys, joka määrittelee joukon vakiotavalla nimettyjä attribuutteja omaan XML-nimiavaruuteensa (esim. viittausosoitetta kuvataan attribuutilla `xlink:href`). Näitä attribuutteja voidaan käyttää uusissa XML-pohjaisissa kielissä ja XLink-kieltä ymmärtävät ohjelmat voivat tunnistaa linkit muista kielen elementeistä riippumatta. XLink on URI-linkkien yleistys siinä mielessä, että linkkiin voidaan määritellä kohteen ja kuvauksen lisäksi omia rooleja ja näyttötapaan liittyviä tietoja. Voidaan myös määrittää ns. laajennettuja linkkejä, jotka sallivat usean eri tietokohteen linkityksen yhteen samanaikaisesti. XPointer on yleistys HTML-dokumenttien fragmenttitunnisteille. Osoittimessa käytetään XPath-

kieleen perustuvaa syntaksia, mikä mahdollistaa id-viittausten lisäksi osoituksen yksittäisen elementin tarkkuudella. Menettely on joustava, koska tällöin viitattavia elementtejä ei tarvitse merkitä dokumenttiin id-attribuuteilla [7, sivut 143-154]. Viittauksia voi tehdä pelkkien rakennetietojen perusteella, jolloin id-arvoista ei tarvitse välttämättä edes tietää (esim. viite XHTML-dokumentin pääotsikkoon merkittäisiin `xpointer(//h1)`). Lisäksi XPointer-osoitus voi osoittaa samanaikaisesti moneen kohtaan dokumenttia. XLink- ja XPointer -suositukset ovat monipuolisia ja ilmaisuvoimaisia, mutta valitettavasti vain harvat sovellukset tukevat niitä. XLinkin ongelma on, että monissa XML-pohjaisissa kielissä määritellään oma linkkimekanismi, joka ei ole yhteensopiva XLink-määrityksen kanssa (joko eri nimiavaruuden tai erinimisen attribuutin takia). Näin on jopa muutamien W3C:n suositusten osalta, tärkeimpänä XHTML.

Linkkejä analysoimalla voidaan arvioida tietyn dokumentin luotettavuutta tai linkitettyjen dokumenttien samanlaisuutta [46]. Lisäksi uudemmat web-hakukoneet (esim. Google) hyödyntävät linkkejä arvioidessaan hakutulosten relevanssia [9]. Esimerkiksi, jos useat sivut viittaavat samoille sivustoille, viittaavat sivut ovat todennäköisesti aiheeltaan samankaltaisia. Toisaalta sivusto, johon monet samanaiheiset sivut viittaavat, sisältää todennäköisesti merkittävää tietoa viittaavien sivujen aiheesta.Linkkien analysointia käsitellään tarkemmin luvussa 4.3.

3.2.3 Metatieto

Metatieto (metadata) on tietoa tiedosta. Tarkka määrittely riippuu määrittelijästä: kirjastoalalla metatieto merkitsee indeksejä, lyhennelmiä, luokitussääntöjä ja yleensäkin tietokannoissa olevaa, kirjastojen kokoelmia kuvaavaa tietoa. Tietokantojen kannalta taas skeemat ovat metatietoa [25, sivut 4-5]. HTML-dokumenteissa metatieto on `<meta>` -elementeissä (joiden väärinkäytön johdosta eräät hakukoneet jättävät tosin metatiedot kokonaan indeksoimatta) [11, 11-12], semanttisessa webissä metatieto on RDF-kielisiä kuvauksia mistä tahansa URI-osoitteella viitattavissa olevassa kohteesta [43]. Yleisimmän mahdollisen määritelmän antaa Gilliland-Swetland [33], jonka mukaan metatieto on *summa kaikesta, mitä informaatio-objektista voidaan sanoa millä tahansa koostetasolla*. Informaatio-objekti tarkoittaa tässä mitä tahansa, mikä on ihmisen tai järjestelmän viitattavissa ja käsiteltävissä omana yksikkönään (jos viittaus voidaan tehdä URI-tunnisteella, tämä vastaa RDF-terminologiassa *resurssia*). Metatieto sijaitsee yleensä kuvaamansa kohteen ulkopuolella.

Ollakseen käyttökelpoista metatietoa käyttävällä järjestelmällä täytyy olla tiedossaan metatietorakenteiden merkitykset. Yleisessä käytössä oleva standardi on esim. Dublin Core[9], joka tarjoaa 15 määrämuotoista kenttää ”minkä tahansa” dokumentin kuvaamiseen. Dublin Coren lisäksi on olemassa runsaasti[10] sovellusaluekohtaisia standardeja metatietorakenteiden määrittelyyn. Näitä rakennekuvauksia kutsutaan ontologioiksi. Gruber [36] määrittelee ontologian olevan *formaali, eksplisiittinen määrittely yhteisestä käsitteistöstä* tietämyksen kuvaamiseen. Yksinkertaisimmillaan tämä tarkoittaa sanalistaa, yleensä käsite- tai tyyppihierarkiaa, kehittyneimmillään kyse on loogisesta sovellusalueen teorista, jossa käsitteet — metatietotyypit suhteineen, yksittäiset kentät ja niiden sisällöt — määritellään formaalisti. Tämä mahdollistaa koneellisen päättelyn meta-

[9] http://dublincore.org/

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

tietokuvausten perusteella, mikä edelleen on edellytys semanttiselle webille [5]. Ontologiat ovat ”skeemoja metatiedolle”.

WWW-ympäristössä tärkein yleiskäyttöinen tapa metatiedon kuvaamiseen on W3C:n määrittelemä RDF-kieli (*Resource Description Framework*) [43]. RDF mahdollistaa HTML-kielen `<meta>` -elementtejä monipuolisemman tiedon liittämisen resursseihin; jopa niin, että tiedon kuvaamisessa käytettävät käsitteet voivat olla eri lähteistä. RDF-kuvaukset ovat kolmikkoja

$$(Resurssi, Ominaisuus, Arvo),$$

joista jokainen voi edelleen olla resurssi. Kielitieteen käsittein metakuvaukset ovat lauseita, joissa resurssi on subjekti, ominaisuus predikaatti ja arvo objekti. Predikaattilogiikassa kolmikko vastaa kaksipaikkaista predikaattia. Resurssikuvausten joukko voidaan tulkita myös suunnattuna, tyypitettynä graafina (poiketen XML:stä, jonka tietomalli on oleellisesti puumainen). RDF-kielessä on oma linkitystoimintonsa, jolla kuvaukset liitetään resursseihin. Kielessä on vakiona yksinkertaisia säiliöluokkia, kuten jonot ja laukut. Käyttäjät voivat myös määritellä omia rakenteisia tietotyyppejä (yksinkertaisia ontologioita) RDFS-skeemakielen avulla. Varsinaisten resurssikuvausten lisäksi RDF tulee konkretisointia (reification), väitteiden tekemistä muista RDF-väitteistä. RDF-kieltä varten on määritelty kaksi XML-syntaksia (tavallinen ja lyhennetty) sekä erityinen N-triples -notaatio, joka soveltuu XML-syntaksia paremmin ihmisen luettavaksi. Merkittävää RDF-kielessä ei ole kuitenkaan syntaksi, vaan tietomalli, joka mahdollistaa monimuotoiset metakuvaukset. [43]

Rakenteisuuden ja hyperlinkkien lisäksi myös metatietoa on mahdollista käyttää apuna tiedonhaussa. Shah *et al.* [55] esittävät semanttisen- ja kokotekstihaun yhdistämistä siten, että dokumenttitekstistä eristetyt käsitteet muunnetaan RDF-kuvauksiksi ja yhdistetään saatavilla olevan metatietoon. Tuloksena saatavasta indeksistä olisi mahdollista hakea indeksitermien ohella metatiedossa olevia semanttisia suhteita. Tässä oletetaan, että metatieto on joukko nimettyjä tekstikenttiä, mutta kentillä ei ole sisäistä rakennetta tai formaalisti määriteltyjä suhteita. Tämä mahdollistaa esim. Dublin Core -määritysten mukaisen, RDF-kielellä tai `<meta>` -elementeillä merkityn metatiedon käytön, mutta ei rakenteisia tyyppejä tai käsitehierarkiaa sisältävien ontologioiden hyödyntämistä eikä automaattista päättelyä.

4 Tekstitiedon analysointitekniikat

Kappaleessa käydään läpi esimerkkejä eri haku- ja indeksointimenetelmistä tekstitiedon, linkkien ja rakenteisten dokumenttien osalta.

4.1 Dokumenttien esikäsittely

Riippumatta käytetystä menetelmästä edellytys toimivalle tekstianalyysijärjestelmälle on oleellisen tiedon erottelu dokumentista indeksiä varten (käsitellään yksinkertaista hakumallia, jossa dokumentit esitetään indeksitermien kokoelmana). Dokumenttien esikäsittely on prosessi, jossa kontrolloidaan indeksitermien määrää ja laatua. Tällöin indeksin koko pienenee ja haun tarkkuus paranee. Esikäsittelyn taustalla on intuitiivisesti järkevä oletus, että kaikki sanat eivät ole

merkityssisältönsä kannalta samanarvoisia. Dokumentin esitys kaikkien sanojensa samanarvoisena joukkona on yleisesti käytetty, mutta epätarkka esitys, jota esim. web-hakukoneet käyttävät nopeus- ja helppokäyttöisyyssyistä. Haun tarkkuus heikkenee, mutta peruskäyttäjälle saattaa olla helpompaa käyttää kokotekstihakua kontrolloidun sanaston sijaan. Dokumenttien esikäsittely voidaan jakaa seuraaviin vaiheisiin: [3, sivut 163-173]

- **Leksikaalinen analyysi** on prosessi, jossa merkkivirta (dokumenttiteksti) muunnetaan sanavirraksi (termikandidaatit). Välilyöntien lisäksi analyysissa on otettava huomioon numerot, tavutus, välimerkit sekä mahdollinen isojen ja pienten kirjainten käsittely. Leksikaalinen analyysi on suoraviivainen operaatio, jossa dokumentteihin sovelletaan yhtenäistä sanojen erottelukäytäntöä. Poikkeukset voidaan ilmaista esim. säännöllisinä lausekkeina.
- **Sulkusanojen poisto**. Jopa 80% dokumenteissa olevista sanoista on niin yleisiä, että ne ovat käyttökelvottomia tiedonhaun kannalta. Näitä sulkusanoja (*stopwords*) ovat kielestä riippuen esim. artikkelit, prepositiot ja konjunktiot. Ne suodatetaan yleensä pois termikandidaattien joukosta. Tämä pienentää indeksin kokoa ja tehostaa hakua. Sulkusanojen poisto saattaa heikentää saantia, jos hakulause koostuu pääasiassa sulkusanoista.
- **Stemmaus** (*stemming, vartalointi*) on kieliriippuvainen prosessi, jossa sanojen taivutusmuodot pyritään palauttamaan perusmuotoon. Ilman stemmausta indeksiin saattaa päätyä monikko- ja eri sijamuotojen takia monta eri muotoa samasta sanasta. Stemmaus pienentää indeksin kokoa ja tehostaa hakua, tosin kirjallisuudessa on ristiriitaisia tuloksia stemmauksen toimivuudesta käytännön hakutehtävissä. Jälkiliitteen poisto on yksinkertaisin ja tärkein stemmausmenetelmä.
- **Indeksitermien valinnalla** pyritään löytämään kandidaattitermien joukosta merkityksellisimmät. Jos indeksoitavien sanojen sanaluokka pystytään tunnistamaan, kannattaa keskittyä substantiiveihin, koska ne sisältävät eniten tietoa dokumentin aiheesta. Peräkkäisiä substantiiveja voidaan ryhmitellä, koska monet käsitteet ilmaistaan sanojen yhdistelmällä.
- **Tesaurus** (asiasanasto) on rakenteinen esitystapa tietyn sovellusalueen käsitteistölle. Se on kontrolloitu sanasto, joka sisältää tietoa termien välisistä suhteista. Termit ovat yleensä (tarvittaessa adjektiiveilla tarkennettuja) substantiiveja. Manualisesti määritellyssä rakenteessa termeillä voi olla myös luonnollisella kielellä annetut määrittelyt. Tesaurus tarjoaa standardisanaston indeksointiin ja hakuun, auttaa käyttäjää löytämään kyselyyn oikeat termit ja mahdollistaa haun muokaamisen tarkemmaksi tai yleisemmäksi. Tesaurusten ongelma on joustamattomuus dynaamisen sisällön tai lyhyellä aikavälillä muuttuvien termien suhteen. Lisäksi kokemattomalla käyttäjällä saattaa olla vaikeuksia löytää haun kannalta oleellisimpia termejä laajasta tesauruksesta.

4.2 Tekstianalyysi

Luvussa käsitellään rakenteettomasta tekstistä koostuvien dokumenttien esittämistä tiedonhaun näkökulmasta. XML-dokumentteja voidaan käsitellä tekstinä joko kokonaisuudessaan tai poistamalla elementtatagit. Rakenteisten dokumenttien kannalta puhdas teksti on rajoittunut esitystapa, koska se ”latteistaa” dokumentin. Vastaavasti myös tiedonlouhinnassa käytetty datamatriisi voi olla monen toisistaan jo riippuvan tietokantataulun liitos. Rakenteisilla dokumenteilla indeksirakenne voi olla tekstidokumentteja monimutkaisempi. Indeksoinnissa voidaan käyttää hyväksi

esim. termien sijaintitiheyksiä dokumenttipuun eri osissa, mutta paikallisesti (dokumenttipuun ”lehdissä”) rakenteiset dokumentit ovat oleellisesti tekstidokumentteja.

Merkittävin tekstidokumenttien esitystapa (sekä tiedonhaussa että klusteroinnissa) on vektorimalli. Myös vaihtoehtoisia tapoja tekstidokumenttien (laskennalliseen) mallintamiseen käsitellään lyhyesti. Käsittelyn ulkopuolelle jäävät varsinaiset luonnollisen kielen käsittelyn menetelmät, kuten tekstin semanttisten suhteiden analyysi (katso esim. Aunimo [2]).

4.2.1 Vektorimalliin perustuvat menetelmät

Vektorimalli on Saltonin *et al.* [53] kehittämä tiedonhakumalli, joka perustuu dokumenttien ja kyselyjen esittämiseen vektoreina. Malli on suosittu erityisesti web-hakukoneissa, koska se on helppo toteuttaa, nopea käyttää, mahdollistaa hakutulosten järjestämisen ja osittaiset täsmäykset. Mallia on kritisoitu mm. täsmällisen tilastollisen perustan puutteen vuoksi ja siksi, että dokumenttien termit oletetaan toisistaan riippumattomaksi. Yksinkertaisuudeestaan huolimatta se on osoittautunut käyttökelpoiseksi käytännön sovelluksissa [3, sivut 27-30]. Formaalisti malli voidaan määritellä seuraavasti:

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

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

ovat dokumenttivektoreita ja rivit

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

edustavat tietyn termin painoja dokumenteissa. Matriisin elementti w_{ij} on siis dokumentin j termin i paino. Matriisin rivit ja sarakkeet esitettäisiin käänteisesti (W^T), jos sanamatriisi toteutettaisiin relaatiotietokannan tauluna. Edellä kuvattu tapa on kuitenkin kirjallisuudessa yleisemmin käytetty.

Painot merkitsevät termin suhteellista tärkeyttä dokumentissa ja ne normalisoidaan yleensä välille $[0, 1]$ siten, että painolla 0 termi ei esiinny dokumentissa lainkaan ja 1 tarkoittaa mahdollisimman hyvin tiettyä dokumenttia kuvaavaa termiä. Painotus on yleistys vanhasta boolen mallista, jossa painojen arvo voi olla vain 0 tai 1 [3, sivut 25-27]. Termien painotukseen on useita eri tapoja, joista yleisimmät pohjautuvat termifrekvenssiin (*term frequency, tf*) ja käänteiseen dokumenttifrekvenssiin (*inverted document frequency, idf*). Näissä $tf \times idf$-malleissa sanamatriisin arvot määritellään kaavalla $w_{ij} = tf_{ij} idf_i$ [11, sivut 56-57]. Termifrekvenssi on

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

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

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

missä $|\mathbf{D_i}|$ on niiden dokumenttien lukumäärä, joissa termi i esiintyy. IDF:ssä käytetään logaritmia, jotta mitta olisi riippumaton dokumenttien kokonaismäärästä $|\mathbf{D}|$ [37, sivu 463].

Dokumenttien tapaan myös kysely voidaan esittää painotettuna vektorina q, jonka painot voivat poiketa dokumenttivektorien painoista [52]. Ilman painotusta kyselyyn kuuluvat termit merkitään 1:llä ja muut 0:lla. Frekvenssien tarkat määritelmät vaihtelevat kirjallisuudessa jonkin verran painotustavasta riippuen. Salton & Buckley [52] ovat tehteet perusteellisen vertailun erilaisista $tf \times idf$ -tyylisistä painotustavoista. Vaihtoehtoinen lähestymistapa on käyttää painotukseen esim. termien entropiaa [24]. Tarkasta kaavasta riippumatta termifrekvenssin perusideana on, että usein tietyssä dokumentissa olevat termit kuvaavat kyseistä dokumenttia hyvin. Käänteisen dokumenttifrekvenssin idea on, että dokumentit voidaan erotella toisistaan parhaiten termeillä, jotka esiintyvät vain pienessä määrässä dokumentteja.

Hakutulokset täsmäytetään vertaamalla kyselyvektoria dokumenttivektoreihin. Yleisin käytössä oleva vertailufunktio on vektorien pituuden suhteen normalisoitu sisätulo, joka voidaan tulkita myös vektorien d_i ja q välisenä kulmana $\mathbf{R^n}$:ssä. Tästä syystä samanlaisuusmittaa kutsutaan usein kosinimitaksi.

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

Muita mahdollisuuksia on käyttää vertailuun esim. euklidista etäisyyttä, laajennettua Jaccardin mittaa, Dicen kerrointa tai Pearsonin korrelaatiota [57]. Kyselyjen ja dokumenttien lisäksi myös yksittäisiä dokumentteja voidaan verrata toisiinsa, mikä mahdollistaa dokumenttien klusteroinnin. Etäisyysmitan valinta ei vaikuta merkitsevästi klusterointiin, kunhan dokumenttien pituudet normalisoidaan vertailussa [60, 24-28].

Vektorimallissa indeksitermit ovat toisistaan riippumattomia ja ortogonaalisia. Yleistetyssä vektorimallissa [65] täsmäytyksessä huomioidaan indeksitermien väliset riippuvuudet ja esitetään menetelmä niiden estimointiin. Riippuvuuksista saadaan symmetrinen matriisi $T_{n\times n}$, jonka arvot vaihtelevat välillä $[-1, 1]$ siten, että arvolla 0 termit ovat ortogonaaliset. Huomattavaa on, että termit voivat olla toisistaan riippumattomia, vaikka ne eivät olisi ortogonaalisia. Täsmäytyksessä käytettävää samanlaisuusmittaa voidaan laajentaa niin, että riippuvuusmatriisin arvot otetaan huomioon. Jos oletetaan, että vektorit on normalisoitu, saadaan dokumentin ja kyselyn välille seuraava samanlaisuusmitta (jos riippuvuusmatriisi oletetaan identiteetiksi, saadaan kaavasta erikoistapauksena vektorimallissa käytetty sisätulo):

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

Latentti semanttinen indeksointi (*latent semantic indexing, LSI*) on indeksointimenetelmä ja tiedonhakumalli, jonka tarkoituksena on hyödyntää dokumenttien ja termien välillä olevaa implisiittistä semanttista (latenttia) rakennetta. Yleistetyn vektorimallin tapaan tavoitteena on löytää relevantteja dokumentteja, jotka sisältävät käyttäjälle tuntemattomia termejä. Tämä on erityisen tärkeää haettaessa tietoja monikielisestä dokumenttijoukosta [39]. Menetelmän kehittäjien [20] mukaan semanttinen rakenne voidaan löytää diagonalisoimalla sanamatriisi, eli etsiä sen singu-

laariarvohajotelma. Diagonalisointi on lineaarialgebran perustekniikka, jonka avulla sanamatriisi W voidaan hajottaa osiin siten, että seuraava yhtälö pätee [11, sivu 97]:

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

missä r on semanttisen avaruuden ulottuvuuksien määrä, U ja V ovat kannanvaihtomatriiseja, joille pätee $U^TU = V^TV = I$. Σ on matriisi, jonka diagonaalilla ovat W:n ominaisarvot laskevassa järjestyksessä. Täsmäytyksessä dokumentit ja kyselyt on muunnettava semanttiseen avaruuteen, jonka jälkeen niitä voidaan verrata kuiten vektorimallissa. Yleensä vain k suurinta ominaisarvoa huomioidaan, jolloin mallin dimensio pienenee. Tämä voidaan tulkita niin, että sanamatriisista poistetaan ”kohinaa” [11, sivu 98]. Nimestään huolimatta LSI:ssä ei ole mitään semanttista, vaan kyseessä on lineaarialgebran perusmenetelmän soveltaminen ilman vahvoja teoreettisia perusteluja. LSI on oleellisesti sama kuin tilastollisessa hahmontunnistuksessa ulottuvuuksien vähentämiseen käytetty pääkomponenttianalyysi. LSI vähentää sanamatriisin dimensiota projisoimalla dokumenttivektorit lähimpään k-ulotteiseen lineaariseen aliavaruuteen, pääkomponenttianalyysi projisoi ne lähimpään affiiniin aliavaruuteen (origo siirtyy dokumenttivektorien keskiarvoon) [58, sivut 7-9].

Baeza-Yates *et al.* [3, sivut 41-45] luokittelevat yleistetyn vektorimallin ja LSI:n omiksi tiedonhakumalleikseen, mutta koneoppimisen tai tiedonlouhinnan näkökulmasta ne ovat vain erilaisia tapoja piirteiden valintaan ja muuntamiseen. Täsmäytysfunktio ja dokumenttien vertailu perustuvat kaikissa malleissa dokumentti- ja kyselyvektoreihin. Jiang & Littman [39] esittävät yleisen mallin vektoripohjaisten tiedonhakumallisen kuvaamiseen, joka kattaa klassisen vektorimallin, yleistetyn vektorimallin ja LSI:n. Jos oletetaan, että vektorit on normalisoitu ja vertailufunktiona käytetään sisätuloa, saadaan dokumentin ja kyselyn välille seuraava yleinen samanlaisuusmitta:

$$sim(d_j, q) = ((P^Td_j)|(P^Tq)),$$

missä P on muunnosmatriisi. Klassisessa vektorimallissa P on identiteetti, yleistetyssä mallissa termien riippuvuuksista johdettu johdettu muunnosmatriisi T ja LSI:ssa ominaisarvohajotelmasta johdettu I_kU^T. I_k on identiteetti, jossa vain diagonaalin k ensimmäistä arvoa ovat ykkösiä.

4.2.2 Erikoismenetelmiä

Vektorimallin ohella teoreettisesti tärkeäksi, mutta käytännössä harvemmin käytetyksi malliksi ovat nousseet probabilistiset hakumallit. Tarkoituksena on mallintaa *todennäköisyys* kunkin dokumentin relevanssille tiettyä kyselyä vastaan (poiketen esim. vektorimallista, jossa dokumentin relevanssi määrätään suoraan vertaamalla sen esitystä kyselyn esitykseen). Tilastollisten hakumallien pohjalla on oletus, jonka mukaan *dokumentin relevanssi on riippumaton muista kokoelmassa olevista dokumenteista*. Tämän oletuksen pohjalta on muotoiltu tilastollinen järjestysperiaate (*probability ranking principle, PRP*). Sen mukaan *todennäköisen relevanssin mukaan järjestetty tuloslista on optimaalinen edellyttäen, että todennäköisyydet on estimoitu parhaalla mahdollisella tarkkuudella käytettävissä olevista tiedoista*. Järjestysperiaate ei ole täysin ongelmaton: sen pohjalla oleva riippumattomuusoletus on täysin päinvastainen klusterointihypoteesin kanssa [60, sivut 87-93] eikä periaate ei kerro, *miten* todennäköisyydet voidaan estimoida [3, sivut 30-34].

Lisäksi periaatteelle on löydetty vastaesimerkkejä käytännön tiedonhakujärjestelmistä, joten oikeastaan pitäisi puhua heuristiikasta. PRP:n eduksi voidaan kuitenkin lukea intuitiivinen tulkinta dokumentin käyttökelpoisuudesta ja mahdollisuus tulkita tiedonhaun vaiheet yhtenäisessä teoreettisessa kehyksessä [15].

Tärkeimmät tiedonhakumallit voidaan tulkita tilastollisina, vektorimalli mukaan lukien. Tässä mielessä probabilistiset mallit ovat yleistys vanhemmille hakumalleille [30]. Esimerkkinä yksinkertaisesta tilastollisesta hakumallista käsitellään binäärinen riippumattomuusmalli (*binary independence retrieval, BIR*). Dokumentti d esitetään bittivektorina, jonka kukin komponentti vastaa vektorimallin tapaan yhtä termiä. Määritellään poissulkevat tapahtumat R *dokumentti on relevantti* ja $\overline{R}$ *dokumentti ei ole relevantti*. Nyt Bayesin kaavan perusteella saadaan dokumentin d relevanssin todennäköisyys kyselyn q suhteen

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

ja $\overline{R}$ vastaavasti. $P(R)$ on priori-todennäköisyys mielivaltaisen dokumentin relevanssille ja

$$P(d) = P(x|R)P(R) + P(x|\overline{R})P(\overline{R}).$$

Koska malli olettaa termit riippumattomiksi toisistaan, saadaan $P(d|R)$:lle (vast. myös $P(d|\overline{R})$) kaava

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

missä $\mathbf{T}_d$ on niiden termien joukko, jotka esiintyvät dokumentissa d ja $P(t|R)$ on todennäköisyys sille, että termi t esiintyy relevantissa dokumentissa [3, sivut 30-34]. $P(t|R)$:n esimointiin on useita menetelmiä, mutta niitä ei käsitellä tässä tarkemmin. Bayesilaisen päättelyn tuloksena dokumentti d sisällytetään relevantteihin hakutuloksiin, kun epäyhtälö $P(R|d) > P(\overline{R}|d)$ pätee [60, 89-93].

Tarkempi tapa dokumenttien mallintamiseen on kuvata ne stokastisten prosessien avulla. Stokastinen prosessi on indeksoitu jono satunnaismuuttujia (dokumenttien kannalta muuttujat voivat olla peräkkäisiä kirjaimia tai sanoja). Yleisessä muodossaan muuttujien välillä voi olla mielivaltaisia riippuvuuksia, mutta laskennallisesti käsiteltävämpi on malli, jossa vain n edellistä muuttujaa vaikuttaa lopputulokseen. Tällaista mallia kutsutaan n. asteen Markovin ketjuksi. 1. asteen Markovin ketju satunnaismuuttujille $X_1, X_2, ...$ voidaan määritellä yhtälöllä [16, sivut 60-62]

$$P(X_{n+1} = x_{n+1}|X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n).$$

Malli on oleellisesti satunnaista binäärimallia tarkempi. Esim. 3. tai 4. asteen malleja käyttämällä pystytään generoimaan dokumentteja, jotka yksittäisiltä lauseiltaan näyttävät luonnolliselta kieleltä [16, 131-136]. Markovin ketjuja on perinteisesti käytetty laskennallisessa kielitieteessä mallinnettaessa kielen yleisiä ominaisuuksia (esim. entropiaa voidaan estimoida), mutta mallin ideoita voidaan soveltaa myös tiedonhakuun ja klusterointiin. n-grammit ovat n peräkkäisen kirjaimen tai sanan muodostamia yksiköitä, joita voidaan käyttää indeksoinnissa termien sijaan. Indeksointi suoritetaan liu'uttamalla n yksikön mittaista ”ikkunaa” dokumentin yli. Koska käytetty

aakkosto on rajallinen, voidaan laskea kaikkien käytettävissä olevien n-grammien määrä G ja esittää dokumentti G-ulotteisena vektorina (käytännössä esim. hajautustauluna), jonka painot saadaan kaavalla

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

missä $\sum_{j=1}^{G} x_j = 1$ ja m_j on j:nnen n-grammin lukumäärä dokumentissa. Tämän jälkeen dokumenteista voidaan hakea tietoja ja niitä voidaan klusteroida aivan kuten vektorimallissa. N-grammipohjaisen indeksointimallin etuna termipohjaiseen on kieliriippumattomuus (esim. sulkusanoja ei tarvitse poistaa esikäsittelyvaiheessa) ja sopeutumiseen kohinaiseen (esim. kirjoitusvirheet) tekstiin [19].

Luonnollisen kielen käsittelyä esikäsittelyvaiheessa hyödyntävä, mutta lopputulokseltaan n-grammien kaltainen malli ovat leksikaaliset suhteet (*lexical affinity, lexical relation*), joilla kuvataan kielitieteessä kahden kielen yksikön yhteisesiintymiä. Maarekin *et al.* [45] lähestymistavassa yhteisesiintymillä kuvataan dokumenttikokoelmaa koko kielen sijaan. Leksikaaliset suhteet kerätään 2-5 sanan etäisyydellä samassa lauseessa olevista, perusmuotoon palautetuista avoimiin sanaluokkiin (verbit, adjektiivit, substantiivit) kuuluvista sanoista. Dokumenttikokoelmassa esiintyvät suhteet kootaan sanatason 2-grammeiksi, joita painotetaan entropiaa muistuttavalla muunnoksella, mutta periaatteessa muutkin painotustavat käyvät.

4.3 Linkkianalyysi

Linkkianalyysia voidaan käsitellä verkkoteorian, informaatiotutkimuksen tai hypertekstikokoelmien näkökulmasta. Dokumenttien välisten suhteiden mittaamiseen on kehitetty lukuisia eri mittoja, jotka voidaan jakaa [23] verkkoteoreettisiin, merkittävyyttä mittaaviin, samanlaisuusmittoihin, hakumittoihin, käyttöä mittaaviin ja informaatioteoreettisiin.

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

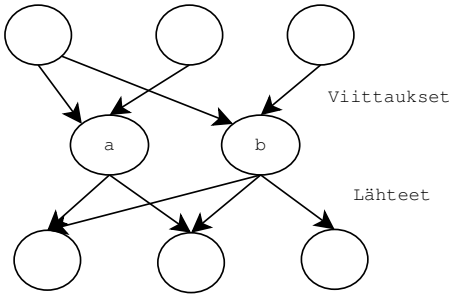
Luvussa keskitytään vain tiedonhaun kannalta keskeisimpiin samanlaisuus- ja merkittävyysmittoihin, jotka käyttävät pelkästään linkkitietoa tai linkkien selitteitä.

4.3.1 Bibliometriset samanlaisuusmitat

WWW-sivujen ja muiden rakenteisten dokumenttien välisten samanlaisuussuhteiden arviointi pohjautuu jo vuosikymmeniä informaatiotutkimuksen alaan kuuluvaan bibliometriikkaan (myös yleisempää nimitystä informetriikka käytetään). Osareh [49] määrittelee bibliometriikan tarkoituksena olevan tieteellisen dokumentaation, tiedon ja viestinnän parantamisen kirjastojen kokoelmien ja palvelujen määrällisellä analyysilla. Lisäksi hän esittää lukuisia eri ajankohtina annettuja tarkempia määrittelyjä, mutta tämän harjoitustyön kannalta bibliometriikka on matemaattisten ja tilastollisten menetelmien soveltamista kirjallisuusviitteisiin. Viiteanalyysin avulla voidaan arvioida dokumenttien samanlaisuutta tai tietyn dokumentin vaikuttavuutta (olettaen, että viitettä toiseen

dokumenttiin pidetään myönteisenä merkkinä). Suomen kielessä dokumentissa olevaa lähdeviitettä (lähtevää linkkiä) kutsutaan lähteeksi (*reference*) ja dokumentin toisesta dokumentista saamaa viitettä (tulevaa linkkiä) kutsutaan nimellä viittaus (*citation, sitaatio*) [42, sivut 12-13].

Bibliometriikassa käytettyjä klassisia samanlaisuumittoja ovat lähdeanalyysiin kuuluva bibliografinen kytkentä (*bibliographic coupling*) ja viittausanalyysiin kuuluva yhteisviittaukset (*co-citation*). Bibliografinen kytkentä dokumenteille a ja b on niiden dokumenttien määrää, joihin molemmat viittaavat. Yhteisviittaus on niiden dokumenttien määrä, jotka viittaavat kumpaankin dokumenttiin. Molemmat mitat voidaan normalisoida jakamalla tulevien (tai vastaavasti lähtevien) solmujen kokonaismäärällä. Kuva 2 havainnollistaa bibliometrisiä etäisyysmittoja. Dokumenttien a ja b normalisoitu yhteiviittausarvo on $1/3$ ja kytkentä $2/3$.



Kuva 2: Esimerkki yhteisviittauksista ja kytkennästä.

Normalisoitu kytkentä voidaan laskea linkkimatriisin avulla seuraavasti:

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

Yhteisviittaus voidaan esittää vastaavasti:

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

Jos lähteiden kokonaismäärä kytkennässä on 0, määritellään kytkennäksi 0. Vastaavasti yhteysviittaus määritellään 0:ksi, jos viitteiden määrä on 0. Käyttämällä bibliometrisiä etäisyysmittoja haussa tai klusteroinnissa saadaan yleiskuva dokumenttikokoelmasta edellyttäen, että kokoelma sisältää kunnollisen linkkirakenteen. Tällöin käyttäjän ei tarvitse tietää sovellusalueen keskeisiä hakutermejä — vaikka ne olisivatkin tiedossa, dokumenteissa olevat termit saattavat vaihdella. Linkkitieto on neutraalia termien ilmiasun suhteen [41]. Toisaalta dokumentin laatija päättää enemmän tai vähemmän subjektiivisesti lähteistään.

Kytkeytyminen ja yhteisviittaukset kuvaavat samaa asiaa eri näkökulmista. Tuntuu luontevalta yleistää mittaa niin, molemmat suunnat huomioidaan. Cristo *et al.* [17] kutsuvat kytkeytymisen ja yhteiviitteen yhdistelmää *Amsler*-mitaksi, mutta nimitys ei ole levinnyt laajaan käyttöön – sitä ei löytynyt mistään muusta käytettävissä olevasta lähteestä. Perusteluja mittojen yhdistämiselle sen sijaan löytyi: esimerkiksi Kochtanek [41] toteaa, että pelkkien yhteisviittausten käyttö heikentää vanhojen dokumenttien merkitystä ja kytkeytyminen ei huomioi uusia samanaiheisia

dokumentteja. Yhdistetty etäisyysmitta vähentää riippuvuutta dokumentin julkaisuajankohdasta. Toisaalta Popescul *et al.* [50] huomauttavat, että dokumenttikokoelman kasvaessa yksittäisten dokumenttien kytkeytyminen pysyy samana, mutta yhteisviittausten määrä voi kasvaa, kun uudet dokumentit viittaavat vanhoihin. Tästä syystä yhteisviittaus on pitkällä aikavälillä kytkeytymistä informatiivisempi. Mitoista on esitetty myös useamman askeleen päähän ulottuvia rekursiivisia versioita [23], mutta ne jätetään tämän käsittelyn ulkopuolelle.

Thelwall & Wilkinson [59] ovat tutkineet akateemisten web-sivustojen samanlaisuutta mitaten yhteisviittausten ja kytkeytymisen lisäksi suoria linkkejä. Tarkkuustasona olivat yksittäiset web-palvelimet (yliopistojen laitokset). Tuloksena todettiin, että linkit olivat paras keino samanlaisuuden arviointiin, joskin suurin osa määrä odotetuista yhteyksistä jäi löytämättä linkkien suhteellisen vähäisen määrän vuoksi. Yhteisviittauksista ja kytkeytymisestä oli myös marginaalista hyötyä. Cristo *et al.* [17] tutkivat etäisyysmittoja web-sivujen luokittelun kannalta ja havaitsivat yhteisviittaukset parhaiten erottelevaksi piirteeksi. Dokumenttikokoelmasta riippuen myös yhdistetyllä mitalla päästiin hyviin tuloksiin.

4.3.2 WWW:n suosiomittarit

Luultavasti merkittävin yksittäinen web-hakukoneiden laatua parantanut tekijä on ollut linkkitiedon hyödyntäminen relevanssiarvion laskennassa. Tärkeimmät perusmenetelmät ovat Brinin ja Pagen esittämä ja Google-hakukoneessa käytetty PageRank [9] sekä Kleinbergin HITS (Hyperlink Induced Topic Search) -algoritmi [40]. Molemmat algoritmit ovat kirjallisuudessa runsaastiviitattuja. Erityisesti HITS-algoritmiin on tehty lukuisia parannuksia ja jatkotutkimusta, jota esim. Chakrabarti [11, 209-242] on koonnut yhteen. Tässä käsitellään tiiviyden vuoksi vain alkuperäiset menetelmät.

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

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

Intuitiivisesti kaava mallintaa ”satunnaisen surffaajan” todennäköisyyttä päätyä sivulle a. Oletetaan, että surffaajalle annetaan satunnainen sivu, jonka jälkeen hän klikkaa satunnaisesti jotain sivulla olevaa linkkiä ja jatkaa edelleen satunnaisesti, kunnes kyllästyy todennäköisyydellä d. Tällöin hän aloittaa uudestaan satunnaiselta sivulta. PageRank-arvoista muodostettu vektori approksimoi normalisoidun linkkimatriisin ensimmäistä ominaisvektoria [9]. d:n merkitys on kompensoida WWW:n linkkigraafin harvaa rakennetta: kaikki sivut eivät ole yhteydessä toisiinsa [11, sivu 211].

PageRankista poiketen HITS-algoritmin arvot riippuvat käyttäjän antamasta kyselystä. HITS-algoritmin tavoitteena on erotella ja arvottaa tulossivut auktoriteetteihin ja napoihin. Auktoriteetit ovat sivuja, joihin monet navat viittaavat, navat ovat sivuja, jotka sisältävät paljon linkkejä

auktoriteetteihin. Algoritmin pohjana on *juurijoukko*, alustavat hakutulokset. Juurijoukkoa laajennetaan sivuilla, jotka viittaavat johonkin juurijoukon sivuun sekä sivuilla, joihin juurijoukon sivut viittaavat. Näin saadaan *perusjoukko*, jonka pohjalta voidaan muodostaa kyselyriippuvainen graafi $G_q = (D_q, E_q)$, missä $D_q \subset D$ on perusjoukko ja E_q on niiden linkkien joukko, joiden alku- ja loppuosa ovat perusjoukossa D_q. Jokaisella perusjoukon sivulla on omat auktoriteetti- ja napa-arvot, jotka voidaan kirjoittaa vektoreina a ja h siten, että vektorin i:s komponentti on perusjoukon i:nnen dokumentin vastaava arvo. Arvot lasketaan rekursiivisesti yhtälöiden $a = E^T h$ ja $h = Ea$ avulla. E on G_q:n matriisiesitys (linkkimatriisin C D_q-joukkoon liittyviä dokumentteja vastaavat sarakkeet ja rivit). Huomattavaa on, että yhtälöt riippuvat toisistaan. a-vektori approksimoi $E^T E$:n pääominaisvektoria ja h-vektori vastaavasti EE^T:n pääominaisvektoria [40]. Vektoreilla on myös kiinnostava yhteys tekstianalyysin puolelta tuttuun LSI-malliin: osoittautuu, että HITS-algoritmin soveltaminen linkkimatriisiin on yhtäpitävä operaatio linkkimatriisin diagonalisoinnille, samaan tapaan kuin LSI diagonalisoi sanamatriisin. Tämä mahdollistaa muunnetun linkkimatriisin käytön klusteroinnissa [11, sivu 212-216].

Pelkän linkkirakenteen lisäksi myös dokumenttiin viittaavan linkin läheisyydessä olevaa tekstiä voi käyttää kuvaamaan dokumenttia. Chakrabarti *et al.* [12] kutsuvat kuvausta ankkuri-ikkunaksi (*anchor window*) ja käyttävät sitä parantamaan hakutulosten järjestämistä osana HITS-tyyppistä algoritmia. Ankkuri-ikkunan käyttöä myös samanlaisuuden arvioinnissa puoltaa myös Gloverin *et al.* [34] web-dokumenttien luokittelua koskeva tutkimus. Paras luokittelutulos saatiin, kun dokumenttitekstin ja viittaavan linkkitekstin lisäksi hyödynnetään kappaletta, jossa linkkiteksti sijaitsee, ns. dokumentin kontekstia. Kontekstikappaleet osoittautuivat ratkaisevaksi luokittelussa, koska jo yksistään niiden avulla saatiin parempi luokittelutulos kuin dokumenttien tekstillä. Tiedonlouhinnan näkökulmasta ankkuri-ikkunat ovat tekstimuotoisia piirteitä, jotka voi yhdistää painotetusti varsinaisesta dokumenttitekstistä koostettuun indeksiin tai käyttää sen rinnalla. Myös Google-hakukone hyödyntää ankkuritekstiä, mikä mahdollistaa tiedonhaun myös sivuista, joihin hakukone ei suoraan pääse käsiksi (tietokantahaut, [9]. Tämä voi aiheuttaa myös väärinkäyttöä, jos suuri joukko sivuja linkittyy samaan sivuun mahdollisella asiattomalla linkkitekstillä. Tällöin kohdesivu päätyy ensimmäiseksi hakutulokseksi[11]. Tekniikkaa kutsutaan nimellä (*Google bombing*) [38]. HITS-algoritmia voidaan myös väärinkäyttää vastaavalla tekniikalla [11, 219-225].

4.4 Rakenneanalyysi

Luvussa käsitellään rakenteista indeksointia ja hakumalleja Tarkastelun kohteena ovat lähinnä elementit ja rakenteinen teksti, attribuuttien ja linkkien käsittely sivuutetaan tässä yksinkertaisuuden vuoksi. Rakenteiset hakulausekkeet kuvataan XPath-tyylisellä kielellä.

4.4.1 Rakenteinen indeksointi

Rakenteiset indeksointitekniikat voidaan jakaa tekstipohjaisiin (ei ota huomioon rakennetta), puolirakenteisiin (ei ota huomioon skeemaa) tai rakenteisiin (ottaa huomioon skeeman, käytetään

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

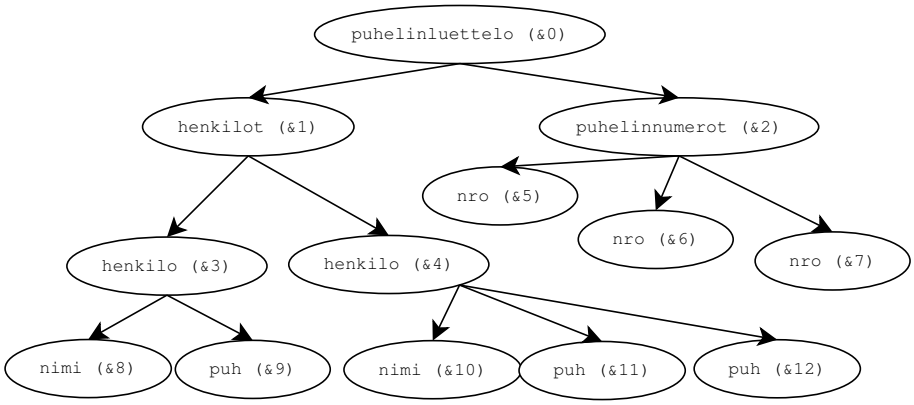
esim. tallennettaessa XML-dokumentteja relaatiotietokantaan) [44]. Tässä käsiteltävät indeksointitekniikat ovat pääasiassa puolirakenteisia: dokumentin rakenne otetaan indeksissä huomioon, mutta tiettyyn skeemaan ei sitouduta. Linkit jätetään yksinkertaisuuden vuoksi huomiotta, eli oletetaan, että dokumentit ovat puita. Lisäksi oletetaan, että dokumentit ovat jossakin puolirakenteisessa tietokannassa, jonka yksittäisiin elementteihin (solmuihin) voidaan viitata solmun tunnisteella. Indeksointirakenteesta riippuen tunniste voi olla yksinkertainen järjestysnumero tai sitten tietue, jonka avulla päästään navigoimaan muihin solmuihin tai esim. päättelemään, onko tietty tunniste elementtihierarkiassa toisen yläpuolella. Puolirakenteinen tietokanta on loogisesti yksi puu tai graafi, joka sisältää kaikki fyysiset dokumentit [62, sivut 34-36]. Fyysisesti dokumentteja voidaan säilyttää suoraan tiedostojärjestelmässä ja pitää ainoastaan indeksi tietokannassa. Tällöin solmujen tunnisteiden täsmääminen fyysisiin XML-elementteihin saattaa tosin aiheuttaa lisätyötä.

Weigel [62, sivut 39-40] jakaa indeksointityypit niiden mutkikkuuden mukaan perusindekseihin, polkuindekseihin ja puuindekseihin, joista kutakin käsitellään tarkemmin edellä. Haku- ja indeksointimalleja voidaan tarkastella myös yleisesti ns. PTN-luokittelun (*Path/Term/Node*) avulla [63]. PTN-luokittelussa mallia kuvataan P (polku), T (termi)- tai N (solmu)-komponenteilla tai niiden yhdistelmillä sen mukaan, mitä tarkkuustasoa malli tukee sisällön tai rakenteen mukaan. P-komponentti kuvaa kyselyn rakenteista osaa (esim. luvun 3 puhelinnumeroesimerkin `<henkilo>` -elementtiin voisi viitata lausekkeella `/puhelinluettelo/henkilot/henkilo`) ja T vastaavasti sisältöosaa. N kuvaa dokumentteja tai niiden osia (solmuja), jotka palautetaan. Tämän luokittelun perusteella vektorimalli $tf \times idf$-tyylisellä painotuksella noudattaa {TN,T,N}-tasoa, koska yksittäisten dokumenttien termifrekvenssit ovat {TN}-tasolla, käännetyt dokumenttifrekvenssit ovat {T}-tasolla ja termifrekvenssin laskennassa käytetty dokumenttikohtainen termien maksimimäärä, $max_{t \in \mathbf{T}}(n(t,j))$ (katso luku 4.2.1) on {N}-tasolla. Toisaalta vektorimallin yhtenä indeksirakenteenä käytetty painottamaton (boolen) sanamatriisi noudattaa vain {TN}-tasoa, koska pelkkää matriisia käyttämällä idf:n tai termien maksimimäärän laskeminen vaatisi matriisin tietyn rivin tai sarakkeen täydellistä läpikäyntiä.

Perusindeksit ovat oleellisesti assosiaatiotauluja tai taulukoita, yleistyksiä tekstidokumenteissa käytetyille käännetyille tiedostoille [62, sivut 39-46]. Perusindeksejä ovat tekstilista eli *käännetty solmulista* [63] (liittää jokaiseen termiin solmujoukon, jossa termi sijaitsee), arvolista (liittää attribuuttien arvot solmujoukkoihin), elementtilista (liittää elementtityypit niitä edustaviin solmujoukkoihin — ei kuitenkaan polkuja) ja vanhempi/lapsi-indeksi (indeksisolmujen hierarkiasuhteet). Tekstilista noudattaa PTN-luokittelussa {TN}-tasoa, muut indeksit eivät sovellu luokitteluun. Myös Luk:n *et al.* [44] puolirakenteisiin indeksointitekniikoihin luokittelema kenttäpohjainen indeksi voidaan laskea perusindekseihin. Kenttäpohjaisessa indeksoinnissa dokumentti kuvataan puun sijasta joukkona kenttiä, joista kukin voidaan indeksoida esim. vektorimallin mukaisesti. Kenttäpohjainen indeksi soveltuu dokumenteille, joissa ei ole syvää hierarkiaa tai jossa karkea rakenteen jako (esim. otsikko, metatietokentät, leipäteksti) riittää. Kenttäpohjaisen indeksin etuna on lähinnä sen yksinkertaisuus, koska mutkikkaita puurakenteita ei tarvitse toteuttaa.Sivun 4 XML-tiedoston osa on esitetty tietokannan solmuina kuvassa 3 (solmuissa mainittu ID-numerot, attribuutit jätetty huomiotta).

Tietokannan käännetty solmulista ja elementtilista on esitetty taulukoissa 1 ja 2

Pelkästään perusindeksejä käyttämällä on mahdollista tehdä hakujärjestelmä, joka tukee samanaikaisia rakenne- ja tekstihakuja. Järjestelmän suorituskyky on kuitenkin alhainen, koska ky-



Kuva 3: Puhelinnumeroesimerkki XML-tietokantana.

Nimi	*Solmujoukko*
A.A	$\{8\}$
N.N	$\{10\}$
435-345345	$\{5\}$
11-343255	$\{6\}$
435-365552	$\{7\}$

Taulukko 1: Sanalista

sely jouduttaisiin käsittelemään monessa vaiheessa: esim. puhelinluetteloesimerkkiin tehtävässä kyselyssä `/puhelinluettelo/henkilot/henkilo/nimi["A. A."]` täytyisi käydä läpi tekstilistasta `A. A.` -arvon sisältävät solmut ja yhdistää ne elementtilistasn `nimi` -elementin sisältäviin solmuihin. Jos `nimi` -elementti voisi olla jonkin muun polun varrella kuin kyselyssä mainitun (tässä esimerkissä ei ole, mutta jos skeema ei ole tiedossa, tästä ei olisi mitään takeita), `nimi` -elementtilistan solmujen polut pitäisi vielä tarkistaa hakemalla `henkilo` -elementtilistan solmut ja vertaamalla niitä `nimi` -solmujen vanhempiin (vanhempi/lapsi-listalla tai indeksirakenteesta riippuen suoraan tunnistenumeroilla) — mahdollisesti rekursiivisesti vielä ylemmille tasoille solmuhierarkiassa. Tästä heikkoudesta huolimatta perusindeksit (erityisesti teksti- ja arvolistoja) soveltuvat käytettäväksi kehittyneempien indeksimekanismien apuindekseinä.

Polkuindeksit käyttävät dokumentin polkuja haun perusyksikköinä solmujen sijaan. Kaikista hakuyksiköistä tallennetaan polku kokonaisuudessaan. Tällöin polkua ei tarvitse rakentaa haun aikana, kuten perusindekseillä. Polkuindeksit liittävät tyypillisesti hakusanoihin ne polut, joissa hakusana esiintyy. Joissakin indeksirakenteissa on mahdollista liittää myös polku hakusanoihin, mutta olennaista on, että rakenne on taulukkomainen — polkujen juuri- ja yläsolmuja toistetaan indeksin sisällä. fcode/puhelinluettelo/henkilot/henkilo/nimi["A. A."] -tyylinen kysely voidaan käsitellä vakioajassa, mutta jokerimerkkejä tai (rakenteesta riippuen) pelkkiä rakennekriteerejä sisältävien kyselyjen käsittely on hitaampaa [62, sivut 39-40]. Esimerkkinä polkuindeksistä käsitellään elementtipaikanninmalli (*element locator scheme*) [62, sivut 47-50], jonka alunperin esittivät

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

Taulukko 2: Elementtilista

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

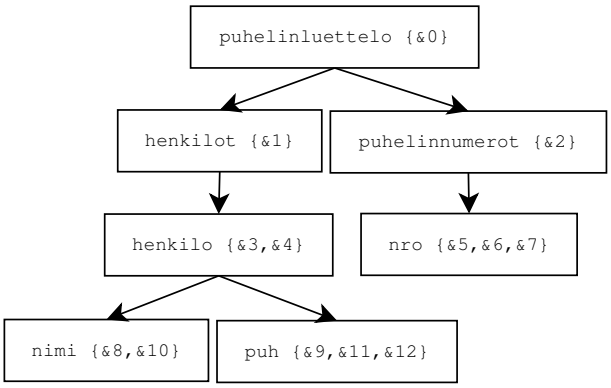
Taulukko 3: Käännetty polkulista

Sacks-Davis *et al.* [51]. Indeksi on toteutettu assosiaatiotauluna, jossa termeihin liitetään solmujen paikallisilla järjestysnumeroilla varustetut polut. Koska malli ei tarvitse solmujen tunnisteita, se on toteutettavissa helposti tapauksessa, jossa XML-dokumentit ovat tiedostojärjestelmässä (tällöin polkuihin on liitettävä myös tiedoston tunniste). Malli noudattaa PTN-luokittelussa{PT}-tasoa. Esimerkki elementtipaikanninmallin mukaisesta ”käännetystä” polkulistasta on esitetty taulukossa 3.

Puuindeksit (tai navigationaaliset indeksit) ovat rakenteisista indeksirakenteista mutkikkaimpia, mutta samalla ilmaisuvoimaltaan, suorituskyvyltään ja yleensä myös tilavaativuudeltaan parhaita. Niiden tietorakenteena on suunnattu graafi tai puu, joka noudattaa muodoltaan dokumenttikokoelman skeemaa ja on näin dokumentin rakenteinen lyhennelmä (tai deskriptiivinen skeema). Tietokannan solmujen tiedot voidaan merkitä suhteessa tähän rakenteeseen, jolloin polkuindeksien kaltaisia moneen kertaan merkittyjä polkuja ei tule. Lisäksi kyselyn rakenteinen osa voidaan yleensä sovittaa dokumentteihin puun syvyyden vaatimassa ajassa [62, sivut 39-46]. Puuindeksit on yleensä suunniteltu puhtaasti rakenteisia kyselyjä silmälläpitäen. Tekstihakua varten puuindeksin apuna voidaan käyttää esim. käännettyä solmulistaa, jolloin kyselyä suoritettaessa puuindeksin ja käännetyn solmulistan tulokset on yhdistettävä (ns. *content/structure-join*) [62, sivut 28-29].

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

jotka täsmäävät solmua vastaavaan polkuun. DataGuide ei ole yksikäsitteinen, vaan se voi olla mikä tahansa dokumentin todellista skeemaa vastaavan epädeterministisen äärellisen automaatin deterministinen esitys (tulkittaessa DataGuide automaattina tilat vastaavat elementtityyppejä ja siirtymät mahdollisuutta tietyn tyyppiseen alielementtiin) [35]. Jos dokumenttitietokanta ei ole puu (sisältää syklejä), DataGuiden solmuja täytyy kopioida huomioiden vaihtoehtoiset reitit. Pahimmassa tapauksessa tämä voi johtaa eksponentiaaliseen graafin koon kasvuun, mutta käytännössä DataGuide on havaittu toimivaksi erilaisilla dokumenttikokoelmilla [62, sivut 69-75]. PTN-luokittelussa DataGuide on {P,PN}-tasolla [63]. Kuvassa 4 on esitetty puhelinnumeroesimerkin rakenne DataGuide-muodossa.



Kuva 4: Puhelinnumeroesimerkin rakenne DataGuide-muodossa.

Kehittyneimmät puuindeksirakenteet sisältävät rakennetietojen lisäksi myös tietoja solmujen sisällöstä, jolloin hakuvaiheessa ei tarvitse tehdä rakenne- ja sisältötulosten liitosta. Tällaisia rakenteita ovat mm. Weigel:n *et al.* Content-Aware DataGuide [63], BUS-hakumallin indeksi sekä IndexFabric-tietorakenne [62, sivut 28-29]. Niiden tarkempi käsittely sivuutetaan.

4.4.2 Rakenteinen haku

Sacks-Davis *et al.* [51] listaavat perusteellisesti vaatimuksia, joita XML (artikkelin kirjoituksen aikana SGML) -kyselykieli voisi täyttää:

1. Sanahaku (kokotekstihaku)
2. Järjestetyt hakutulokset.
3. Kysely rajoitettu dokumentin osiin (esim. XPath-polulla ja termeillä rajattu kysely).
4. Dokumentin osien (alipuiden) haku (esim. datakeskeisillä kielillä).
5. Kysely rakenteen mukaan (esim. XPath-lausekkeet, ei termejä).
6. Samanaikainen haku monesta eri dokumenttityypistä.
7. Haku attribuuttien tai muiden XML-spesifisten ominaisuuksien mukaan (esim. entiteetit).
8. Yhdistetty haku XML-dokumenttien lisäksi muista tietojoukoista (esim. ulkoinen metatieto).

Kokotekstihaku ja järjestetyt hakutulokset ovat perinteisesti kuuluneet tekstitiedonhakuun, kun taas kohdat 3-5 ovat tietokannoille ominaisia. Niiden yhdistelmä voidaan hoitaa indeksoinnin osalta puuindeksillä, jota on mahdollisesti täydennetty käännetyllä solmulistalla. Kohta 6 on

mahdollinen, jos koko dokumenttikokoelma mallinnetaan yhtenä graafina. Attribuutteja lukuunottamatta tämän kirjoittaja ei löytänyt kohtaan 7 soveltuvia kyselykieliä.

Bunemanin [10] mainitsemat lähestymistavat XML-kyselykielten määrittelyyn (SQL:n laajennus tai puolirakenteisen mallin soveltaminen) vastaa XML-pohjaisten kielten jakoa data- ja dokumenttikeskeisiin. Datakeskeisten dokumenttien käsittelyyn soveltuu esim. W3C:n XQuery-kieli [6], joka SQL:n tyyliin sallii XPath-kieltä monipuolisempien rakenteisten ja kenttäpohjaisten ehtojen muodostamisen, mutta hakutulosten järjestely relevanssin mukaan tai kokotekstihaku eivät ole oletuksena tuettuja (kieleen on suunniteltu laajennuksia, mutta ne sivuutetaan tässä). Dokumenttikeskeisiä kieliä varten tarvitaan tiedonhaun käsitteiden pohjalta kehitetty hakukieli, joita voidaan edelleen jakaa rakenteeseen tai sisältöön keskittyviin [31]. Yksittäisiä XML-kyselykieliä on runsaasti, eikä tässä ole mahdollista keskittyä minkään yksittäisen kielen syntaksiin muuten kuin edellisessä luvussa käytettyjen XPath/hakusana -yhdistelmien tasolla (poikkeuksena hakumallit, joilla kysely voidaan antaa suoraan dokumenttifragmenttina). Sen sijaan käydään läpi yleisiä hakumalleja, termien ja elementtien painotustapoja ja relevenssin arviointitapoja. Kuvauksessa on käytetty alkuperäisten lähteiden lisäksi apuna Luk:n *et al.* [44] kattavaa kirjallisuuskatsausta.

Yksinkertainen, mutta ilmaisuvoimaltaan rajoitettu tapa rakenteiseen tiedonhakuun on käyttää Foxin [28] laajennettua vektorimallia. Standardissa vektorimallissa dokumentit ja kyselyt esitetään vektorina, laajennetussa ne ovat vektorimonikoita, joista kukin vektori kuvaa tiettyä näkökulmaa (näkökulmia voivat olla esim. sanamatriisi, linkkimatriisi, otsikkotekstit tai metatietokentät) dokumenttiin. Esimerkiksi sana- ja linkkimatriisia käyttävä dokumentti j voitaisiin esittää seuraavana vektoriparina:

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

Foxin artikkelissa indeksoinnin kohteena oli bibliografinen tietokanta, mutta ainakin Crouch *et al.* [18] ovat käyttäneet laajennettua vektorimallia myös XML-dokumenttikokoelman indeksointiin ja hakuun. Hakuvaiheessa jokaista vektoria verrataan mahdollisesti omalla etäisyysmitallaan ja tulokset yhdistetään painotettuna lineaarikombinaationa. Edellisen dokumenttivektorin tapauksessa täsmäytysfunktion R arvoksi kyselyllä q tulee

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

missä sim_1 ja sim_2 ovat etäisyysmitat termi- ja linkkivektoreita varten ja w-vektori sisältää komponenttien painot siten, että niiden summa on 1. Laajennettu vektorimalli soveltuu hyvin käytettäväksi kenttäindeksin kanssa, mutta periaatteessa muitakin indeksityyppejä voi käyttää, kunhan lopputuloksena saadaan useita verrattavia vektoreita. Esimerkiksi Luk *et al.* [44] mainitsevat, että laajennettu vektorimalli soveltuisi myös painotettuihin polkukyselyihin, jos käytettävissä on polkuindeksi. Ideaa voi yleistää eri lähteistä saatavien ja esitystavaltaan vaihtelevien etäisyysmittojen yhdistelyyn.

Varhaisimmat hierarkkista rakenteista tiedonhakua koskevat tulokset on esitetty hypertekstikirjallisuudessa: haulla pyritään määrittämään paras ”aloitusdokumentti” kokoelman selaamiseen. Myös rakenteinen dokumenttikokoelma voidaan tulkita hypertekstikantana. Tällöin varsinaisten linkkien sijaan sisäkkäiset elementit tulkitaan linkittyneiksi solmuiksi. Frissen [29] sovelluksessa

lääketieteellinen käsikirja jaettiin hierarkkisiin ”kortteihin”, joista kukin vastasi kirjan tiettyä kappaletta tai lukua. Jokainen kortti sisälsi edelleen linkit omiin ”alikortteihinsa”. Jokaisella kortilla oli oma otsikko ja leipätekstiä, mutta asiayhteyden puolesta kortit eivät olleet itsenäisiä dokumentteja (hypertekstimallin samankaltaisuus XML-dokumenttiin on selkeä, jos ”kortti” tulkitaan elementiksi). Kortit indeksoidaan tavallisella $tf \times idf$-tyylisellä painotuksella, mutta täsmäytyksessä otetaan huomioon korttien hierarkia. Olk. $\mathbf{S}_j$ dokumentin j alidokumenttien indeksijoukko. Tällöin täsmäytysfunktion R arvoksi kyselyllä q tulee

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

missä sim on etäisyysfunktio (esim. kosinimitta). Täsmäyttäminen aloitetaan dokumenttipuun lehdistä, jonka jälkeen edetään rekursiivisesti juuridokumenttiin asti. Jakotermin $|\mathbf{S}_j|$ tarkoitus on vähentää yksittäisen alidokumentin merkitystä, jos alidokumentteja on suuri määrä. Frissen mukaan tulokset pitäisi näyttää käyttäjälle muuten relevanssijärjestyksessä, mutta jos listaan on jo merkitty korkeammalle tasolle tämänhetkisen kortin ”ylikortti”, se voidaan jättää pois. Weigelin [62] terminologialla Frissen malli tarvitsee ”lattean” sanamatriisin lisäksi vanhempi/lapsi-indeksin täsmäytysfunktion laskennassa. Fuller [32] kehittää Frissen ideoita edelleen SGML-dokumenteille, mutta perusidea painojen laskemisesta on sama. Uudemmista XML-hakumalleista XIRQL [31] muistuttaa Frissen mallia siinä mielessä, että dokumentit on ositettu itsenäisiksi solmuiksi, joitsa kuhunkin on koottu sopivaksi katsottu dokumentin alipuu. Termit on tallennettu käänteiseen solmulistaan, joka on painotettu $tf \times idf$-mallilla siten, että jokaista solmua pidetään dokumenttina. XIRQL:n täsmäytys perustuu kuitenkin Frissen mallista täysin poiketen probabilistiseen päättelymaliin [44].

BUS (Bottom Up Scheme) [56] on indeksointi- ja hakumalli, joka sallii yksinkertaisina polkuina tehtävät rakenteiset kyselyt, termien painotukset ja järjestetyt hakutulokset. BUS-mallin indeksirakenne on puu, joka säilyttää alkuperäisen dokumentin rakenteen (jokaisessa indeksisolmussa on tieto siitä, mitä elementtityyppiä solmu vastaa). Termien esiintymät merkitään vain lehtisolmuihin. Lisäksi malli sisältää käännetyn solmulistan, joka liittää termit lehtisolmuihin [62, 60-64]. Käsiteltäessä kyselyä tutkitaan aluksi puuindeksistä, mitä elementtityyppejä palautetaan (esim. kyselyllä `/puhelinluettelo/henkilot/` palautetaan henkilösolmuja). Palautettavan elementtityypin alla olevat elementtityypit otetaan muistiin ja niitä verrataan solmulistaan. Tuloksena saadaan kyselyyn sopivat lehtisolmut. Löydettyjen solmujen määrä on rakenteinen vastine käännetylle dokumenttifrekvenssille, termifrekvenssit on merkitty suoraan indeksiin. Relevanssiarvio lasketaan Frissen mallin tapaan rekursiivisesti ”alhaalta ylös” ja painotuksia päivitetään jokaisella askeleella, esim. lehtisolmujen ylisolmun termifrekvenssi koostuu alisolmujensa termifrekvenssien summasta. Rekursiivista käsittelyä jatketaan, kunnes päästään palautettavan elementtityypin tasolle.

XPRES [64] on rakenteinen laajennus luvussa 4.2.2 mainitulle probabilistiselle hakumallille (laajennuksen johtaminen sivuutetaan). Dokumentin rakenne kuvataan käyttäjälle joukkona *rakenteisia rooleja*, joiden tarkkuus voi vaihdella karkeasta XIRQL-tyylisestä jaottelusta yksittäisiin XPath-polkuihin. Jokainen rooli voi sisältää useita dokumentin elementteja ja toisaalta jokainen elementti voi kuulua useampaan rooliin. Kyselyt esitetään $(termi, rooli)$-parien joukkona. XPRES käyttää käännettyä solmulistaa termien hakuun ja puuindeksiä rakenteen (elementtien ja

roolien) esittämiseen. Tietyn elementin e relevanssiarvo termistä t_i ja roolista s_k koostuvaan kyselyyn voidaan esittää seuraavasti:

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

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

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

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

Schliederin ja Meussin [54] puumalli yhdistää rakenteiset kyselyt todennäköisen relevanssin mukaan järjestettyihin tuloksiin ja dynaamisiin dokumentteihin, joiden granulariteetti riippuu kyselystä. Kyselyt annetaan XML-dokumenttifragmentteina. Dokumenttikokoelma tulkitaan yhdeksi puuksi, jonka alipuita sanotaan *loogisiksi dokumenteiksi*. Puun solmuja ovat dokumentin elementit, attribuutit ja varsinainen teksti. Merkkijonot paloitellaan niin, että jokainen sana muodostaa oman solmunsa. Palautettavan dokumentin tyyppi riippuu kyselypuun juuresta: jokainen looginen dokumentti, jolla on samanniminen juuri kuin kyselypuulla, voidaan palauttaa hakutuloksena. Näitä dokumentteja kutsutaan soveltuviksi dokumenteiksi (*admissible documents*).

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

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

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

Yhdistetyn mallin keskeinen käsite on *rakenteinen termi*, joka tarkoittaa mitä tahansa dokumentin tai kyselyn alipuuta. Rakenteinen termi on yleistys vektorimallin (vain yhteen sanaan viittaavalle) termille. Rakenteisten termien *esiintymät* kyselyssä ovat kaikki sen alipuut. Esiintymät dokumentissa ovat kaikki rakenteiset termit, jotka täsmäävät dokumenttiin. Rakenteisille termeille voidaan vektorimallin termien tapaan määritellä termifrekvenssi ja käänteinen dokumenttifrekvenssi. Olkoon T rakenteinen termi, D looginen dokumentti, $freq_T(D)$ T:n esiintymien määrä D:ssä ja $maxfreq(D)$ mielivaltaisen rakenteisen termin suurin esiintymien määrä D:ssä. Termifrekvenssi määritellään kaavalla

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

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

$$idf_T^t = \log \frac{|D^t|}{n_T} + 1.$$

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

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

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

5 Yhteenveto

Puolirakenteinen tieto on joustava tietomalli tietokannoissa olevan rakenteisen tiedon ja rakenteettoman datan välimaastossa. Esimerkki puolirakenteisesta tiedosta ovat rakenteiset dokumentit, jotka sisältävät tekstin lisäksi rakennetietoja, linkkejä ja metatietoa. Rakenteisuus tarjoaa tiedonhakuun lisätietoa. Dokumenttien ja kyselyjen esittäminen on mutkikkampaaa verrattuna puhtaaseen tekstiin, mutta hakuja pystytään tekemään suuremmalla granulariteetilla ja tuloksina pystytään esittämään dokumenttien relevanteiksi havaitut osat. Linkkitiedon avulla voidaan arvioida dokumenttien samankaltaisuutta tai relevanssia. XML mahdollistaa monipuoliset indeksointirakenteet ja hakumallit, jotka vaihtelevat yksinkertaisista taulukoista ja polkulistoista puurakenteisiin. Kehittyneimmät indeksit pitävät kirjaa yksittäisten dokumenttisolmujen ilmentymistä rakennepuussa ja painottavat termejä ottaen huomioon niiden suhteelliset määrät elementeissä. Luonteva hakutapa on esittää kysely dokumenttifragmenttina, jolloin haku tehdään vertaamalla kyselyfragementin rakenteellista samanlaisuutta tulosdokumenttiin.

Tenttikysymykset

- K1: Mitä ominaisuuksia tiedonhakua tukevalla XML-kyselykielellä pitäisi olla?
- V: Katso luku 4.4.2. Esim. kokotekstihaku, haku polkulausekkailla, haku attribuuteilla, hakutulosten järjestäminen relevanssin mukaan, dokumentin relevantin osan palauttaminen, osittaiset täsmäytykset.

- K2: XML-indeksirakenteiden päätyypit ja tarvittavat tietorakenteet?
- V: Katso luku 4.4.1. Perusindeksit (jaoteltuna), polkuindeksit, puuindeksit.

Lähteet

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

[2] L. Aunimo: *Tekstifragmenttien välisen semanttisen samanlaisuuden tunnistaminen*. Master's thesis, Helsingin yliopisto, Yleisen kielitieteen laitos, 2002. http://ethesis.helsinki.fi/julkaisut/hum/yleis/pg/aunimo/.

[3] R. Baeza-Yates and B. Ribeiro-Neto: *Modern Information Retrieval*. Addison-Wesley, 1999.

[4] T. Berners-Lee: *Www: Past, present, and future*. Computer, 29(10):69–77, 1996.

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

[6] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon: *Xquery 1.0: An xml query language, w3c working draft*. Techn. rep., W3C, 2004. http://www.w3.org/TR/2004/WD-xquery-20041029/.

[7] N. Bradley: *The XML Companion*. Addison-Wesley, 2000.

[8] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau: *Extensible markup language (xml) 1.0 (third edition), w3c recommendation*. Techn. rep., W3C, 2004. http://www.w3.org/TR/2004/REC-xml-20040204.

[9] S. Brin and L. Page: *The anatomy of a large-scale hypertextual web search engine*. In Enslow, Jr., P. H. and Ellis [26], pp. 107–117. http://decweb.ethz.ch/WWW7/1921/com1921.htm.

[10] P. Buneman: *Semistructured data*. In A. Mendelzon and Z. M. Özsoyoglu (eds.): *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 117–121. ACM Press, 1997. http://doi.acm.org/10.1145/263661.263675.

[11] S. Chakrabarti: *Mining the web – Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.

[12] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg: *Automatic resource compilation by analyzing hyperlink structure and associated text*. In Enslow, Jr., P. H. and Ellis [26], pp. 65–74. http://decweb.ethz.ch/WWW7/1898/com1898.htm.

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

[14] J. Clark and S. DeRose: *Xml path language (xpath) version 1.0, w3c recommendation*. Techn. rep., W3C, 1999. http://www.w3.org/TR/xpath.

[15] W. S. Cooper: *The formalism of probability theory in ir: a foundation or an encumbrance?* In W. B. Croft and C. J. van Rijsbergen (eds.): *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 242–247. Springer, 1994. http://portal.acm.org/citation.cfm?id=188562.

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

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

[18] C. Crouch, S. Apte, and H. Bapat: *An approach to structured retrieval based on the extended vector model*. In N. Fuhr, S. Malik, and M. Lalmas (eds.): *INEX 2003 Workshop Proceedings*, pp. 89–93. INEX, 2003. http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf.

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

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

[21] S. DeRose, E. Maler, and R. D. Jr.: *Xml pointer language (xpointer) version 1.0, w3c candidate recommendation*. Techn. rep., W3C, 2001. http://www.w3.org/TR/2001/CR-xptr-20010911/.

[22] S. DeRose, E. Maler, and D. Orchard: *Xml linking language (xlink) version 1.0, w3c recommendation*. Techn. rep., W3C, 2001. http://www.w3.org/TR/xlink.

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

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

[25] R. A. Elmasri and S. Navathe: *Fundamentals of Database Systems*. Addison-Wesley, 2000.

[26] P. H. Enslow, Jr. and A. Ellis (eds.): *Proceedings of the seventh international conference on World Wide Web 7*. Elsevier, 1998.

[27] D. C. Fallside: *Xml schema part 0: Primer*. Techn. rep., W3C, 2001. http://www.w3.org/TR/xmlschema-0/.

[28] E. A. Fox, G. L. Nunn, and W. C. Lee: *Coefficients of combining concept classes in a collection*. In Y. Chiaramella (ed.): *Proceedings of the 11th annual international ACM SIGIR*

conference on Research and development in information retrieval, pp. 291–307. ACM Press, 1988. http://doi.acm.org/10.1145/62437.62465.

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

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

[31] N. Fuhr and K. Großjohann: *Xirql: a query language for information retrieval in xml documents*. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel (eds.): *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 172–180. ACM Press, 2001. http://doi.acm.org/10.1145/383952.383985.

[32] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson: *Structured answers for a large structured document collection*. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 204–213. ACM Press, 1993. http://doi.acm.org/10.1145/160688.160720.

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

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

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

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

[37] D. Hand, H. Mannila, and P. Smyth: *Principles of Data Mining*. MIT Press, 2001.

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

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

[40] J. M. Kleinberg: *Authoritative sources in a hyperlinked environment*. In H. Karloff (ed.): *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pp. 668–677. Society for Industrial and Applied Mathematics, 1998. http://portal.acm.org/citation.cfm?id=315045.

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

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

[43] O. Lassila and R. R. Swick: *Resource description framework (rdf) model and syntax specification, w3c recommendation*. Techn. rep., W3C, 1999. http://www.w3.org/TR/REC-rdf-syntax/.

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

[45] Y. Maarek, D. Berry, and G. Kaiser: *An information retrieval approach for automatically constructing software libraries*. IEEE Transactions on Software Engineering, 17(8):800–813, 1991. http://dx.doi.org/10.1109/32.83915.

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

[47] T. H. Nelson: *Structure, tradition and possibility*. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pp. 1–1. ACM Press, 2003. http://doi.acm.org/10.1145/900051.900053.

[48] C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel, and L. Seligman (eds.): *Proceedings of the eleventh international conference on Information and knowledge management*. ACM Press, 2002.

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

[50] A. Popescul, L. H. Ungar, G. W. Flake, S. Lawrence, and C. L. Giles: *Clustering and identifying temporal trends in document databases*. In *Proceedings of the IEEE Advances in Digital Libraries 2000*, p. 173. IEEE Computer Society, 2000. http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=848380&isnumber=18432&punumber=6863&k2dockey=848380@ieeecnfs.

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

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

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

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

[55] U. Shah, T. Finin, and A. Joshi: *Information retrieval on the semantic web*. In Nicholas, C. *et al.* [48], pp. 461–468. http://doi.acm.org/10.1145/584792.584868.

[56] D. Shin, H. Jang, and H. Jin: *Bus: an effective indexing and retrieval scheme in structured documents*. In I. Witten, R. Akscyn, and F. M. Shipman, III (eds.): *Proceedings of the third ACM conference on Digital libraries*, pp. 235–243. ACM Press, 1998. http://doi.acm.org/10.1145/276675.276702.

[57] A. Strehl: *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, 2002. http://strehl.com/download/strehl-phd.pdf.

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

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

[60] C. J. van Rijsbergen: *Information Retrieval*. London: Butterworths, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[61] K. Wang and H. Liu: *Discovering typical structures of documents: a road map approach*. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel (eds.): *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 146–154. ACM Press, 1998. http://doi.acm.org/10.1145/290941.290982.

[62] F. Weigel: *A survey of indexing techniques for semistructured documents, institute of computer science, lmu, munich*. Project thesis, 2002. http://www.pms.ifi.lmu.de/publikationen/#PA_Felix.Weigel.

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

[64] J. E. Wolff, H. Flörke, and A. B. Cremers: *Searching and browsing collections of structural information*. In *Proceedings of the IEEE Advances in Digital Libraries 2000*, p. 141. IEEE Computer Society, 2000. http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=848377&isnumber=18432&punumber=6863&k2dockey=848377@ieeecnfs.

[65] S. K. M. Wong, W. Ziarko, and P. C. N. Wong: *Generalized vector spaces model in information retrieval*. In J. M. Tague (ed.): *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 18–25. ACM Press, 1985. http://doi.acm.org/10.1145/253495.253524.