Moniarvoisen relevanssin hyödyntäminen XML-tiedonhakujen evaluoinnissa

Pro gradu -tutkielma
Mikko Saari (67578)
Tampereen yliopisto
Informaatiotutkimuksen laitos
2006

TAMPEREEN YLIOPISTO
Informaatiotutkimuksen laitos
SAARI, MIKKO: Moniarvoisen relevanssin hyödyntäminen XML-tiedonhakujen evaluoinnissa
Pro gradu -tutkielma, 64 s., 3 liites.
Informaatiotutkimus
Huhtikuu 2006

Tutkimus käsittelee moniarvoisen relevanssin käyttämistä XML-tiedonhaun kontekstissa. Erityisesti työ liittyy INEX-konferenssin (Initiative for the Evaluation of XML Retrieval) puitteissa tehtävään tiedonhaun evaluoinnin tutkimukseen ja sen ongelmiin. Työssä pureudutaan kahteen vakavimpaan XML-tiedonhaun evaluoinnissa havaittuun ongelmaan: tuloslistan päällekkäisyyteen ja ylikansoitetun saantikannan ongelmaan.

Näiden ongelmien luonne kuvataan INEXin kontekstissa. Ratkaisuksi ongelmiin esitellään erilaisia vaihtoehtoisia tiedonhaun evaluoinnin mittareita. Olennaisin käsiteltävä evaluoinnin apuväline on XCG-mitta, jonka toiminta kuvataan yksityiskohtaisesti sen taustalla olevaa kertynyt hyöty -mittaa myöten. Lisäksi esitellään työtä varten kehitetty XCG-mittaa käyttävä evaluointityökalu ja sillä saatuja evaluointituloksia, jotka havainnollistavat ongelmia ja niiden vaikutuksia tiedonhaun evaluoinnissa.

Evaluointityökalun tulosten perusteella voidaan todeta, että kertynyt hyöty -mittaan perustuva XCG-mitta toimii hyvin XML-tiedonhakujen evaluoinnissa. Tarkastellut ja työn puitteissa kehitetyt ratkaisuehdotukset päällekkäisyyden ja ylikansoitetun saantikannan ongelmiin toimivat ja antavat käyttäjälle mielekkäämpiä tuloksia XML-tiedonhakuja evaluoitaessa. Kehitetty evaluointityökalu toimii halutulla tavalla rankaisten käyttäjän näkökulmasta vähemmän hyödyllisiä päällekkäisiä tuloksia tuottavaa järjestelmää.

Avainsanat: tiedonhakujärjestelmät, arviointi, moniarvoinen relevanssi, XML

Sisällysluettelo

1 Johdanto ... 1
1.1 Tutkimuskysymykset ... 2
2 Tiedonhaun evaluointi ... 3
2.1 Tiedonhaun evaluoinnin konteksti ... 3
2.2 Mitä evaluoidaan? ... 3
2.3 Tiedonhaun tutkimuksen menetelmiä ... 4
2.3.1 Edellytykset evaluointitutkimukselle ... 4
2.3.2 Tiedonhaun tutkimuksen laboratoriomalli ... 6
2.4 Relevanssi ... 7
2.4.1 Relevanssin määritelmiä ... 7
2.4.2 Moniarvoinen relevanssi ... 9
2.5 Tiedonhaun tehokkuuden mittaaminen ... 11
2.5.1 Saanti ja tarkkuus ... 11
2.5.2 Tuloslistan järjestyksen merkitys: normalisoitu saanti ja muita samankaltaisia mittoja ... 13
2.5.3 Kertynyt hyöty ... 16
2.6 Moniarvoisen relevanssin merkitys tiedonhaun evaluoinnille ... 20
3 Rakenteiset dokumentit ... 21
3.1 Rakenteiset ja rakenteettomat dokumentit ... 21
3.2 Laajennettava merkintäkieli XML ... 22
3.2.1 Taustaa ... 22
3.2.2 XML-dokumentteihin liittyviä keskeisiä käsitteitä ... 23
4 XML tiedonhaussa ... 24
4.1 Initiative for the Evaluation of XML retrieval ... 24
4.2 Kyselyt ja hakuaiheet ... 25
4.3 Testikokoelma ... 27
4.4 Moniulotteisen relevanssin käsittely INEXissä ... 28
4.4.1 Relevanssin ulottuvuudet ... 28
4.4.2 Kvantisointifunktiot ... 30
4.5 Järjestelmien tehokkuuden mittarit ... 31
4.5.1 INEXissä käytetyt mittarit ... 31
4.5.2 Päällekkäisyysongelma ... 31
4.5.3 XCG-mittaan perustuva mittari ... 33
5 Evaluointityökalun kehitystyö ... 38

5.1 Lähtökohdat ... 38
5.1.1 Evaluointityökalun komponentit ... 38
5.1.2 INEXin dokumenttityyppimäärittelyt ... 40
5.2 Evaluointityökalun käytännön toteutus ... 42
5.2.1 Käyttöympäristö ... 42
5.2.2 Ohjelman rakenne ja toiminta ... 42
5.3 Ohjelman toteutukseen liittyviä ratkaisuja ... 44
5.3.1 Tuloslistan elementtien päällekkäisyys ... 44
5.3.2 Ideaalisaantikannan muodostaminen ... 48
5.3.3 Arvonvähennysalgoritmi ... 51
5.3.4 Esimerkkejä evaluointityökalun tuloksista ... 52
6 Johtopäätökset ... 56
6.1 XML-tiedonhakujen evaluoinnin ongelmat ... 56
6.1.1 Relevanssiarviot ja kaksiulotteinen relevanssi ... 56
6.1.2 Elementtien päällekkäisyys ... 58
6.1.3 Ratkaisut ongelmiin ... 58
6.2 Yhteenveto ... 60
7 Lähteet ... 61
8 Liitteet ... 65
Liite A: CAS-tyyppinen INEX-hakuaihe ... 65
Liite B: CO-tyyppinen INEX-hakuaihe ... 66
Liite C: TREXML_eval-ohjelman asetustiedosto ... 67

1 Johdanto

Noin kymmenen viimeisen vuoden ajan *XML* on ollut käsite, jonka merkitys ja johon kohdistuneet odotukset ovat kasvaneet tasaisesti. Kyseessä ei ole niinkään uusi idea kuin jo 1980-luvulla toteutetuista ideoista muovattu käyttökelpoinen tekniikka. XML:stä ja rakenteisista dokumenteista on hiljalleen kehittynyt ilmiö, johon on pakko tutustua ja ottaa kantaa, mikäli haluaa olla automaattisessa tietojenkäsittelyssä ajan tasalla.

XML:n suosio perustuu sen tarjoamiin uusiin mahdollisuuksiin. Myös tiedonhakututkimus on ollut kiinnostunut rakenteisista dokumenteista. Kyse on tavallaan perinteisten pitkään tutkittujen tekstitietokantojen laajennuksesta. Vuonna 2002 järjestettiin ensimmäinen INEX-konferenssi (*Initiative for the Evaluation of XML Retrieval*), tarkoituksenaan tarjota XML-tiedonhakua tutkiville tahoille mahdollisuus arvioida hakujärjestelmiensä tehokkuutta käyttäen yhtenäistä aineistoa ja yhteisiä arviointiperiaatteita (Gövert ja Kazai 2003).

Tiedonhaun näkökulmasta XML:n ja rakenteisen tekstin kiinnostavuus piilee siinä, että dokumentteja voidaan käsitellä isojen yhtenäisten tekstimassojen sijasta hienovaraisemmin niiden oman loogisen rakenteen mukaisesti. Perinteisessä tiedonhaussa vastauksena kyselyihin tiedonhakijoille annetaan kokonaisia dokumentteja; vaikka hakija etsisi täsmällistä vastausta tiukasti rajattuun kysymykseen, vastauksena on kokonainen artikkeli, josta vastaus löytyy jostain kohtaa.

XML-dokumentista voidaan sen sijaan palauttaa haluttaessa tarkasti se dokumentin osa, esimerkiksi kappale, joka vastaa kyselyyn. Myös kyselyjä laadittaessa voidaan hyödyntää dokumenttien rakennetta: haku voidaan kohdistaa esimerkiksi kuvateksteihin, otsikoihin tai muihin dokumenttiin merkittyihin elementteihin.

Toinen kyseenalaistettu perinteisen tiedonhaun ilmiö on relevanssin binäärinen arviointi (katso esimerkiksi Borlund 2000, 79). Binäärisessä arvioinnissa dokumentin relevanssi nähdään kaksiulotteisena: dokumentti joko on relevantti tai ei ole. Vaihtoehtona on moniarvoinen relevanssin arviointi, jossa dokumentin relevanssiarvo voi olla myös jotain täysin relevantin ja ei-relevantin väliltä. Vaikka moniarvoista relevanssia on tutkittu jo vuosikymmenten ajan, binäärinen tulkinta on hallinnut tiedonhakujärjestelmien evaluointiin liittyvää tutkimusta (Kekäläinen ja Järvelin 2002a, 11). Moniarvoinen

relevanssi tarjoaa kuitenkin binääristä paremmat mahdollisuudet tiedonhakujärjestelmien tarkempaan vertailuun.

Pyrin työssäni kartoittamaan molempia ilmiöitä. Esittelen sekä XML-tiedonhaun että moniarvoisen relevanssin taustaa ja yhdistän ne sitten XML-tiedonhakujärjestelmien evaluoinnissa, jossa käytetään moniarvoista ja -ulotteista relevanssia. Käytännön sovelluksena tiedonhaun evaluoinnista moniarvoisen relevanssin avulla esittelen kertynyt hyöty –mitan.

Tutkimuksen teoreettinen puoli perustuu moniarvoisen relevanssin ja XML-dokumenttien merkityksen tiedonhaun evaluoinnissa selvittämiseen. Perehdyn myös tiettyihin XML-tiedonhakuun keskittyneessä INEX-konferenssissa esiinnousseisiin ongelmiin ja esitän niihin oman ratkaisuni.

Kirjallisen työn taustalla vaikuttaa ohjelmointiprojekti, jonka tuloksena syntyi TREXML_eval-työkalu. Työkalulla voidaan evaluoida XML-tiedonhakuja käyttäen tässä työssä esiteltyä kertynyt hyöty –mittaa ja sen sovellusta XCG:tä. Käsittelen työssäni ohjelman suunnittelun ja toteutuksen yksityiskohtia ja esittelen ohjelmalla saavutettuja tuloksia.

1.1 Tutkimuskysymykset

Tutkimukseni tiivistyy seuraaviin kysymyksiin:

- Mitä ongelmia liittyy XML-tiedonhakujen evaluointiin?
- Kuinka näitä ongelmia voidaan ratkaista?
- Miten kertynyt hyöty –mittaa pitää soveltaa, jotta sitä voi käyttää XML-tiedonhakujen evaluointiin?

Tutkimukseni kontekstina ja materiaalina toimii XML-tiedonhakuun keskittynyt INEX-konferenssi. Perehdyn XML-tiedonhaun ongelmiin ja moniarvoisen relevanssin sovelluksiin kehittämällä evaluointityökalun, jolla voi evaluoida XML-tiedonhakuja käyttäen mittana moniarvoista relevanssia käyttävää kertynyttä hyötyä.

2 Tiedonhaun evaluointi

2.1 Tiedonhaun evaluoinnin konteksti

Tiedonhaun (*information retrieval*) prosessi on osa laajempaa toimintaa, *tiedonhankintaa* (*information seeking*). Siinä missä tiedonhankinta voi olla joko aktiivista toimintaa tai passiivisempaa vastaanottamista, tiedonhaku on aktiivista toimintaa, jossa käyttäjä pyrkii täyttämään *tiedontarpeensa*. (Järvelin 1995, 8.)

Tiedonhaku kohdistuu tyypillisesti *tietokantaan*. Tietokanta on kokoelma tiettyä kohdetta kuvaavia tietoja, joita tietojärjestelmä käyttää (Järvelin 1995, 11). Tiedonhakuun käytettävä ohjelmisto, joka toimii välittäjänä käyttäjän hakupyyntöjen ja tietokannan välillä on *tiedonhakujärjestelmä*.

Tämän tutkielman aihepiiri, tiedonhaun evaluointi, liittyy tiiviisti tiedonhakuun ja sen järjestelmäsuuntautuneeseen perinteeseen. Tarkastelun kohteena on yksinomaan tiedonhakujärjestelmän toiminta ja sen tehokkuuden arviointi. Evaluoinnin kohteeksi on määritelty järjestelmän toiminta siten, että oikeita käyttäjiä ei tarvita järjestelmien evaluoinnin osana.

2.2 Mitä evaluoidaan?

Tiedonhakua evaluoidessa tarkastellaan tiedonhaun tuloksellisuutta ja kustannuksia. Evaluointia voidaan tehdä makro- tai mikrotasolla. Makrotasolla tarkastellaan hakuprosessin kokonaisuutena tuottamia tuloksia, kuten saadun tiedon laatua ja määrää suhteessa kustannuksiin. Mikroevaluoinnissa tutkitaan, mitkä tekijät vaikuttavat hakuprosessiin. Evaluointi voi kohdistua sekä hakujärjestelmiin, hakuihin, tietokantoihin että hakijoihin. (Järvelin 1995, 48-49.)

Tässä tutkimuksessa keskitytään tiedonhakujärjestelmien suodatuskyvyn evaluoimiseen analysoimalla järjestelmien tuottamien tuloslistojen relevanssia. Eri järjestelmiä vertailtaessa käytetään samoja hakuja samasta tietokannasta, jolloin myös hakijan osuus jää tyystin pois. Ongelmanasettelu on tietoisesti tarkkaan rajattu; tutkimuksen kohteena ovat nimenomaan tiedonhakujärjestelmien ominaisuudet.

Ensimmäinen valinta tiedonhaun evaluointia tehtäessä kohdistuu siihen, mitä evaluoidaan. Tietokonepohjaisen tiedonhakujärjestelmän evaluointi jakautuu kuuteen eri tasoon (Saracevic 1995, 140-141):

1. Teknisellä tasolla tutkitaan laitteistoihin ja ohjelmistojen suorituskykyyn liittyviä kysymyksiä (luotettavuus, virhealttius, nopeus, joustavuus ja niin edelleen). Erityisesti tutkitaan hakumenetelmien ja -algoritmien laskennallista tehokkuutta.
2. Syötetasolla tutkitaan järjestelmän syötteisiin ja sisältöön liittyviä kysymyksiä. Kiinnostuksen kohteena voi olla esimerkiksi sisällön kattavuus.
3. Prosessitasolla tutkitaan, kuinka syötteet käsitellään. Tähän liittyy esimerkiksi algoritmien ja tekniikoiden suorituskyvyn arviointi.
4. Tulostasolla tutkitaan vuorovaikutusta järjestelmän kanssa ja järjestelmän tuottamia tulosteita. Tutkimuksen kohteena voivat olla esimerkiksi hakuprosessi, palaute ja tulokset.
5. Käyttö- ja käyttäjätasolla tutkitaan kysymyksiä järjestelmän soveltuvuudesta annettujen ongelmien ja tehtävien ratkaisemiseen.
6. Sosiaalisella tasolla tutkitaan ympäristövaikutuksia. Voidaan esimerkiksi tutkia, kuinka järjestelmä vaikuttaa tutkimustyöhön, tuottavuuteen tai päätöksentekoon.

Käytännössä tiedonhaun evaluointi keskittyy tutkimaan järjestelmää prosessitasolla: kuinka onnistuneesti tiedonhakujärjestelmän algoritmit palauttavat hyviä dokumentteja vastauksena käyttäjän kyselyihin? Oikeilla käyttäjillä ei ole kuitenkaan juurikaan tekemistä tutkimuksessa, vaan käyttäjän osuus on korvattu vakioiduilla kyselyillä ja hakuaiheilla.

2.3 Tiedonhaun tutkimuksen menetelmiä

2.3.1 Edellytykset evaluointitutkimukselle

Tiedonhakujärjestelmän evaluoinnille on olemassa tietyt vaatimukset (Saracevic 1995, 142):

- *Järjestelmä*, tai sen representaatio, ja *prosessi*. Esimerkiksi testikokoelma ja tietyt hakualgoritmit.

- *Kriteerit*, jotka kuvaavat järjestelmän tavoitteita. Tiedonhakututkimuksessa järjestelmän tehokkuuden kriteerinä on relevanttien dokumenttien palauttaminen.
- *Mitta*, joka perustuu kriteereihin, esimerkiksi saanti ja tarkkuus relevanttien dokumenttien palauttamisen osoittimena.
- *Mittausväline*, jolla kriteereiden täyttyminen havaitaan (relevanssiarviot) tai johdettuja mittoja (saanti ja tarkkuus) lasketaan.
- *Metodologia*, jolla mittaukset suoritetaan ja evaluointi tapahtuu.

Testikokoelmia käsittelen enemmän seuraavassa luvussa.

Tiedonhakututkimuksen pääasialliseksi kriteeriksi on vakiintunut relevanssi – nyt on kulunut jotakuinkin tasan 50 vuotta siitä, kun ajatus alunperin esitettiin. Muitakin ehdotuksia on esitetty, mutta ne eivät ole yleistyneet. Relevanssilla on puolensa kriteerinä, mutta toisaalta se on toivottoman monimuotoinen ilmiö, jota on tutkittu paljon ja jonka luonteesta ei edelleenkään vallitse selkeä yksimielisyys. (Saracevic 1995, 143.) Käsittelen relevanssia – sekä perinteistä että moniarvoista – tarkemmin luvussa 2.4.

Kun relevanssi oli valittu kriteeriksi, saannista ja tarkkuudesta tuli luonteva mitta relevanttien dokumenttien palauttamisen tehokkuuden mittaamiseen (saanti ja tarkkuus esitellään luvussa 2.5.1). Ne ovat ensisijaiset mitat useimmissa merkittävissä tiedonhakututkimuksissa. (Saracevic 1995, 143.) Saantiin ja tarkkuuteen liittyy kuitenkin erinäisiä ongelmia, jonka vuoksi vaihtoehtoisia mittoja on esitetty vuosien varrella. Tässä tutkimuksessa käsitellään yhtä vaihtoehtoista mittaa, joka soveltuu saantia ja tarkkuutta paremmin moniarvoisen relevanssin kanssa käytettäväksi.

Mitta tarvitsee jonkun mittarin, jolla sitä tutkitaan. Dokumenttien relevanssia tutkittaessa välineenä käytetään ihmisiä, jotka tekevät relevanssiarviot dokumenteista. Oikeita käyttäjiä käytetään harvoin; dokumenttien relevanssia arvioivat yleensä asiantuntijaraadit, joiden katsotaan edustavan käyttäjiä jollain tapaa. (Saracevic 1995, 144.)

Asiantuntija-arvioiden luotettavuuteen (kuten yleensä ihmisiä mittareina käyttäviin tutkimuksiin) liittyy ongelmia, jotka usein ohitetaan vähällä pohtimisella. Jos relevanssi on kriteeri ja saanti ja tarkkuus mittana, jonkun on arvioitava dokumenttien relevanssi;

helpointa on vain olettaa, että tämä joku arvioi relevanssin oikein ja edustavasti. Jos relevanssiarvioissa on vinoumaa, se koskee tasaisesti kaikkia evaluoituja dokumentteja. (Saracevic 1995, 144.)

Kun tarkastellaan sitä, kuinka tehokkaasti hakujärjestelmä palauttaa relevantteja dokumentteja, mittarina käytetään evaluointityökalua, joka analysoi hakujärjestelmän palauttamat dokumentit. Tämä mittaus edellyttää relevanssiarvioiden olemassaoloa.

Tiedonhakututkimusten metodologiaa on pohdittu ja kyseenalaistettu tasaisesti alusta pitäen. Tutkimuksiin liittyy paljon kysymyksiä, joihin ei ole selviä vastauksia. Kuinka edustavia kokoelmat ovat? Kuinka kyselyt luodaan, ovatko ne sopivia? Kuinka hakeminen tapahtuu? Onko se realistista? Kuinka tulokset saadaan? Kuka arvioi ja miten? Kuinka analyysi tehdään? Miten tilastollinen testaus tehdään, entä vertailut? Mitkä ovat johtopäätökset? Kuinka yleistettäviä ne ovat? (Saracevic 1995, 144.)

2.3.2 Tiedonhaun tutkimuksen laboratoriomalli

Tiedonhaun tutkimuksen laboratoriomalli on lähtöisin Cranfield II -projektista (Cleverdon 1967). Laboratoriomallin peruskomponentteihin kuuluu testikokoelma, jonka muodostavat dokumenttitietokanta, joukko hyvin määriteltyjä kyselyitä ja joukko relevanssiarvioita, jotka tunnistavat kuhunkin kyselyyn nähden aiherelevantit dokumentit. Testeissä tiedonhakujärjestelmiä arvioidaan sillä perusteella, kuinka hyvin ne pystyvät palauttamaan relevantteja dokumentteja vastauksina kyselyihin. (Robertson 1981, 11.)

Laboratoriomalli on suosittu, koska se mahdollistaa erilaisten ulkoisten muuttujien vaikutuksen minimoimisen. Tämä on tärkeää, mikäli halutaan johonkin tiettyyn kysymykseen mahdollisimman yksiselitteinen vastaus. Mikäli tutkimuksen tavoitteena on vastata kysymyksiin, jotka liittyvät suoraan todellisiin tiedonhakujärjestelmien suunnittelemisen ongelmiin, kannattaa testaus suorittaa mieluummin operationaalista järjestelmää käyttäen. (Robertson 1981, 11-12.)

Testikokoelmat olivat pitkään varsin pienikokoisia (esimerkiksi alkuperäinen Cranfield-kokoelma tai SMART-tutkimusten tietokannat), yleensä käytännön syistä. Pienet laboratoriokokoelmat tarjosivat tutkijoille paljon kontrollia ulkoisten muuttujien

karsimiseen, mutta herättivät epäilyksiä tulosten yleistettävyydestä isoihin todellisen elämän tietokantoihin. (Saracevic 1995, 142.)

Apuun tuli TREC, *Text Retrieval Conference*, jonka testitietokannat ovat merkittävästi laajempia kuin aikaisemmat kokoelmat. Kokoelman koon suhteen ei ole enää juurikaan valittamista; sen sijaan voidaan kysyä, onko TRECin dokumenttivalikoima edustava ja onko vaarallista, että valtaosa tekstitiedonhaun tutkimuksesta on keskittynyt nimenomaan TREC-aineistoon. TRECin merkitys on joka tapauksessa suuri. (Saracevic 1995, 142.)

Viime vuosina TREC-konferenssit ovat olleet tiedonhaun laboratoriomalliin perustuvan tutkimuksen pääfoorumi. TREC-konferenssien tavoitteena on edistää suuriin, yli miljoonan dokumentin testikokoelmiin perustuvaa tutkimusta ja lisätä keskustelua luomalla avoin foorumi tutkimusideoiden vaihtamiselle. Lisäksi halutaan nopeuttaa uuden teknologian siirtymistä laboratorioista kaupallisiin tuotteisiin ja parantaa asianmukaisten evaluointitekniikoiden saatavuutta. (Voorhees ja Harman 2001.)

TREC jakautuu alaosioihin, trackeihin, jotka keskittyvät erityyppisiin tiedonhakutehtäviin. TREC:n fokusalueita ovat muun muassa webtiedonhaku, videotiedonhaku, kieltenvälinen tiedonhaku, interaktiivinen tiedonhaku ja kysymyksiin vastaaminen. Tyypillisin ja perinteisin alue on kuitenkin *ad hoc* –tiedonhaku (TREC:n *ad hoc* –track ei tosin ole enää aktiivinen). Siinä taustaoletuksena on tyypillinen tiedonhakutilanne, tutkija tekemässä tiedonhakua kirjastossa. Hakujärjestelmä tietää, mikä haun kohteena oleva dokumenttijoukko on (kirjaston kokoelma), mutta haettava aihe voi olla mitä tahansa. Haku on siis kestoltaan lyhyt ja aiheeltaan mielivaltainen. Hyvin tyypillinen esimerkki tämäntyyppisestä hakutehtävästä on Internetin hakukoneilla tehdyt haut. (Voorhees ja Harman 2001.)

2.4 Relevanssi

2.4.1 Relevanssin määritelmiä

Moderni informaatiotiede syntyi käsittelemään informaatioräjähdyksen ongelmaa. Alan pioneerit, jotka kehittivät ensimmäisiä tiedonhakujärjestelmiä 1950-luvulla määrittelivät tiedonhaun päätavoitteeksi *relevantin* informaation löytämisen. Sama päämäärä hallitsee

tiedonhakututkimusta tänä päivänäkin. Relevanssi on informaatio-objektien hakemisen tehokkuuden arvioimisen ylin kriteeri. (Saracevic 1996, 201-202.)

Relevanssin määritteleminen ei ole aivan yksiselitteistä. Erilaisia näkökulmia relevanssiin on paljon, mutta seuraava jaottelu on kuitenkin varsin yleisesti hyväksytty:

- *Systeemi- eli algoritminen relevanssi*: Hakulauseen ja dokumentin välinen suhde, järjestelmän käyttämän algoritmin mukaan. Järjestelmän näkemys siitä, mitkä dokumentit ovat relevantteja käyttäjän esittämään hakulauseeseen nähden.
- *Aiherelevanssi*: Hakulauseen aiheen ja dokumenttien esittämien aiheiden välinen suhde. Sekä hakulause että dokumentit kuvaavat jotain aihetta; dokumentti on relevantti, jos se kuvaa samaa aihetta kuin hakulause.
- *Kognitiivinen relevanssi, pertinenssi*: Käyttäjän tietämyksen tilan ja kognitiivisen informaation ja dokumenttien välinen suhde. Dokumenttien sisältämän tiedon uutuusarvo ja laatu vaikuttavat niiden kognitiiviseen relevanssiin.
- *Tilannerelevanssi, hyödyllisyys*: Tilanteen, tehtävän tai ongelman ja dokumentin välinen suhde. Dokumentti on relevantti, mikäli se auttaa ongelmanratkaisussa, päätöksenteossa tai epävarmuuden vähentämisessä.
- *Motivationaalinen tai affektiivinen relevanssi*: Käyttäjän tarkoituksen, tavoitteiden tai motivaation ja dokumentin välinen suhde. Relevantti dokumentti edistää tyytyväisyyttä, menestystä tai suoriutumista. (Saracevic 1996, 212.)

Tässä tutkimuksessa relevanssi ymmärretään järjestelmiin keskittyneelle tiedonhakututkimukselle ominaiseen tapaan aiherelevanssina. Relevanssi nähdään tavallaan systeemin ominaisuutena, joka riippuu siitä, kuinka hyvin järjestelmä kuvaa, organisoi ja täsmäyttää dokumentteja kyselyihin (Saracevic 1996, 206).

Korfhage (1997, 192-193) esittää hieman toisenlaisen jäsennyksen relevanssille. Korfhage määrittelee relevanssin Saracevicin systeemi- ja aiherelevanssin tapaan vastauksena kysymykseen “kuinka hyvin dokumentti vastaa esitettyyn hakulauseeseen?”. Dokumentteja voitaisiin suhteuttaa myös tiedontarpeeseen tai sen pohjalta esitettyyn kysymykseen,

mutta hakulause on järjestelmiä evaluoitaessa tärkein. Suhteuttamista kysymykseen tai tiedontarpeeseen Korfhage nimittää pertinenssiksi.

Pertinenssin ja relevanssin rinnalle Korfhage nostaa vielä käyttökelpoisuuden. Dokumentin käyttökelpoisuus ei liity sen relevanssiin: dokumentti voi olla relevantti, käyttökelpoinen, molempia tai ei kumpaakaan. Käyttökelpoinen informaatio on sellaista, joka ei ole välttämättä relevanttia tai pertinenttiä, mutta jonka käyttäjä haluaa säilyttää siitä huolimatta. (Korfhage 1997, 193.)

2.4.2 Moniarvoinen relevanssi

Perinteisesti relevanssi on nähty binäärisenä ominaisuutena: löydetty dokumentti on joko relevantti tai epärelevantti (esimerkiksi Järvelin 1995, 55). Kaikki relevantit dokumentit eivät kuitenkaan ole käyttäjälle yhtä arvokkaita, vaan jaottelua on mahdollista tehdä esimerkiksi marginaalisesti relevantteihin, relevantteihin ja erittäin relevantteihin dokumentteihin. Ympäristössä, jossa kaikki vähänkään relevantit dokumentit (vaikka dokumentin hyödyllinen sisältö rajoittuisi vain yhteen lauseeseen) luokitellaan binäärisen relevanssiarvioinnin mukaan relevanteiksi, erot huolimattomien ja tehokkaiden hakumenetelmien välillä eivät näy evaluoinnissa (Järvelin ja Kekäläinen 2002, 423).

Nykyaikaisissa tiedonhakuympäristöissä ongelmana on usein tulosten paljous. Dokumenttien tulvan alle joutuvalle käyttäjälle tulisikin esittää ensin relevanteimmat dokumentit. Jotta tiedonhakujärjestelmät voisivat kehittyä tähän suuntaan, tarvitaan evaluointimenetelmä, joka palkitsee hakujärjestelmiä relevanteimpien dokumenttien esittämisestä ensin. (Järvelin ja Kekäläinen 2002, 422-423.)

Käytännön syistä käytetään yleensä korkeintaan viisiportaista relevanssiasteikkoa. Käyttäjien on vaikeaa tehdä eroa kovin moniportaisten relevanssitasojen välille. (Korfhage 1997, 194.) Tampereen yliopistossa on kehitetty seuraavanlainen neliportainen relevanssiasteikko (Sormunen 2002, 325):

(0) Dokumentti ei sisällä mitään informaatiota aiheesta.
(1) Dokumentti ainoastaan viittaa aiheeseen. Se ei sisällä muuta informaatiota kuin aiheen kuvaukset. Tyypillisesti aiheesta mainitaan yksi lause tai fakta.

(2) Dokumentti sisältää muutakin kuin vain aiheen kuvauksen, mutta käsittely ei ole kattavaa. Jos aihe on monitahoinen, vain joitain näkökulmia tai osia käsitellään. Tyypillinen laajuus on yksi tekstikappale, muutama lause tai joitain faktoja.
(3) Dokumentti käsittelee aiheen teemoja kattavasti. Monitahoisen aiheen kaikki tai ainakin useimmat näkökulmat käsitellään. Tyypillinen laajuus on useita tekstikappaleita, ainakin neljä lausetta tai faktaa.

Marginaalisesti relevantit dokumentit (taso 1) ovat niin heikosti relevantteja, että dokumentin hyöty käyttäjälle on lähes merkityksetön. Tason 2 *relevanttien* dokumenttien odotetaan sen sijaan tuovan jotain lisäinformaatiota käyttäjälle, jolla on jo jotain näkemystä aiheesta. Tason 3 *erittäin relevanttien* dokumenttien avulla on mahdollista saada hyvä ymmärrys aiheesta. (Sormunen 2002, 325.)

TREC:n webtiedonhaussa on käytetty kolmiportaista relevanssiasteikkoa (ei relevantti, relevantti, erittäin relevantti), mutta sen erottelukyky ei ole riittävä. Relevanttien, mutta mahdollisesti hyödyttömien (taso 1) ja relevanttien ja hyödyllisten (taso 2) dokumenttien välinen ero häviää kolmiportaista relevanssiasteikkoa käytettäessä. Kolmiportainen asteikko on edistysaskel binääriseen asteikkoon verrattuna, mutta ei sittenkään tarpeeksi joustava. (Sormunen 2002, 329.)

Sittemmin TRECin webtiedonhaussa on siirrytty neliportaiseen relevanssiasteikkoon, joka vastaa yllä esitetty jaottelua (TREC-2004 Web Track Guidelines 2004).

Ongelmaa korostaa TREC:n erittäin liberaali relevanssiarviointi. Tason 1 marginaalisesti relevantti dokumentti on TREC:n binäärisellä asteikolla relevantti. Ero siihen, että relevanteiksi lasketaan vain selvästi relevantit dokumentit eli tasot 2 ja 3 (esim. Sormunen 2000, 63), on huomattava. Tutkimuksessa, jossa tarkasteltiin 38 hakuaiheen ja noin 5700 dokumentin otosta ja arvioitiin niiden relevanssitasoa neliportaisella relevanssiasteikolla, jopa neljäsosa TREC:n arvioinnissa relevanteiksi määritellyistä dokumenteista arvioitiin uudelleen täysin irrelevanteiksi. Vain noin 40% relevanteista dokumenteista arvioitiin tasolle 2 tai 3. (Sormunen 2002, 326, 329.)

Pelkkä moniarvoinen relevanssiasteikko ei kuitenkaan riitä. Tarvitaan myös menetelmiä, jotka ottavat huomioon eritasoisen relevanssin. Usein analysointivaiheessa moniarvoinen relevanssi tiivistetään kahteen kategoriaan – relevantti ja ei relevantti – saannin ja tarkkuuden laskemista varten (Kekäläinen ja Järvelin 2002b, 1120).

2.5 Tiedonhaun tehokkuuden mittaaminen

Käyttökelpoisen tiedonhakujärjestelmän tulee palauttaa tehokkaasti dokumentteja, jotka vastaavat käyttäjän tiedontarvetta. Useimpien palautettujen dokumenttien tulisi siis olla käyttäjän mielestä hyödyllisiä tiedontarpeen kannalta. Tämä sopii lähtökohdaksi, mutta on turhan epämääräinen antaakseen pohjaa järjestelmien tehokkuuden mittaamiseen tai erilaisten järjestelmien vertailuun. (Korfhage 1997, 191.)

Käytännössä vertailujen tekemiseen tarvitaan mittoja, jotka kuvaavat tiedonhaun tuloksellisuutta. Erilaisia mittoja on paljon, mutta käytännössä suurin osa tutkimuksista käyttää *saantia ja tarkkuutta* jossain muodossa. Tuoreemmista vaihtoehdoista esitellään lisäksi *kertynyt hyöty*, joka huomioi paremmin dokumenttien arvon ja järjestyksen tuloslistalla.

2.5.1 Saanti ja tarkkuus

Saanti ja *tarkkuus* (*recall* ja *precision*) ovat perinteisiä tiedonhaun tuloksellisuuden mittoja. Saanti kuvaa hakutuloksen osumien (eli löydettyjen relevanttien) suhdetta kaikkiin relevantteihin dokumentteihin, eli sitä, missä määrin onnistuttiin löytämään tietokannan sisältämät relevantit dokumentit. Tarkkuus puolestaa kuvaa osumien suhdetta kaikkiin löydettyihin dokumentteihin, eli sitä, kuinka suuri osuus haun tuloksesta oli relevantteja dokumentteja. (Järvelin 1995, 55-56.)

Yleensä ottaen on hyvin vaikeaa saada yhdellä haulla sekä hyvää saantia että hyvää tarkkuutta. Arvojen suhde onkin useimmiten käänteinen. Ääritapaukset valottavat asiaa: jos haetaan vain yksi dokumentti, joka on relevantti, saadaan täydellinen tarkkuus, mutta saanti on useimmissa tapauksissa erittäin heikko. Jos taas haetaan tietokannan kaikki dokumentit, saadaan täydellinen saanti, mutta erittäin heikko tarkkuus. (Korfhage 1997, 195.)

Saanti ja tarkkuus esitetään usein käyränä, jossa kuvataan useiden hakujen keskimääräinen tarkkuus kiinteillä saantitasoilla (tyypillisesti 10 prosentin välein). Vertailemalla saanti/tarkkuus-käyriä voidaan helposti verrata eri tiedonhakujärjestelmien suorituskykyä. Suoritus on sitä parempi, mitä korkeampi tarkkuus saantitasoilla saavutetaan.

Niin suosittuja kuin saanti ja tarkkuus tiedonhaun tehokkuuden mittoina ovatkin, niiden käyttöön liittyy tiettyjä ongelmia. Tarkkuus voidaan laskea täsmällisesti, mutta saantia ei: on mahdotonta tietää tietokannan relevanttien dokumenttien määrää. Ongelma on ratkaistu erilaisissa testikokoelmissa, mutta käytännön ympäristössä saannin tarkka laskeminen on mahdotonta. (Korfhage 1997, 197.)

Testikokoelmissa voidaan etsiä kaikki tiettyyn hakuaiheeseen liittyvät relevantit dokumentit, jolloin pystytään laskemaan absoluuttinen saanti. Testikokoelmien laatiminen on kallista ja vaivalloista ja lisäksi niiden käyttöön liittyy muita ongelmia; monet testikokoelmat ovat esimerkiksi merkittävästi operatiivisia järjestelmiä pienempiä.

Saannin ja tarkkuuden merkitys käyttäjille on sitä paitsi epäselvä; jos ne eivät ole tärkeitä, ne ovat huonoja osoittimia tiedonhaun tehokkuudesta. Useimmat käyttäjät arvostavat korkeaa tarkkuutta ja roskatonta tulosjoukkoa, mutta korkean saannin merkitys on huomattavasti vähäisempi. Tyypillisinä esimerkkeinä korkeista saantivaatimuksista mainitaan usein oikeustapauksen ennakkotapauksia tutkiva asianajaja ja patentteihin liittyvä tiedonhaku. Nämä tapaukset ovat varsin kaukana arkipäivän tiedonhakutilanteista. (Korfhage 1997, 195.)

Useimmissa tapauksissa käyttäjä haluaa varsin pienen joukon hakemaansa aihetta käsitteleviä dokumentteja. Tämä tarve korostuu entisestään Internet-tiedonhaussa, jossa aiheeseen liittyviä dokumentteja voi helposti olla tuhansia. Kukaan ei pysty lukemaan sellaista määrää dokumentteja läpi. Siksi on tärkeää, että parhaat tulokset löytyvät heti tuloslistan kärjestä. Tätä ulottuvuutta binäärinen saanti ja tarkkuus eivät kuitenkaan huomioi; järjestelmä voi saada testissä hyvät pisteet, vaikka tuloslistan kärjessä olisi marginaalisesti relevantteja dokumentteja, joista ei ole käyttäjälle juuri mitään iloa.

Saantia ja tarkkuutta käytettäessä voidaan huomioida eri relevanssitasot jossain määrin. Perinteisesti relevanssitasot on muunnettu binääriseksi relevanssiksi. Sen voi tehdä liberaalisti (tasot 1, 2 ja 3 ovat relevantteja, taso 0 ei) tai tiukemmin (esim. tasot 2 ja 3 ovat relevantteja, tasot 0 ja 1 eivät).

Toisaalta voidaan tehdä erilliset saantikannat eri relevanssitasoille ja tutkia saanti-tarkkuuskäyriä eri relevanssitasoilla (Kekäläinen ja Järvelin 2002b, 1122-1123). Tällöin voidaan huomata esimerkiksi hakujärjestelmien kyky noutaa erittäin relevantteja dokumentteja; tämä erinomainen ominaisuus saattaa tavallisessa tarkastelussa peittyä huonompaan suoritukseen marginaalisissa dokumenteissa. Käyttäjän näkökulmasta kyky noutaa erittäin relevantteja dokumentteja tehokkaasti on kuitenkin huomattavasti hyödyllisempi kuin tehokkuus marginaalisten dokumenttien kohdalla.

2.5.2 Tuloslistan järjestyksen merkitys: normalisoitu saanti ja muita samankaltaisia mittoja

Hakujärjestelmät palauttavat tulokset käyttäjille tyypillisesti järjestettynä listana, jota käyttäjä selaa alusta aloittaen. Tyypillisesti tätä järjestystä ei oteta hakujärjestelmän suorituskykyä arvioitaessa huomioon. Dokumenttien arvo riippuu siitä, missä vaiheessa tuloslistaa ne esiintyvät: mitä kauempana relevantti dokumentti tuloslistalla on, sitä pienempi on todennäköisyys, että se yleensä ottaen tulee luetuksi. Kahdesta samansisältöisestä dokumentista ensin luettu mielletään helposti paremmaksi. (Korfhage 1997, 208.)

Dokumenttien järjestyksen toinen vaikutus nousee relevanttien ja epärelevanttien dokumenttien sekoituksesta. Jos käyttäjä joutuu selaamaan suuren joukon epärelevantteja dokumentteja ennen relevantteja, käyttäjä turhautuu helposti. Toisaalta kauas tuloslistan kärjestä jäävä relevantti dokumentti jää helposti katsomatta. Tämä ei ole mielekästä, minkä vuoksi onkin tarpeen kehittää menetelmiä, jotka huomioivat dokumenttien järjestyksen tuloslistassa. (Korfhage 1997, 208, Järvelin ja Kekäläinen 2000, 422-423.)

Varhaisin tämänsuuntainen malli on *normalized recall*, *normalisoitu saanti* (Rocchio 1966). Siinä tiedonhaun edistymistä kuvataan portaittain kohoavalla kuvaajalla, joka nousee yhdellä yksiköllä aina, kun relevantti dokumentti löydetään. Ideaalitapauksessa

relevantit dokumentit ovat tietysti tuloslistan kärjessä, jolloin saantifunktion kuvaaja nousee nopeasti, kunnes saavuttaa maksimiarvonsa. (Korfhage 1997, 208.)

Muussa kuin ideaalitapauksessa kuvaaja nousee samaan tapaan, mutta hitaammin; kaikki relevantit dokumentit eivät todennäköisesti ole heti tulosjoukon alussa. Asettamalla kuvaajat samaan koordinaatistoon, voidaan nähdä, kuinka kauas ideaalisuorituksesta tutkittu järjestelmä on jäänyt. Numeroksi normalisoitu saanti voidaan muuttaa jakamalla kuvaajien pinta-alojen erotus arvolla $n_1(N - n_1)$, missä n_1 on relevanttien dokumenttien määrä ja N on dokumenttien yhteismäärä, ja vähentämällä saatu tulos luvusta 1. (Korfhage 1997, 208-209.)

Normalisoituun saantiin liittyy sama tietokannan relevanttien dokumenttien tuntemisen ongelma kuin saantiin yleensä. Lisäksi viimeisen relevantin dokumentin sijainnilla on liian suuri vaikutus saatuihin tuloksiin. Jos täydellinen saanti ei ole tärkeää käyttäjille, viimeisen relevantin dokumentin sijainnilla tulosjoukossa ei kuuluisi olla suurta merkitystä. (Korfhage 1997, 209.)

Samalla tavalla voidaan laskea myös normalisoitua tarkkuutta. Sekin kuitenkin vaatii kaikkien relevanttien dokumenttien tuntemista (Korfhage 1997, 209).

Normalisoitua saantia muistuttaa liukuva suhde, *sliding ratio* (Pollack 1968), joka eroaa normalisoidusta saannista kahdella tavalla: se perustuu painotettuihin relevanssiarvioihin ja jonkun tietyn dokumenttimäärän hakemiseen (eikä kaikkien relevanttien dokumenttien, kuten normalisoitu saanti). (Korfhage 1997, 209-210.)

Liukuvan suhteen painotetut relevanssiarviot ovat käytännössä yksi toteutus moniarvoisesta relevanssista. Luonnollisesti ideaalijärjestelmässä dokumentit saataisiin laskevassa järjestyksessä relevanssiarvon mukaan. Käytännössä näin ei tietenkään ole. (Korfhage 1997, 210.)

Jos tuloslistan N:n dokumentin painotetut relevanssiarvot ovat järjestyksessä $p_1, \ldots, p_N$ ja $P_1, \ldots, P_N$ on sama lista, mutta laskevassa relevanssijärjestyksessä, niin silloin liukuva suhde dokumentin n kohdalla saadaan seuraavalla kaavalla (Korfhage 1997, 210):

$$SR(n) = \frac{\sum_{i=1}^{n} p_i}{\sum_{i=1}^{n} P_i}$$

Liukuva suhde muistuttaa normalisoitua saantia, mutta on hieman parempi, kiitos painotettujen relevanssiarvojen. Liukuva suhde on lisäksi riippuvainen ainoastaan noudetuista dokumenteista. (Korfhage 1997, 211.)

Vielä hienostuneempi mittajärjestelmä on *tyytyväisyys/turhautuneisuus* (*satisfaction/frustration*) (Myaeng ja Korfhage 1990), joka perustuu kahteen yhteensopivaan mittaan. Tyytyväisyys huomioi ainoastaan relevantit dokumentit, turhautuneisuus vain ei-relevantit. Nämä voidaan sitten yhdistää yhteismitaksi usein eri tavoin. (Korfhage 1997, 211.)

Tyytyväisyys/turhautuneisuus-mitta tukee moniarvoista relevanssia, mutta edellyttää tavallaan moniarvoisen relevanssin rutistamista binääriseksi: relevanteille dokumenteille asetetaan joku kynnysarvo, jonka ylittävät dokumentit ovat relevantteja. Tyytyväisyyttä laskettaessa kaikkien irrelevanttien dokumenttien relevanssiarvoksi tulee nolla, turhautuneisuutta laskettaessa taas kaikki relevantit dokumentit lasketaan nolliksi. Nollattujen dokumenttien sijainti tuloslistalla kuitenkin huomioidaan. (Korfhage 1997, 211.)

Yhteistulosta laskettaessa mitat voidaan yhdistää eri tavoin. Kuinka paljon käyttäjä nauttii tyytyväisyydestään ja paljonko sietää turhautuneisuutta? Yksinkertaisin tapa yhdistää mitat on vähentää turhautuneisuuden arvo tyytyväisyydestä, mutta muitakin mahdollisuuksia voidaan kehittää tarvittaessa. (Korfhage 1997, 211-212.)

Otetaan esimerkiksi tilanne, jossa dokumentit on arvioitu relevanssiasteikolla 0-4. Dokumentit arvoltaan 0 ja 1 katsotaan irrelevanteiksi, arvot 2, 3 tai 4 taas relevanteiksi. Oletetaan, että järjestelmä palauttaa dokumentteja tällaisessa järjestyksessä: 3, 4, 2, 0, 2, 3, 3, 4, 1, 0. Tästä voidaan laskea tyytyväisyys laskemalla vain relevantin dokumentin rajan ylittävät dokumentit: 3, 4, 2, 0, 2, 3, 3, 4, 0, 0. Täysin irrelevantit dokumentit (arvo 0) pysyivät nollina, mutta myös hyvin vähän relevantti dokumentti (arvo 1) laskettiin nollana.

Näistä arvoista voitaisiin nyt piirtää tyytyväisyys kumulatiivisena kuvaajana. (Korfhage 1997, 212.)

Tuloslistan dokumenteista saadaan turhautuneisuuden arvo laskemalla vain irrelevantit dokumentit. Niiden arvoksi lasketaan pienin relevantti relevanssiarvo miinus dokumentin relevanssiarvo. Tuloksena saadaan: 0, 0, 0, 2, 0, 0, 0, 0, 1, 2. Tästä voidaan piirtää turhautuneisuuskuvaaja. (Korfhage 1997, 212.)

Yhteisarvo voitaisiin nyt siis laskea vaikkapa suoralla vähennyslaskulla kumulatiivisista summista: 3, 7, 9, 7, 9, 12, 15, 19, 18, 16. Tuloksia voidaan nyt verrata esimerkiksi ideaalitulokseen (4, 4, 3, 3, 3, 2, 2, 1, 0, 0). Haluttaessa yhteisarvoa laskettaessa sekä tyytyväisyyden että turhautuneisuuden arvoa voidaan korjata sopivalla kertoimilla. (Korfhage 1997, 212.)

Tyytyväisyys/turhautuneisuus on mielenkiintoinen mitta, joka huomioi myös moniarvoista relevanssia. Kaikki dokumentit eivät ole yhtä arvokkaita, vaan toisten saamisesta tulee enemmän iloa. Jako relevantteihin ja epärelevantteihin dokumentteihin viittaa kuitenkin binäärisen relevanssin maailmaan, vaikka toki tässä on enemmän vivahteita: täysin irrelevantit dokumentit tuottavat enemmän turhautuneisuutta kuin ei-niin-irrelevantit.

2.5.3 Kertynyt hyöty

Tyytyväisyysmitassa summattiin dokumenttien relevanssiarvoja. Jos käytetään binääristä relevanssia, tuloksena on käytännössä sama kuvaaja kuin normalisoidussa saannissa (kuvaaja nousee yhden askeleen jokaista löytynyttä relevanttia dokumenttia kohden). Jos binäärisen relevanssin sijaan käytetäänkin moniarvoista relevanssia, saadaan huomattavasti kiinnostavampia tuloksia (löydetyn relevanssin kuvaaja nousee jyrkemmin, kun kohdataan erittäin relevantteja dokumentteja). Tällöin ideaalitapauksessa saadaan ensin kaikista relevanteimmat dokumentit, sitten vähemmän relevantit ja vasta niiden jälkeen ei-relevantit.

Tälle ajatukselle perustuu tyytyväisyysmitan lisäksi Cumulated Gain -mitta (CG) eli kertynyt hyöty, joka on melko tuore vaihtoehto perinteisille tiedonhaun evaluoinnin mitoille. Kertynyt hyöty pyrkii arvioimaan käyttäjälle järjestetyn tuloslistan selailusta

kertyvää relevanssihyötyä ja ottaa huomioon dokumenttien järjestyksen. (Järvelin ja Kekäläinen 2002, 423-424.)

Laskettaessa kertynyttä hyötyä dokumenttien relevanssiarvoa käytetään saadun hyödyn mittana. Hyötyä lasketaan kumulatiivisesti yhteen aloittaen listan ensimmäiseltä sijalta. Näin järjestetty dokumenttilista muutetaan saadun hyödyn listaksi vaihtamalla dokumenttien tunnusnumerot niiden relevanssiarvoihin. Listasta saadaan hyötyvektori, jonka komponentit ovat yksittäisten dokumenttien relevanssiarvoja. (Järvelin ja Kekäläinen 2002, 424.)

Kertynyt hyöty -vektori CG muodostetaan hyötyvektorista G rekursiivisesti (Järvelin ja Kekäläinen 2002, 424-425):

$$CG[i] = \begin{cases} G[1], & \text{jos } i = 1 \\ CG[i-1] + G[i], & \text{muuten.} \end{cases}$$

Oletetaan neliportainen relevanssiasteikko 0-3 (3 tarkoittaa korkeaa relevanssia, 0 ei mitään relevanssia). Tällöin järjestetystä tuloslistasta voitaisiin saada esimerkiksi hyötyvektori G = <3, 2, 3, 0, 0, 1, 2, 2, 3, 0, ...>. Tästä hyötyvektorista saataisiin taas kertynyt hyöty -vektori CG = <3, 5, 8, 8, 8, 9, 11, 13, 16, 16, ...>. (Järvelin ja Kekäläinen 2002, 424-425.)

Hyötyvektorista voidaan piirtää kuvaaja, jossa kertynyt hyöty on y-akselilla ja dokumenttien määrä x-akselilla. Kahta käyrää voidaan verrata sekä pysty- että vaakasuoraan. Pystysuoraan vertaillessa voidaan verrata kertyneen hyödyn määrää, kun tietty määrä dokumentteja on luettu. Vaakasuoraan vertaillessa voidaan tarkastella, montako dokumenttia enemmän käyttäjän pitää tutkia huonompaa järjestelmää käyttäessään saavuttaakseen saman hyödyn kuin paremmalla järjestelmällä (Järvelin ja Kekäläinen 2000, 47).

Kertynyt alennettu hyöty

Dokumentin arvo käyttäjälle laskee sen mukaan, miten kaukana tuloslistalla dokumentti on. Mitä kauempana dokumentti listan alkupäästä on, sitä todennäköisempää on, ettei käyttäjä koskaan lue sitä. Kertyneeseen hyötyyn periaate voidaan toteuttaa sakottamalla

dokumentteja niiden sijoituksen mukaan. Tästä mitasta käytetään nimitystä Discounted Cumulated Gain eli kertynyt alennettu hyöty. (Järvelin ja Kekäläinen 2002, 425.)

Sopiva alennusfunktio on jakaminen sijaluvun logaritmilla, sillä se kasvaa tasaisesti, mutta ei liian jyrkästi (kuten esimerkiksi suora sijaluvulla jakaminen). Käytettäessä esimerkiksi kaksikantaista logaritmia, sijaluvulla 1024 oleva dokumentti saa vielä kymmenesosan arvostaan ($^2\log 1024 = 10$). Tällöin alennettu hyötyvektori muodostetaan

$$DCG[i] = \begin{cases} CG[i], & \text{jos } i < k \\ DCG[i-1] + G[i] / \log_k i, & \text{muuten.} \end{cases}$$

Kantalukua pienemmällä sijalla oleville dokumenteille alennusta ei tehdä, jotta dokumentti ei saa arvoonsa lisäystä $\log_k i$ -funktion arvon ollessa alle yksi. (Järvelin ja Kekäläinen 2002, 425.)

Muuntelemalla logaritmin kantalukua saadaan matalampia tai jyrkempiä alennuksia, joilla voidaan mallintaa erilaisten käyttäjien käyttäytymistä. Matalampi kantaluku saa dokumenttien arvon laskemaan nopeasti, mikä vastaa kärsimätöntä käyttäjää, joka ei jaksa lukea muita kuin listan kärjessä olevia dokumentteja. (Järvelin ja Kekäläinen 2002, 425.)

Voidaan tietysti todeta, että jos käyttäjä todella selaa tuloslistaa pitkälle ja löytää kauas kärjestä jääneitä relevantteja dokumentteja, hän saa niistä täyden arvon. Näin onkin, mikäli asiaa tarkastellaan käyttäjän näkökulmasta. DCG on kuitenkin ensisijaisesti työkalu järjestelmien vertailemiseen. Järjestelmä tai metodi, joka järjestää arvokkaimmat dokumentit listan kärkeen on yleisesti ottaen parempi, sillä suurin osa käyttäjistä ei kuitenkaan selaa tuloslistoja pitkälle. (Järvelin ja Kekäläinen 2000, 46.)

Erilaisten hyötyarvojen käyttäminen

Erittäin relevanttien dokumenttien arvoa kertynyttä hyötyä laskettaessa voidaan vaihdella. Korkeamman hyötyarvon käyttäminen (esim. 10, 100 tai jopa 1000) lisää erittäin relevanttien dokumenttien merkitystä kokonaishyötyä laskettaessa. Kun hyötyarvo on matalampi (esim. 1, 2, 3 tai 5), vähemmän relevantit dokumentit vaikuttavat tulokseen enemmän. (Voorhees 2001, 78.)

Mitä korkeampi ero relevanttien ja erittäin relevanttien dokumenttien arvossa on, sitä epävakaammaksi mittaus muuttuu, jos arviointi perustuu varsin pieneen määrään dokumentteja. Pienemmillä erittäin relevanttien dokumenttien arvoilla (3-5) DCG pystyy huomioimaan pisteissä kaiken relevanssi-informaation. Tällöin tulokset ovat vakaita, mutta palkitsevat silti järjestelmiä, jotka palauttavat erittäin relevantit dokumentit ensimmäisenä. (Voorhees 2001, 78.)

Yhtenä vahvuutenaan DCG tarjoaa kaksi muuttujaa, joita vaihtelemalla voidaan mallintaa erilaisia tilanteita. Kasvattamalla enemmän ja vähemmän relevanttien dokumenttien hyötyarvojen välistä eroa, voidaan korostaa relevantimpien dokumenttien merkitystä. Tämä kuitenkin tekee tuloksista epävakaampia, kun yksittäisten arvokkaiden dokumenttien merkitys kasvaa. Käyttämällä tasaisempia pistearvoja saavutetaan vakaampia tuloksia.

Toisaalta mittauksissa voidaan korostaa tulosjoukon kärkipään dokumenttien merkitystä muuttamalla logaritmin kantalukua. Jos tutkitaan hakujärjestelmää, jonka käyttäjien tiedetään pitäytyvän tulosjoukon ensimmäisissä dokumenteissa (esimerkiksi tyypilliset webtiedonhaut), voidaan käyttää matalampaa kantalukua, jolloin kärkijoukon dokumentit saavat paremmat pisteet. Äärimmäisissä tapauksissa voidaan käyttää logaritmin sijasta suoraan dokumentin sijalukua sellaisenaan jakajana.

Normalisoitu kertynyt hyöty

Perinteiset saanti-tarkkuus-käyrät voidaan suhteuttaa ideaalisuoritukseen, 100% tarkkuuteen kaikilla saantitasoilla. Kertynyttä hyötyä ei ole samalla tavalla suhteutettu ideaalisuoritukseen, joten kahden CG- tai DCG-kuvaajan vertaileminen keskenään ei ole mahdollista, ellei niitä ole tuotettu samoissa olosuhteissa. (Järvelin ja Kekäläinen 2002, 426-427.)

Ideaalisuoritus voidaan kuitenkin mallintaa hyötyvektorina, jossa saantikannan dokumenttien relevanssiarvot on järjestetty laskevaan järjestykseen. Ideaalitapauksessa relevanteimmat dokumentit tulevat siis ensin ja vähiten relevantit viimeisenä. Aiemman esimerkin mukainen ideaalivektori olisi siis I = <3, 3, 3, 2, 2, 2, 1, 0, 0, 0, ...> ja siitä muodostettu ideaali kertynyt hyöty CG_I = <3, 6, 9, 11, 13, 15, 16, 16, 16, 16, ...> (Järvelin ja Kekäläinen 2002, 426-427.)

Kun ideaalivektori on muodostettu, voidaan kertynyt hyöty normalisoida suhteessa ideaalitapaukseen. Normalisoitu hyötyvektori muodostetaan jakamalla hyötyvektori vastaavalla ideaalivektorilla. Näin saadaan vektori, jossa arvo 1 kuvaa täydellistä suoritusta ja arvot välillä [0,1) osuutta täydellisestä suorituksesta, jonka kyseinen menetelmä on saavuttanut. Aikaisempia esimerkkejä käyttäen hyötyvektori CG:stä saataisiin normalisoitu kertynyt hyöty -vektori nCG = <1, 0,83, 0,89, 0,73, 0,62, 0,6, 0,69, 0,76, 0,89, 0,84,...>. (Järvelin ja Kekäläinen 2002, 427.)

2.6 Moniarvoisen relevanssin merkitys tiedonhaun evaluoinnille

Moniarvoisen relevanssin merkitys tiedonhakujen evaluoinnissa on sitä suurempi, mitä laajemmista aineistoista on kyse. Tehtäessä tiedonhakuja suurista kokoelmista, ei riitä, että tiedonhakujärjestelmä pystyy palauttamaan jollain tavalla relevantteja dokumentteja jotakuinkin hyvässä järjestyksessä.

Tiedonhakujärjestelmän kyky palauttaa erittäin relevantteja dokumentteja aivan tuloslistan kärjessä on hakijan ajankäytön ja vaivannäön kannalta hyvin tärkeä. Tämä kyky ei erotu, mikäli evaluoinnissa ei huomioida relevanttien dokumenttien vaihtelevaa relevanssitasoa ja dokumenttien järjestystä tuloslistoilla.

Pelkkää saantia ja tarkkuutta laskemalla ei siis päästä pitkälle. Saannin merkitystä tiedonhaun evaluoinnin kriteerinä on muutenkin syytä tarkastella kriittisesti – tarvitseeko kukaan lopulta suurta määrää relevantteja dokumentteja? Erityisesti Internetin kaltaisista hyvin laajoista tietomassoista haettaessa korostuu, ettei lopulta ole suurtakaan merkitystä sillä, onko tulosjoukossa 200 vai 500 relevanttia dokumenttia. Olennaista on pikemminkin se, mitä kymmenen ensimmäisen dokumentin joukossa on.

Kertynyt hyöty palvelee tässä suhteessa tiedonhakijaa oikeita asioita korostavana mittana. Erityisesti kertyneen alennetun hyödyn laskeminen soveltuu tähän tarkoitukseen hyvin, sillä se korostaa tuloslistan kärjen dokumenttien merkitystä vieläkin enemmän. Hakujärjestelmä, joka saa hyviä tuloksia kertynyttä hyötyä mitattaessa tuottaa selvää hyötyä käyttäjälle.

3 Rakenteiset dokumentit

3.1 Rakenteiset ja rakenteettomat dokumentit

Elektronisessa muodossa oleva informaatio on joko rakenteista tai rakenteetonta, useimmiten jonkin verran molempia. Tietokannan tietueet ovat rakenteista informaatiota, sillä tieto on jaettu kenttiin, joiden sisältämä informaatio on muodoltaan tarkoin määrätty. Rakenteiseen informaatioon voidaan kohdistaa täsmällisiä kyselyjä: tietokannasta voidaan esimerkiksi hakea kaikki vuoden 2001 jälkeen julkaistut artikkelit, jotka on julkaistu *Journal of the American Society for Information Science and Technology* -lehdessä.

Kokotekstitietokannassa dokumentin metatiedot (dokumenttia kuvaavat tiedot, kuten kirjoittaja, julkaisuvuosi, julkaisun tiedot ja niin edelleen) ovat usein erillisiin kenttiin järjestettyä rakenteista tietoa. Dokumentin sisältö on yhdessä kentässä ilman rakennetta: teksti on esitetty yksinkertaisena merkkijonona. Myös dokumentin abstrakti on rakenteetonta tietoa, tosin pieneen tekstikatkelmaan ei usein sisälly monimutkaista rakennetta.

Dokumenteilla on kuitenkin rakennetta, jota on mahdollista hyödyntää dokumentteja käsiteltäessä. Useimmat dokumentit jakautuvat jollain tavalla osiin: kappaleisiin, alalukuihin, lukuihin, osiin ja niin edelleen. Jos dokumentti esitetään pelkkänä merkkijonona, tämä rakenne ja sen mahdollisuudet tiedonhaussa menetetään.

Dokumentin rakenne voidaan kuvata käyttäen dokumentin kuvauskieltä, jolla dokumenttiin merkitään sen rakenneosat. Tutuin esimerkki tällaisesta kielestä on Internet-sivujen luomiseen käytettävä HTML *(Hypertext Markup Language),* jolla voidaan erottaa dokumentista muun muassa otsikoita ja kappaleita. HTML perustuu monipuolisempaan SGML:ään (*Standard Generalized Markup Language*), jolla pystytään kehittämään huomattavasti yksityiskohtaisempia dokumenttirakenteita.

Kun dokumentin rakenne on merkitty, dokumentin sisältämä informaatio voidaan varsin helposti esittää eri tavoin esimerkiksi tulostettuna ja ruudulla – Internetissä sama sisältö voidaan vaikkapa muotoilla eri lailla isolle kuvaruudulle ja matkapuhelimen pikkunäytölle. Myös tiedonhaussa saavutetaan etuja, kun hakuja voidaan kohdistaa tarkasti dokumentin haluttuihin elementteihin.

3.2 Laajennettava merkintäkieli XML

3.2.1 Taustaa

XML on lyhenne sanoista *Extensible Markup Language*, laajennettava merkintäkieli. XML on rajoitettu muoto SGML-kielestä. XML-kieltä käytetään kuvaamaan XML-dokumenteiksi kutsuttuja dataobjekteja. Vuonna 1996 World Wide Web Consortiumin XML Working Groupin kehittämän XML-kielen suunnittelun tavoitteina oli kehittää Internet-käyttöön sopiva merkintäkieli, joka on muun muassa monikäyttöinen, SGML-yhteensopiva, ihmisille luettava ja helppo käsitellä ja tuottaa. (Bray, Paoli, Sperberg-McQueen, Maler ja Yergeau 2004.)

XML:ää käytetään pääasiassa rakenteisten dokumenttien tuottamiseen. XML ei ole ohjelmointikieli, vaan Internet-sivujen luomiseen käytettävän HTML-kielen tapainen tageihin eli tunnisteisiin (oikeammin elementteihin; tageilla osoitetaan elementtien esiintyminen dokumentissa) ja niiden ominaisuuksiin perustuva dokumenttien kuvauskieli. XML:ää käyttäen kuvataan dokumentin rakenne; esimerkiksi lehtiartikkelin sisältävä XML-dokumentti voisi sisältää elementtejä kuten otsikko, ingressi, kappale, kirjoittaja ja niin edelleen (elementit voitaisiin esittää esimerkiksi tageilla <otsikko>, <ingressi> ja niin päin pois). (W3C Communications Team 2003.)

Siinä missä HTML-kielen tagit on ennalta määritelty ja rajattu tiettyihin merkityksiin, XML ei sisällä itsessään mitään merkitystä tageille. Sovellusten kehittäjät voivat luoda omia tagejaan ja tulkita niitä haluamallaan tavalla. Siitä juontaakin XML:n nimitys Extensible, laajennettava; kieli venyy käyttäjänsä tarpeisiin. XML onkin tarkkaan ottaen metakieli, jolla voi luoda uusia merkintäkieliä; XML vain määrittelee, kuinka näitä uusia kieliä voidaan luoda. (W3C Communications Team 2003.)

Idea ei ole uusi: XML:n taustalla oleva SGML-kieli kehitettiin 1960-luvulla ja sen standardi on vuodelta 1986. XML on huomattavasti SGML:ää säännönmukaisempi ja yksinkertaisempi, ja soveltuu siksi paremmin muihinkin kuin teknisiin dokumentaatioprojekteihin, joihin SGML:ää pääasiassa käytetään. (W3C Communications Team 2003.)

XML:n suosio perustuu vapauteen rajoittavista lisensseistä, alustariippumattomuuteen ja laajaan tukeen. XML on käyttäjälleen täysin ilmainen formaatti, joka toimii missä tahansa ympäristössä, jossa pystytään käsittelemään yksinkertaisia tekstidokumentteja; ne ovat tietynlainen pienin yhteinen nimittäjä tietojenkäsittelyssä. XML:n ympärille on rakentunut laaja kieltä käyttävä yhteisö, joten projektiin kuin projektiin löytyy ilmaisia työkaluja ja asiantuntevaa tukea. (W3C Communications Team 2003.)

XML-kielen syntaksi on kuvattu yksityiskohtaisesti World Wide Web Consortiumin suosituksessa (Bray ja muut 2004), joka on kielen virallinen ja normatiivinen määritelmä.

3.2.2 XML-dokumentteihin liittyviä keskeisiä käsitteitä

Tiedosto on *XML-dokumentti*, mikäli se on *hyvinmuodostettu (well-formed)*. Dokumentti on *hyvinmuodostettu*, mikäli se sisältää yhden juurielementin ja muut elementit alkavat ja päättyvät saman elementin sisällä, dokumentti noudattaa XML-määrittelyn mukaisia hyvinmuodostetun dokumentin rajoituksia ja jokainen sen sisältämä jäsennetty entiteetti on hyvinmuodostettu. (Bray ja muut 2004.)

XML-dokumentit muodostuvat *elementeistä*, joiden sisältöä rajaavat *alku-* ja *lopputagit* (esimerkiksi <elementti> … </elementti>). Jos elementti on tyhjä, se merkitään *tyhjän elementin tagilla* (esimerkiksi <tyhjäelementti />). Elementeillä on tyyppi (jota osoittaa sen nimi) ja mahdollisesti joukko *attribuutteja* (esimerkiksi <elementti attribuutti=”attribuutin arvo”>). (Bray ja muut 2004.)

Dokumentin looginen rakenne voidaan määritellä kirjoittamalla sille *dokumenttityypin määrittely* (*document-type definition* eli *DTD*). Se kuvaa dokumentin loogisen kieliopin, eli sen, millaisia elementtejä (tageja) dokumentissa voidaan käyttää, miten elementtejä voi käyttää toistensa sisällä ja millaisia attribuutteja elementit voivat saada. DTD:n avulla voidaan tarkistaa, onko XML-dokumentti validi eli noudattaako se DTD:n määrityksiä. (Bray ja muut 2004.)

4 XML tiedonhaussa

4.1 Initiative for the Evaluation of XML retrieval

XML-tiedonhakujärjestelmien määrän kasvaessa kasvaa myös tarve niiden evaluoimiselle. Perinteisen tiedonhakututkimuksen puolella hallitsevaksi menetelmäksi on noussut dokumenteista, kyselyistä ja relevanssiarvioista koostuvien testikokoelmien käyttäminen, ja sama menetelmä on omaksuttu myös XML-tiedonhaun tutkimukseen. (Kazai, Lalmas, Fuhr ja Gövert 2004, 552.)

Perinteiset tiedonhakututkimuksen testikokoelmat eivät kuitenkaan sovi sisältösuuntautuneen XML-tiedonhaun tutkimukseen: dokumenteissa ei ole XML-rakennetta ja relevanssiarviot on tehty kokonaisten dokumenttien perusteella. (Kazai ja muut 2004, 552.)

Perinteisiin testikokoelmiin liittyy myös tiettyjä olettamuksia, joita ei tavallisesti kyseenalaisteta (Gövert, Fuhr, Kazai ja Lalmas 2003, 2-3):

1. Dokumentit ovat itsenäisiä yksiköitä, eli dokumentin relevanssi on riippumaton muiden dokumenttien relevanssista.
2. Dokumentti on selvästi erottuva erillinen yksikkö.
3. Dokumentit ovat suunnilleen saman kokoisia. Tehtäessä tiettyjä mittauksia, käyttäjien oletetaan viettävän yhtä paljon aikaa kunkin dokumentin parissa.
4. Jos käyttäjälle annetaan järjestetty tuloslista, tämä katsoo dokumentit läpi järjestyksessä, kunnes lopettaa johonkin kohtaan. Ei-lineaarisia tuloslistoja ei siis huomioida.

Sisältösuuntautuneessa XML-tiedonhaussa nämä olettamukset eivät välttämättä pidä paikkaansa (Gövert ja muut 2003, 3):

1. Koska tiedonhaku tapahtuu dokumenttien elementtien tasolla, useita saman dokumentin elementtejä ei millään voi pitää itsenäisinä yksikköinä.
2. Kun sallitaan dokumenttien elementtien palauttaminen, pitää huomioida elementtien päällekkäisyys. Voidaan esimerkiksi hakea useista kappaleista koostuva osio ja joku näistä kappaleista. Palautettuja elementtejä ei siis voi aina pitää erillisinä yksiköinä.

3. Palautettujen elementtien koko tulisi huomioida, koska se voi vaihdella periaatteessa otsikosta kokonaiseen kirjaan asti.
4. Kun palautetaan useita elementtejä samasta dokumentista, tulosten järjestäminen lineaarisesti ei ole välttämättä mielekästä. Koska yksittäiset elementit ovat usein hyvin kontekstisidonnaisia, jatkuvat kontekstin vaihdokset tuloslistassa sekoittavat käyttäjää helposti. Tuloslistaa kannattaa siis todennäköisesti ryhmitellä jollain tavalla.

Initiative for the Evaluation of XML retrieval eli INEX perustettiin vuonna 2002 tavoitteenaan kehittää infrastruktuuria ja keinoja sisältösuuntautuneen XML-tiedonhaun evaluointiin, edellämainitut ongelmat huomioiden. Vuoden 2002 aikana kehitettiin IEEE Computer Societyn lehtien artikkeleista koostuva testikokoelma, 60 hakuaihetta ja relevanssiarviot hakuaiheiden tueksi. Lisäksi kehitettiin yhteinen arviointijärjestelmä, jolla osallistujaryhmien tiedonhakujärjestelmien tehokkuutta voitiin arvioida. (Gövert ja Kazai 2003.)

Evaluoinnin kohteeksi valittiin järjestelmien hakutehokkuuden arviointi. Määritelmän mukaan tehokas järjestelmä tyydyttää käyttäjän kyselystä sekä sisältöön että rakenteeseen liittyvät vaatimukset. Käytännössä järjestelmän siis odotetaan palauttavan tarkin (käsittelee mahdollisimman vähän muita aiheita) relevantti elementti, joka on riittävän kattava (käsittelee tarpeeksi haluttua aihetta). (Kazai ja muut 2004, 552.)

Suoritettavaksi hakutehtäväksi valittiin XML-dokumenttien TREC:issäkin käytetty ad-hoc-hakeminen. Osallistujaryhmät hakivat siis staattisesta dokumenttijoukosta tietoa erilaisilla vaihtuvilla kyselyillä. (Kazai ja muut 2004, 552.)

4.2 Kyselyt ja hakuaiheet

XML-tiedonhaun erityispiirteiden vuoksi on mahdollista käyttää monimutkaisempia kyselyitä kuin tavallisista tekstidokumenteista haettaessa. Kyselyt voivat hyödyntää dokumenttien rakennetta rajoittamaan kyselyitään. Tämän mahdollisuuden tulee heijastua järjestelmien evaluointiin käytettävissä kyselyissä. Toisaalta sisältösuuntautuneiden XML-tiedonhakujärjestelmien on tuettava myös kyselyitä, joissa ei määritellä rakenteellisia

ehtoja. Käyttäjäthän eivät välttämättä tunne haun kohteena olevien dokumenttien rakennetta. (Gövert ja Kazai 2003.)

Tämän johdosta INEXissä käytetään kahdenlaisia kyselyjä. Sisältö ja rakenne -kyselyissä (*Contents and structure*, CAS) on suoria viittauksia dokumenttien rakenteeseen. Sisältökyselyt (*Contents only*, CO) eivät huomioi dokumenttien rakennetta ja muistuttavat siten pintapuolisesti perinteisen tiedonhaun kyselyjä. (Gövert ja Kazai 2003.)

Hakuaiheet ovat konferenssin osallistujien tekemiä. Vuoden 2004 konferenssiin kultakin osallistujalta edellytettiin kolmea sisältökyselyä ja kolmea sisältö ja rakenne -kyselyä. Kyselyn rakentamisen prosessi alkaa yksinkertaisella kuvauksella haetusta tiedosta. Alustavien tiedonhakujen tulokset arvioidaan ja kyselyä jalostetaan tuloksena saatujen elementtien perusteella. (Sigurbjörnsson, Larsen, Lalmas ja Malik 2004, 215-216.)

Vaiheen tavoitteena on arvioida sata parasta elementtiä. Näiden perusteella kirjoitetaan kyselyn taustatarina, jonka tarkoitus on selittää, mikä erottaa relevantit ja irrelevantit elementit tässä kyselyaiheessa. Sen jälkeen voidaan laatia lopullinen XML-muotoinen kyselyaihe käytössä olevan dokumenttityypin määrittelyn mukaan.

```
<!ELEMENT inex_topic    (title, description, narrative, keywords)>
<!ELEMENT title         (#PCDATA>
<!ELEMENT description   (#PCDATA)>
<!ELEMENT narrative     (#PCDATA)>
<!ELEMENT keywords      (#PCDATA)>
<!ATTLIST inex_topic
  topic-id        CDATA #REQUIRED
  query-type      CDATA #REQUIRED
>
```

Kuvio 1: INEX-hakuaiheiden dokumenttityypin määrittely (Trotman ja Sigurbjörnsson 2004, 221).

Kuviossa 1 esitetään INEX-hakuaiheen dokumenttityypin määrittely, joka oli käytössä vuonna 2004. Hakuaihe koostuu neljästä pääelementistä, jotka ovat otsikko (title), aiheen kuvaus (description), tausta (narrative) ja avainsanat (keywords). (Trotman ja Sigurbjörnsson 2004, 221.)

Hakuaiheen otsikko on NEXI-kyselykielellä (*Narrow Extended XPath I*) esitetty kysely, tyypiltään joko pelkkä sisältö tai sisältö ja rakenne riippuen query-type-attribuutin arvosta. Aiheen kuvaus on parin lauseen mittainen luonnollisella kielellä esitetty määritelmä

käyttäjän tiedontarpeesta. Hakuaiheen tausta on yksityiskohtaisempi kuvaus aiheen taustalla olevasta tiedontarpeesta ja siitä, mikä tekee dokumentista tai elementistä relevantin haun kannalta. Avainsanaelementti listaa hakuun liittyviä mahdollisia hakutermejä. (Trotman ja Sigurbjörnsson 2004, 221.)

Liitteissä A ja B on esimerkit hakuaiheista. Liitteessä A on esimerkki sisältö ja rakenne -tyyppisestä hakuaiheesta, jossa hakulauseessa (”*//article[about(.//bb, Rumbaugh Jacobson Booch) and about(.//abs, formal methods logic)]//sec[about(., UML formal logic)]*”) on määritelty erilaisia viittauksia dokumentin XML-rakenteeseen. Artikkelin lähdeluettelossa (elementti *bb*) tulee olla viittaus Rumbaughiin, Jacobsoniin tai Boochiin ja abstraktissa (*abs*) pitää olla maininta formaalista metodilogiikasta. Käyttäjä on kiinnostunut saamaan tuloksena vain ne osiot (*sec*) artikkelista, joissa puhutaan formaalista UML-logiikasta. Tämä on luettavissa hakulauseesta, mutta hakuaiheen narrative-elementti taustoittaa informaatiotarpeen vielä perusteellisemmin.

Liitteen B esimerkki on pelkkä sisältö -tyyppinen hakuaihe. Siinä hakulauseessa (”*+"open standards" +"digital video" +"distance learning"*”) ei välitetä ollenkaan XML-dokumentin rakenteesta, vaan haetaan pelkästään yksinkertaisia fraaseja. Ne voivat esiintyä dokumentissa missä kohtaa tahansa, ja dokumentista voidaan palauttaa mikä relevantti osa tahansa.

4.3 Testikokoelma

INEXin testikokoelma koostuu tyypilliseen tapaan kolmesta osasta: joukosta dokumentteja, hakuaiheista ja relevanssiarvioista. INEXin käyttämän dokumenttikokoelman on lahjoittanut IEEE Computer Society. Aineisto koostuu 12107 kokotekstiartikkelista IEEE:n julkaisuista vuosilta 1995-2002. Artikkeleissa on keskimäärin 1532 XML-elementtiä kussakin. (Gövert ja Kazai 2003.)

Dokumentit on merkitty käyttäen yhteistä DTD:tä. Tyypillinen artikkeli muodostuu etuosasta (front matter, *fm*), vartalosta (body, *bdy*) ja loppuosasta (back matter, *bm*). Etuosassa on metadataa artikkelista: otsikko, tekijä, julkaisutiedot ja abstrakti. Artikkelin vartalo on jaettu osioihin (*sec*), alaosioihin (*ss1*) ja niiden alaosioihin (*ss2*). Nämä yksiköt alkavat otsikolla, jota seuraa joukko kappaleita (*p*). Artikkeleihin on lisäksi merkitty

viittauksia (lainauksia, tauluja, kuvioita), listoja ja asettelua (korostuksia, kursiivia ja niin edelleen). Loppuosassa on tyypillisesti lähdeluettelo ja tietoa artikkelin kirjoittajista. (Gövert ja Kazai 2003.)

4.4 Moniulotteisen relevanssin käsittely INEXissä

4.4.1 Relevanssin ulottuvuudet

Hakuaiheiden saantikantoja varten ehdotetut dokumentit arvioitiin INEX 2002:ssa kahden ulottuvuuden mukaan. *Aiherelevanssi* kuvaa, kuinka dokumentin elementin sisältämä informaatio tyydyttää hakuaiheen kuvaaman tiedontarpeen. *Elementin kattavuus* kuvaa, kuinka hyvin elementti on keskittynyt tiedontarpeeseen. (Gövert ja Kazai 2003.)

Elementtien relevanssin kaksi piirrettä on erotettu omiksi ulottuvuuksikseen kontrollin lisäämiseksi. Käsitys oli, että kaksiulotteinen ratkaisu tuottaa vakaampia relevanssiarvioita. Yksiulotteista relevanssia arvioitaessa on mahdotonta tietää, kuin arvioija painottaa eri ulottuvuuksia antaessaan yhden relevanssiarvon. (Kazai ja Lalmas 2005, 25.)

Jakoa puolustaa käytännön kokemus tutkimuksesta, jossa rakennettiin pieni testikokoelma XML-merkityistä Shakespearen näytelmistä. Arvioijia pyydettiin merkitsemään relevantit tekstikatkelmat korostusvärillä. Eri arvioijat merkitsivät hyvin erisuuruisia katkelmia relevanteiksi samoihin kyselyihin. Kattavuuden ja spesifisyyden merkitys vaihtelee siis henkilöltä toiselle paljon relevanssiarvioita tehtäessä. (Kazai ja Lalmas 2005, 26.)

Molemmat ulottuvuudet arvioitiin neliportaisella asteikolla. Relevanssin mittaamisessa käytettiin Tampereen yliopistossa kehitettyä asteikkoa (katso luku 2.4.2). Kattavuuden kohdalla asteikko oli ei käsittele aihetta (N; ei käsittele aihetta ollenkaan), liian suuri (L; käsittelee liian paljon muuta kuin aihetta), liian pieni (S; käsittelee pääasiassa aihetta, mutta on kooltaan liian pieni) ja täydellinen kattavuus (E; käsittelee pelkästään aihetta ja on kooltaan mielekkään kokoinen). (Gövert ja Kazai 2003.)

Moniasteista relevanssia tarvittiin kuvaamaan dokumenttien elementtien ja niiden alaelementtien suhteellista relevanssia. Dokumentin elementti voi esimerkiksi olla kattavampi kuin alaelementtinsä yhteensä, sillä se sisältää kaiken, mitä yksittäiset alaelementit sisältävät. Toisaalta alaelementti voi olla spesifimpi kuin yläelementti, koska se voi sisältää vähemmän turhaa informaatiota. (Kazai, Lalmas ja de Vries 2004a, 73.)

Tavoitteena oli pystyä erottamaan ja palkitsemaan järjestelmät, jotka pystyvät palauttamaan kaikista tarkimpia tuloksia. Kärjistettynä kysymys on siitä, palauttaako järjestelmä vastauksena kysymykseen kokonaisen tietosanakirjan vai pelkästään relevantin, kysymykseen vastaavan kappaleen. Ilman kattavuusulottuvuutta tulisi ongelmia, sillä aiherelevanssin kynnys on hyvin matala: jo aiheen mainitseminen tekstissä riittää alimman relevanssitason saavuttamiseen. (Kazai ja muut 2004, 553.)

INEX 2003:ssa ulottuvuudet nimettiin *kattavuudeksi (exhaustivity)* ja *spesifisyydeksi (specifity)* (Kazai, Lalmas ja de Vries 2004a, 73). Molemmat arvioidaan asteikolla 0-3. Kattavuus kertoo, kuinka laajasti elementti käsittelee haun aihetta (eli kuinka paljon siinä on relevanttia informaatiota), spesifisyys taas sen, kuinka fokusoitunut elementti on aiheeseensa (eli kuinka paljon siinä on irrelevanttia informaatiota) (Kazai ja Lalmas 2005, 25). Ulottuvuudet lyhennetään tyypillisesti *e* (kattavuus) ja *s* (spesifisyys).

INEXissä on noussut ajoittain argumenttejä yksinkertaisen binäärisen relevanssin puolesta. Koska INEXissä on evaluoitu hakutuloksia tiukan relevanssiluokituksen mukaan, jossa vain täysin kattavat ja täysin spesifit (3,3)-luokitetut elementit lasketaan relevanteiksi, muiden relevanssiarvioiden tekemistä on toisiaan kyseenalaistettu. (Kazai 2003.)

Ad-hoc-XML-tiedonhaun määritelmä kuitenkin on, että järjestelmien pitää löytää kaikki relevantti informaatio, ei vain erittäin relevanttia. Pelkästään (3,3)-elementteihin perustuva luokittelu ei siis riitä. Tehokkaasti erittäin relevantteja elementtejä löytävä järjestelmä ei välttämättä toimi hyvin laajaan saantiin perustuvassa hakutehtävässä. (Kazai 2003.)

4.4.2 Kvantisointifunktiot

Kaksiulotteinen relevanssi täytyy kuitenkin muuttaa yhdeksi arvoksi, jotta sitä voidaan käyttää tiedonhakujen arvionnissa. Muunnokseen käytetään kvantisointifunktiota, joka muuttaa kaksi arvoa yhdeksi relevanssipistearvoksi välille [0,1]. Erilaisia käyttäjiä mallintamaan voidaan käyttää erilaisia funktioita. (Kazai,Lalmas ja de Vries 2004a, 74.)

INEX 2003:ssa käytettiin kahta erilaista funktiota, tiukkaa ja yleistä. Tiukka funktio (f_{strict}) arvioi hakumenetelmiä sen mukaan, kuinka hyvin ne pystyvät noutamaan erittäin kattavia ja spesifejä elementtejä. Tämän funktion mallintamaa käyttäjää kiinnostavat vain erittäin relevantit elementit, joissa ei ole mitään turhaa. Yleinen funktio (f_{gen}) antaa dokumenteille pisteitä niiden relevanssin asteen mukaan ja mallintaa siten käyttäjän epätäydellisistä, mutta relevanteista elementeistä saamaa tyytyväisyyttä. (Kazai, Lalmas ja de Vries 2004a, 74.)

$$f_{strict}(e, s) = \begin{cases} 1 & \text{jos } e = 3 \text{ ja } s = 3, \\ 0 & \text{muuten.} \end{cases}$$

$$f_{gen}(e, s) = \begin{cases} 1 & \text{jos } (e, s) = (3, 3), \\ 0{,}75 & \text{jos } (e, s) \in \{(2, 3), (3, 2), (3, 1)\}, \\ 0{,}5 & \text{jos } (e, s) \in \{(1, 3), (2, 2), (2, 1)\}, \\ 0{,}25 & \text{jos } (e, s) \in \{(1, 2), (1, 1)\}, \\ 0 & \text{jos } (e, s) = (0, 0). \end{cases}$$

Yleinen kvantisointifunktio on kuitenkin hieman ongelmallinen. Se suosii kattavuutta enemmän kuin spesifisyyttä. Kattavimmat elementit ovat usein suuria, esimerkiksi artikkelitason elementtejä. Tällöin hakujärjestelmä voi saavuttaa kohtalaisen hyvät tulokset vain artikkeleja listaamalla. Se ei kuitenkaan ole mielekäs lopputulos XML-ympäristössä. Siksi Kazai, Lalmas ja de Vries ehdottavat *specificity-oriented generalised* -kvantisointia, joka suosii spesifisyyttä kattavuuden kustannuksella. (Kazai, Lalmas ja de Vries 2004a, 77.)

$$f_{sog}(e, s) = \begin{cases} 1 & \text{jos } (e, s) = (3, 3), \\ 0{,}9 & \text{jos } (e, s) = (2, 3), \\ 0{,}75 & \text{jos } (e, s) \in \{(3, 2), (1, 3)\}, \\ 0{,}5 & \text{jos } (e, s) = (2, 2), \\ 0{,}25 & \text{jos } (e, s) \in \{(1, 2), (3, 1)\}, \\ 0{,}1 & \text{jos } (e, s) \in \{(2, 1), (1, 1)\}, \\ 0 & \text{jos } (e, s) = (0, 0). \end{cases}$$

4.5 Järjestelmien tehokkuuden mittarit

4.5.1 INEXissä käytetyt mittarit

INEX-arviointien tavoitteena on mitata järjestelmien tiedonhakutehokkuutta. Tehokkuus määriteltiin järjestelmän kyvyksi palauttaa tarkimmat dokumenttien elementit, jotka ovat kattavia tiedonhaun aiheen suhteen. (Gövert ja Kazai 2002.)

Ensimmäisessä INEXissä käytettiin *inex-2002*-mittaria (*inex_eval*), joka laskee kullekin elementille todennäköisyyden *P(rel|retr)*, jolla käyttäjän katsoma elementti on relevantti. (Kazai 2004.)

Seuraava mittari, *inex-2003* eli *inex_eval_ng*, pyrki korjaamaan inex-2002-mittarin puutteita. Ongelmat liittyvät elementtien päällekkäisyyden ongelmaan, joka on hyvin keskeinen tekijä XML-elementtien tiedonhaussa. Tarkempi kuvaus inex-2003-mittarista löytyy esimerkiksi Kazailta, Lalmasilta ja de Vriesiltä (2004a, 74).

Ehdotuksena päällekkäisyysongelman ratkaisuun Kazai, Lalmas ja de Vries esittelivät *XCG*:n, kertyneen hyödyn mittaamiseen perustuvan mitan ja sitä käyttävän *EvalJ*-mittarin (Kazai, Lalmas ja de Vries 2004a, 78-79). Esittelen sen tarkemmin, mutta sitä ennen on hyvä selvittää, mistä päällekkäisyysongelmassa oikeastaan on kyse.

Muitakin ratkaisuja on esitetty. Kazai ja Lalmas esittelevät ja vertailevat artikkelissaan (Kazai ja Lalmas 2005) mittarit inex-2002 ja inex-2003, sekä mitat XCG (kertynyt hyöty), *PRUM* (*Precision Recall with User Modelling*) ja T_2I (*Tolerance to Irrelevance*). Kazain ja Lalmasin EvalJ-mittari sisältää toteutukset XCG- ja PRUM-mitoille. T_2I-mitta on toistaiseksi puhtaasti teoreettinen. En käsittele näitä vaihtoehtoisia mittoja ja mittareita sen tarkemmin, koska Kazai ja Lalmas toteavat keskittyvänsä jatkossa XCG-mittaan lupaavimpana vaihtoehtona.

4.5.2 Päällekkäisyysongelma

XML-dokumenttien evaluointi tuottaa uudenlaisia vaatimuksia evaluointimenetelmille. Merkittävä ongelma on päällekkäisten tulosten palauttaminen. Tulosjoukossa saattaa esimerkiksi olla artikkelin osa ja sen sisältämiä kappaleita, jotka palautetaan käyttäjälle eri kohdissa tulosjoukkoa (Kazai, Lalmas ja de Vries 2004a, 72). Käyttäjän kannalta tilanne on

turhauttava, sillä ylempi elementti (artikkelin osa) sisältää alemmat elementit (kappaleet) kokonaisuudessaan.

Tulosjoukon lisäksi päällekkäisyys vaivaa saantikantaa. Koska relevanssiarviot on tehty yksittäisten elementtien tasolla, saantikannassa on päällekkäisiä elementtejä. Käytetyissä mittareissa on mekanismeja, jotka huomioivat tulosjoukon elementtien päällekkäisyyttä, mutta saantikannan elementtien päällekkäisyyden seurauksia ei huomioida. (Kazai, Lalmas ja de Vries 2004a, 72.)

Koska dokumentin elementin kattavuus on aina suurempi tai yhtä suuri kuin sen alaelementtien (koska elementti sisältää kaikki alaelementtinsä), koko polku relevantista elementistä artikkelin juureen asti on relevantti (relevanssin aste luonnollisesti vaihtelee polun varrella). Tämän vuoksi saantikanta koostuu sarjoista päällekkäisiä elementtejä. (Kazai, Lalmas ja de Vries 2004a, 73.)

Alkuperäinen inex-2002-mittari ei huomioi tulosjoukon päällekkäisyyttä millään tavalla (Kazai, Lalmas ja de Vries 2004b, 33). Järjestelmä, joka palauttaa voimakkaasti päällekkäisen tulosjoukon, voi saada hyvät pisteet, vaikka tällainen tulos on käyttäjälle varsin epäkäytännöllinen. INEX 2003 -workshopissa päätettiin, että tällaisesta toiminnasta järjestelmää tulisi rangaista, eikä palkita (Kazai 2004).

On esitetty, että järjestelmäsuuntautuneesta näkökulmasta päällekkäisyyttä ei tarvitse nähdä ongelmana, koska järjestelmien voidaan olettaa selviävän siitä jollain tavalla, esimerkiksi suodattamalla päällekkäiset tulokset käyttäjälle annettavista tuloksista. Tämän vuoksi päällekkäisyys tulisi sallia tuloslistoilla kun tehdään järjestelmäsuuntautunutta evaluointia. Se on hyväksyttävää, mutta päällekkäisyys ei saa olla tekijä, jota voi hyödyntää parempien tulosten saamiseksi. Mittarin tulisi siis sallia järjestelmille päällekkäisyyteen perustuvien hakustrategioiden käyttö, mutta sellaisesta strategiasta ei saa antaa etua vertailussa. (Kazai ja Lalmas 2005, 23-24.)

Toisena ongelmana inex-2002-mittari laskee saantia koko saantikannan perusteella. Koska saantikannassa on valtava määrä päällekkäisiä elementtejä, järjestelmä voi saavuttaa täydellisen saannin vain palauttamalla kaikki nämä päällekkäisyydet. (Kazai, Lalmas ja de Vries 2004b, 33.)

Ilmiöstä käytetään nimitystä *overpopulated recallbase*, ylikansoitettu saantikanta. Ongelman ydin on siinä, että saantikannassa on enemmän elementtejä kuin täydellisesti toimivan järjestelmän tulisi oikeastaan palauttaa. Täydellisen saannin saavuttaakseen järjestelmän on siis palautettava kaikki relevantit elementit, myös päällekkäiset, mikä on vastoin tehokkaan XML-tiedonhaun määritelmää. (Kazai, Lalmas ja de Vries 2004a, 74-75.)

Havainnollistaakseen ongelmaa Kazai, Lalmas ja de Vries muodostivat täydelliset hakutulokset valitsemalla parhaat elementit jokaisesta päällekkäisten elementtien ryhmästä. Näin saadussa joukossa oli vain relevanteimmat elementit, eikä lainkaan päällekkäisyyttä. Tällaisella tulosjoukolla saannin ja tarkkuuden tulisi olla huippuluokkaa, mutta kun täydellistä tulosta verrattiin ylikansoitettuun saantikantaan, tulos oli kaukana täydellisestä. (Kazai, Lalmas ja de Vries 2004a, 75.)

Uudempi inex-2003-mittari pyrki huomioimaan päällekkäisyyttä hakutuloksissa. Se ei anna lainkaan pisteitä kokonaan nähdylle elementille ja vähentää osittain nähdyn elementin saamia pisteitä. Tämäkään mittari ei kuitenkaan ratkaise ylikansoitetun saantikannan ongelmaa.

4.5.3 XCG-mittaan perustuva mittari

Kazai, Lalmas ja de Vries (2004a, 78-79) esittelevät kertynyt hyöty –mittaan perustuvan XCG-mitan (ja sitä käyttävän EvalJ-mittarin) ratkaisuna ongelmiin, jotka liittyvät moniasteiseen relevanssiin ja päällekkäisyyteen. Alunperin yksiulotteista relevanssia ja perinteisiä dokumentteja varten kehitetty mitta vaatii kuitenkin pientä hienosäätöä toimiakseen moniulotteisen relevanssin ja XML-dokumenttien maailmassa.

Esitetty ratkaisu on vaikuttanut suuresti siihen, miten olen kertynyt hyöty -mittaa työssäni käsitellyt, joten sen esittely tarkemmin on paikallaan.

Relevanssiarvofunktiot

Kahden relevanssiulottuvuuden muuntamiseen yhdeksi relevanssiskaalaksi on käytetty kvantisointifunktioita, joita käsiteltiin luvussa 4.4.2. Nämä muunnokset ovat aina

tietynlaisia mallinnoksia käyttäjien käyttäytymisestä ja siten olennaisia parametrejä evaluointiprosessissa.

Kazai, Lalmas ja de Vries lähtevät laajentamaan tästä lähtökohdasta määrittelemällä relevanssiarvofunktion (*relevance value function) r(c_i)*. Funktio palauttaa arvon [0,1] elementille c_i järjestetyssä tuloslistassa. Arvo kuvaa elementin relevanssia tai hyödyllisyyttä käyttäjälle. (Kazai, Lalmas ja de Vries 2004a, 77.)

Relevanssiarvofunktio (lyhyesti RV-funktio) voi käyttää parametreinään esimerkiksi elementin relevanssiarvioinnin ulottuvuuksia, aiemmin nähdyn osan määrää tai mahdollisesti luettuja osia. Tarpeiden mukaan voidaan kehittää useita erilaisia RV-funktioita, jotka kuvaavat kukin eri tyyppistä käyttäjämallia. (Kazai, Lalmas ja de Vries 2004a, 77.)

Kazai, Lalmas ja de Vries esittelevät kaksi RV-funktiota, joista kiinnostavampi on tuloslistan järjestyksestä riippuva funktio. Sen tavoitteena on rangaista järjestelmiä, jotka palauttavat päällekkäisiä tuloksia. Funktio edustaa käyttäjää, jonka mielestä aikaisemmin nähdyt elementit ovat irrelevantteja. (Kazai, Lalmas ja de Vries 2004a, 77.)

RV-funktion tarkoituksena on arvioida käsillä oleva elementti siten, että elementin relevanssiarvoa vähennetään ennennäkemättömän ja jo nähdyn sisällön suhteessa. Kokonaan nähtyjen elementtien vähennyksen määrään vaikuttaa myös tekijä α, joka määrittelee sen, kuinka paljon käyttäjä sietää toistoa tuloksissa. Tyypillisellä arvolla α = 1 aikaisemmin nähtyjen elementtien pistearvoksi tulee automaattisesti 0. (Kazai, Lalmas ja de Vries 2004a, 77-78.)

Tuloslistasta riippumaton yksinkertaisempi RV-funktio palauttaa relevanssiarvon, joka riippuu ainoastaan elementin saamasta relevanssiarviosta:

$$r(c_i) = f(assess(c_i))$$

jossa f on kvantisointifunktio (kts. luku 4.4.2), joka sovittaa kattavuus-spesifisyys-parin asteikolle [0, 1] ja *assess(c_i)* on funktio, joka palauttaa elementin c_i relevanssiarvion (siis kattavuuden ja spesifisyyden, (*e, s*)), jos arvio saantikannasta löytyy, muussa tapauksessa (0, 0). (Kazai, Lalmas ja de Vries 2004a, 77.)

Inex-2003-mittari rankaisee päällekkäisistä tuloksista normalisoimalla elementin pistearvon elementin vielä näkemättömän osan koon ja elementin koko koon suhteella. Sama vaikutus saadaan seuraavalla RV-funktiolla:

$$r(c_i) = \frac{f(assess(c_i)) \cdot |c'_i|}{|c_i|}$$

jossa $|c_i|$ on elementin c koko ja $|c'_i|$ vielä näkemättömän osuuden koko. (Kazai, Lalmas ja de Vries 2004a, 77.)

Koska ennennäkemättömän osuuden koko lasketaan hyvin suoraviivaisesti, tämä kaava olettaa, että relevantti informaatio on jakautunut tasaisesti elementin sisällä, eli se antaa vain karkean arvion elementin osien relevanssiarvosta. Esimerkki tilanteesta, jossa tämä ei toimi, on relevantti luku, jossa on yksi relevantti kappale ja yhdeksän ei-relevanttia. Tällöin tuntuisi oudolta antaa luvulle 90% sen relevanssiarvosta, jos ainoa relevantti kappale on jo nähty. (Kazai, Lalmas ja de Vries 2004a, 77.)

Toisenlainen arvio elementin osan relevanssiarvosta saadaan määrittelemällä uusi arviointifunktio *assess*′(·), joka toimii ennennäkemättömän elementin kohdalla samoin kuin *assess*(·): palauttaa arvioparin (*e, s*) mikäli sellainen saantikannasta löytyy, muuten (0, 0). Jos elementti on nähty kokonaisuudessaan aikaisemmin, funktio määritellään:

$$assess'(c_i) = (1-\alpha) \cdot assess(c_i)$$

jossa α on aikaisemmin mainittu tekijä, joka kuvaa käyttäjän kärsivällisyyttä aiemmin nähtyjä elementtejä kohtaan. Jos α on 1, aikaisemmin nähty elementti saa arvokseen (0, 0). (Kazai, Lalmas ja de Vries 2004a, 77-78.)

Jos parametrinä on elementti, josta osa on jo näytetty käyttäjälle, arvio relevanssiarvosta lasketaan seuraavalla kaavalla:

$$assess'(c_i) = \alpha \cdot \frac{\sum_{j=1}^{m} (assess'(c_j) \cdot |c_j|)}{\sum_{j=1}^{m} |c_j|} + (1-\alpha) \cdot assess(c_i)$$

jossa $|\cdot|$ on elementin koko, m elementin c_i lapsielementtien määrä ja c_j elementti, joka voi olla ennennäkemätön lapsielementti, elementin osa tai kokonaan aikaisemmin nähty elementti. (Kazai, Lalmas ja de Vries 2004a, 78.)

Tällöin siis kokonaan aikaisemmin nähty elementti saa arvokseen jotain nollan ja elementin täyden arvon väliltä, riippuen α-tekijästä. Osittain nähtyjen elementtien tapauksessa relevanssiarvo riippuu elementin sisältämien osien relevanssiarvosta ja koosta. Mitä suurempi osuus on jo nähty, sitä pienempi on relevanssiarvo.

Kaksiulotteisen relevanssin muuttaminen yksiulotteiseksi relevanssiasteikoksi tapahtuu siis RV-funktion avulla. Tuloslistan järjestyksestä riippuva RV-funktio auttaa myös päällekkäisyydessä tuloslistan elementtien osalta, koska samojen elementtien toistuminen tuloslistalla pudottaa pistearvoa nopeasti. (Kazai, Lalmas ja de Vries 2004a, 78.)

Saantikannan päällekkäisyyden käsittely

Saantikannan päällekkäisyyden ongelma ratkaistaan rakentamalla ideaalisaantikanta (Kazai, Lalmas ja de Vries 2004a, 78). Saantikannan rakentamisesta on kerrottu tarkemmin luvussa 5.3.2, koska toteutan ratkaisun myös omassa evaluointityökalussani.

Ideaalisaantikannan käytöstä seuraa pari huomionarvoista seikkaa. Ensinnäkin, ideaalisaantikannasta muodostetussa ideaalihyötyvektorissa voi olla vähemmän elementtejä kuin varsinaisissa hakuajoissa (jos tuloksissa on esimerkiksi päällekkäisiä elementtejä). Toiseksi, ideaalivektorilla on tietty maksimihyötyarvo, jonka hakutuloksen hyötyvektori voi ylittää, jos hyöty lasketaan koko saantikannan perusteella. (Kazai, Lalmas ja de Vries 2004a, 78.)

Ensimmäinen ongelma ratkeaa täyttämällä ideaalihyötyvektoria irrelevanteilla elementeillä. Se ei vaikuta haun evaluointiin, koska irrelevanteista dokumenteista ei tule ylimääräistä hyötyarvoa. Toinen ongelma ratkeaa määrittelemällä ideaalivektorin maksimihyötyarvo ylärajaksi, jota tulosjoukosta muodostettu hyötyvektori ei voi ylittää. (Kazai, Lalmas ja de Vries 2004a, 78.)

Yhteenveto

Kertyneen hyödyn käyttäminen hakujärjestelmien evaluointiin XML-ympäristössä ratkaisee päällekkäisyyteen liittyvät ongelmat. Tulosten vertaaminen koko saantikantaan luo joustavan järjestelmän, joka ei huomioi pelkästään täydellisiä osumia, vaan erottelee ne elementit, jotka järjestelmän pitäisi löytää (ideaalisaantikannan sisältö) ja ne osittaiset

osumat, joiden löytämisestä voidaan palkita (ideaalisaantikannan ulkopuoliset relevantit elementit). (Kazai, Lalmas ja de Vries 2004a, 79.)

Tulosjoukkojen päällekkäisyys liittyy myös käyttäjien tyytyväisyyteen. Kehitetty mittari tarjoaa mahdollisuuksia soveltaa erilaisia käyttäjämalleja; mittari itsessään ei ota kantaa käyttäjien tarpeisiin. Kazai, Lalmas ja de Vries ilmoittavatkin aikovansa selvittää todellisten käyttäjien toimintaa INEXin interaktiivisen tiedonhaun avulla. (Kazai, Lalmas ja de Vries 2004a, 79.)

5 Evaluointityökalun kehitystyö

5.1 Lähtökohdat

Yhtenä osana tutkimustani kehitin Tampereen yliopiston informaatiotutkimuksen laitoksen tarpeisiin evaluointityökalun, jota voi käyttää XML-aineistosta INEX-konferenssien kontekstissa tehtyjen tiedonhakujen evaluointiin.

Uutta evaluointityökalua tarvitaan, sillä aikaisemmin käytettyjen mittareiden ominaisuudet eivät ole riittäviä. Mittarit eivät selviä XML-tiedonhakuun liittyvistä erikoisuuksista, kuten päällekkäisyydestä, riittävän hyvin. Siksi on tarpeen luoda uusi evaluointityökalu, joka selviää päällekkäisyydestä ja joka hyödyntää moniarvoista relevanssia paremmin. Hyödylliseksi havaitun kertynyt hyöty -mittarin hyödyntäminen edellyttää niin ikään uutta mittaria.

Tarkoitukseni on esitellä, millaisia ratkaisuja evaluointityökalun kehittäminen edellytti ja kuinka työkalun käytännössä toteutin. Esittelen ohjelman toimintaa lopuksi muutamalla esimerkillä, jotka havainnollistavat ohjelman avulla saavutettavia etuja.

5.1.1 Evaluointityökalun komponentit

Tiedonhakujen evaluointiin tarvitaan seuraavat komponentit:

1. evaluointityökalu
2. evaluointimitta
3. relevanssiarviot aineistosta, johon haut kohdistuvat
4. evaluoitavat hakutulokset

Käsittelen seuraavaksi näiden komponenttien piirteitä evaluointityökalun toteutuksen yhteydessä.

Evaluointityökalu

Evaluointityökalu, kehitystyön lopullinen tulos, on tietokoneohjelma, joka laskee saamiensa syötteiden perusteella tulokset, joista käy ilmi ohjelmalle annettujen tiedonhakujen tehokkuus hakuaiheissa, joiden relevanssiarvioita työkalu käyttää vertailupohjana. Evaluointityökalu on siis mittari, jolla mitataan tiedonhakujärjestelmien tehokkuutta. Mitä tämä tehokkuus käytännössä on, sen määrittelee käytetty mitta.

Evaluointimitta

Mittari mittaa jotain, jonka pitäisi osoittaa tiedonhakujen tehokkuus. Koska käytettävissä ovat relevanssiarviot, tehokkuuden mitaksi voidaan ottaa se, kuinka hyvin tiedonhakujärjestelmä palauttaa relevantteja dokumentteja. Kun mielenkiinnon kohteena on nimenomaan se, kuinka moniarvoinen relevanssi ja tuloslistan järjestyksen hyödyntäminen vaikuttaa tiedonhakujen evaluoinnissa, käyttää työkalu moniarvoista relevanssia hyödyntävää kertynyt hyöty -mittaa. Sen toiminta on kuvattu tarkemmin luvussa 2.5.3.

Kehitystyöni lähtökohtana on toiminut XCG-mitta (katso luku 4.5.3). Olen kuitenkin tehnyt mitan käytännön toteutuksessa ja päällekkäisyyden käsittelyssä omia tulkintojani ja ratkaisujani, enkä ole tutustunut sen tarkemmin EvalJ-mittarin käytännön toteutukseen.

Relevanssiarviot aineistosta, johon haut kohdistuvat

Tiedonhaut kohdistuvat INEXin testiaineistoon, joka muodostuu IEEE:n julkaisemien lehtien artikkeleista. Aineistoon on laadittu valmiita hakuaiheita ja aineiston relevanssia on arvioitu näiden aiheiden kannalta. Tiedonhakujen tehokkuutta arvioidaan näitä relevanssiarvioita vastaan: tehokas hakualgoritmi palauttaa aineiston relevanteimmat elementit ensimmäisenä.

Relevanssiarviot on käytännössä tallennettu XML-tiedostoina käyttäen tarkkaan määriteltyä rakennetta. Työkalun onkin pystyttävä lukemaan tätä DTD:tä noudattavia XML-muotoisia saantikantoja.

Evaluoitavat hakutulokset

INEX-konferenssin osallistujat vastaavat annettuihin hakuaiheisiin kehittämiensä hakukoneiden tuottamilla järjestetyillä tuloslistoilla. Tuloslistassa on lueteltu ohjelman palauttamat dokumenttielementit järjestyksessä. Tehtävä evaluointi perustuu näiden tuloslistojen suorituksen tason määrittelyyn, minkä vuoksi evaluointityökalun on luonnollisesti kyettävä lukemaan tuloslistoja. Myös tuloslistat ovat XML-tiedostoja, jotka käyttävät tiettyä, ennaltamäärättyä DTD:tä.

5.1.2 INEXin dokumenttityyppimäärittelyt

Evaluoitavien hakutulosten formaatti

INEXissä käytetään kuvion 2 mukaista DTD:tä hakutulosten esittämiseen. Hakutulos koostuu `inex-submission`-elementistä, jolla on attribuutit `participant-id` (INEXin osallistujatunnus), `run-id` (hakuajon tunnus), `task` (tehtävätyyppi, sisältökysely vai sisältö ja rakenne -kysely) ja `query` (hakutyyppi, automaattinen vai manuaalinen).

```
<!ELEMENT inex-submission       (description, topic+)>
<!ATTLIST inex-submission
  participant-id         CDATA          #REQUIRED
  run-id                 CDATA          #REQUIRED
  task                   (CO | VCAS)    #REQUIRED
  query          (automatic | manual)   #REQUIRED
>
<!ELEMENT description           (#PCDATA)>
<!ELEMENT topic                 (result*)>
<!ATTLIST topic
  topic-id                     CDATA          #REQUIRED
>
<!ELEMENT result                (file, path, rank?, rsv?)>
<!ELEMENT file                  (#PCDATA)>
<!ELEMENT path                  (#PCDATA)>
<!ELEMENT rank                  (#PCDATA)>
<!ELEMENT rsv                   (#PCDATA)>
```

Kuvio 2: Evaluoitavien hakutulosten dokumenttityypin määrittely (Lalmas ja Malik 2004.)

`Inex-submission`-elementin sisällä on yksi `description`-elementti, joka sisältää vapaamuotoisen kuvauksen tehdystä ajosta. Sen lisäksi elementti sisältää yhden tai useampia `topic`-elementtejä, jotka sisältävät varsinaiset tulokset.

`Topic`-elementin attribuutti `topic-id` kertoo hakuaiheen numeron. Tulokset ovat `result`-elementeissä, joista jokainen sisältää yhden tuloslistan rivin. `Result`-elementin tulee sisältää `file`- ja `path`-elementit, jotka yksilöivät tuloksena saadun elementin: `file` määrittelee tiedoston, jossa elementti sijaitsee ja `path` polun tiedoston sisällä.

`Rank`-elementti kertoo dokumentin sijoituksen tuloslistalla. On syytä huomata, että `rank` ei ole pakollinen elementti. Siinä tapauksessa, että `rank`-elementit puuttuvat, ohjelmani olettaa, että tulosjoukon rivit on lueteltu järjestyksessä ensimmäisestä viimeiseen. Viimeinen elementti, `rsv`, kertoo elementin hakujärjestelmältä saaman pistearvon; tällä tiedolla ei ole käytännön merkitystä.

Evaluoinnin kannalta tärkeimmät tiedot ovat hakuaiheen numero, jolla tuloslista yhdistetään oikeisiin relevanssiarvioihin ja varsinaiset tuloselementit, jotka listaavat palautetut elementit. On kätevää, mikäli tuloslistan laatija on numeroinut listan, mutta siihen ei voi luottaa. Tarkastelemissani tuloslistoissa esiintyi erilaisia dokumenttityypin määrittelyn mahdollistamia esitystapoja.

Relevanssiarvioiden formaatti

INEXiin tehdyt relevanssiarviot ovat saatavilla seuraavan DTD:n mukaisessa formaatissa.

```
<!ELEMENT assessments           (file*)>
<!ATTLIST assessments
  pool                  CDATA           #REQUIRED
  topic                 CDATA           #REQUIRED
>
<!ELEMENT file                  (path*)>
<!ATTLIST file
  file                  CDATA           #REQUIRED
>
<!ELEMENT path                  EMPTY>
<!ATTLIST path
  path                  CDATA           #REQUIRED
  exhaustiveness        (0|1|2|3|U)     "U"
  specificity           (0|1|2|3|U)     "U"
  inpool                (true|false)    "false"
  inferred              (true|false)    "false"
  inconsistant          (true|false)    "false"
>
```

Kuvio 3: Relevanssiarvioiden dokumenttityypin määrittely. (INEX 2004)

Relevanssiarviot ovat `assessments`-elementissä. Sen attribuuteista tärkeämpi on `topic`, joka kertoo hakuaiheen, johon relevanssiarviot liittyvät. `Pool`-attribuutilla ei ole tässä yhteydessä merkitystä.

Relevanssiarviot on listattu dokumenttielementeittäin. Elementit on jaoteltu ensin tiedoston mukaan `file`-elementteihin (attribuutti `file` kertoo, mistä tiedostosta on kyse) ja niiden sisällä `path`-elementteihin. `Path`-elementin attribuutti `path` kertoo polun, jota arvio koskee.

Itse relevanssiarvio on attribuuteissa `exhaustiveness` ja `specificity`, jotka kertovat tiedoston ja polun määrittämän elementin kattavuuden ja spesifisyyden. Attribuutit `inpool`, `inferred` ja `inconsistant` ovat relevanssiarvioiden luonteeseen liittyviä piirteitä, joilla ei ole tässä yhteydessä merkitystä.

5.2 Evaluointityökalun käytännön toteutus

5.2.1 Käyttöympäristö

Aikaisempi INEX-evaluointityökalu *inex_eval* on toteutettu perl-ohjelmointikielellä ja toimii siten pääasiassa Unix-ympäristössä (perl sinänsä on käännetty useimmille tietokonealustoille, mutta etenkin Windows-pohjaisissa koneissa sen käyttäminen on hieman hankalaa). Tämä on vähimmäisvaatimus uutta evaluointityökalua kehitettäessä, mutta laajempi toimivuus on tavoittelemisen arvoista. Erityisesti helppokäyttöisyys Windows-ympäristössä olisi merkittävä parannus aikaisempaan verrattuna.

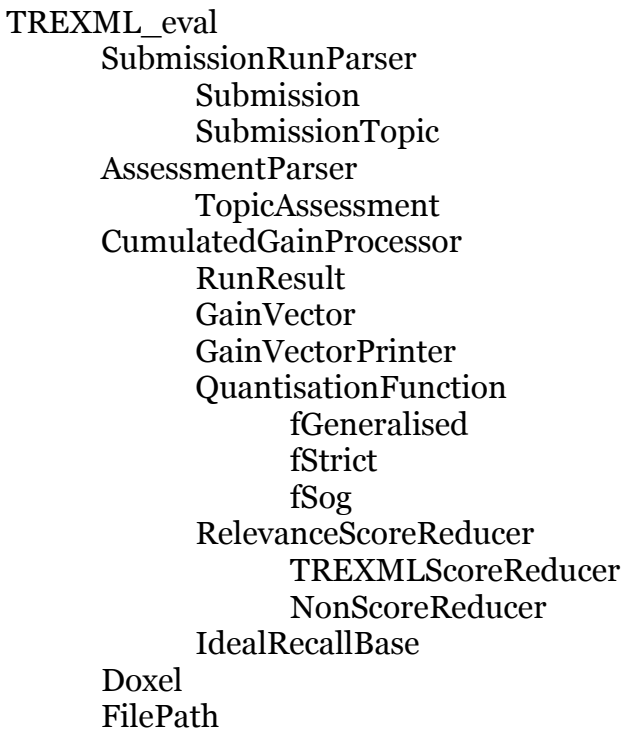
Ottaen huomioon omat ohjelmointitaitoni ja järjestelmälle asetetut vaatimukset, päädyin toteuttamaan evaluointityökalun Sunin kehittämällä Java-ohjelmointikielellä (*http://java.sun.com/*). Java on perlin tapaan tulkattava ohjelmointikieli, eli millä tahansa alustalla kirjoitettu ohjelma on ajettavissa missä tahansa, kunhan kyseiselle alustalle on toteutettu Java-tulkki. Käytännössä tulkki löytyy kaikille tyypillisimmille ympäristöille. Mikä olennaisinta, Java on Windows-ympäristössä perliä vaivattomampi ohjelmointikieli.

Toteutin ohjelman komentorivipohjaisesti, eli ohjelma ajetaan komentoriviltä eikä siinä ole minkäänlaista graafista käyttöliittymää. Käyttöliittymä on suhteellisen vaivatonta lisätä ohjelmaan myöhemmin; nyt en kokenut sitä ohjelman tarkoituksen kannalta mielekkääksi. Jos ohjelmaa haluaisi jatkossa kehittää, graafisen käyttöliittymän rakentaminen olisi hyvä idea. Se helpottaisi ohjelman syötteiden antamista huomattavasti.

Graafisen käyttöliittymän puuttumisesta huolimatta olen kehittänyt evaluointityökalun käytettävyyttä inex_evaliin nähden. Kehittämäni ohjelman syötteitä ei anneta komentoriviparametreinä, vaan erillisen asetustiedoston kautta (esimerkki asetustiedostosta on liitteessä C). Tämä on helpompaa ja vähemmän altista virheille kuin komentoriviparametrien käyttäminen.

5.2.2 Ohjelman rakenne ja toiminta

Koska Java on olio-ohjelmointikieli, ohjelman rakenteesta muodostui väistämättä oliopohjainen. Käytännössä ohjelma koostuu pienemmistä osista, luokista, jotka ovat erikoistuneet jonkun tietyn tehtävän suorittamiseen. Näitä osia luodaan ja kutsutaan tarpeen mukaan.



Kuvio 4: TREXML_eval-ohjelman rakennekaavio.

Kuviossa 4 on esitelty ohjelman rakenne. Ohjelman ytimenä on `TREXML_eval`-luokka, joka huolehtii koko evaluointiprosessin etenemisestä. Se lukee ohjelman asetukset, lukee tuloslistan, hakee tuloslistan aiheita vastaavat relevanssiarviot, laskee ideaalisaantikannat ja varsinaiset tulokset sekä tulostaa tulokset.

Käytännössä luokka ei tee kaikkea tätä itse, vaan delegoi alatehtävät muille luokille. `SubmissionRunParser` lukee XML-muotoisen tuloslistan (tuloslistan XML-muoto eli dokumenttityypin määrittely on esitelty luvussa 5.1.2) ja tulkitsee sen ohjelman ymmärtämään muotoon, eli luo siitä `Submission`-luokkaa edustavia olioita. `Submission` on ohjelmassa määritelty tietorakenne, joka on suunniteltu nimenomaan INEX-tuloslistojen käsittelyyn.

Relevanssiarvioiden tulkitseminen on `AssessmentParser`-luokan tehtävä. Tämä tulkki luo XML-muotoisista relevanssiarvioista `TopicAssessment`-luokkaa edustavia tietorakenteita. Niiden avulla annetun dokumenttielementin tietyssä aiheessa saama relevanssiarvio on yksinkertaista tarkistaa.

Varsinaisen työn tekee `CumulatedGainProcessor`-luokka. Se käy saamansa tuloslistan läpi elementti kerrallaan. Kunkin elementin relevanssiarvo tarkastetaan ja muunnetaan kaksiulotteisesta relevanssista yhdeksi relevanssipistearvoksi. Muunnos riippuu käytetystä kvantisointifunktiosta (katso luku 4.4.2). Toinen lopulliseen pistearvoon vaikuttava tekijä on arvonvähennysalgoritmi, jonka toiminnasta kerron tarkemmin luvussa 5.3.3.

Kun dokumenttielementin arvo on tiedossa, se voidaan lisätä hyötyvektorin jatkoksi. Ohjelma laskee samalla myös alennettua hyötyä, normalisoitua hyötyä ja alennettua normalisoitua hyötyä. Hyötyvektorien tietorakenteena toimiva `GainVector`-luokka osaa tehdä vaadittavat laskutoimitukset näille vektoreille, mikäli ideaalisaantikanta on käytettävissä.

Ohjelman tietorakenteiden ytimessä on `Doxel`-luokka. Doxeleihin varastoidaan yksittäiset elementit (sana *doxel* on lyhenne sanoista *document element*) ja niiden tiedot. `FilePath` on vastaava alkeistason apuluokka, jota käytetään elementtien sijaintien tallentamiseen. Elementin sijaintihan muodostuu kahdesta osasta: tiedostosta eli dokumentista, josta se löytyy (sisältäen tiedostopolun kyseiseen dokumenttiin) ja polusta dokumentin sisällä.

Javan oliorakenne tarjoaa joitain hyvin käytännöllisiä mahdollisuuksia ohjelman käytössä. Esimerkiksi uusien kvantisointifunktioiden lisääminen on yksinkertaista, eikä edellytä kajoamista ohjelman lähdekoodiin. Riittää, että haluttu kvantisointifunktio laaditaan siten, että se perii ohjelmaan sisältyvän `QuantisationFunction`-luokan (eli toteuttaa sen sisältämän muunnosfunktion, joka ottaa kaksi relevanssiarvoa ja palauttaa yhdistetyn relevanssiarvon).

Uusi kvantisointifunktio otetaan käyttöön antamalla sen nimi parametrinä asetustiedostossa, jolloin Java osaa liittää sen mukaan ohjelmakoodiin ohjelmaa ajettaessa. Samalla periaatteella toimii myös arvonvähennysalgoritmi.

5.3 Ohjelman toteutukseen liittyviä ratkaisuja

5.3.1 Tuloslistan elementtien päällekkäisyys

Tuloslistan elementtien päällekkäisyyden ongelma keskittyy muutamaan kysymykseen. Ensinnäkin, jos elementillä on vain yksi relevantti alaelementti, joka palautetaan ensin,

ratkaisu on selvä: ylemmän elementin palauttaminen ei tuo lisää tietoa, joten siitä ei kuulu palkita ollenkaan.

Toisaalta tilanteessa, jossa ylempi elementti on palautettu ensin, alemman elementin palauttaminen ei tuo lisäinformaatiota. Voidaan pohtia, onko sillä merkitystä sisältääkö ylempi elementti jotain turhaa, mitä alemmassa elementissä ei ole, mutta toisaalta turhan, ei-relevantin tiedon mukanaolon pitäisi näkyä relevanssiarviossa (ainakaan spesifisyys ei voi olla yhtä hyvä kuin asiaan tarkemmin keskittyvällä elementillä). Voidaan siis olettaa, että toimiva hakujärjestelmä palauttaisi paremman, eli tarkemman, elementin muutenkin ensin.

Kolmanneksi, tilanne muuttuu hankalaksi erityisesti silloin, kun ensin palautetaan alempi elementti ja sitten ylempi elementti, jolla on monta relevanttia alaelementtiä. Jos yksi niistä on jo nähty, miten yläelementin palauttamisesta tulisi palkita? Täydet pisteet eivät tule kysymykseen, sillä osa elementistä on jo nähty. Elementtiä ei voi myöskään jättää pisteittä, sillä se sisältää ennennäkemätöntä relevanttia informaatiota. Sen pisteistä on siis leikattava osa, mutta leikattavan osuuden määritteleminen ei ole aivan yksiselitteistä.

Toteutan evaluointityökalussani menetelmän, jossa lasketaan yhteen elementin kaikkien alielementtien (ja niiden alielementtien aina kaikista pienimpiin elementteihin asti) relevanssipisteet. Näin saadaan selville elementin kokonaisuudessaan sisältämä relevanssi, rel_{kaikki}. Lisäksi lasketaan yhteen näkemättömien alaelementtien relevanssiarvot, $rel_{näkemättä}$. Dokumentin relevanssiarvosta voidaan nyt vähentää jo nähdyn relevanssin osuus seuraavan kaavan mukaan:

$$vrel_{elementti} = rel_{elementti} * (rel_{näkemättä} / rel_{kaikki}) * kärsivällisyys$$

Mitä enemmän elementin alla olevia relevantteja elementtejä on nähty, sitä vähemmän pisteitä se saa. Kaavassa on mukana myös *kärsivällisyys*, eli arvo, joka kuvaa käyttäjän kärsivällisyyttä jo nähtyä aineistoa kohtaan. Jos kärsivällisyyden arvo on nolla, kaikki elementit, jotka sisältävät yhtään ennen nähtyä aineistoa, saavat relevanssiarvokseen nollan. Tämä tavallaan simuloi käyttäjää, joka ei lainkaan siedä nähdä aikaisemmin näkemäänsä aineistoa.

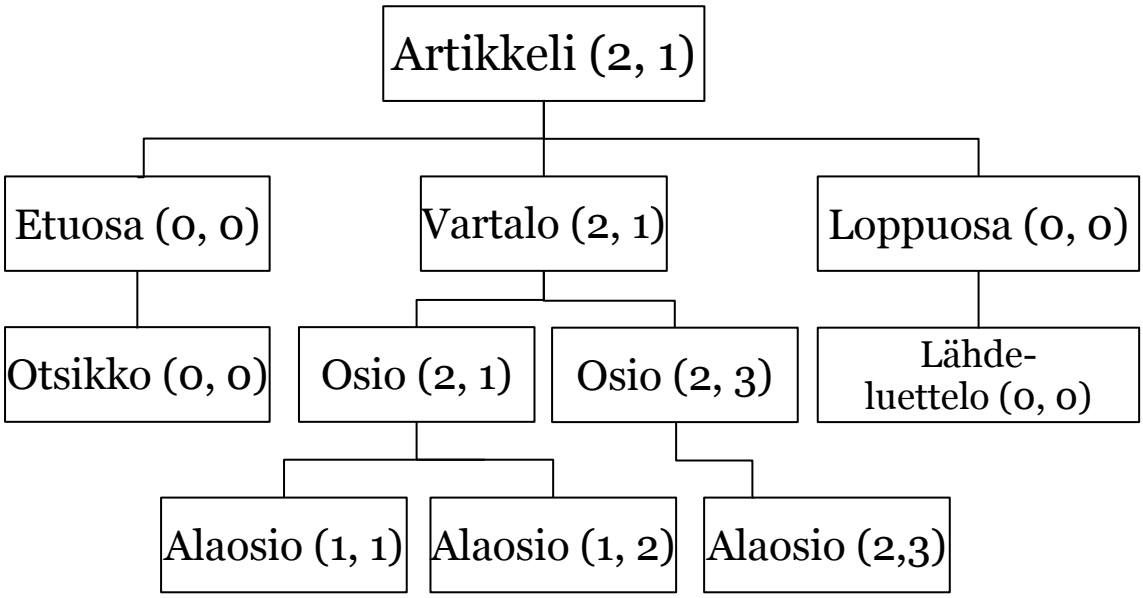
Jos *kärsivällisyys* on yksi, vähennys suoritetaan normaalisti. Tämä on useimmissa tilanteissa mielekkäämpi arvo, koska muuten melko hyväkin elementti voidaan hylätä, jos siitä on nähty edes pieni osa.

Esimerkki

Oletetaan kuvion 5 mukainen XML-dokumentti. Esimerkki pysähtyy selvyyden vuoksi alaosioiden tasolle; todellisessa dokumentissa sisältö olisi jaoteltu alaosioiden sisällä edelleen kappaleiksi ja jopa sitä pienemmiksi elementeiksi. Dokumentin rakennetta havainnollistaa puumuotoinen esitys kuviossa 6.

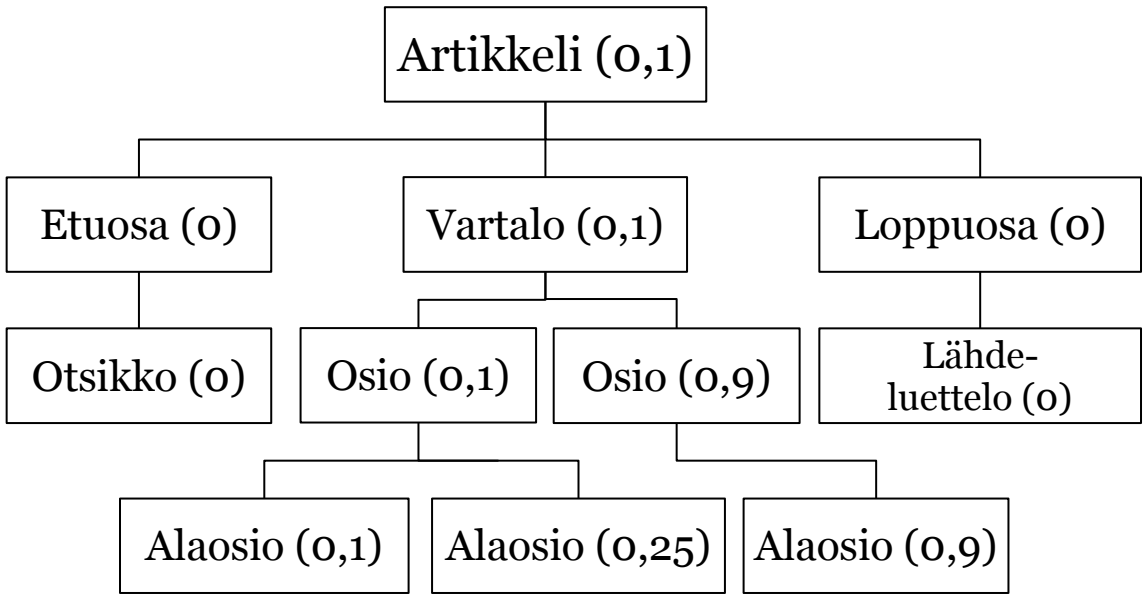
```
<artikkeli>
  <etuosa>
    <otsikko>
    </otsikko>
  </etuosa>
  <vartalo>
    <osio>
      <alaosio>
      </alaosio>
      <alaosio>
      </alaosio>
    </osio>
    <osio>
      <alaosio>
      </alaosio>
    </osio>
  </vartalo>
  <loppuosa>
    <lähdeluettelo>
    </lähdeluettelo>
  </loppuosa>
</artikkeli>
```

Kuvio 5: Esimerkki XML-dokumentin rakenteesta.



Kuvio 6: XML-dokumentin rakenne puumuodossa.

Artikkelin osiin on kuvassa merkitty niiden relevanssiarvo (kattavuus, spesifisyys)-pareina. Hakuaiheeseen löytyisi siis paras vastaus jälkimmäisen osion alaosiosta. Kun eri elementtien relevanssiarviot muunnetaan yhdeksi luvuksi käyttäen Kazain, Lalmasin ja de Vriesin *specificity-oriented generalised* -kvantisointia, saadaan kuvion 7 mukaiset relevanssiarviot.



Kuvio 7: Dokumentin elementtien relevanssiarvot.

Tässä tapauksessa elementin `artikkeli/vartalo/osio[2]/alaosio` palauttaminen ensimmäisenä olisi paras ratkaisu. Sen jälkeen ylemmän elementin

(`artikkeli/vartalo/osio[2]`) palauttamisesta ei tule antaa yhtään pistettä, sillä elementin kaikki sisältö on jo nähty.

Jos parhaan osion jälkeen seuraavana palautetaan koko artikkelin vartaloelementti (`artikkeli/vartalo`), voidaan sen relevanssiarvo laskea yllä mainitun kaavan mukaan. Elementin alaelementtien koko relevanssi (rel_{kaikki}) on 2,25. Näkemättömän osan relevanssi ($rel_{näkemättä}$) on 1,35. Kyseisen elementin relevanssi ($rel_{elementti}$) on 0,1. Sijoittamalla nämä kaavaan saadaan

$$vrel_{elementti} = 0{,}1 * 1{,}35 / 2{,}25 * kärsivällisyys$$

Jos *kärsivällisyys* oletetaan arvoon 1, saadaan vähennetyksi relevanssiarvoksi 0,06. Elementin relevanssiarvosta katoaa siis melko suuri osa, kun sen paras sisältö on jo nähty.

5.3.2 Ideaalisaantikannan muodostaminen

Ideaalisaantikanta on joukko elementtejä, jotka on valittu saantikannasta edustamaan parasta mahdollista tulosjoukkoa. Ideaalisaantikanta on riippuvainen käytetystä kvantisointifunktiosta. (Kazai, Lalmas ja de Vries 2004b, 35.)

Ideaalisaantikanta muodostetaan käymällä läpi kaikki relevantit polut (relevantti polku on XML-artikkelin polku, jonka alussa on *article*-elementti ja lopussa relevantti elementti, jolla ei ole lapsielementtejä tai jonka lapsielementit ovat irrelevantteja). Jokaiselta polulta valitaan relevantein elementti eli se, jonka relevanssipistemäärä on suurin valitun kvantisointifunktion mukaan. (Kazai, Lalmas ja de Vries 2004b, 35.)

Mikäli kahdella elementillä on yhtä korkea relevanssipistemäärä, valinta riippuu elementtien paikasta hierarkiassa (Kazai, Lalmas ja de Vries 2004b, 35). Kazai, Lalmas ja de Vries ovat käyttäneet hierarkiassa alempana olevaa (eli pienempää) elementtiä, mutta kokeilleet myös korkeammalla olevan (eli suuremman) elementin valitsemista. TREXML_eval-työkaluun valitsin poimittavaksi alemman elementin.

Kun relevanteimmat elementit on valittu, saatu ideaalisaantikanta suodatetaan vielä kerran päällekkäisyyksien varalta – sama elementtihän voi olla useamman polun paras

elementti ja silti huonompi kuin joku toinen elementti jollain polulla. Tässä kohtaa ideaalisaantikantaan valitaan elementti, jonka polku on lyhyempi. (Kazai, Lalmas ja de Vries 2004b, 35.)

Artikkelissaan Kazai, Lalmas ja de Vries eivät perustele, miksi nimenomaan lyhyempipolkuinen elementti valitaan. Nopeasti ajateltuna voisi kuvitella, että pidemmän polun valitseminen johtaisi pienempiin ja spesifisimpiin elementteihin, mikä on tietynlainen kantava ajatus koko XML-tiedonhaussa.

Karsintaa tehdään kuitenkin elementeistä, jotka on jo aikaisemmin todettu polkujensa parhaiksi elementeiksi. Se muuttaa tilannetta olennaisesti. Oletetaan kaksi relevanttia polkua, jotka yhdistyvät elementin A kohdalla – elementti A sisältää siis molempien polkujen sisältämät elementit. Suodatettavaa päällekkäisyyttä syntyy, jos elementti A on toisen polun (polku 1) paras elementti, toisen polun (polku 2) parhaan elementin ollessa pienempi ja syvemmällä hierarkiassa.

Tällöin ideaalisaantikantaan valittaisiin sekä elementti A (polun 1 parhaana elementtinä) ja alempana oleva elementti B (polun 2 parhaana elementtinä). Nyt elementti A sisältää sekä polun 1 että polun 2 parhaan sisällön, kun taas elementti B sisältää vain polun 2 parhaan sisällön. Sen takia suodatusvaiheessa on parempi valita lyhyempipolkuinen elementti A: se sisältää enemmän relevanttia informaatiota.

Prosessin tuloksena saadaan ideaalisaantikanta, joka sisältää parhaat elementit käyttäjälle palautettavaksi, olettaen, että päällekkäisyyttä halutaan välttää ja että käytetty kvantisointifunktio kuvaa käyttäjän mieltymyksiä ja odotuksia. (Kazai, Lalmas ja de Vries 2004b, 35.)

Ideaalisaantikantaa tarvitaan, koska muuten saantikannan elementtien päällekkäisyys vääristää tuloksia. Hyvin toimiva tiedonhakujärjestelmä voi saada huonot pisteet, jos koko saantikantaan tuloksia suhteuttava mittari edellyttää, että tuloslistalta löytyy koko joukko päällekkäisiä elementtejä, joiden merkitys käyttäjälle on vähäinen.

Asiaa on tutkittu muodostamalla täydellinen tuloslista, johon valittiin relevanteimmat elementit, jotka olivat parhaita mahdollisia yksiköitä käyttäjälle palautettavaksi. Elementit

valittiin edellä kuvatulla tavalla muodostetusta ideaalisaantikannasta. Täydellisellä tuloslistalla olisi pitänyt saada erittäin hyvä tulos, mutta käytännössä saanti-tarkkuus-käyrät olivat paljon odotettua kehnommat. Huonot tulokset selittää ylikansoitetun saantikannan aiheuttama vääristymä. (Kazai, Lalmas ja de Vries 2004a, 75-76.)

Pelkkää ideaalisaantikantaa voi käyttää myös perinteisten evaluointimittojen kuten saannin ja tarkkuuden kanssa, mutta tällöin arviointi voi olla hieman turhan tylyä. Ideaalisaantikannasta puuttuva elementti voi kuitenkin sisältää jonkun ideaalin elementin (ja sen lisäksi ylimääräistä sisältöä), jolloin sen palauttaminen ei olisi täysi huti. Ideaalisaantikannan käyttäminen mahdollistaa sen, että ideaalisaantikannasta puuttuvat relevantit elementit voidaan luokitella puolittaisiksi osumiksi; silloin niiden palauttamisesta voidaan palkita jonkin verran, kun taas ideaalisaantikannasta löytyvien elementtien palauttaminen voidaan katsoa hyvin toimivan tiedonhakujärjestelmän kriteeriksi. (Kazai, Lalmas ja de Vries 2004a, 76.)

Ideaalisaantikannan muodostamiseen on esitetty myös toista tapaa. Benjamin Piwowarskin PRUM-mittarinsa yhteydessä esittämässä mallissa elementti valitaan ideaaliksi, mikäli:

1. sen relevanssipistearvo on suurempi kuin 0 ja
2. kaikkien sen alla olevien elementtien relevanssipisterarvo on pienempi kuin sen arvo ja
3. kaikkien sen yllä olevien elementtien relevanssipistearvo on pienempi tai yhtä suuri kuin sen arvo, tai jos yllä olevalla elementillä on toinen alempana puussa oleva elementti, jonka relevanssipistearvo on suurempi tai yhtä suuri. (Kazai ja Lalmas 2005, 28.)

Tämä ratkaisu suosii puussa alempana olevia pienempiä elementtejä ja voi siten ikään kuin peittää relevanssiarvioissa olevia puutteita. Kazain, Lalmasin ja de Vriesin malli luottaa siihen, että relevanssiarvioiden tekijät toimivat luotettavasti ja oikein. (Kazai ja Lalmas 2005, 28.)

5.3.3 Arvonvähennysalgoritmi

Arvonvähennysalgoritmi on evaluointityökalussani osa, joka laskee dokumenttielementin todellisen relevanssiarvon. Laskettu arvo perustuu relevanssiarvioon ja siitä kvantisointifunktion avulla saatuun relevanssipistearvoon, jota sitten vähennetään tarpeen mukaan. Arvonvähennys liittyy päällekkäisyysongelmaan ja on osa evaluointityökaluni ratkaisua ongelmaan (ideaalisaantikannan käyttäminen on toinen osa). Käytännössä arvonvähennysalgoritmin tehtävä on toteuttaa luvussa 5.3.1 esitetty periaate.

Yksinkertaisin mahdollinen vähennysalgoritmi (toteutettu ohjelmassa luokkana `NonScoreReducer`) laskee vain elementin relevanssiarvon annetun kvantisointifunktion ja relevanssiarvioinnin mukaan eikä puutu pistearvoon muuten. Tällä tavalla toimii myös päällekkäisyysongelmasta tyystin piittaamaton mittari.

Luokkana `TREXMLScoreReducer` toteuttamani ratkaisu on astetta monimutkaisempi. Ensimmäinen tarkistus on, onko elementtiä nähty vai ei – `CumulatedGainProcessor`-luokka merkitsee kaikki käsittelemänsä elementit nähdyiksi. Se ei kuitenkaan merkitse nähtyjen elementtien alaelementtejä nähdyiksi; tämä päällekkäisyyden kannalta tärkeä vaihe on jätetty arvonvähennysalgoritmin tehtäväksi, sillä sen käsittelyyn on erilaisia lähestymistapoja.

Toteuttamani algoritmi etsii kaikki käsiteltävän elementin alaelementit, eli ne elementit, joiden polku alkaa nykyisen elementin polulla. Tämä vaihe on helppo ja tehokas; XML-dokumenteissa käytetty looginen polkurakenne, jonka käsittelyssä selviää yksinkertaisilla merkkijononkäsittelyfunktioilla on suuri ilo ohjelmoijalle.

Alaelementit käydään läpi, ja niiden relevanssipistearvot lasketaan yhteen, jaettuna jo nähtyjen ja ennennäkemättömien elementtien osiin. Kun alaelementti on käsitelty, se merkitään nähdyksi – jos se kohdataan tuloslistalla myöhemmin, se saa pistearvokseen nolla.

Kerättyjen relevanssipistearvojen perusteella voidaan laskea näkemättömän relevanssin osuus yhteisrelevanssista. Jos mitään ei ole vielä nähty, osuus on yksi; jos kaikki on nähty, osuus on nolla. Elementin relevanssiarvo kerrotaan tällä luvulla, jolloin saadaan lopullinen, vähennetty relevanssipistearvo.

5.3.4 Esimerkkejä evaluointityökalun tuloksista

Koska kehittämäni evaluointityökalun olennaisimmat ratkaisut liittyvät päällekkäisyyden ongelmaan ja sen ratkaisemiseen, etsin INEX 2004 -konferenssin tuloslistoista kaksi ääripäätä: paljon päällekkäisyyttä ja ei lainkaan päällekkäisyyttä.

Päällekkäisyyttä edustamaan löytyi IBM Haifa Research Labin CO-0.5-tuloslista, jonka päällekkäisyysprosentti on 81,46 % (Kazai, Lalmas ja de Vries 2004b, 40). Toisena ääripäänä on Tampereen yliopiston UTampere_CO_average-tuloslista, jossa ei ole lainkaan päällekkäisyyttä.

Osoittaakseni evaluointityökalun ja erityisesti sen arvonvähennysalgoritmin toimintaa, vertailen eri tuloslistoilla saatuja tuloksia keskenään. Tarkastelen IBM Haifan tuloslistaa sekä ilman arvonvähennystä että arvonvähennyksen kanssa. Tampereen yliopiston tuloksissa riittää kumpi tahansa: koska tuloksissa ei ole päällekkäisyyttä, mistä rangaista, arvonvähennyksellä saadaan tismalleen samat tulokset kuin ilman arvonvähennystä.

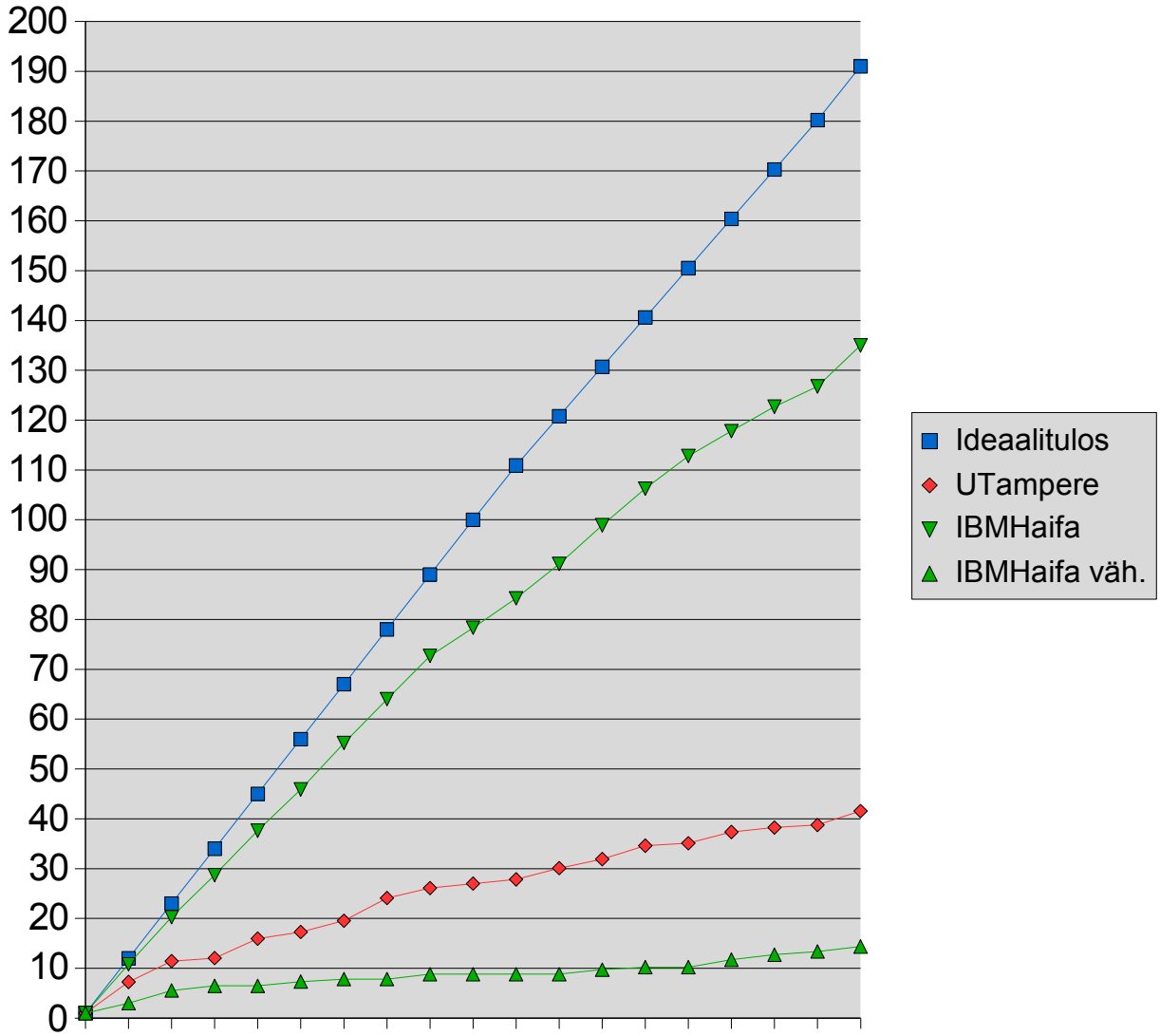
Poimin esimerkiksi kyselyaiheen 164. Aiheen relevanssiarviot ovat hyvin runsaita: täydellisiä (3, 3)-luokiteltuja elementtejä on yli 200. Yksin dokumentissa ex/1998/x3034 on 22 tällä luokituksella varustettua elementtiä (dokumentti on mielestäni arvioitu hieman huolimattomasti: tuntuu kummalliselta, että koko dokumentin spesifisyys voi olla 3, kun dokumentissa on kuitenkin vähemmän spesifejä osia). Ideaalisaantikantaan näistä elementeistä on valittu yksitoista parasta, jotka hyvin toimivan hakujärjestelmän pitäisi siis palauttaa.

IBM Haifan tuloslistalla on 46 eri elementtiä tästä dokumentista. Jo toisena elementtinä esiintyy dokumentin juurielementti, jonka jälkeen jokainen saman dokumentin elementti on päällekkäistä informaatiota ja hyödyltään kyseenalainen käyttäjälle. Tampereen yliopiston päällekkäisyyttä välttävässä tuloslistassa on yhdeksän elementtiä tästä dokumentista.

n	Ideaalitulos	UTampere	IBM Haifa	IBM Haifa väh.
10	10,00	6,75	8,80	2,00
50	50,00	16,05	41,15	6,48
100	100,00	27,00	78,35	8,83
200	191,00	41,55	135,00	14,37
keskiarvo	98,45	25,33	74,93	8,66

Taulukko 1. XCG-mittarin tulokset hakuaiheelle 164.

Taulukossa 1 nähdään XCG-mittarin tulokset hakuaiheen 164 osalta (tulokset ovat myös kuviossa 8). Taulukko esittää kertyneen hyödyn arvon tuloslistalla tietyissä elementtien määrän leikkauspisteissä; mitä suurempi arvo, sitä enemmän käyttäjä on siihen mennessä saanut hyötyä tuloslistan selaamisesta. Kuten voidaan huomata, 200 parhaimman elementin joukossa melkein kaikki ovat täydellisiä relevanssiarvon 1 elementtejä (maksimiarvo kertyneelle hyödylle 200 elementin kohdalla on 200).



Kuvio 8. XCG-mittarin tulosten kuvaajat hakuaiheelle 164.

Tampereen yliopiston tuloslista jää varsin paljon täydellisestä tuloksesta, kun taas IBM Haifan tulos on merkittävästi lähempänä ideaalitulosta. Todellisuudessa tämä hyvä tulos perustuu kuitenkin päällekkäisten elementtien sisällyttämiseen tulosjoukkoon. Strategiaa, jossa näin tehdään tarkoituksellisesti parempien hakutulosten saamiseksi, on ryhdytty INEXissä kutsumaan lypsämiseksi (*milking*) (Kazai ja Lalmas 2005, 23).

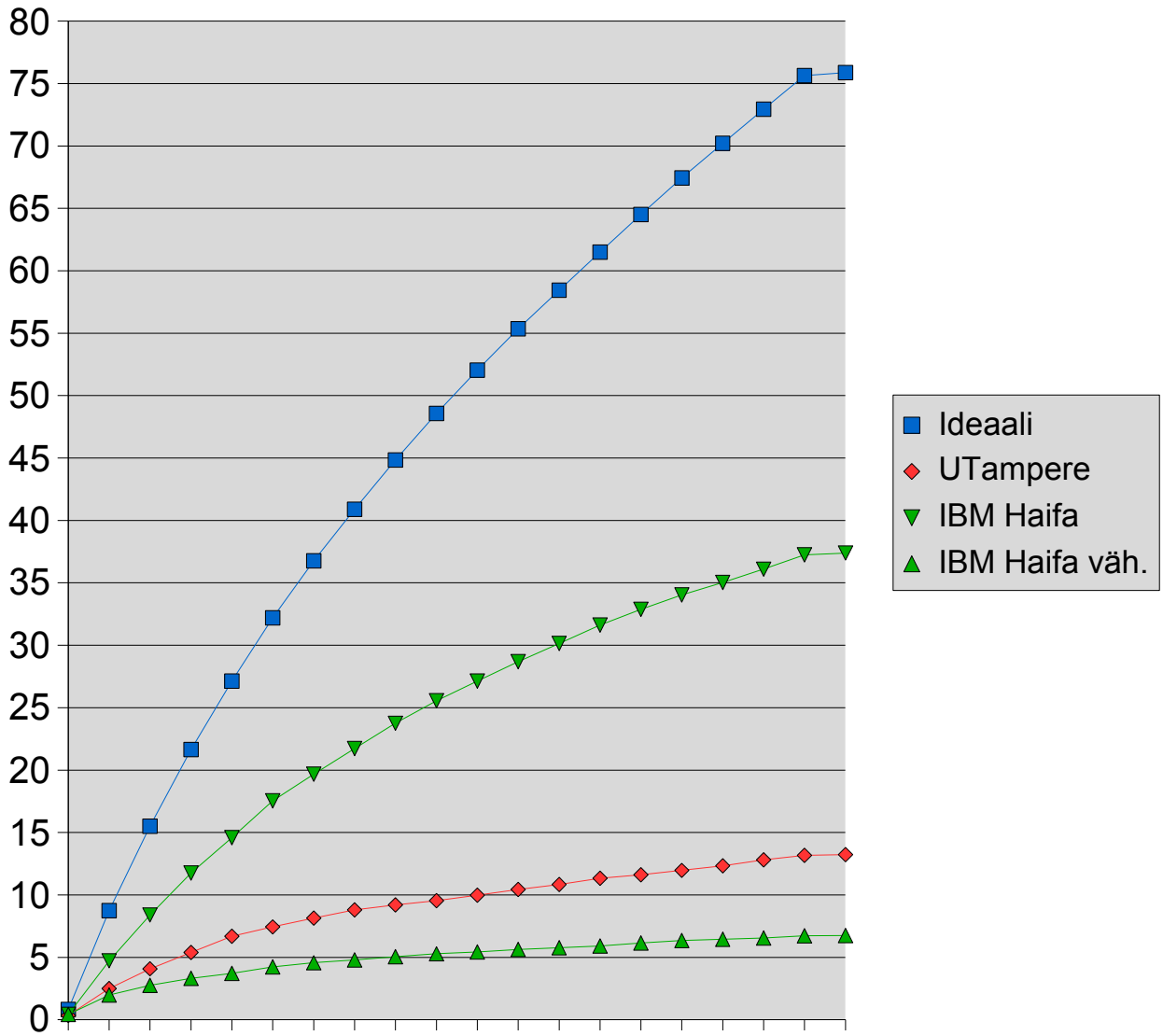
IBM Haifan tuloslistan todellinen tila paljastuukin, kun evaluointi suoritetaan arvonvähennysalgoritmia käyttäen (taulukossa kohta "IBM Haifa väh."). Tällöin päällekkäisistä elementeistä saatava hyöty häviää ja tulosten taso romahtaa. Kun kuvittelen itseni tiedonhakijan paikalle selaamaan tuloslistaa, pidän vähennettyä tulosta huomattavasti realistisempana kuvana käyttäjälle tulevasta hyödystä.

Tulosten luonne herättää ajatuksen: vähennyksessä voisi mennä jopa pidemmälle luvussa 2.5.2 esitellyn tyytyväisyys/turhautuneisuus-mitan tapaan ja sakottaa turhiksi käyneistä päällekkäisistä elementeistä, jotka lisäävät käyttäjän turhautuneisuutta. Tällainen ratkaisu vähentäisi IBM Haifan tuloslistan kaltaisen vahvasti päällekkäisen listan tuloksia vielä ankarammin.

Kyse ei ole pelkästään yhdestä yksittäisestä hakuaiheesta erikoisuuksineen, vaan sama kuvio toteutuu kaikissa hakuaiheissa. Taulukko 2 esittelee XCG-mittarin tulokset kaikkien hakuaiheiden tulosten keskiarvona eri leikkauspisteissä. Kuvio 9 esittää vastaavat kertynyt hyöty -kuvaajat, jotka muistuttavat pääpiirteiltään kuviossa 8 esiteltyjä, joskin IBM Haifan vähentämätön tulos ei ole aivan yhtä hyvä kuin hakuaiheen 164 kohdalla. Selvästikin runsaasti erittäin relevanteiksi arvioituja elementtejä sisältävä hakuaihe sopii hyvin päällekkäisiä elementtejä palauttavalle järjestelmälle, joka pystyy löytämään korkeasti relevanttia palautettavaa päällekkäisyyttä välttävää järjestelmää enemmän.

n	Ideaalitulos	UTampere	IBM Haifa	IBM Haifa väh.
10	7,42	2,36	3,96	1,81
50	29,52	7,10	15,91	3,95
100	48,58	9,55	25,55	5,29
200	75,88	13,24	37,40	6,74
keskiarvo	45,53	8,91	23,50	4,87

Taulukko 2. Kaikkien hakuaiheiden XCG-tulosten keskiarvo.



Kuvio 9. Kaikkien hakuaiheiden XCG-tulosten keskiarvo.

6 Johtopäätökset

6.1 XML-tiedonhakujen evaluoinnin ongelmat

6.1.1 Relevanssiarviot ja kaksiulotteinen relevanssi

XML-dokumentit tuovat uusia haasteita relevanssiarvioiden tekemiseen. Dokumentteja tarkastellaan hyvin monella eri tasolla ja niiden sisältämä relevantti sisältö nousee esiin eri tavoin. Luonnollisesti koko dokumentin kokoinen elementti sisältää kaiken dokumentin sisältämän relevanssin ja on siten käypä elementti palautettavaksi. Sama sisältö voi kuitenkin löytyä kokonaisuudessaan pienemmänkin elementin sisältä.

Tällaisessa tilanteessa en näe mitään syytä palauttaa koko dokumenttia. XML-dokumenttien rakenteesta ei saada mitään etua, mikäli sitä ei käytetä. Tarjoamalla käyttäjälle mahdollisuus hakea vain se osa dokumentista, joka sisältää relevantin tiedon säästää käyttäjältä aikaa ja vaivaa; se on tavoite, johon nykyajan ruuhkaisessa informaatiomaailmassa on mielekästä pyrkiä.

On kuitenkin riskialtista ajatella, että kaikki käyttäjät haluaisivat aina tarkimman elementin. On hyvin mahdollista, että jonkun tarpeisiin sopii parhaiten kokonaisten dokumenttien selailu; joissain tapauksissa koko dokumentin konteksti voi olla tarpeen aiheen ymmärtämiseksi, vaikka dokumentin sinänsä relevantti informaatio sisältyisikin pienempään osaan dokumenttia.

INEXin käytäntö arvioida dokumenttien relevanssia kahdella akselilla on toimiva lähestymistapa tähän ristiriitaan. Kattavuus kertoo sen, mistä kohdin dokumenttia relevantti sisältö löytyy. Koska säännön mukaan elementin yläelementin kattavuus on aina vähintään yhtä suuri kuin elementin itsensä kattavuus (koska ylempi elementti sisältää alemman elementin kokonaisuudessaan), kattavuus nousee kätevästi dokumentin hierarkiapuussa ylöspäin.

Tämän periaatteen johdosta dokumenttien kattavuudesta saa varsin hyvän kuvan. Dokumentin juurielementti kertoo sen, kuinka kattavasti aihetta dokumentissa käsitellään, sillä sen kattavuusarvo on dokumentin korkein arvo. Tarkastelemalla juurielementistä haarautuvien elementtipolkujen kattavuutta pystytään nopeasti selvittämään, missä osassa dokumenttia relevantti sisältö piilee. Kun kattavuusarvot laskevat alaspäin mentäessä

(pienemmät osat dokumenttia ovat todennäköisesti vähemmän kattavia kuin suuremmat), voidaan helposti tunnistaa ylin taso, jossa koko kattavuus on mukana.

Pelkkä kattavuus ei kuitenkaan riitä. Tarkasteltaessa kattavuutta ylimmän elementin palauttaminen on aina paras vaihtoehto, sillä dokumentin juurielementti on varmasti dokumentin kattavin elementti. Tässä tavallaan palataan perinteiseen dokumentin ja relevanssin ajatukseen, jossa dokumentilla on tietty, koko dokumentin kattava relevanssiarvo ja tiedonhakujärjestelmä palauttaa mahdollisimman relevantteja dokumentteja.

Siksi tarvitaan täydentävä ulottuvuus, spesifisyys. Se kertoo, kuinka tarkasti elementti käsittelee kaivattua aihetta. Toisin kuin kattavuudella, spesifisyydellä on tapana lisääntyä kun mennään alemmas hierarkiassa, kohti pienempiä elementtejä. Mitä pienemmästä elementistä on kysymys, sitä vähemmän siihen mahtuu irrelevanttia sisältöä.

Spesifisyyskään ei tietysti riitä yksinään. Hyvin pieni elementti voi sisältää erittäin täsmällisen tiedonhiukkasen, joka ei kuitenkaan ole riittävän kattava. Toimiva kokonaisuus saadaan, kun otetaan jonkinlainen ristileikkaus kattavuutta ja spesifisyyttä. Miten eri ulottuvuudet yhdistellään, onkin sitten eri asia. Paras lähestymistapa riippuu paljon siitä, mitä halutaan saavuttaa.

Erilaiset INEXissä käytetyt kvantisointifunktiot yhdistävät relevanssin ulottuvuuksia eri tavoin. Parhaimmalta vaikuttaa mielestäni *specificity-oriented generalised* -funktio, joka painottaa spesifisyyttä enemmän kuin kattavuutta; se korostaa hakujärjestelmän kykyä palauttaa pieniä ja tarkkoja elementtejä suurten ja kattavien sijasta. Täydellinen elementti on tietysti mahdollisimman kattava ja spesifi, mutta jos toisesta ulottuvuudesta on tingittävä, pelkkä kattavuus ei riitä.

Oli paras kvantisointifunktio sitten mikä tahansa, mahdollisuus yhdistää relevanssin ulottuvuudet halutulla tavalla yhdeksi relevanssipistemääräksi on hyvä asia. Se mahdollistaa erilaisten käyttäjäprofiilien mallinnuksen; mikäli huomataan, että käytetty tapa laskea relevanssipistemäärää ei vastaa käyttäjien tarpeita ja käyttäytymistä, koko mittaristoa ei tarvitse laittaa remonttiin. Riittää, että käyttäjän mallinnusta muokataan säätämällä kvantisointifunktion tai kertynyt hyöty -mitan parametrejä.

6.1.2 Elementtien päällekkäisyys

XML-tiedonhaussa elementtien päällekkäisyys tuloslistalla ja saantikannassa on ehdottomasti huomioimisen arvoinen tekijä. On vaikeaa kuvitella tilannetta, jossa käyttäjä iloitsisi saatuaan tuloslistan kärkijoukkoon nipun elementtejä yhdestä ja ainoasta dokumentista. Jos elementit ovat eri poluilta eli eri osista laajempaa dokumenttia, ongelmaa ei ole, mutta jos polut ovat päällekkäisiä, lopputulos ei varmastikaan auta käyttäjää.

Ongelmaan on sinänsä hyvin selkeä ja elegantti ratkaisu: tuloslistalle tulee päätyä saman polun elementeistä sen, jossa spesifisyys ja kattavuus kohtaavat käyttäjälle mieluisimmalla tavalla. Tämän tavan käyttäjä on ilmaissut kvantisointifunktion muodossa. Evaluoinnissa paras sijoitus kuuluu ehdottomasti tiedonhakujärjestelmälle, joka kykenee sijoittamaan listan kärkeen parhaimman dokumentin tämän kriteerin mukaan. Muista samalle relevantille polulle osuvista elementeistä voi palkita – ovathan ne lähellä ideaalitapausta – mutta useamman elementin palauttamisesta samalta polulta sietää jo antaa nollapisteet.

6.1.3 Ratkaisut ongelmiin

Sovelsin evaluointityökalussani Kazain, Lalmasin ja de Vriesin kehittämiä menetelmiä. Ideaalisaantikannan rakentamisessa seurasin täsmällisesti samaa menettelyä, tuloslistan päällekkäisyyden osalta vein ratkaisua hieman yksinkertaisempaan suuntaan. Molemmat toimivat tulosten valossa hyvin: mittarini antoi selvästi kehnompia tuloksia tuloslistoille, joissa oli voimakasta päällekkäisyyttä. Vähemmän päällekkäisyyttä sisältävät tuloslistat saivat paremmat pisteet. Tällaisista listoista on selvästi enemmän hyötyä käyttäjälle, joten tulos on oikea.

Ideaalisaantikannan muodostamisen idea on yksinkertainen: tarkistetaan kaikki relevantit polut, poimitaan paras elementti kultakin polulta ja tarkistetaan sen jälkeen, ettei näiden parhaiden elementtien kesken ole päällekkäisyyttä. Menetelmän toteuttaminen on helppoa ja se tuottaa vaivattomasti saantikannan, johon on valittu hakuaiheen kannalta parhaimmat elementit. Tässä saantikannassa ovat ne elementit, joiden kuuluisi olla tuloslistalla. Niiden puuttuminen tarkoittaa huonoa suoritusta hakujärjestelmältä.

Ideaalisaantikannan seurana on kuitenkin hyvä käyttää täyttä saantikantaa. Niukasti ideaalisaantikannasta karsiutunut elementti voi olla lähes täydellinen, jos samalta polulta löytyy pistearvoltaan täydellinen elementti. Tällaisen elementin palauttaminen ei ehkä ole ideaalipalautus, mutta käyttäjän kannalta kuitenkin hyvä, huomattavasti parempi kuin monen muun elementin palauttaminen. Relevantit elementit, jotka eivät kuitenkaan ole ideaalisaantikannassa ovat joka tapauksessa relevantteja, eikä niiden palauttamisesta tule missään tapauksessa rangaista hakujärjestelmää.

Kehittämäni ratkaisu päällekkäisyyteen tuloslistoilla perustuu jossain määrin samalle ajatukselle kuin Kazain, Lalmasin ja de Vriesin esittelemä. Tarkoituksena on tutkia, kuinka suuri osa käyttäjälle esiteltävän elementin sisällöstä on jo nähty aikaisemmissa elementeissä. Tämän nähdyn osion relevanssiarvo vähennetään sitten elementin ja sen alaelementtien yhteisestä relevanssiarvosta. Mitä useampi alielementti on jo nähty, sitä heikommaksi jää elementin pistearvo.

Ratkaisu on yksinkertainen, mutta vaikuttaisi kokeideni perusteella toimivan riittävän hyvin. Se sakottaa päällekkäisiä tuloksia tarjoavaa hakujärjestelmää tehokkaasti, samalla kun päällekkäisyyttä välttävä hakujärjestelmä saa täydet pisteet kaikista tuloslistan elementeistä. Vaikutuksen voi havaita selvästi luvussa 5.3.4 esitellyistä kuvioista 8 ja 9, joissa päällekkäisyyksiä sisältävän tuloslistan suoritus romahtaa, kun päällekkäisyydestä rangaistaan.

Kokeilemisen arvoisena ideana näkisin rakentaa arvonvähennysalgoritmiin mahdollisuuden säätää päällekkäisistä elementeistä annettavaa rangaistusta. Nyt kärsivällisyyselementtiä säätämällä on mahdollista lieventää rangaistusta päällekkäisyydestä, mutta toinenkin suunta olisi kiinnostava: jos käyttäjä kokee päällekkäiset tulokset ongelmallisiksi (mikä riippuu tietysti hyvin paljon siitäkin, kuinka tulokset esitetään, mutta ainakin suoran tuloslistan tapauksessa päällekkäisyys on turhauttava ongelma), niiden palauttamisesta voisi rangaista nollatulosta voimakkaammin vähentämällä pistearvoa. Tämä idea toteutuu tyytyväisyys/turhautuneisuus-mitassa, jossa ei-relevanttien dokumenttien palauttaminen vähentää pisteitä.

6.2 Yhteenveto

Pidän tämän tutkimuksen tehtyäni kertynyt hyöty -pohjaista mittaria oivallisena valintana INEXin tiedonhakujen evaluointiin. Se ei ole ainoa mahdollinen mittari, mutta toteuttaa ne kriteerit, jotka hyvälle mittarille asettaisin. Pohjimmiltaan kertynyt hyöty pureutuu oikeisiin asioihin tiedonhaun tuloksellisuuden mittaamisessa (mahdollisimman relevanttien dokumenttien palauttamiseen mahdollisimman lähellä tuloslistan huippua) ja sen esitetyt sovellukset XML-ympäristöön sekä hoitavat päällekkäisyysongelmat että antavat evaluointiin joustavuutta ja mukautuvuutta.

7 Lähteet

Borlund, P. 2000. Experimental components for the evaluation of interactive information retrieval systems. *Journal of Documentation*, 56 (1), s. 71-90.

Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. ja Yergeau, F. (toim.) 2004. *Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004.* [online] World Wide Web Consortium. Päivitetty 4.2.2004. Viitattu 19.4.2006. Saatavissa: <http://www.w3.org/TR/2004/REC-xml-20040204>.

Gövert, N. ja Kazai, G. 2003. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. [online] Teoksessa Fuhr, N., Gövert, N., Kazai, G. ja Lalmas, M. (toim.) *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, s. 1-17. Viitattu 19.4.2006. Saatavissa: <http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf>.

Gövert, N., Fuhr, N., Kazai, G. ja Lalmas, M. 2003. *Evaluating the effectiveness of content-oriented XML retrieval.* [online] Technischer bericht, University of Dortmund, Computer Science 6. Viitattu 2.4.2006. Saatavissa: <http://www.is.informatik.uni-duisburg.de/bib/fulltext/ir/Goevert_etal:03a.pdf>.

INEX 2004. *Relevance assessments DTD*. [online] Viitattu 13.3.2006. INEXin sisäiset sivut, ei julkisesti saatavissa.

Järvelin, K. 1995. *Tekstitiedonhaku tietokannoista*. Espoo: Suomen ATK-Kustannus.

Järvelin, K. ja Kekäläinen, J. 2000. IR evaluation methods for retrieving highly relevant documents. Teoksessa Belkin, N., Ingwersen, P. ja Leong, M.-K. (toim.) *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, s. 41-48. New York: ACM Press.

Järvelin, K. ja Kekäläinen, J. 2002. Cumulated Gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (ACM TOIS)*, 20 (4), s. 422-446.

Kazai, G. 2003. Report of the inex 2003 metrics working group. [online] Teoksessa Fuhr, Lalmas, Malik (toim.) *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, Dagstuhl, Germany, December 2003, s. 184-190. Viitattu 19.4.2006. Saatavissa: <http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf>.

Kazai, G., Lalmas, M. ja de Vries, A. 2004a. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. Teoksessa *SIGIR '04: Proceedings of the 27th annual international conference on research and development in information retrieval*, s. 72-79. Sheffield, UK: ACM Press.

Kazai, G., Lalmas, M. ja de Vries, A. 2004b. Reliability Tests for the XCG and inex-2002 metrics. [online] Teoksessa Fuhr, N., Lalmas, M., Malik, S. ja Szlávik, Z. (toim.) *INEX 2004 Workshop Pre-Proceedings*, s. 33-40. Viitattu 14.3.2006. Saatavissa: <http://inex.is.informatik.uni-duisburg.de:2004/pdf/INEX2004PreProceedings.pdf>.

Kazai, G., Lalmas, M., Fuhr, N. ja Gövert, N. 2004. A report on the first year of the INitiative for the Evaluation of XML retrieval (INEX'02), European research letter. *Journal of the American Society for Information Science and Technology* 55(6), s. 551-556.

Kazai, G. ja Lalmas, M. 2005. Notes on what to measure in INEX. [online] Teoksessa Trotman, A., Lalmas, M. ja Fuhr, N. (toim.) *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, July 2005, s. 22-38. Viitattu 11.3.2006. Saatavissa: <http://www.cs.otago.ac.nz/inexmw/Proceedings.pdf>.

Kekäläinen, J. ja Järvelin, K. 2002a. Evaluating information retrieval systems under the challenges of interaction and multidimensional dynamic relevance. Teoksessa Bruce, H., Fidel, R., Ingwersen, P. ja Vakkari, P. (toim.) *Proceedings of the 4th CoLIS Conference*, s. 253-270. Greenwood Village, CO: Libraries Unlimited.

Kekäläinen, J. ja Järvelin, K. 2002b. Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology*, 53 (13), s. 1120-1129.

Korfhage, R. 1997. *Information Storage and Retrieval*. New York: Wiley & Sons.

Lalmas, M. ja Malik, S. 2004. INEX 2004 Retrieval Task and Result Submission Specification. [online] Teoksessa Fuhr, N., Lalmas, M., Malik, S. ja Szlávik, Z. (toim.) *INEX 2004 Workshop Pre-Proceedings*, s. 237-240. Viitattu 14.3.2006. Saatavissa: <http://inex.is.informatik.uni-duisburg.de:2004/pdf/INEX2004PreProceedings.pdf>.

Myaeng, S. ja Korfhage, R. 1990. Integration of user profiles: Models and experiments in information retrieval. *Information Processing and Management* (26) 6, s. 719-738.

Pollack, S. 1968. Measures for the comparison of information retrieval systems. *American Documentation* 19 (4), s. 387-397.

Robertson, S. 1981. The methodology of information retrieval experiment. Teoksessa Sparck-Jones, K. (toim.) *Information retrieval experiment*, s. 9-31. Butterworths.

Rocchio, J. 1966. *Document retrieval systems – Optimization and evaluation*. Ph.D. diss., report no. ISR-10, Harvard Computation Laboratory, Harvard University.

Saracevic, T. 1995. Evaluation of evaluation in information retrieval. Teoksessa *Proceedings of the 18th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, s. 138-151. New York: ACM Press.

Saracevic, T. 1996. Relevance reconsidered. Teoksessa Ingwersen, P. ja Pors, N.O. (toim.) *Proceedings of the second international conference on conceptions of library and information science: Integration in perspective*, s. 201-218. Kööpenhamina: The Royal School of Librarianship.

Sigurbjörnsson, B., Larsen, B., Lalmas, M. ja Malik, S. 2004. INEX04 Guidelines for Topic Development. [online] Teoksessa Fuhr, N., Lalmas, M., Malik, S. ja Szlávik, Z. (toim.) *INEX 2004 Workshop Pre-Proceedings*, s. 212-218. Viitattu 14.3.2006. Saatavissa: <http://inex.is.informatik.uni-duisburg.de:2004/pdf/INEX2004PreProceedings.pdf>.

Sormunen, E. 2000. *A method for measuring wide range performance of Boolean queries in full-text databases*. [online] Väitöskirja. Tampereen yliopisto, Acta Electronica Universitatis Tamperensis 34. Viitattu 9.9.2005. Saatavissa: <http://acta.uta.fi/pdf/951-44-4732-8.pdf>

Sormunen, E. 2002. Liberal relevance criteria of TREC – Counting on negligible documents? Teoksessa *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR '02)*, Tampere, Finland, August 11-15, 2002, s. 324-330. New York: ACM Press.

TREC-2004 Web Track Guidelines. 2004 [online] Commonwealth Scientific and Industrial Research Organisation. Päivitetty 16.7.2004. Viitattu 24.1.2006. Saatavissa: <http://es.csiro.au/TRECWeb/guidelines_2004.html>

Trotman, A. ja Sigurbjörnsson, B. 2004. Narrow Extended XPath I (NEXI). [online] Teoksessa Fuhr, N., Lalmas, M., Malik, S. ja Szlávik, Z. (toim.) *INEX 2004 Workshop Pre-Proceedings*, s. 219-236. Viitattu 14.3.2006. Saatavissa: <http://inex.is.informatik.uni-duisburg.de:2004/pdf/INEX2004PreProceedings.pdf>.

Voorhees, E. 2001. Evaluation by Highly Relevant Documents. Teoksessa Croft, W.B. ja muut (toim.) *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, s. 74-82. New York: ACM Press.

Voorhees, E. ja Harman, D. 2001. Overview of TREC 2001. [online] Teoksessa *The Tenth Text REtrieval Conference (TREC 2001)*. Viitattu 2.4.2006. Saatavissa: <http://trec.nist.gov/pubs/trec10/papers/overview_10.pdf>.

W3C Communications Team. 2003. *XML 10 kohdan tiivistelmänä*. [online] Suomentanut Ossi Nykänen. W3C Suomen toimisto. Päivitetty 28.8.2003. Viitattu 7.4.2005. Saatavissa: <http://www.w3c.tut.fi/translations/xml/xmlin10pts/>.

8 Liitteet

Liite A: CAS-tyyppinen INEX-hakuaihe

```
<inex_topic query_type="CAS">
<title>
      //article[about(.//bb, Rumbaugh Jacobson Booch) and about(.//abs,
      formal methods logic)]//sec[about(., UML formal logic)]
</title>
<description>
      Find information on the use of formal logics to model or reason
      about UML diagrams.
</description>
<narrative>
      My main interest is the application of formal methods and logics in
      software development. I choose to search for its application to UML
      diagrams because I think it is an interesting application area. To
      be relevant, a document/component must discuss the use of formal
      logics, such as first-order-, temporal-, or description-logics, to
      model or reason about UML diagrams. I'm only interested in proper
      formal logics, Business-logics and Client-logics do not have a proof
      system and are therefore not considered to be formal logics. I think
      that sections are the most appropriate unit of retrieval for this
      fairly specific topic, since I'm not really interested in reading a
      lot about UML stuff in general. I want to focus in on the document
      parts that talk about logic. I think it is useful for the search
      engine to look for citation to the UML trio: Rumbaugh, Jacobson and
      Booch. Similarly think that it might be useful to put the formal
      methods constraints on the abstract to stress that I'm only
      interested in this particular subset of UML articles. Of course a
      relevant article need not have this sort of reference or abstract,
      therefore the relevance of an element will be judged on basis of how
      well it explains the use of formal logics to model or reason about
      UML diagrams.
</narrative>
<keywords>
      unified modeling language, first order logic, temporal logic,
      description logic, formal verification, theorem proving, model
      checking, Kripke structures, Spin, Promela, SMV
</keywords>
</inex_topic>
```

(Sigurbjörnsson ja muut 2004, 218.)

Liite B: CO-tyyppinen INEX-hakuaihe

```
<inex-topic query_type="CO">
<title>
      +"open standards" +"digital video" +"distance learning"
</title>
<description>
      We are looking for articles that discuss open standards behind media
      streaming and distribution of digital video that may be useful for
      distance learning projects.
</description>
<narrative>
      Our aim is to use our acquired knowledge and experience with digital
      video in a distance learning project. We prefer, however, to use
      software and methods that are based on open standards only. We are
      looking for articles/components discussing methodologies of digital
      video production and distribution that respect free access to media
      content through internet or via CD-ROMs or DVDs in connection to the
      learning process. Discussions of open versus proprietary standards
      of storing and sending digital video will be appreciated.
</narrative>
<keywords>
      media streaming, video streaming, digital video, distance learning,
      open standards, open technologies, free access
</keywords>
</inex-topic>
```

(Sigurbjörnsson ja muut 2004, 218.)

Liite C: TREXML_eval-ohjelman asetustiedosto

```
#TREXML_eval properties
#Thu Mar 14 14:18:06 EEST 2006
#
# Hakemisto, josta relevanssiarviot löytyvät
assessmentDir=/assessmentsII-3.0/
#
# Hakemisto, josta arvioitavat tuloslistat löytyvät
submissionDir=/submissions/utampere/
#
# Käytetty kvantisointifunktio
quantisationFunction=fSog
#
# Hakemisto, johon tulokset kirjoitetaan
resultDir=/TREXML/results/
#
# Tulosten esittämisformaatti
resultPattern=0.000
#
# Käytettävät tuloslistat sisältävän tiedoston nimi (johon
# lisätään pääte .xml)
submissionRunFile=0
#
# Käytettävä arvonvähennysalgoritmi
relevanceScoreReducer=TREXMLScoreReducer
#
# XML-parserin Java-luokan nimi
SAXParserName=org.apache.xerces.parsers.SAXParser
#
# Tulosten leikkauspiste
cutoffPoint=200
```