# XTerm-korpuskyselykielen kehittäminen ja korpuskyselykielten vertailu

Hanna Tuomisto

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu –tutkielma
Ohjaaja: Marko Junkkari
Kesäkuu 2012

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Hanna Tuomisto: XTerm-korpuskyselykielen kehittäminen ja korpuskyselykielten vertailu
Pro gradu -tutkielma, 78 sivua, 13 liitesivua
Kesäkuu 2012

---

Tässä työssä esitellään Tampereen yliopistossa monialaisena projektina toteutettu erityisesti termien louhintaan tarkoitettu korpuskyselykieli ja verrataan sitä muihin korpuskyselykieliin. Suurin osa korpuskyselykielistä on kehitetty yliopistoissa, eikä kaupallisia korpustyökaluja juurikaan ole saatavilla. Vertailtavat korpuskyselyjärjestelmät ja -kielet ovat: tekstihakujärjestelmä Emdros ja sen kyselykieli MQL, työkalusarja NITE XML Toolkit ja sen sisältämä kyselykieli NXT Query Language, IMS Corpus Workbench ja sen kyselykieli Corpus Query Processor Language sekä BNCweb-käyttöliittymän Simple Query Syntax-kyselykieli, joka sekin pohjautuu IMS Corpus Workbenchin Corpus Query Processoriin.

Korpuskyselykielten vertailu on tehty esittämällä kymmenen erilaista kyselyesimerkkiä ja tämän jälkeen esitetty jokainen näistä kullakin kyselykielellä. Esimerkkikyselyjen on tarkoitus kuvata niitä erilaisia tilanteita, joita kielentutkijalla saattaa olla ja joihin korpuskyselykielen tulisi vastata. Esimerkkikyselyissä kuvataan mallin sovittamista, säännöllisten lausekkeiden hyödyntämistä, termien louhintaa, tilastotietojen johtamista korpuskyselykielen avulla, kyselyn tulostusmuodon määrittelyä sekä tekstin rakenteen ja elementtien etäisyyksien määrittelyä ja rajaamista hauissa. Vertaillut korpuskyselykielet eroavat toisistaan huomattavasti. Kullakin korpuskyselykielellä on selvästi omat vahvuutensa ja heikkoutensa.

Tämän työn yhteydessä toteutetun XTerm-kyselykielen vahvuus on sen yksinkertaisuus ja soveltuvuus erityisesti helppoon ja nopeaan termien louhintaan. MQL-kyselykielen vahvuus on haettavien elementtien sisäkkäisyyden ja peräkkäisyyden kuvaamisessa. NXT-kyselykieli on ehkä vertailluista kyselykielistä monipuolisin ja mahdollistaa hyvin monenlaiset haut, mutta vaatii käyttäjältä tietojenkäsittelyyn liittyvien periaatteiden ymmärtämistä ja opiskelua. Corpus Query Processor-kyselykieli on sekin monipuolinen ja selvästi kielitieteellisiin tarkoituksiin soveltuva. Simple Query Syntaxin vahvuus on sen yksinkertaisuus. Se on helppokäyttöinen ja nopeasti opittava ja se mahdollistaa kuitenkin monet yleiset kielentutkijan hakutilanteet. On mahdotonta yksiselitteisesti valita parasta ja helppokäyttöisintä korpuskyselykieltä. Yleistäen voidaankin todeta, että mitä monipuolisempi korpuskyselykieli on ominaisuuksiltaan, sitä enemmän se vaatii käyttäjältään opiskelua tai jo ennalta tietojenkäsittelyyn liittyvien paradigmojen ymmärtämisen.

Avainsanat: korpus, korpuskyselykieli, kielitiede, XML, XPath, XQuery.

## Sisällys

# 1 Johdanto

Korpuksiin perustuva kielitiede (corpus linguistics) on vielä suhteellisen nuori tieteenala, vaikka ensimmäisiä korpuksia alettiin kerätä ja muodostaa jo 1960-luvulla. Sivistyssanakirja [2003] määrittelee korpuksen olevan ”kielitieteessä tutkimus- tai vertailuaineistona käytettävä kokoelma kirjoitettua tai äänitettyä kielen ainesta”. Korpuslingvistiikka tarkoittaa kielitieteen tutkimuksia, jossa tietoa kerätään suurista tietomassoista, eli korpuksista. Tutkimusten perusteella voidaan esimerkiksi tehdä johtopäätöksiä sanojen esiintymisistä eri tilanteissa ja sanojen eri käyttötarkoituksista.

Tietokone on muuttunut olennaiseksi työvälineeksi kielitieteen tutkimuksissa viimeisen kahdenkymmenen vuoden aikana, mutta vasta aivan viimeisten vuosien aikana yleinen tekninen kehitys on mahdollistanut tehokkaan korpustutkimuksen. Nykyisin voidaan tallentaa erittäin suuria tekstimassoja ilman merkittäviä taloudellisia investointeja. Erilaiset haut voidaan suorittaa nopeasti, vaikka kyseessä olisi tuhansien tai kymmenien tuhansien sanojen tekstikorpukset. Lisäksi www-pohjaisten ohjelmien yleistyminen ja monet www-tekniikat sekä parantuneet tietoliikenneyhteydet mahdollistavat ohjelmien käyttämisen nopeasti ja vaivattomasti mistä tahansa. Näin korpukset eivät jää vain yhden yliopiston tutkijoiden ja opiskelijoiden käyttöön.

Korpustutkimuksiin on saatavilla sekä kaupallisia että avoimen lähdekoodin ohjelmia. Näiden käyttö ei kuitenkaan ole aina yksinkertaista. Ohjelma saattaa rajoittaa tekstin muotoilua ja merkintätapoja. Lisäksi se saattaa olla liian suppea tai liian laaja tiettyihin tutkimustavoitteisiin. Korpustutkimukseen liittyvien ohjelmien käytössä ongelmana on myös se, että ohjelmat antavat mahdollisuuden tutkia vain tiettyjä ennakolta tiedettyjä asioita ja rajaavat näin kielentutkijoiden mahdollisuuksia tehdä omia uusia löydöksiä. Useimmat olemassa olevat korpustutkimuksiin soveltuvat ohjelmistot ovat innokkaiden amatööriharrastajien kehittelemiä, kuten WordSmith (http://www.lexically.net/wordsmith/), tai tekstinkäsittelyyn perehtyneiden tietojenkäsittelyn tutkijoiden kehittelemiä, kuten IMS Corpus Workbench. Kaupalliset ohjelmistot ovat harvinaisia [Mason, 2008].

Tässä tutkimuksessa esitellään Tampereen yliopistossa kehitelty korpuskyselykieli XTerm. Kyselykieli on kehitelty kieltentutkijoiden tarpeiden pohjalta projektissa, jossa oli mukana informaatiotutkimuksen, tietojenkäsittelytieteen ja kielitieteen tutkijoita. Kielitieteen tutkijoilla oli tarve poimia suomen ja venäjän kieltä sisältävästä lakitekstikokoelmasta termiehdokkaita eli kahden tai useamman sanan yhteisesiintymiä. Termiehdokkaita voidaan poimia tekstistä sanojen perusmuotojen eli lemmojen tai sanoihin liitettyjen kielitieteellisten merkintöjen avulla. Korpus on tallennettu XML-tiedostoiksi ja jokainen sana on merkattu sanaluokkainformaatiolla ja sanan lemmoilla. XML kuvaa tieto-objekteja, joita kutsutaan XML-dokumenteiksi. XML-dokumentit koostuvat elementeistä, jotka sisältävät joko jäsennettyä tai jäsentämätöntä dataa.

Jäsennetty data koostuu merkkidatasta ja merkinnöistä. Merkintöjen avulla kuvataan dokumentin tallennusmalli ja looginen rakenne. XML tarjoaa mekanismin, jolla voidaan rajoittaa tallennuksen esittämistä ja loogista rakennetta. [W3C, 2006.]

XTerm-kyselykielen suunnittelussa on huomioitu erityisesti termien louhinnan tarpeet. Kieli on suunniteltu helpoksi ja nopeaksi työvälineeksi kielen tutkijalle. Kieli on yksinkertainen ja sillä on rajattu ilmaisuvoima. Kielitieteilijöiltä saadun palautteen perusteella se soveltuu tarkoitukseensa erittäin hyvin eikä vaadi käyttäjältään esimerkiksi erilaisten tietokanta- tai ohjelmointikielten käsitteiden ja rakenteiden ymmärtämistä, toisin kuin osa muista korpuskyselykielistä. Muista korpuskyselykielistä poiketen XTerm sisältää projektion, joka antaa käyttäjälle mahdollisuuden vaikuttaa tulosten esittämismuotoon.

Tässä työssä perehdytään siihen, millaisia korpuskyselykieliä olemassa olevissa ohjelmissa käytetään, millaisia ominaisuuksia korpuskyselykielillä on, ja miten kielet eroavat toisistaan. Vertailtavat korpuskyselykielet ovat Emdros-järjestelmän MQL-kyselykieli, NITE XML ToolKit -järjestelmän NXT-kyselykieli, IMS Corpus Workbenchin kyselykieli Corpus Query Processor Language sekä Simple Query Syntax, joka sekin toimii IMS Corpus Workbenchin päällä. Yksikään näistä kielistä ei ole kaupallinen, vaan ne kaikki on kehitetty yliopistojen ja julkisten organisaatioiden varoin. Sekä Emdrosin, NITE XML ToolKit -ohjelmiston että IMS Corpus Workbenchin kehittäminen jatkuu tällä hetkellä avoimen lähdekoodin ohjelmana. Kielten vertailu on toteutettu kartoittamalla erilaisia korpuksiin liittyviä tarpeita ja muodostamalla erilaisia esimerkkikyselyjä, joiden avulla kielten erot pyritään havainnollistamaan. Vertailun pohjalta pyritään arvioimaan korpuskyselykielten tärkeimpiä ominaisuuksia sekä kielten hyödynnettävyyttä kielitieteellisissä tutkimuksissa ja erityisesti termien louhinnassa.

Työn toisessa luvussa kerrotaan lyhyesti, mitä korpukset ovat ja perehdytään korpusten käyttöön ja korpustutkimukseen. Saman luvun lopussa esitellään lyhyesti, mitä korpuskyselykielet ovat ja mitä vaatimuksia niille voidaan asettaa. Luvussa kolme esitellään tämän työn yhteydessä toteutettu erityisesti termien louhintaan suunniteltu korpuskyselykieli XTerm. Seuraavassa luvussa kerrotaan, miten XTerm on teknisesti toteutettu. Muiden vertailtavien korpuskyselykielten perusrakenteet ja tärkeimmät ominaisuudet esitellään luvussa viisi. Luku kuusi sisältää varsinaisen kielten vertailun ja johtopäätökset tehdään luvussa seitsemän. Työn yhteenveto on viimeisessä, kahdeksannessa luvussa.

# 2 Korpukset ja niiden hyödyntäminen

## 2.1 Korpus

Korpus on elektroninen tekstikokoelma, joka voi sisältää joko kirjoitettua tai puhuttua kieltä. Korpus sisältää luonnollista kieltä, joka on valittu kuvaamaan kielen tilaa tai vaihtelua. Korpukset voidaan koota vain tietyn aihepiirin teksteistä tai ne voidaan tarkoituksella muodostaa sellaisiksi, että ne sisältävät mahdollisimman erityyppisiä ja eri aihealueiden tekstejä. Korpuksen käyttötarkoitus vaikuttaa siihen, millaista materiaalia korpukseen kerätään. Modernissa laskennallisessa kielitieteessä korpus tyypillisesti sisältää miljoonia sanoja. Kielen luova käyttö johtaa niin äärettömään ilmaisujen vaihteluun, että pienemmästä aineistosta on vaikeaa rajata toistuvia malleja, joiden avulla voitaisiin saada vihjeitä kielen sanastollisesta rakenteesta [Sinclair, 1991].

Korpukselle löytyy kirjallisuudesta useita määritelmiä. McEnery ja Wilson [1996, 2001] esittävät kolme erilaista korpusmääritelmää: ”(1) (loosely) any body of text; (2) (most commonly) a body of machine-readable text; (3) (more strictly) a finite collection of machine readable text; sampled to be maximally representative of a language or variety.” Näistä Pearson [1998] valitsee sopivimmaksi kolmannen, eli korpus on määrätty, koneluettavaa tekstiä sisältävä kokoelma, joka on koottu sellaiseksi, että se edustaa mahdollisimman hyvin kielen vaihtelua. Tämä määritelmä sopii myös sellaisiin tapauksiin, joissa korpusta ei ole tarkoitettu vain kielitieteellisiin analyyseihin vaan myös esimerkiksi luonnollisen kielen prosessointijärjestelmiin.

Ensimmäinen konelukuinen tekstikorpus oli Brownin yliopistossa Yhdysvalloissa muodostettu Brownin korpus. Korpuksen muodostamiseen liittyvä hanke aloitettiin vuonna 1963. Jo tätä ensimmäisestä konelukuista korpusta käytettiin esimerkiksi frekvenssilaskelmiin, morfologisiin tutkimuksiin ja korrelaatioanalyyseihin. [Lehtinen et al., 1996.] Muita merkittäviä korpuksia ja korpushankkeita ovat British National Corpus (http://www.natcorp.ox.ac.uk/), Bank of English (http://www.titania.bham.ac.uk/) ja Corpus of Contemporary American English (http://corpus.byu.edu/coca/).

## 2.2 Korpustyypit

Korpuksia voidaan luokitella monella eri tavalla, esimerkiksi käyttötarkoituksen ja kattavuuden perusteella. Korpusta, jonka tarkoitus on tarjota kokonaisvaltaista informaatiota kielestä, ja joka pyrkii olemaan riittävän suuri edustaakseen kielen kaikkia olennaisia vaihteluja, kutsutaan referenssikorpukseksi (general reference corpus). Näytekorpus (sample korpus) on lopullinen kokoelma tekstejä, jotka on valittu huolellisesti ja tutkittu yksityiskohtaisesti. Kun näytekorpus on muodostettu, siihen ei lisätä mitään eikä sitä muokata jälkeenpäin [Sinclair, 1991]. Hoffmann [2008] käyttää tällaisesta lopullisesta korpuksesta nimitystä staattinen korpus. Sen sijaan

seurantakorpuksella (monitor corpus) ei ole lopullista tilaa vaan se kehittyy ja muuttuu kuten kieli itse. Sen sisältö on jatkuvasti päivityksen ja lisäysten kohteena. Tällöin suurin osa materiaaleista kerätään lähteistä, jotka voidaan lukea koneellisesti ja tarkastella rutiininomaisesti. Materiaalia kerättäessä huomion keskipisteenä tulee olla se, millaista informaatiota teksteistä on tarkoitus poimia ja miten tekstit vastaavat tutkijoiden aihepiirejä. Tekstinäytteet voidaan ottaa tutkijoiden ja tutkimusten tarpeiden perusteella valtavista jatkuvasti muuttuvista tekstivarastoista ja todisteet kielen kehityksestä voidaan pitää hallussa tehokkaasti. Seurantakorpuksella voidaan tuottaa sellaista informaatiota, jota näytekorpuksella ei saada aikaan. Tasapainoisen tekstivirran saavuttaminen on vaikeampaa kuin näytekorpuksissa. [Sinclair, 1991.] Joissain tapauksissa vanhoja tekstejä saatetaan poistaa korpuksesta [Hoffmann, 2008].

Luokittelu voi perustua myös ajalliseen ulottuvuuteen, jolloin korpus voidaan määritellä synkroniseksi tai diakroniseksi. Synkroninen korpus sisältää tekstiä vain tietyltä ajanhetkeltä kun taas diakronisen eli tapahtumien aikajärjestystä tarkkailevan korpuksen on tarkoitus kattaa pitkä ajanjakso. Täten diakroninen korpus soveltuu hyvin kielen muutoksen ja kehityksen tutkimiseen. Sen sijaan diakroniset korpukset saattavat olla liian suppeita antamaan kuvaa kielen erilaisista käyttötilanteista, koska valinnassa on kiinnitetty huomiota pääasiassa tekstien ajalliseen kattavuuteen. [Hoffmann, 2008.]

Korpus voi koostua kokonaisista teksteistä, esimerkiksi kirjoista tai artikkeleista, tai vain tekstinäytteistä. Kun korpus sisältää kokonaisia, eripituisia tekstejä, saattavat esimerkiksi tietyn kirjoittajan tyyli ja ominaispiirteet olla yliedustettuina korpuksessa. Koottaessa korpukseen tekstinäytteitä ongelmaksi voi nousta se, että elementit, jotka vain harvoin esiintyvät tietyn tyyppisissä teksteissä, eivät tule riittävän edustetuiksi korpuksessa. [Hoffmann, 2008.]

Korpus voi sisältää puhdasta tekstiä (plain) tai se voi olla merkattu (marked-up, annotated). Suurin osa korpuksista sisältää jonkinlaista merkintää. Merkitsemisen avulla voidaan korpukseen liittää erilaista metadataa ja liittää mukaan kielitieteellistä informaatiota (linguistic annotation) tekstistä sekä tekstin rakenteesta. Metadataa voi olla esimerkiksi tieto tekstin tuottajasta. [Hoffmann, 2008.] Tekstin tuottajasta voidaan kertoa esimerkiksi ikä, sukupuoli, ammatti tai muita seikkoja, jotka saattavat olla kiinnostavia tekstin kannalta. Tekstin rakenteesta saatava informaatio kertoo kappaleiden rakenteesta, otsikoista, lauseista ja niin edelleen. Puhutun tekstin merkinnöissä saatetaan ilmaista puhujien vaihdokset, häiriöt tai puheiden päällekkäisyys. Kielitieteellinen informaatio sisältää yleensä lisätietoja, jotka ovat jo jonkin analyysin tulosta. Kielitieteellisen informaation ilmaisemiseen käytetään usein tunnisteita (tags). Ne voidaan lisätä tekstiin joko manuaalisesti tai tietokoneohjelman avulla. Merkinnät voivat sisältää tietoa esimerkiksi sanaluokista tai lauseenjäsenistä.

## 2.3 Korpusten kokoaminen

Korpuksen keräämisvaiheessa on huomioitava korpukselle asetetut vaatimukset ja korpuksen käyttötarkoitus. Nämä asiat vaikuttavat siihen, mitä aineistoa korpukseen kerätään ja mitä rajataan pois. Korpusten muodostamisessa ongelmana on se, että korpuksesta tulisi muodostaa mahdollisimman kattava. Kuitenkin usein vasta tekstien tutkiminen ja niiden analysoiminen, mahdollistaa tekstien perusteellisen luokittelun [Lehtinen et al., 1995].

Jos korpusta on tarkoitus käyttää esimerkiksi yleiskieltä selittävien sanakirjojen kokoamisessa, on tarkoituksenmukaista kerätä korpukseen neutraalia yleiskieltä, joka ei ole suunnattu vain asiantuntijoille. Puhuttua kieltä on kerättävä silloin, jos halutaan tarkastella esimerkiksi yleiskielen normijärjestelmää kokonaisuutena. [Lehtinen et al., 1995.] Erilaisten tekstien kerääminen vaatii erilaisia huomioita. Esimerkiksi lehtitekstejä kerättäessä tulee pohtia, luokitellaanko artikkelit lehden osastojaon mukaan vai luokitellaanko ne jollakin muulla tavalla korpukseen.

Korpuksen koko vaikuttaa muun muassa siihen, millaisia tilastollisia analyyseja siitä voidaan muodostaa. Pienet korpukset riittävät esimerkiksi tekstilajien tilastollisiin analyyseihin ja morfologisiin tutkimuksiin. Sen sijaan semantiikan, sanojen frekvenssien, nykymerkitysten ja -kontekstien selvittämiseen, esimerkiksi sanakirjojen kirjoittajille, aineiston on oltava merkittävästi suurempi. Korpukseen kerätyt kokonaiset tekstit mahdollistavat erilaisia tekstilingvistisiä tutkimuksia kuten koheesion, koherenssin ja intertekstuaalisuuden tutkimisen. [Lehtinen et al., 1995] Edellä esitettyjen asioiden lisäksi korpuksen suunnittelussa on huomioitava esimerkiksi erilaiset lupa-asiat, tallennusmuoto sekä tekstin prosessointi ja siihen valittavat työkalut ja ohjelmistot.

## 2.4 Korpusten merkitseminen

Kuten jo edellä mainittiin, korpukset voivat olla joko puhdasta tekstiä tai varustettu lisäinformaatiolla. Lisäinformaatio parantaa korpusten käyttömahdollisuuksia. Se samaan aikaan sekä mahdollistaa monipuolisempien hakujen tekemisen että tekee hauista nopeampia ja luotettavampia. Korpusten koodaus- ja merkintämenetelmät voivat olla korpuskohtaisia, vaikka niihin on kehitelty suosituksia. Usein myös hakuohjelmat ovat korpuskohtaisia. Itse korpukset eivät kuitenkaan yleensä ole laitteisto- tai käyttöjärjestelmäriippuvaisia.

Korpukseen liitettävä informaatio voidaan jakaa kolmeen tasoon. Ensimmäiseen kuuluu asemallinen (positional) informaatio, esimerkiksi sanaluokkainformaatio, morfosyntaktinen informaatio tai tieto sanojen perusmuodoista. Toinen taso kattaa rakenteellisen informaation eli esimerkiksi fraasien, lauseiden ja kappaleiden merkitseminen. [Christ et al., 1999.] Nämä kaksi tasoa on merkitty myös tämän tutkimuksen kohteena olevaan lakitekstiä sisältävään XTerm-korpukseen.

Tutkimuksessa kehitetty ja toteutettu XTerm-kyselykieli hyödyntää sekä asemallista informaatiota että rakenteellista informaatiota. Sen sijaan Christin ja muiden [1999] mainitsema kolmas, monikerroksellinen taso puuttuu XTerm-korpuksesta. Monikerroksellisella tasolla korpus voidaan merkitä suhteessa toiseen korpukseen ja tämä kolmas taso voidaan nähdä yleistyksenä rakenteellisesta informaatiosta. Lehtinen et al. [1995] esittävät, että tekstin koodaamisen (vertaa merkkaus) ohjeiston tulisi:

- olla yleinen, joustava ja laajennettava,
- tarjota standardoitu muoto, joka mahdollistaa tekstin vaivattoman siirtämisen laitteesta ja käyttöjärjestelmästä toiseen ja sen käyttämisen uudessa ympäristössä,
- esittää ne yhtenäiset periaatteet, joiden mukaisesti teksti koodataan ja
- esittää yhtenäiset tavat erilaisten tekstien erilaisten piirteiden merkitsemiseen.

McEnery ja Wilson [2001] listaavat alunperin Leechin vuonna 1993 mainitsemat perusohjeet tekstikorpusten lisäinformaation merkitsemiseen. Korpuksesta tulisi voida poistaa lisäinformaatio tai muutoin tarkastella korpuksen puhdasta tekstiä ilman lisämerkintöjä. Sekä tutkimuksen kohteena olevan korpuksen että monien muiden korpusten tallennusmuotona käytetty XML-muoto mahdollistaa tämän erittäin hyvin. Käyttäjä voi halutessaan tarkastella joko tekstiä tai sekä tekstiä että XML-elementtien ja niiden attribuuttien sisältämää lisäinformaatiota. Lisäinformaatiota tulisi voida laajentaa muista lähteistä, esimerkiksi relaatiotietokannoista. Huomioimalla edellä mainitut kohdat voidaan todeta, että korpuksen lisäinformaation tulisi olla käyttäjän muokattavissa. Korpuksesta tulee tehdä merkintäsuunnitelma (annotation scheme). Loppukäyttäjällä tulee olla käytössään tieto siitä, millaista lisäinformaatiota korpus sisältää ja miten sitä voidaan hyödyntää. Loppukäyttäjän tulee myös tietää miten lisäinformaatio on syntynyt. Olennaista voi olla esimerkiksi, onko merkinnät tehty manuaalisesti vai koneellisesti. Käyttäjälle tulee tehdä selväksi, etteivät lisämerkintätyökalut ole erehtymättömiä, mutta ovat silti kuitenkin hyödyllisiä. Merkintäsuunnitelmien tulee perustua mahdollisimman pitkälle laajasti hyväksyttyihin ja teorianeutraaleihin periaatteisiin.

Korpusten merkkauksessa ristiriita saattaa muodostua sen välille, mikä olisi loppukäyttäjälle hyödyllisintä ja sen, mikä olisi merkitsijälle helpointa. Aina ei ole mahdollista toteuttaa kaikkea ajan tai rahan puutteesta johtuen. Esimerkiksi korpuksen manuaalista läpikäyntiä vaativat merkinnät voivat olla mahdottomia toteuttaa. Useimmat merkintäsuunnitelmat ovatkin näiden kahden asian kompromissi. [McEnery and Wilson, 2001.]

### 2.4.1 Kielitieteelliset merkinnät

Kuten on jo mainittu, korpuksiin voidaan liittää monenlaista kielitieteellistä informaatiota, joka lisää korpuksen käyttömahdollisuuksia. Kun tekstiin liitetään määrättyjen koodien avulla kielitieteellistä informaatiota, kutsutaan tätä leimaamiseksi tai merkitsemiseksi (tagging) mieluummin kuin annotoinniksi eli sisällön kuvailuksi. Koodeja, jotka tekstiin liitetään, kutsutaan tunnisteiksi (tags). Kielitieteellisiä merkintöjä voivat olla sanaluokkamerkinnät sekä lemmatisointiin, jäsentämiseen, semantiikkaan, diskursiiviseen ja lingvistiseen merkintään, foneettiseen puhtaaksikirjoitukseen, prosodiaan ja ongelmasuuntautuneeseen leimaamiseen liittyvät merkinnät. [McEnery and Wilson, 2001.]

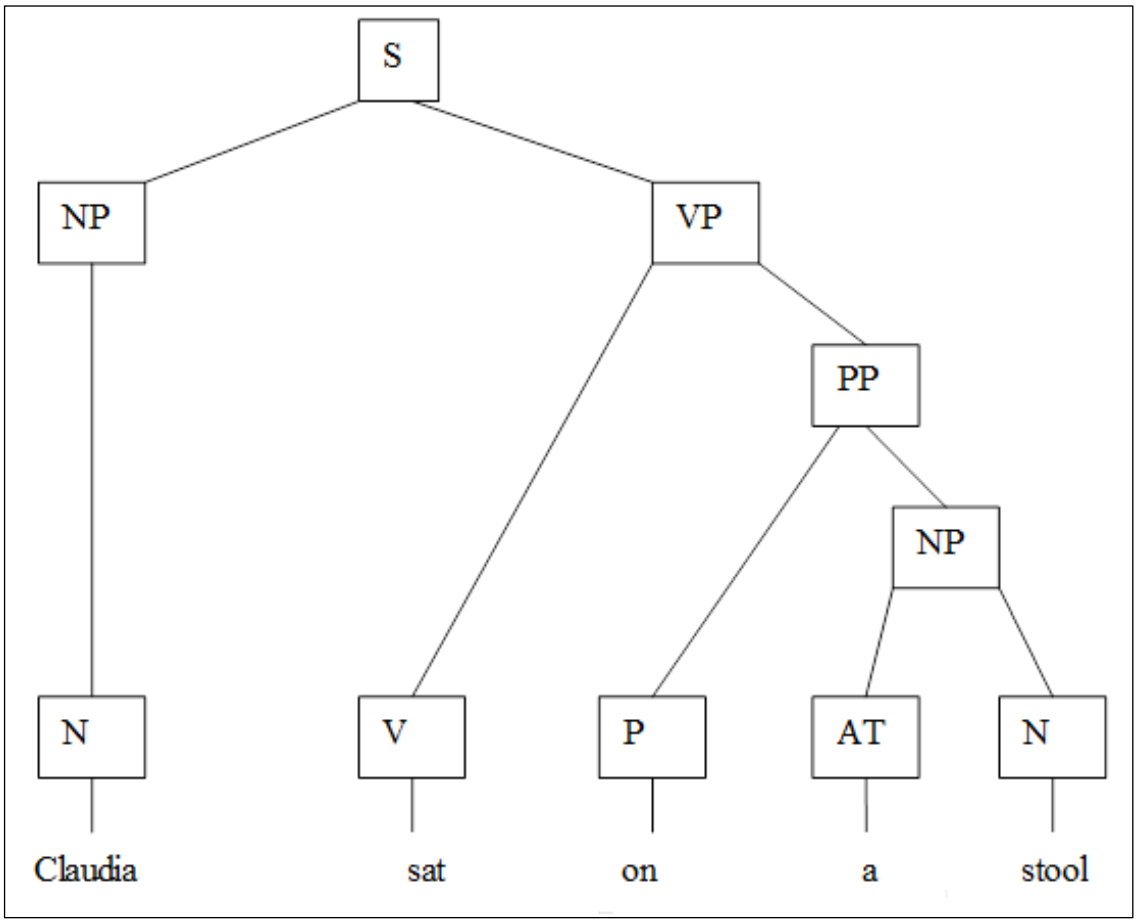
*Sanaluokkamerkinnät* ovat yleisimpiä kielitieteellisiä korpusmerkintöjä, jotka tunnetaan myös nimellä kieliopillinen leimaaminen tai morfosyntaktinen merkintä. Tarkoituksena on liittää jokaiseen tekstin sanastolliseen yksikköön sen sanaluokka. Sanaluokkamerkinnät lisäävät merkittävästi korpustiedonhakujen tarkkuutta ja muodostavat olennaisen perustan myöhemmille analyyseille kuten lauserakenteelliselle jäsentämiselle (syntactic parsing) ja semanttiselle osa-aluemerkkaukselle (semantic field annotation). Sanaluokkamerkintä voidaan liittää sanojen perään alaviivalla, kuten lovely_JJ eyes_NNS. [McEnery and Wilson, 2001.] Sanaluokkainformaatio oli ensimmäisiä korpuksiin liitettyjä merkintöjä ja on tällä hetkellä myös yleisin näistä merkinnöistä. Riittävän tarkka sanaluokkainformaatio voidaan yleensä muodostaa ilman manuaalista käsittelyä. [McEnery and Wilson, 2001.] Leimaamisohjelmia ovat esimerkiksi CLAWS (http://ucrel.lancs.ac.uk/claws/) ja Amalgam (http://www.scs.leeds.ac.uk/amalgam/amalgam/amalghome.htm). Brownin korpus oli alun perin merkattu TAGGIT-ohjelman avulla.

Sanaluokkainformaation tulisi olla helppokäyttöistä. Esimerkiksi tunnisteiden merkitys tulisi olla selvä ilman, että niitä pitää tarkistaa mistään. Sanaluokkainformaatiota merkittäessä muodostuu ristiriita sen välillä, milloin saavutetaan suurin hyöty käyttäjälle ja milloin taas saavutetaan mahdollisimman suuri tarkkuus [McEnery and Wilson, 2001]. EAGLES-projektin (http://www.ilc.cnr.it/EAGLES/home.html) yhteydessä on tehty suosituksia sanaluokkatunnisteista Euroopan kielille. Suosituksissa ominaisuuksilla on kolme tasoa: 1) pakolliset ominaisuudet, jotka tulee olla olemassa aina kun sanaluokkainformaatiota merkitään, 2) suositellut ominaisuudet, jotka tulee merkitä, jos vain mahdollista, sekä 3) valinnaiset ominaisuudet, joita voidaan käyttää yksityiskohtaisissa tilanteissa. Pakollisiin ominaisuuksiin kuuluvat pääsanaluokat kuten substantiivit, verbit, adjektiivit, pronominit. Suositeltaviin ominaisuuksiin kuuluvat esimerkiksi substantiiveille merkittävät, luku (number), suku (gender), sija (case) ja tyyppi (type). Valinnaisia ominaisuuksia on kahdenlaisia. Yleiset ominaisuudet, joita on kaikissa kielissä, sekä vain tietyissä kielissä esiintyvät ominaisuudet. Yleisiä ominaisuuksia ovat

esimerkiksi substantiivien alakategoriat. Vain tietyissä kielissä olevia ominaisuuksia ovat esimerkiksi määräisyys, joka käsittelee loppuliitteisiä määräisiä artikkeleja tanskan kielen substantiiveissa. [McEnery and Wilson, 2001.]

*Lemmatisointi* tarkoittaa tekstissä esiintyvien sanojen lekseemien löytämistä. Lekseemi on sanan perusmuoto, jota käytetään esimerkiksi sanakirjoissa. Sanan muita esiintymismuotoja kutsutaan lekseemin lemmoiksi. Lemmatisointi on korpusperustaisessa tutkimuksessa tärkeä toimenpide. Sanastotutkimuksessa (vocabulary studies) ja sanakirjatyössä (lexicography) lemmatisointi antaa tutkijalle mahdollisuuden poimia ja tarkastella kaikkia tietyn lekseemin muunnoksia tarvitsematta syöttää kaikkia mahdollisia sanan esiintymismuotoja hakujärjestelmään. Lisäksi lemmatisoinnin avulla voidaan muodostaa lekseemin frekvenssit ja jakaumat. [McEnery and Wilson, 2001.]

Kun morfosyntaktiset kategoriat on tunnistettu, on mahdollista tuoda näihin kategorioihin korkeamman tason syntaktisia suhteita toisiinsa. Toimintoa, joka tämän tekee, kutsutaan *jäsentämiseksi* (parsing). Jäsentäminen on yleisin korpusmerkintätapa sanaluokkainformaation merkitseminen jälkeen. Korpuksia, jotka on jäsennelty, kutsutaan englanniksi myös nimellä treebanks. [McEnery and Wilson, 2001.] Esimerkiksi lauseen ’Claudia sat on a stool’ rakenne voidaan jäsentää ja kuvata alla olevalla puudiagrammilla.




Kaavio 1. Korpuslauseen jäsentäminen puurakenteena [McEnery and Wilson, 2001]

Kaaviossa S vastaa lausetta, NP substantiivifraasia, VP verbifraasia, PP prepositiofraasia ja niin edelleen. Yleensä kyseinen informaatio merkitään hakasulkein. Esimerkiksi ylläesitetty lause olisi muotoa [S [NP Claudia_NP1 NP] [VP sat_VVD [PP on_II [NP a_AT1 stool_NN1 NP] PP] VP] S]. Sanaluokkainformaatio on liitetty alaviivoin. Sama voitaisiin esittää myös muodossa

```
[S
  [NP Claudia NP]
    [VP sat
      [PP on
        [NP a stool NP]
      PP]
    VP]
S].
```

Jäsentäminen voidaan jakaa täydelliseen jäsentämiseen (full parsing) ja runkojäsentämiseen (skeleton parsing). Täydellinen jäsentäminen pyrkii analysoimaan lauseen rakenteen niin tarkasti kuin mahdollista. Runkojäsentäminen taas on vähemmän yksityiskohtaista. Se pyrkii käyttämään vähemmän tarkkoja syntaktisia rakenneosatyyppejä ja jättää huomiotta esimerkiksi tiettyjen rakenneosien tyyppien sisäisen rakenteen. [McEnery and Wilson, 2001.]

Sanaluokkamerkinnät tehdään yleensä automaattisesti tietokoneen avulla, kun taas jäsentämisen tekemisessä on paljon enemmän vaihtelua. Jäsentäminen voidaan tehdä täysin automaattisesti, sen voi tehdä analyytikko käyttäen apunaan jäsentämisohjelmistoa tai se voidaan tehdä täysin manuaalisesti. Jäsentäminen on vähemmän yleistettävissä eri kielille kuin sanaluokkainformaation merkitseminen eikä siihen voida määritellä pakollisia ominaisuuksia. McEnery ja Wilson [2001] suosittelevat, että sulkujen käyttäminen rakenneosien kanssa pitäisi olla pakollista, jos on käytetty fraasirakennekielioppia (phrase-structure grammar). Lisäksi he suosittelevat, että tietyt perusrakennetyypit tulee erotella. Näitä ovat lause, virke, substantiivifraasi, verbifraasi, adjektiivinen fraasi, adverbiaalinen fraasi ja prepositionaalinen fraasi.

*Semanttiset merkinnät* voidaan jakaa kahteen luokkaan, tekstielementtien välisten semanttisten suhteiden merkitsemiseen ja tekstin sanojen semanttisten ominaisuuksien merkitsemiseen eli sanojen merkitysten merkitsemiseen tavalla tai toisella. Myöskään semantiikassa ei voida antaa universaaleja sopimuksia siitä, mitkä sanojen ominaisuudet tulisi merkitä. [McEnery and Wilson, 2001.] Sanat voidaan jakaa semanttisiin kategorioihin, jotka edelleen voidaan jakaa pienempiin kategorioihin. Kategoriat voidaan kuvata numerosarjoilla, joissa esimerkiksi kaksi ensimmäistä numeroa kertovat, mihin yläkategoriaan sana kuuluu. Esimerkiksi avain 252505 voi tarkoittaa värejä ja 210510 tarkoittaa ihmisruumiinosia. [McEnery and Wilson, 2001.]

Edellä mainittujen korpusmerkintöjen lisäksi voidaan tehdä myös muita merkintöjä. Näitä ovat esimerkiksi foneettiseen puhtaaksikirjoittamiseen (phonetic transcription) ja prosodiaan (prosody) liittyvät merkinnät sekä tekstin tasoihin ja diskurssiin liittyvät merkinnät. Diskurssitunnisteisiin voidaan merkitä esimerkiksi anteeksipyynnöt, tervehdykset ja kohteliaisuudet. Diskurssiin liittyvät merkinnät ovat kuitenkin harvinaisia, koska niiden merkitseminen on hankalaa. Diskurssimäärittelyt ovat riippuvaisia tekstin sisällöstä ja niiden tunnistaminen tekstistä on kiistanalaista. [McEnery and Wilson, 2001.]

Merkintöjä voidaan tehdä myös käyttäjän tarpeiden mukaan korpukseen, joka voi olla etukäteen merkitty tai merkitsemätön. Tällöin korpustutkija saa käyttöönsä juuri sellaiset merkinnät, jotka hänen tutkimuksensa kannalta ovat olennaisia. Tätä kutsutaan ongelmasuuntautuneeksi leimaamiseksi. Ongelmasuuntautunut leimaaminen eroaa

muista merkitsemistavoista. Se ei ole tyhjentävä, koska vain ilmiö, joka tutkimuksen kannalta on relevantti, merkitään. [McEnery and Wilson, 2001.]

## 2.5 Korpusten hyödyntäminen

Korpuslingvistiikka on menetelmä, jota voidaan käyttää lähes kaikilla kielitieteen aloilla, mutta korpuksia voidaan hyödyntää myös muilla tieteenaloilla. Korpuksista voidaan johtaa hyvin monenlaista tilastollista tietoa ja niitä voidaan käyttää hyvin erilaisiin tarkoituksiin.

### 2.5.1 Frekvenssilistat

Frekvenssilista on yksinkertainen tilastollinen menetelmä, jolla voidaan tarkastella esimerkiksi tiettyjen sanojen tai sanaluokkien esiintymiskertoja tekstissä. Frekvenssilista koostuu kohteena olevasta sanasta tai useammista sanoista sekä luvusta, joka kertoo kyseisen sanan tai tekstin esiintymiskerrat tutkimuksen kohteena olevassa korpuksessa. Frekvenssilistat voidaan järjestää joko sanojen esiintymisjärjestykseen, aakkosjärjestykseen tai frekvenssien mukaiseen järjestykseen. Frekvenssilistat eivät sisällä kohteena olevan sanan tai tekstin ulkopuolista kontekstia. Aakkosjärjestyksessä olevaa frekvenssilistaa voidaan käyttää hypoteesien muodostamiseen ja tehtyjen oletusten tarkistamiseen. Usein aakkosjärjestyksessä olevaa frekvenssilistaa käytetään tietyn sanamuodon frekvenssin tarkistamiseen. [Sinclair, 1991.] Frekvenssin mukaan järjestettyjä listoja voidaan vertailla toisiinsa. Tavallisesti yleisimmät sanat tai sanayhdistelmät muodostavat vakaan jakauman, joten huomattava poikkeus tästä on yleensä merkittävä. Frekvenssilistojen perusteella saadaan vain vihjeitä tekstin luonteesta. Tutkimalla listaa voidaan saada vihjeitä siitä, mitä lisäinformaatiota tulisi hankkia, tai tehdä olettamuksia tekstin rakenteesta ja tutkimuksen keskipisteestä. [Sinclair, 1991.]

Jos korpuksen tutkittava teksti on pitkä, frekvenssilistoista, KWIC-listoista (keyword in context) ja muista tutkittavista kokonaisuuksista saattaa tulla niin pitkiä, ettei niitä ole mielekästä tarkastella kokonaisuudessaan. Sinclair [1991] ehdottaakin tekstin rajaamista erilaisin perustein. Rajaaminen tai valinta voi tapahtua frekvenssin tai esiintymän muodon perusteella. Frekvenssin perusteella voidaan tulokseen ottaa mukaan useimmin esiintyvät kokonaisuudet tai esimerkiksi esiintymät, joiden frekvenssi on yksi. Esiintymät voidaan myös luokitella portaittain frekvenssin mukaan. Sanat voidaan rajata ja luokitella myös niiden muodon tai rakenteen mukaan. Lisäksi tietysti voidaan yhdistää frekvenssien ja sanojen mukaan rajaaminen.

### 2.5.2 Termien ja kollokaatioiden louhinta

Koska tässä tutkimuksessa toteutettu kyselykieli on suunniteltu erityisesti termien louhintaa varten, perehdytään seuraavaksi tarkemmin korpusten käyttöön juuri termien louhinnassa. Sanojen lähekkäin esiintymisestä käytetään nimitystä kollokaatio. Sanaa

kollokaatio käytetään erityisesti kielitieteessä ja sillä tarkoitetaan sanojen normaalia suurempaa taipumusta esiintyä lähekkäin. Kollokaatio koostuu kahdesta tai useammasta sanasta, jotka esiintyvät yhdessä ja, joiden merkitys vastaa tiettyä tavanomaista tapaa ilmaista asioita [Manning and Schütze, 1999]. Esimerkki tiukasta kollokaatiosta on sana laimin, joka esiintyy yhdessä vain verbin lyödä kanssa. Kielellisesti erityisen kiinnostavaa kollokaatioissa on hienovaraiset ja hankalasti selitettävät sanojen käyttötavat, jotka äidinkielenään kieltä puhuvat ymmärtävät. Luonnollisen kielen ilmaisuja kutsutaan kompositionaalisiksi, jos ilmauksen merkityksen voi päätellä sen osien merkityksistä. Kollokaatiot eivät yleensä ole täysin kompositionaalisia vaan osien merkitys tarkentuu.

Sanalle kollokaatio löytyy erilaisia määritelmiä. Laskennallinen lähestymistapa ja tilastollinen lähestymistapa määrittelevät kollokaatiot yleensä vierekkäisiksi sanoiksi, mutta kielitieteellinen lähestymistapa laskee kollokaatioiksi myös fraasit, joissa sanat eivät välttämättä esiinny peräkkäin. [Manning and Schütze, 1999.] Manning and Schütze [1999] esittävät esimerkit a) ‘she knocked on his door’, b) ‘they knocked at the door’, c) ‘100 women knocked on Donaldson’s door’ ja d) ‘a man knocked on the metal front door’. Lauseista voidaan huomata, että sanat knock ja door esiintyvät yleensä yhdessä, mutta näiden kahden sanan etäisyys voi vaihdella.

Termi tarkoittaa sanaa tai muuta ilmaisua, jolla on täsmällinen ja eriytynyt, usein erikseen määritelty merkitys. Termit ovat monisanayksikköjä (multi-word units MWUs), ja ne muodostetaan yleensä useista sanoista. Monisanayksiköiden pituus voi vaihdella välillä 1-n. [Oakes, 1998.] Tekninen termi tarkoittaa sanaryhmää, joka esiintyy yhdessä ja jolla on jokin merkitys, joka ei välttämättä ole sama kuin termien osana olevien sanojen yhteismerkitys. Fraasi tarkoittaa kieltentutkimuksessa jotakin kielelle ominaista, kiteytynyttä ja vakiintunutta ilmaisua. Salton [1989] käyttää nimitystä ’term phrase’ ja toteaa sen koostuvan peräkkäisistä sanoista ja olevan merkitykseltään tarkempi kuin fraasin yksittäiset sanat.

Kahden sanan suhdetta toisiinsa voidaan tutkia esimerkiksi laskemalla etäisyyksien keskiarvo ja varianssi. Keskiarvon ja otoshajonnan avulla voidaan tutkia kahden sanan etäisyyksien jakaumaa korpuksessa. Pieni otoshajonta tarkoittaa, että kaksi sanaa esiintyy yleensä samalla etäisyydellä toisistaan. Varianssi kuvaa tilastollista hajontaa. Jos kahden sanan etäisyyksien varianssi on korkea, sanat esiintyvät yhdessä vain sattumanvaraisesti [Oakes, 1998].

### 2.5.3 Termien ja kollokaatioiden hyödyntäminen

Kollokaatioita ja termejä voidaan hyödyntää kielitieteessä ja tiedonhaussa. Kielitieteessä kollokaatioita tunnistamalla voidaan kehittää monipuolisempia sanakirjoja, jotka käsittelevät kielen luonnollista käyttöä. Kollokaatioita tunnistamalla voidaan etsiä oikeita ilmauksia, esimerkkinä tästä ’vahva tee’, joka on oikeampi ilmaus

kuin ’voimakas tee’. Jos toinen kollokaatioehdokkaista esiintyy selvästi useammin, voidaan sen olettaa olevan oikea. Lisäksi kollokaatioita voidaan käyttää hyväksi valittaessa sanoja sanakirjaan. Kollokaatioita ja termejä voidaan hyödyntää myös luonnollisen kielen generoinnissa ja kielten välisessä tiedonhaussa. Informaatiotutkimuksessa termejä hyödynnetään muun muassa indeksointeihin, hakujen suorittamiseen ja hakujen painotuksiin.

#### 2.5.3.1 Termien ja kollokaatioiden tunnistamismenetelmät

Kollokaatioita ja termejä voidaan löytää ja tunnistaa erilaisin tavoin. Yksinkertaisin menetelmä kollokaatioiden löytämiseen on lemmaparien tai -yhdistelmien frekvenssien laskeminen. Koska pelkkiä frekvenssejä laskemalla ei välttämättä saada hyviä tuloksia, voidaan yhdistää yksinkertainen määrällinen menetelmä, lingvistinen tietämys sekä käyttää hyväksi sulkusanalistaa (stop word list). Tuloksia voidaan parantaa käyttämällä hyödyksi sanaluokkainformaatiota ja valitsemalla vain tiettyihin malleihin sopivat sanaparit ja -yhdistelmät. Pelkästään matemaattisiin ja tilastollisiin menetelmiin tukeutuminen tuo mukanaan paljon kohinaa eli esimerkiksi sanapareja, jotka esiintyvät yhdessä, mutta jotka eivät ole termiehdokkaita. Tällaisia voivat muodostaa sanat, jotka esiintyvät usein kaikissa teksteissä.

Sanaluokkainformaation avulla kandidaattitermipareiksi voidaan valita esimerkiksi seuraavia pareja:

- Adjektiivi-substantiivi
- Substantiivi-substantiivi
- Adjektiivi-adjektiivi-substantiivi
- Adjektiivi-substantiivi-substantiivi
- Substantiivi-adjektiivi-substantiivi
- Substantiivi-substantiivi-substantiivi
- Substantiivi-prepositio-substantiivi [Manning and Schütze, 1999].

Daille [1994] on esitellyt tavan louhia teknisiä termejä korpuksesta yhdistämällä kielellinen data ja tilastolliset menetelmät. Ensin valitaan kandidaattitermit. Kandidaattitermit ovat sanojen sarjoja, jotka muodostavat lauseopillisia malleja, jotka tunnetaan teknisinä termeinä. [Oakes, 1998.] Kandidaattitermit valitaan esimerkiksi yllä mainitusti sanaluokkien avulla. Tämän jälkeen hyödynnetään erilaisia tilastollisia menetelmiä. Yksinkertaisimmat menetelmät ovat frekvenssien, keskiarvojen ja varianssien laskeminen. Lisäksi termien löytämiseen voidaan käyttää useita tilastollisia menetelmiä kuten hypoteesitestaus, t-testi ja Pearsonin korrelaatiokerroin. [Manning and Schütze, 1999.]

Daille [1994] jakaa tilastolliset mittarit neljään luokkaan: frekvenssit, assosiaatiokriteerit (association criteria), Shannonin diversiteetti (Shannon diversity) sekä etäisyysmitat (distance measures). Frekvenssit toimivat parametreina assosiaatiokriteereille. Assosiaatiokriteerien avulla voidaan laskea kahden sanan tai lemman sidoksen voimakkuutta. [Daille, 1994.]

Käytettäessä kollokaatioiden ja termien tunnistamiseen tilastollisia menetelmiä, parin kahta lemmaa voidaan käsitellä kahtena kvalitatiivisena muuttujana, joiden yhteyttä testataan. Kullekin parille ($L_i$ ja $L_j$) voidaan määritellä kontingenssitaulu (contingency table). [Oakes, 1998.] Kontingenssitaulun luvut ovat frekvenssejä tutkittaville ilmentymille [Cooper and Weekes, 1983]. Taulukossa 1 annetaan esimerkki tästä.

Taulukko 1. Kontingenssitaulu

| | $L_j$ | $L_{j'}$ with $j' \neq j$ |
|---|---|---|
| $L_i$ | a | b |
| $L_{i'}$ with $i' \neq i$ | c | d |

Taulukossa 1 jokainen lemmapari koostuu lemmoista, jotka on merkitty $L_i$:llä ja $L_j$:llä.

- a tarkoittaa frekvenssiä lemmaparille, joka sisältää sekä $L_i$:n että $L_j$:n.
- b tarkoittaa frekvenssiä lemmaparille, joka sisältää $L_i$:n mutta ei $L_j$:tä.
- c tarkoittaa frekvenssiä lemmaparille, joka ei sisällä $L_i$:tä mutta sisältää $L_j$:n.
- d tarkoittaa frekvenssiä lemmaparille, joka ei sisällä $L_i$:tä eikä $L_j$:tä.

Nyt esimerkiksi summa a + b + c + d on niiden parien kokonaismäärä, jotka toteuttavat annetun mallin, esimerkiksi kaikki adjektiivi-substantiiviparit. [Oakes, 1998.] Tilastotiede tarjoaa paljon mittareita, joita voidaan käyttää yhdessä kontingenssitaulun kanssa laskemaan kahden muuttujan välisen sidoksen voimaa. Nämä voidaan jakaa tilastollisiin testeihin ja yhteystuloksiin (link scores). Yhteystuloksiin kuuluvat esimerkiksi yksinkertainen yhteensopivuuskerroin (SMC), Fagerin ja McGowanin kerroin (FAG) sekä Yulen kerroin (YUL). [Oakes, 1998; Daille 1994.]

Salton [1989] esittelee menetelmän fraasien muodostamiseen indeksointia varten. Fraasien muodostusprosessissa muunnetaan korkeafrekvenssisiä fraaseja keskifrekvenssin fraaseiksi, jotka ovat parempia hakuavaimia. Menetelmä sisältää seuraavat vaiheet:

1. Ensisijaiseksi fraasikomponentiksi eli fraasin pääksi valitaan sana, jonka dokumenttifrekvenssi ylittää tietyn asetetun rajan.

2. Muiden fraasin komponenttien tulee olla keski- tai matalan frekvenssitason sanoja, jotka esiintyvät yhdessä fraasin pään kanssa.
3. Yleisiä sulkusanoja ei oteta huomioon fraasien muodostusprosessissa.
4. Sanoihin liitetään sanaluokkainformaatio sopivan sanakirjan avulla ja tämän jälkeen fraasien muodostus voidaan rajata esimerkiksi substantiivi-substantiivi sarjoihin.
5. Yksinkertaista lauseopillista analyysiprosessia voidaan käyttää myös tunnistamaan synteettisiä lauseyksiköitä kuten subjektifraaseja, objektifraaseja ja verbifraaseja. Fraasielementit voidaan tällöin kaikki valita samasta syntaktisesta yksiköstä.

## 2.6 Korpuskyselykielet

### 2.6.1 Korpuskyselykielet

Korpushakujärjestelmiin kuuluu usein korpuskyselykieli (corpus query language). Monet järjestelmät antavat käyttäjän muodostaa haut graafisen käyttöliittymän avulla, mutta toisena vaihtoehtona on usein kyselykieli, jolla haut muodostetaan. Korpuskyselykielet ovat erityisesti kielten tutkijoille tarkoitettuja kyselykieliä. Ne saattavat tarjota käyttäjille lisäominaisuuksia ja yksityiskohtaisempia rajausmahdollisuuksia kuin graafiset käyttöliittymät. Tässä työssä vertailtavien korpuskyselykielten lisäksi on olemassa muita korpuskyselykieliä, esimerkiksi Stuttgartin yliopiston TIGERSearch query language (http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/ html/QueryLanguage.html).

Yleisesti kyselykielet muodostuvat komennoista, jotka esitetään komentosanojen avulla. Komentosanojen lisäksi kyselykieliin kuuluu määritteitä, jotka kertovat esimerkiksi, mihin komento kohdistuu tai, miten komento suoritetaan. [Järvelin, 1995.] Korpuskyselykielet ovat kyselykieliä, jotka on kehitetty kieltentutkijoiden tarpeiden pohjalta ja suunniteltu erityisesti soveltuviksi korpuksiin kohdistettuihin hakuihin. Kyselykielet rajaavat usein käyttäjän hakumahdollisuuksia vähemmän kuin hakujen muodostamiseen ja suorittamiseen tarkoitettu graafinen käyttöliittymä.

Järvelin [1995] esittää kyselykielten komentojen ryhmittelyn. Komennot jakautuvat tämän ryhmittelyn mukaan kohdennuskomentoihin ja käsittelykomentoihin. Kohdennuskomennoilla valitaan tarkastelun kohteeksi osia tietokannan sisällöstä ja nämä komennot jaetaan rajaus-, valinta- ja siirtymäkomentoihin. Rajauskomentojen avulla kohdistetaan haut vain tiettyyn osaan kohdetietokannassa. Valintakomennoilla käsitellään rajattua osaa. Valintakomentoihin kuuluvat joukko-opin operaatiot, tekstihakuoperaatiot sekä tilastomatemaattiset operaatiot. Siirtymäkomentoja ovat dokumenttien ja tekstin selaukseen liittyvät komennot. Korpuskyselykielissä rajauskomennoilla voidaan rajata haut kohdistumaan vain tiettyyn osaan korpuksessa.

Valintakomennoilla valitaan mukaan esimerkiksi vain virkkeet, jotka sisältävät relatiivilauseen. Erilaisilla operaattoreilla voidaan kyselyn rajauksia yhdistellä, ryhmitellä ja merkitä vaihtoehtoisiksi.

Käsittelykomennoilla käsitellään tarkastelun kohteena olevia osia tietokannan sisällöstä. Käsittelykomentoihin kuuluvat saanti- ja esityskomennot. Tulostuskomennot ovat saantikomentoja ja tulostuksen muotoilukomennot ovat esityskomentoja. [Järvelin, 1995.] Käsittelykomentojen avulla korpuskyselykielissä voidaan esimerkiksi valita haun tulokset esitettäviksi kokonaisina lauseina tai määrätä tulostusmuoto KWIC-tulosteeksi.

### 2.6.2 Korpuskyselykielille asetetut vaatimukset

Korpuskyselykieliä käyttävät kielitieteen tutkijat eivätkä suinkaan yleensä tietojenkäsittelytieteen, informaatiotutkimuksen tai matematiikan asiantuntijat. Korpushakujärjestelmien ja korpuskyselykielten tulisikin olla helppokäyttöisiä eivätkä ne saisi vaatia käyttäjältään esimerkiksi ohjelmointitaitoja. Myös Lehtinen et al., [1995] toteavat, että korpukset eroavat toisistaan siinä, miten helppoa tai vaikeaa niistä on etsiä haluamaansa tietoa ja jos korpuksen hakuominaisuudet koetaan vaikeiksi, aineistot jäävät kokonaan käyttämättä. Heidän mukaan suomalaisten Kotimaisten kielten tutkimuskeskuksen korpusten ja Waltari-korpusten käytön huono puoli on se, että niissä käyttäjä joutuu itse kirjoittamaan tarkat, määrämuotoiset komennot [Lehtinen et al., 1995]. Tässä työssä on pyritty suunnittelemaan yksinkertainen kyselykieli, joka ei vaadi käyttäjältä erityistä kielen opiskelua.

# 3 XTerm – tutkimuksessa kehitelty kyselykieli

## 3.1 Tausta ja tavoitteet

XTerm-kyselykielen kehittäminen lähti liikkeelle tarpeesta louhia termejä lakitekstiä sisältävästä korpuksesta. Korpus on rinnakkaiskorpus, jonka kielinä ovat suomi ja venäjä. Käännöksen ja tekstien vertailun suurimmaksi ongelmaksi oli todettu vastintermien löytäminen ja käsittely. Hakujen mahdollistamiseksi korpus on merkattu. Kielitieteelliset tunnisteet on lueteltu liitteessä 2. Korpus voidaan määritellä myös synkroniseksi korpukseksi, koska se sisältää tekstiä vain tietyltä kolmen vuoden ajanjaksolta. Lisäksi se voidaan määritellä otantakorpukseksi, koska tekstiä ei ole tarkoitus lisätä. Hakujärjestelmä mahdollistaa kuitenkin tekstin lisäämisen edellyttäen että lisättävän tekstin merkinnät on tehty samalla tavalla kuin alkuperäisen tekstin merkinnät.

Kyselykielelle oli tarve Tampereen yliopiston kielitieteen laitoksella. Kielen tarkoitus on olla helppo ja tehokas työkalu, jolla voidaan hakea sanoja ja sanayhdistelmiä. Kieli on tarkoitettu erityisesti termien tunnistamiseen. Kysely muodostaa mallin, jonka perusteella ohjelma palauttaa tekstikokoelmasta malliin sopivat termiehdokkaita. Kielen kehittäminen on tapahtunut kielitieteen laitoksen tutkijoiden tarpeiden pohjalta ja sen kehittäminen on tapahtunut Suomen Akatemian projektissa numero 115480, jossa on ollut mukana sekä informaatiotutkimuksen että tietojenkäsittelytieteen tutkijoita.

## 3.2 Korpustietomalli

Ullman [1998] määrittelee, että tietomalli on matemaattinen formalismi, joka koostuu kahdesta osasta: 1. Kuvauksesta, joka määrittelee datan. 2. Joukosta operaatioita, joita voidaan käyttää tämän datan käsittelyssä. Korpustietomalli voidaan esitellä koostuvaksi merkkaussäännöistä ja kielestä, joka pystyy hyödyntämään kyseiset merkkaukset. Korpustietomallin taustalla on pyrkimys korpusriippumattomaan käsittelyyn. Seuraavaksi tutustutaan XTerm-tietomalliin.

XTerm-korpus on tallennettu hierarkkisiksi XML-tiedostoiksi. Jokainen yksittäinen XML-tiedosto käsittää yhden itsenäisen dokumentin. Esimerkki yhdestä korpukseen kuuluvasta XML-tiedostosta löytyy liitteestä 3. Yksi XML-dokumentti on yksi tiedosto ja juurena on doc-niminen elementti. Dokumentti jakautuu kappale-elementteihin (p), jotka sisältävät lauseita (s). Lauseet sisältävät sanoja (w) ja välimerkkejä (punct). Kyselyiden kannalta olennaisia ovat sanojen lapsielementit eli lemmat. Haut kohdistetaan lemma-elementtien sisältöihin ja lemmojen tags-nimisiin attribuutteihin.

Tutkimuksen kohteena olevassa korpuksessa kaikki kielitieteelliset merkinnät on liitetty sanoihin yhtenäisellä tavalla. Lemma-nimisen XML-elementin tags-nimiseen

attribuuttiin liitetään erilaiset kielitieteellisten merkintöjen lyhenteet. Tutkimuksen kohteena olevassa korpuksessa nämä lyhenteet voidaan jakaa seuraaviin ryhmiin sanaluokka (part of speech), vertailumuoto (comparation), sija (case), luku (number), suku (gender), omistusliitteet (possessive suffixes), tapaluokka (mood), aikamuoto (tense), pääluokka (voice), adjektiivin muoto (form of the adjective), persoonamuoto (person), kielto (negative), infinitiivit (infinitives), partisiipit (participles) ja jälkiliitteet (clitics). Esimerkiksi tekstissä esiintyvän sanan ‘sovelleta’ merkintä on XML-muodossa `<lemma tags="|V|PRES|PSS|NEG|">soveltaa</lemma>.` Kaikki korpukseen merkityt kielitieteelliset tunnisteet löytyvät liitteestä 2.

XTerm-kielen kysely koostuu kolmesta osasta. Alkuosa sisältää mallin, jonka perusteella tulosjoukko muodostetaan. Toinen osa sisältää projektio-osan, jossa määritellään, miten sanat tuloksessa esitetään. Kolmas osa on valinnainen ja koskee koko mallia. Siinä määrätään, halutaanko tulokseen mukaan koko tulosrivin frekvenssi. Nämä kaikki kolme osaa erotetaan toisistaan => -merkinnällä.

XTerm-kyselykieli on suunniteltu ja toteutettu siten, että se ei ole sidottu vain olemassa oleviin merkintöihin. Merkintöjä voidaan lisätä tai poistaa tarpeiden mukaan. Tietomalli käsittää tällä hetkellä luvussa 2.4.1. esitellyistä kielitieteellisistä merkinnöistä sanaluokkamerkinnät ja lemmatisoinnin. Korpus kyllä on jäsennetty esimerkiksi kappaleisiin ja lauseisiin, mutta kyselyissä voidaan hyödyntää vain sanoja (w) ja lemmoja (lemma). Korpustietorakenne mahdollistaa nyt käytössä olevien merkintöjen lisäksi muiden merkintöjen lisäämiseen. Tallennusmalliin voitaisiin lisätä esimerkiksi luvussa 2.4.1. esitellyt semanttiset merkinnät tai halutut diskurssitunnisteet ilman muutoksia kyselykieleen.

### 3.3 Ominaisuudet

Kyselyn malliosa koostuu osista, jotka on eroteltu |-merkein. Kukin osa vastaa yhtä sanaa tai lemmaa kohdetekstissä. Osien määrää ei ole rajoitettu. Kysely muodostetaan kyselyyn kuuluvien merkkien sekä korpukseen merkattujen tunnisteiden ja lemmojen avulla. Tunnisteet ovat kielitieteellisiä merkintöjä. Näitä merkintöjä ovat esimerkiksi sanaluokat ja sijamuodot. Mallin yksittäinen osa voidaan jättää tyhjäksi, tällöin kyseiselle sanalle ei aseteta mitään ehtoja. Tyhjiä osia voi olla useampia ja ne voivat sijaita missä kohdassa tahansa. Malliosa voi myös sisältää useamman tunnisteen. Tunnisteiden ja lemmojen yhdistäminen tapahtuu automaattisesti kyselyä suoritettaessa. Tai-operaattoria ei ole käytössä kyselyosan yksittäisessä yhtä sanaa vastaavassa osassa. Käyttäjille tarkoitetut XTerm-kyselykielen käyttöohjeet on esitelty yksityiskohtaisesti liitteessä 1.

Jos halutaan rajata tuloksesta pois sanat, joilla on määrätty kielitieteellinen tunniste, voidaan tunnisteeseen liittää negaatiomerkintä –. Vastaavasti voidaan rajata pois jokin tietty lemma. Esimerkiksi haku `A -’vakava’ => l` palauttaisi tuloksenaan kaikki

korpuksen adjektiivit, lukuun ottamatta adjektiivia vakava. Yksi tekstissä esiintyvä sana voi sisältää useita lemmoja, kuten perinteinen hauissa-esimerkki osoittaa. Tekstissä esiintyvä muoto ’hauissa’, saa kaksi lemmaa haku ja hauki. Kahdella negaatiomerkillä ’– –’ voidaan rajata tulosjoukosta pois kaikki sellaiset sanat, joilla yksikin sanan lemmoista on annettu lemma.

Valinnaisuus merkitään ?-operaattorilla. Tällöin kyseinen osa rajoituksineen voi esiintyä tai olla esiintymättä tuloksessa. Esimerkiksi kysely `?A | A | N` palauttaa kaikki sellaiset lemmayhdistelmät, jotka koostuvat kahdesta adjektiivista ja näitä seuraavasta substantiivista tai yhdestä adjektiivista ja sitä seuraavasta substantiivista.

Kokonaisia kyselyjä voidaan yhdistää toisiinsa OR-operaattorin avulla. Tällöin tulokseen otetaan mukaan kaikki lemmat tai lemmayhdistelmät, jotka täyttävät ehdot jollekin OR-operaattorilla erotelluista kyselylausekkeista. Käytettäessä valinnaisuusoperaattoria tai OR-operaattoria, palautettavan tulosjoukon tietueet voivat olla keskenään eripituisia.

Projektio-osassa määrätään, halutaanko lemmat esittää tuloksessa lemmamuodossa vai tekstissä esiintyvässä muodossa. Kirjain l merkitsee lemmamuodossa palauttamista ja kirjain w esiintymismuodossa palauttamista. Lisäksi voidaan määrätä, halutaanko nähdä yksittäisen lemman frekvenssi koko kokoelmassa tai tulosjoukossa. Kunkin lemman määreet erotellaan |-merkillä noudattaen samaa järjestystä kuin malliosassa. Useampi määre erotellaan toisistaan pilkuilla. Frc-määre (frequency in collection) palauttaa lemman frekvenssin kokoelmassa ja frr (frequency in result set) lemman frekvenssin tulosjoukossa. Esimerkiksi kysely `PREP | N => l, frr | l, frr` palauttaisi kaikki prepositio-substantiivi -sanaparit lemmamuodossa ja näin lemmojen frekvenssit tulosjoukossa.

Tuloksessa voidaan palauttaa myös koko tietueen frekvenssi. Tämä merkitään projektiosan jälkeen valinnaisena kolmantena osana liittämällä kyselyyn `=> FR`. Tietueella tarkoitetaan tässä kyselyn palauttamaa tekstikokonaisuutta eli yhtä riviä tuloksessa. Tietueen pituus on minimissään yksi sana tai lemma, ja se voi olla lemma- tai esiintymismuodossa. Tällaisella haulla voidaan selvittää, kuinka monta kertaa tekstissä esiintyy tietty sanapari tai sanayhdistelmä. Jälkimmäinen projektio-osa ei ole pakollinen, mutta ensimmäinen on.

XTerm-kyselykielen syntaksi esitettynä EBNF-merkinnällä (Extended Backus-Naur Form) [ISO/IEC 14977, 1996]:

```
<lause> ::= <kyselyosa> (”OR” <kyselyosa>)* (”=>FR”)?
<kyselyosa> ::= <kysely>
<kyselyosa> ::= <alkuosa> ”|” <kyselyosa> ”|” <loppuosa>
<kysely> ::= <alkuosa> ”=>” <loppuosa>
<alkuosa> ::= ((<kysymysm> | <numero>+)? <negaatio>?
<teksti>)*
<loppuosa> ::= <loppumerkki> (”,” (<loppumerkki> |
<frekvenssi>))*
<negaatio> ::= ”-” | ”--”
<kysymysm> ::= ”?”
<numero> ::= (”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” |
”8” | ”9” )
<teksti> ::= <tag> | ”’”<kirjain>+”’”
<tag> ::= (“A” | “ABBR”| “AD-A” | “ADV” | “ART” | “C”|
“INTJ” , …)
<kirjain> ::= <pieniKirjain> | <isoKirjain>
<pieniKirjain> ::= (“a” | “b” | “c” | “d” | “e”, “f” | “g”
| “h” | “i” | “j” | “k” | “l” | “m” | “n” | “o”, “p” | “q”
| “r” | “s” | “t” | “u” | “v” | “x” | “y” | “z”, “å” | “ä”
| “ö”)
<isoKirjain> ::= (“A” | “B” | “C” | “D” | “E”, “F” | “G” |
“H” | “I” | “J”, “K” | “L” | “M” | “N” | “O”, “P” | “Q” |
“R” | “S” | “T” | “U” | “V” | “W” | “X” | “Y”, “Z” | “Å” |
“Ä” | “Ö”)
<loppumerkki> ::= (”w” | ”l”)
<frekvenssi> ::= (”frc” | ”frr”)
```

Tässä merkinnässä ei ole lueteltu erikseen venäjän kielen aakkosia (pieniKirjain ja isoKirjain) ja tag-elementtiin on lueteltu vain suomen kieleen liittyvät kielitieteelliset merkinnät. Venäjän kielen merkinnät löytyvät kuitenkin liitteestä 2. Suomen kielen kielitieteellisistä tunnisteista on niistäkin lueteltu vain seitsemän ensimmäistä.

### 3.4 Puutteet ja ja tärkeimmät ominaisuudet

XTerm-kielen puute on se, että sanan katkaisuja ei voida tehdä eikä kyselyn ehtoja voida merkitä vaihtoehtoisiksi. Vaihtoehtoisuus on kuvattava aina kahdella tai useammalla erillisellä kyselyllä, jotka on yhdistetty OR-operaatiolla. Tämä saattaa tehdä joistakin kyselyistä pitkiä ja monimutkaisia. Kielen tärkein ominaisuus on sen soveltuvuus peräkkäisten sanojen palauttamiseen ja näiden frekvenssien esittämiseen. Frekvenssien esittämisen lisäksi kielen etuna on ehdottomasti se, että tulos voidaan palauttaa joko perusmuodossa tai siinä muodossa kuin se alkuperäisessä tekstissäkin esiintyy. XTerm-kyselykieli on helppo oppia ja se soveltuu käyttötarkoitukseensa, termien louhintaan, erittäin hyvin. Tutkimuksessa mukana olleet kielitieteilijät omaksuivat kielen helposti.

# 4 XTerm-kyselykielen tekninen toteutus

XTerm-tietomalli on implementoitu Oraclen Berkeley DB XML-kirjastoon. Toteutettu ohjelma, joka muuntaa käyttäjän syöttämän kyselyn Berkeley DB XML-tietokantatuotteen ymmärtämäksi XQuery-kyselyksi, on ohjelmoitu PHP (Hypertext Preprocessor) -ohjelmointikielellä. Ohjelma sisältää graafisen käyttöliittymän, jota voidaan käyttää www-selaimella. Käyttöliittymä on toteutettu HTML-elementtejä ja JavaScript-kielen mahdollistamaa vuorovaikutteisuutta hyödyntäen. Toteutus mahdollistaa sen, ettei ohjelmaa tarvitse asentaa käyttäjien koneelle vaan se on yleisesti käytettävissä selainohjelman kautta. Seuraavassa esitellään tarkemmin ohjelman toteuttaminen sekä XQuery ja XPath -kyselykielet.

## 4.1 Berkeley DBXML

XML-dokumentteja voidaan tallentaa monella eri tavoin. Kokonaiset XML-dokumentit voidaan tallentaa tekstikenttinä tietokannanhallintajärjestelmien avulla. Toinen vaihtoehto on tallentaa tietokannanhallintajärjestelmillä dokumenttien sisältö tietoelementteinä. Yksi vaihtoehto on luoda ja julkaista XML-dokumentit olemassa olevien relaatiotietokantaan tallennettujen tietojen pohjalta. Neljäs vaihtoehto on suunnitella erityinen järjestelmä XML-datan tallentamiseen. Tässä tutkimuksessa XML-dokumentit on tallennettu XML-dokumenteille tarkoitettuun tietokantaan. Tällaisia järjestelmiä kutsutaan natiiveiksi XML-tietokannanhallintajärjestelmiksi. Nämä järjestelmät sisältävät tavallisesti yleisiä kaikille XML-dokumenteille sopivia indeksointi- ja hakutekniikoita. Ne voivat sisältää myös datan tiivistämismenetelmiä dokumenttien koon pienentämiseksi tallennusta varten. [Elmasri and Navathe, 2009.]

Oraclen Berkeley DB XML -kirjasto on natiivi XML-tietokantatuote. Se on upotettavissa oleva tietokantamoottori (database engine), joka tarjoaa tuen dokumenttien kyselyyn XQuery-kyselykielen ja sitä kautta myös XPathin avulla [Oracle, 2009]. Vaikka BDB XML ei olekaan perinteinen relaatiotietokantapohjainen tietokantajärjestelmä tai tietokantapalvelin, se tarjoaa käyttäjilleen yleiset tietokantaominaisuudet, kuten tiedon tallentamisen, indeksoinnit, kyselyt, lataamisen, poiminnat ja varmuuskopioinnin sekä palauttamisen. Yleisimmin BDB XML:a käytetään kirjastona, joka linkitetään suoraan haluttuun sovellukseen. Sanalla kirjasto tarkoitetaan tässä yhteydessä sitä, että BDB XML:a voidaan käyttää suoraan ohjelmointikielen kautta ja se mahdollistaa XQuery-kyselyjen kohdistamisen tietokantamoottoriin tallennettuun dataan. [Oracle, 2009b.] Berkeley DB XML sisältää lisäksi funktioita, jotka laajentavat XQuery-spefikaatiota. Näitä ovat esimerkiksi dokumentin osiin, indekseihin ja metadataan liittyvät funktiot. [Oracle, 2009.]

Berkeley DB XML tallentaa kaiken datan tiedostoihin, joita kutsutaan säiliöiksi (containers). Järjestelmän XML-kehikko tarjoaa yksinkertaisen tavan hallita näitä

tiedostoja. XML-dokumenttien lisäksi säiliöt sisältävät tietoa kyseisistä dokumenteista. Tällaista tietoa ovat esimerkiksi erilaiset indeksit, joita hyödynnetään dokumenttien hallinnassa. XML-dokumentit tallennetaan säiliöihin joko kokonaisina dokumentteina tai solmuina. Kun dokumentit tallennetaan solmuina, XML-dokumentti puretaan pienempiin osiin, solmuihin, ja nämä tallennetaan säiliöihin erillisinä. Haettaessa dokumentteja ne palautuvat kokonaisina alkuperäisessä muodossaan. Dokumenttien tallentaminen solmuina parantaa suorituskykyä, joten siksi solmutallentaminen on säiliöissä oletustallennusmuotona. [Oracle, 2009b.] Seuraavaksi tutustutaan yksityiskohtaisemmin XML-merkintäkieleen ja -tietomalliin.

### 4.1.1 XML-merkintäkieli

XML, Extensible Markup Language, 1.1. on W3C-konsortion (World Wide Web Consortium) suositus rakenteisten dokumenttien merkitsemiseen. XML on yksinkertainen ja joustava tekstin tallennusmuoto ja tiedon kuvauskieli, joka on johdettu SGML-merkintätavasta. XML on alun perin suunniteltu suurten tekstimassojen elektroniseen julkaisemiseen, mutta sitä hyödynnetään paljon myös tiedon siirrossa [W3C, 2008]. XML:n merkitys tiedon tallennus- ja hakumallina on kasvamassa, vaikka toistaiseksi XML-tietokantoja on olemassa melko vähän. Connolly ja Begg [2002] määrittelevät XML:n metakieleksi, eli muita kieliä kuvailevaksi kieleksi, jolla kehittäjät voivat määritellä omia tunnisteita (tags) tarjoten näin toiminnallisuutta, joka ei ole mahdollista HTML:n avulla.

XML-dokumentteihin tallennettu tieto on puolirakenteista. Connolly ja Begg [2002] määrittelevät puolirakenteisen datan tiedoksi, joka voi olla epäsäännöllistä tai epätäydellistä ja, jolla voi olla rakenne, joka saattaa kuitenkin muuttua nopeasti ja odottamatta. Datalla on siis rakenne, mutta rakenteen ei tarvitse olla tiukkaa, säännöllistä tai täydellistä kuten esimerkiksi relaatiotietokannoissa.

### 4.1.2 XML-tietomalli

XML-tietomallin perusobjekti on XML-dokumentti joka koostuu elementeistä ja attribuuteista. Elementti voi sisältää muita elementtejä ja attribuutteja. Attribuutti sen sijaan ei voi sisältää muita rakenteisia komponentteja. Dokumentin ensimmäinen elementti on juurielementti, joka sisältää epäsuorasti kaikki muut elementit. Elementeillä on aina järjestys, kun taas attribuuteilla ei ole.

Elementti koostuu sen sisällöstä sekä alkutunnisteesta, `<elementin_nimi>` ja lopputunnisteesta, `</elementin_nimi>`. Attribuutti-arvo -parit liitetään elementtien aloitustunnisteisiin muodossa `attribuutti=”arvo”`. Elementtien nimien tarkoitus on kuvata niiden sisältöä. HTML-merkintätavasta poiketen XML-elementeille voi antaa mitä tahansa nimiä. Nämä nimet voidaan myös luoda ohjelmallisesti. Kun elementit sisältävät toisia elementtejä, niitä kutsutaan

kompleksisiksi (complex elements). Jos elementti sisältää vain dataa, sitä kutsutaan yksinkertaiseksi elementiksi (simple element). [Elmasri and Navathe, 2009.] XML-mallia voidaan kutsua puumalliksi tai hierarkkiseksi malliksi sen sisäkkäisen rakenteen takia. Sisäkkäisten tasojen määrää ei ole rajoitettu.

Elmasri ja Navathe [2009] jakavat XML-dokumentit kolmeen ryhmään, datakeskeisiin, dokumenttikeskeisiin ja hybrideihin. Datakeskeiset XML-dokumentit koostuvat pienistä tietoelementeistä, jotka noudattavat tiettyä rakennetta ja, jotka on voitu johtaa rakenteisista tietokannoista. Tavallisesti nämä dokumentit on muunnettu XML-dokumenteiksi, jotta ne voitaisiin jakaa internetissä. Dokumenttikeskeiset XML-dokumentit taas koostuvat suurista tekstimassoista, joita voivat olla esimerkiksi kirjat ja erilaiset artikkelit. Tällöin dokumentissa on yleensä vain muutamia elementtejä. Hybrididokumenteissa on rakenteellista dataa sekä osia, jotka ovat pääosin tekstuaalista tai rakenteetonta. Tässä työssä tutkittavana oleva XML-dokumenttikokoelma sisältää datakeskeisiä XML-dokumentteja. Vaikka tekstiä on paljon, se on jaettu hyvin pieniin osiin. Jokainen sana on oma elementtinsä ja sisältää vielä muita elementtejä. Jokaisella sanalla on sisältönä yksi tai useita lemma-nimisiä lapsielementtejä ja yhdyssanojen lemma-elementit sisältävät segm-lapsielementtejä, jotka sisältävät yhdyssanan osat.

### 4.1.3 XML Document Type Definition

Dokumentin tyyppimääritys XML DTD (Document Type Definition) määrittelee XML-dokumentin validin syntaksin eli dokumentin elementit ja niiden sisäkkäisen rakenteen [Connolly and Begg, 2002]. DTD:n määrittelemiseen on oma syntaksinsa. DTD:ssa luetellaan dokumentissa esiintyvät elementit, miten elementtejä voidaan yhdistää, mitkä elementit voivat olla sisäkkäisiä, mitä attribuutteja on käytössä ja niin edelleen. Toisto-operaattorit *, + ja ? määrittelevät sen, kuinka monta kertaa elementti toistuu dokumentissa. Operaattori * tarkoittaa sitä, että elementti voi esiintyä nolla, yksi tai monta kertaa. Operaattori + tarkoittaa sitä, että elementin on esiinnyttävä vähintään yhden kerran ja operaattori ? merkitsee elementin esiintymistä ei yhtään tai yhden kerran. Vaihtoehtoisuutta kuvataan |-merkillä. [Elmasri and Navathe, 2009.]

Tässä tutkimuksessa käytettäviin dokumentteihin ei ole tehty alun perin DTD:a, mutta rakenteen analyysin perusteella DTD on seuraavanlainen:

```
<DOCTYPE doc [
        <!ELEMENT doc (p)* | (s)+ >
        <!ELEMENT p (s)+>
        <!ELEMENT s (w+, punct*)>
        <!ELEMENT w (lemma)* (#PCDATA)>>
        <!ELEMENT lemma (segm)* (#PCDATA)>
        <!ELEMENT segm (#PCDATA)>
        <!ATTLIST doc n CDATA #REQUIRED>
        <!ATTLIST lemma tags CDATA #REQUIRED>
]>
```

Jokainen dokumentti (doc) koostuu kappaleista (p) tai lauseista (s), jotka eivät ole kappaleiden sisällä. Jokainen kappale sisältää yhden tai useampia lauseita. Lause voi sisältää yhden tai useampia sanoja (w) ja mahdollisesti välimerkkejä (punct). Sanat sisältävät mahdollisesti lemma-elementtejä sekä tekstiä. Jos sanat ovat yhdyssanoja, lemma-elementit sisältävät segm-elementtejä. PCDATA (parseable character data) tarkoittaa merkkidataa, jossa ei ole elementtejä. Elementit voivat sisältää sekä muita elementtejä että merkkidataa, kuten tässä tapauksessa elementti w. Elementtien attribuutit luetellaan attlist-määritelmässä. Tässä tapauksessa vain dokumentti- ja lemma-elementeillä on attribuutteja. Jokainen attribuuttimääritelmä koostuu kolmesta osasta, nimestä, tyypistä ja valinnaisesta oletusarvosta. Attribuutit voivat olla tyypiltään

- merkkidataa (cdata),
- yksilöiviä tunnisteita (id),
- viittauksia yksilöivien tunnisteattribuuttien arvoihin tai lista näitä viittauksia (idref/idrefs),
- entiteetin nimi tai lista entiteettien nimiä (entity/entities),
- tunnistemerkkijono tai lista tunnistemerkkijonoja (nmtoken/nmtokens) tai
- lista nimiä eli arvot, jotka attribuutti voi saada [Connolly and Begg, 2002].

Required-määritelmä attribuutilla, tässä tapauksessa attribuutilla tags, tarkoittaa sitä, että lemma-elementillä on aina oltava attribuutti. Jos attribuuttia ei tarvitse olla, määritellään attribuutti käyttäen avainsanaa #IMPLIED. Jos taas elementillä on aina sama arvo, merkitään attribuutti määreellä #FIXED. [W3C, 2006.] Lemma-elementin tags-attibuutti voisi olla tyypiltään myös lista ja tällöin siinä voitaisiin luetella kaikki mahdolliset arvot, jotka kyseinen elementti voi saada. Tässä tapauksessa lemma-elementin tageja voisivat olla kielitieteelliset lyhenteet N (substantiivi), NOM

(nominatiivi), SG (yksikkö), A (adjektiivi), ADV (adverbi), C (konjunktio), V (verbi) ja PSP (postpositio), jotka kaikki sisältävät informaatiota kielestä kuten sanaluokan, sijamuodon tai aikamuodon.

### 4.1.4 Toteutuksessa käytetyt XML-kyselykielet

Erilaisia XML-kyselykieliä on kehitelty runsaasti. Näistä kaksi, XPath ja XQuery ovat muodostuneet standardeiksi. XPath perustuu polkuilmauksiin, joilla voidaan hakea ja palauttaa XML-dokumentin haluttuja elementtejä. XQuery taas on yleisempi ja polkuilmausten lisäksi siinä voidaan käyttää myös muita hakurakenteita ja se sisältää myös ohjelmointikielten primitiivejä, kuten toisto-operaatiot ja muuttujien käytön. Tässä työssä tehty ohjelma muuntaa käyttäjän syöttämät XTerm-kyselyt XQuery-kyselyiksi, jotka voidaan suorittaa Berkeley DB XML-tietokannassa.

### 4.1.5 XML Path Language

XPath (XML Path Language) on XML:n oma polku- ja lausekieli [Nykänen, 2001]. Se on deklaratiivinen kyselykieli, joka tarjoaa yksinkertaisen syntaksin hakujen kohdistamiseksi XML-dokumentteihin. XPath-kyselykielen hakemistopolkuja muistuttavilla ilmauksilla voidaan palauttaa tuloksena joukko elementtejä, jotka vastaavat kyselyn mallia. XPath käsittelee XML-dokumenttia loogisena järjestettynä puuna, jossa on solmu jokaiselle elementille, attribuutille, tekstille, prosessointiohjeelle, kommentille, nimiavaruudelle ja juurelle. Kyselyt koostuvat kontekstisolmusta (context node) ja lokaatiopolusta (location path), joka kuvaa polun XML-dokumentin sisällä. Polkuilmauksissa käytetään XML-dokumentissa esiintyviä nimiä, jotka voivat olla joko elementtien tai attribuuttien nimiä. [Connolly and Begg, 2002.]

XPathin osoittamismekanismissa lähdetään liikkeelle kontekstisolmusta ja siirrytään eteenpäin lokaatiopolun avulla. Kunkin vaiheen tulos on solmujen lista, joka toimii aloituspisteenä seuraavalle vaiheelle. [Connolly and Begg, 2002.] Lokaatiopolku koostuu lokaatioaskelista (location step). Jokainen lokaatioaskel edustaa siirtymistä dokumentissa tiettyyn suuntaan. Askelten erottimina käytetään yhtä kauttaviivaa (/) tai kahta kauttaviivaa (//), joiden avulla siirrytään XML-dokumenttipuussa. Yksi kauttaviiva ennen elementtiä tarkoittaa, että kyseisen elementin tulee olla suoraan edellisen elementin lapsi. Kaksi kauttaviivaa merkitsee sitä, että kyseinen elementti on edellisen elementin jälkeläinen millä tahansa tasolla.

Yksittäinen lokaatioaskel koostuu akselista (axis), solmutestistä (node test) ja yhdestä tai useammasta predikaatista (predicate). Jokaisella askeleella voidaan poistaa tuloksesta solmuja soveltamalla yhtä tai useampaa predikaattia. [Nykänen, 2001.] Predikaatti on Boolen ehto, joka valitsee alijoukon solmuista. Predikaatit merkitään hakasulkeiden sisään. XPath sisältää 13 akselia, joita ovat esimerkiksi esivanhempi (ancestor), attribuutti (attribute) ja lapsi (child). Joillekin akseleille on määritelty

lyhennysmerkintä, kuten attribuutille, johon voidaan viitata @-symbolilla. Solmutesteissä voidaan käyttää myös erilaisia XPathin valmiita funktioita. Valmiit funktiot voidaan jaotella liittyväksi solmujoukkoihin, merkkijonoihin, Boolen arvoihin ja numeroihin [W3C, 1999]. Esimerkiksi XPath-ilmaus `/dokumentti/child::kappale [otsikko="esipuhe"]` palauttaisi kaikki dokumentti-elementtien kappale-nimisten lapsielementtien, otsikko-nimiset lapset, joiden arvo on esipuhe.

Polkuilmauksissa voidaan tuloksia rajata myös attribuuttien avulla. Attribuutit merkitään @-symbolilla ja niiden arvo voidaan määrätä. Esimerkiksi lauseke `doc("luettelo.xml")/luettelo /tuote[@osasto="viihde"]` palauttaisi vain sellaiset tuotteet, joiden osasto-attribuutin arvo on viihde. Elementtejä voidaan palauttaa myös niiden sijainnin perusteella. Lauseke `doc("luettelo.xml")/luettelo/tuote[1]` palauttaisi luettelo-elementin ensimmäisen tuote-nimisen lapsielementin.

Polkuelementit ovat loogisia ja helppoja muistaa. Niiden avulla ei voida kuitenkaan vaikuttaa haun palauttaman tuloksen elementtien esitysmuotoon ja sisältöön. Kun valitaan polussa tietty elementti, palautetaan tulokseen myös tämän elementin sisällä olevat elementit sisältöineen. Koska XML-dokumentti on järjestetty, myös XPath palauttaa tuloksenaan solmut samassa järjestyksessä kuin ne ovat XML-dokumenttipuussa [Elmasri and Navathe, 2009].

### 4.1.6 XQuery-kyselykieli

XQuery on W3C Query Working Groupin suunnittelema kyselykieli. XQuery on funktionaalinen kieli, jossa kysely esitetään lausekkeena. Se mahdollistaa elementtien valitsemisen, uudelleen järjestämisen sekä muokkaamisen ja sen avulla voidaan tulos palauttaa halutun rakenteen mukaisena. XML-kyselykieleen liittyy useita operaatioita, joita voidaan käyttää sisäkkäin. Informaatio voidaan valita käyttäen erilaisia kriteerejä ja informaatio, jota ei haluta mukaan tulokseen, voidaan suodattaa pois. Informaatiota voidaan etsiä dokumentin sisältä tai dokumenttijoukosta. Dataa voidaan yhdistellä useista dokumenteista tai dokumenttikokoelmista. Tulokseen tuleva data voidaan myös lajitella, ryhmitellä ja yhdistellä. XML-data voidaan muokata ja jäsentää uudelleen toisen XML-rakenteen mukaisesti. [Walmsley, 2007.]

XQuery on määritelty formaalin tietomallin, ei XML-tekstin, suhteen. XQueryn tietomallissa jokainen dokumentti esitetään puuna, joka koostuu solmuista. Solmuja voivat olla dokumentti, elementti, attribuutti, teksti, nimiavaruus, käsittelyohje ja kommentti. Solmujen lisäksi tietomalli sisältää atomisia arvoja, joita ovat merkkijonot, Boolen totuusarvot, desimaali- ja kokonaisluvut, liukuluvut sekä päivämäärät. Elementti (item) on yksittäinen solmu tai atominen arvo. Joukko elementtejä muodostaa sarjan (sequence). XQuery-kyselykielessä jokainen arvo on sarja eikä yksittäinen

elementti ja yhden pituinen sarja eroa toisistaan. Sarja voi sisältää ainoastaan solmuja tai atomisia arvoja, se ei voi sisältää muita sarjoja. XQuery-tietomallissa dokumentin elementti-, kommentti-, ja käsittelyohjesolmut esiintyvät samassa järjestyksessä kuin ne ovat XML-dokumentissakin. Attribuutteja ei pidetä elementtien lapsina, mutta niille on määritelty sijainti dokumenttijärjestyksessä. Ne esiintyvät elementtinsä jälkeen ennen elementin lapsia. [Katz, 2004.]

Yksinkertaisin tapa valita tietoa XML-elementeistä ja -attribuuteista on käyttää polkuilmaisuja. XQueryn polkuilmauksissa käytetään XPath-syntaksia. XQueryssa polkuilmauksen tulos on järjestetty lista solmuja sisältäen myös solmujen jälkeläiset. Polkuilmauksen tuloksen solmujen ylin taso on järjestetty solmujen alkuperäisen järjestyksen mukaan, ylhäältä alas, oikealta vasemmalle. Polkuilmauksen tulos voi sisältää duplikaattiarvoja, eli solmuja, joilla on sama typpi ja sisältö. [Connolly and Begg, 2002.]

XQueryn FLWOR-rakenne tarjoaa monipuoliset mahdollisuudet erilaisten kyselyjen muodostamiseen ja lajitteluun sekä antaa mahdollisuuden vaikuttaa myös kyselyn tulostusmuotoon. FLOWR-rakenteinen kysely koostuu osista, joita ovat for, let, where, order by ja return. FLOWR-ilmaus sitoo arvot yhteen tai useampaan muuttujaan ja käyttää sitten näitä muuttujia tuloksen muodostamisessa [Connolly and Begg, 2002].

> **for:** Lauseke käy läpi annetut elementit. Se toimii ohjelmointikielten toisto-operaatioiden tapaan ja sitä seuraavat lausekkeet suoritetaan jokaiselle elementille erikseen. For-lauseen tulos on lista monikkoja.
> **let:** Lausekkeessa annetaan arvo halutuille muuttujille. Myös let-lause sitoo yhden tai useamman muuttujan yhteen tai useampaan ilmaukseen mutta ilman toistoa palauttaen tuloksena vain yksinkertaisen sidoksen jokaiselle muuttujalle.
> **where:** Lausekkeessa annetaan ehtoja for- ja let-lauseissa valittaville elementeille eli rajataan tulokseen tulevia monikkoja.
> **order by:** Lauseke järjestää tuloksen elementit määrättyyn järjestykseen.
> **return:** Lausekkeessa määritellään, mitä tuloksessa palautetaan ja miten tulokset esitetään.

FLOWR-ilmaus voi sisältää useita for- ja let-lauseita ja jokainen näistä lauseista voi sisältää viittauksia muuttujiin, jotka on sidottu edellisissä lauseissa [Connolly and Begg, 2002]. For-rakenteen avulla voidaan muodostaa sisäkkäisiä toisto-operaatioita kuten ohjelmointikielissä. XQuery mahdollistaa siis hakujen kohdistumisen sisällön lisäksi myös dokumenttien rakenteeseen.

XQuery-kyselykieleen on sisäänrakennettu yli sata funktiota moniin eri tarkoituksiin. Omia funktioita voidaan tehdä niin kyselyyn kuin ulkopuoliseen

kirjastoonkin. XQuery-funktioita on esimerkiksi numeeristen arvojen, merkkijonojen, päivämäärien ja Boolen arvojen käsittelyyn. Lisäksi funktiot voivat mahdollistaa esimerkiksi nimien ja nimiavaruuksien, solmujen ja sarjojen käsittelyn sekä vertailun. [W3C, 2010b.]

## 4.2 XTerm-kyselyiden muuntaminen XQuery-kyselyiksi

### 4.2.1 Kyselyn alkuosan muodostaminen ja ehtojen määrittely

Tässä työssä on toteutettu PHP-ohjelmointikielellä ohjelma, joka muuntaa käyttäjän syöttämät XTerm-kyselyt Berkeley DB XML:n ymmärtämiksi XQuery-kyselyiksi, suorittaa tämän kyselyn ja palauttaa tulokset näytölle. Tässä luvussa esitetään, miten käyttäjän syöttämä XTerm-kysely muutetaan ohjelmallisesti XQuery-kyselyksi.

Jokainen kyselyn |-merkillä eroteltu osa vastaa yhtä sanaa kohdetekstissä. Jokaista tällaista kyselyn osaa kohden tulee yksi for-lause XQuery-lausekkeeseen. Ensimmäinen for-lause asetetaan hakemaan kaikki XML-dokumentin lemma-elementit, jotka täyttävät halutut ehdot. Haettu polku asetetaan muuttujaan nimeltä polku1, jotta mahdollinen seuraava sana voidaan löytää liikkumalla seuraavassa for-lauseessa tästä edellisen sanan polusta eteenpäin. Polussa lähdetään liikkeelle kontaineri.dbxml-nimisestä tiedostosta, jonka juurielementti on doc-niminen. Juurielementistä siirrytään sen lapseen nimeltä kappale (p) ja seuraaviin jälkeläisiin, joita ovat lause (s) ja sana (w). Jos haettavalle sanalle ei ole asetettu ehtoja ja kyselyssä haetaan vain yksittäisiä sanoja, ensimmäinen for-lause on muotoa:

```
for $polku1 in collection("kontaineri.dbxml")/
doc/p/s/w/ lemma
```

Jos haettava lemma ei ole viimeinen eli kyseessä on sana, jota seuraa muita sanoja, asetetaan for-lauseessa ehto, jolla rajataan tuloksesta pois sellaiset sanat, jotka esiintyvät lauseen viimeisenä. Tämä tehdään siksi, ettei tulokseen haluta sanayhdistelmiä yli lauserajojen. Polkulausekkeen elementeille voidaan asettaa ehtoja hakasulkeissa elementin nimen perässä. Lemma-elementin ehdoksi asetetaan, että sen vanhemmalla, joka on elementti w, on olemassa seuraava sisarus, elementti, jonka nimi on w ja tällä on oltava lapsena lemma-elementti. Merkintä `*[1]` palauttaa ensimmäisen elementin haetusta sarjasta eli tässä tapauksessa elementin w seuraavista sisaruksista ensimmäisen. Ensimmäisen sanan hakeminen silloin, kun sitä seuraa kyselyssä toinen sana, on:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[parent::w/following-sibling::*[1]
[name()="w"]/child:: lemma]
```

Kyselyn seuraavaa sanaa vastaava lemma haetaan viittaamalla edellisen sanan lemmaan ja lähtemällä polussa liikkeelle tuosta edellisen sanan lemmasta. Tämän jälkeen viitataan jo haetun elementin vanhempaan ja haetaan vanhemman seuraava sisarus. Jos tämän elementin nimi on w, jatketaan polkua ja palautetaan w:n lapsielementti, joka on lemma. Näin saadaan haettua tekstissä seuraavana esiintyvän sanan lemma-elementti. Kyselyn toista sanaa hakeva for-lause olisi:

```
for $polku2 in $polku1/parent::node()/following-
sibling:: *[1][name()="w"]/lemma
```

Vastaavalla tavalla muodostettaisiin seuraavat for-lauseet, jos kysely koostuisi useammasta kuin kahdesta sanasta. Liikkeelle lähdettäisiin tällöin edellisen haetun sanan polusta, haettaisiin taas vanhemman seuraavista sisaruksista ensimmäinen ja tarkistettaisiin, että kyseinen elementti on nimeltään w ja tällä on lapsi nimeltä lemma.

```
for $polku3 in $polku2/parent::node()/following-
sibling::*[1][name()="w"]/lemma
```

Kysely `A | ’laki’ => l | l` muodostuisi edellä kuvatun tavoin kahdesta for-lauseesta. Ensimmäisessä haetaan sellaisia lemma-elementtejä, joiden tags-niminen attribuutti sisältää tekstin ’|A|’ eli on adjektiivi. Attribuutin sisältöä haettaessa voidaan hyödyntää XQueryn valmista funktiota contains, jolla voidaan hakea tietoa siitä, mitä elementin attribuutti sisältää. Lemma-nimisen elementin tags-nimisen attribuutin tulee sisältää teksti ’|A|’. Lisäksi, koska tiedetään, että ollaan hakemassa kahta peräkkäistä sanaa, asetetaan edellä kuvatun tavoin ehto, että nyt palautettavalla lemma-elementillä on oltava vanhemman (w) ensimmäisen seuraavan sisaruksen (w) lapsena myös lemma-elementti. Ensimmäistä sanaa hakeva for-lause olisi muotoa:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[parent::w/following-
sibling::*[1][name()="w"]/child:: lemma and
contains(@tags,"|A|")]
```

Seuraava for-lause määritellään kuten aiemmassakin esimerkissä hakemalla edellisen for-lauseen palauttamien elementtien vanhempien seuraava sisarus ja tämän lapsi. Ehdoksi asetetaan se, että lemma-nimisen lapsielementin tekstinä on oltava laki. Tässä käytetään apuna valmista funktiota text, joka tunnistaa elementin tekstisisällön. Koska yhdyssanoja ei ole tallennettu suoraan lemma-elementin tekstiksi vaan ne muodostuvat segm-nimisistä lapsielementeistä, on kyselyyn lisättävä myös ehto, jossa

liitetään kaikki kyseisen lemma-elementin lapset yhdeksi tekstijonoksi. XQueryn-kyselykielen string-join -funktiolla voidaan yhdistää useampi merkkijono yhdeksi. Tämän yhdistetyn merkkijonon tulee olla haluttu lemma eli tässä tapauksessa laki. Laki ei ole yhdyssana, mutta koska kyselyä muodostettaessa, ei tiedetä, onko kyseessä yhdyssana vai ei, lisätään liitosehto aina, kun hakua rajataan lemmalla.

```
for $polku2 in $polku1/parent::node()/following-
sibling::*[1][name()="w"]/lemma[(text()='laki' or
string-join(child::segm/text(), '')='laki')]
```

XTerm-kielen negaatiot esitetään XQuery-kyselyssä käyttämällä valmista not-funktiota ja yhdistämällä se jo aiemmin esitettyyn contains-funktioon. Jos käyttäjä antaa ehdon –A eli sanaluokka ei saa olla adjektiivi, olisi kyseisen sanan for-osa muotoa:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/lemma[not(contains (@tags, "|A|"))]
```

Vastaavalla tavalla negaatiofunktio not liitetään ehto-osaan myös silloin, kun ei haluta, että mukaan tulee tietty lemma. Jos esimerkiksi on annettu ehto –’laki’, olisi kysely muotoa:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[not(text()='laki' or string-
join(child::segm/ text(),'')='laki')]
```

Jos käyttäjä tekee rajauksen, että mikään kohdetekstin sanan lemma-lapsielementeistä ei saa olla adjektiivi, merkitään XTerm-kielessä --A, käytetään silloinkin valmista not-funktiota. Tällöin määrätään, että lemman vanhemmalla (w) ei saa olla lasta (lemma), jonka tags-attribuutti sisältää (contains) annetun sanaluokka-merkinnän ’A’.

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[not(parent::w/ child::lemma
[contains(@tags, "|A|")])]
```

Jos käyttäjä tekee rajauksen sille, että mikään haetun sanan lemmoista ei saa olla laki, merkitään XTerm-kielessä --‘laki’, käytetään yllä esitettyä tapaa asettaa ehtoja vanhemman muille lapsille. Tähän yhdistetään not-funktion käyttö, eli ei saa olla vanhempaa w, jolla olisi lapsielementti lemma, jonka tekstinä olisi laki. Ei myöskään saa olla vanhempaa w, jolla olisi lapsielementti lemma, jonka segm-nimisten lapsielementtin sisältöjen yhdistelmä olisi laki. Kysely olisi kokonaisuudessaan:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/lemma[not(parent::w/child::lemma[(text()=
'laki' or string-join(child::segm/text(),'')=
'laki')])]
```

### 4.2.2 Kyselyn loppuosan määritteleminen

Käyttäjä määrittelee kyselyn loppuosassa, halutaanko tulos palauttaa lemmamuodossa vai siinä muodossa, jossa sanat esiintyvät tekstissä. XQuery-kyselyssä tämä määritellään return-osassa. Myös return-lauseessa on huomioitava se, että yhdyssanat on tallennettu segm-elementteihin. Ensin return-lauseessa tarkistetaan, onko for-lauseessa asetetulla polkumuuttujan palauttamalla elementillä lapsi, jonka nimi on segm. Jos on, palautetaan segm-elementtien tekstit. Jos segm-elementtiä ei ole eli sana ei ole yhdyssana, palautetaan tulos vain hakemalla kyseisen lemma-elementin sisältö valmiilla funktiolla text eli `$polku1/text()`. Tulosta voidaan myös muotoilla return-lauseessa. Palautettavaan tekstiin voidaan liittää esimerkiksi lemma tai w -tagit riippuen siitä, kumpi tuloksessa on palautettu. Jos tulos halutaan lemmamuodossa, return-osa on muotoa:

```
return <lemma> {if($polku1/child::segm) then
$polku1/segm/text() else $polku1/text()} </lemma>
```

Jos tulos halutaan siinä muodossa, jossa se esiintyy tekstissä, palautetaan polku1-muuttujan eli haetun lemman vanhemman w sisältämä teksti. Elementin vanhempi voidaan hakea merkinnällä ’..’, jolla siirrytään polussa yksi askel ylöspäin. Kun tulokseen halutaan sanan esiintymismuoto tekstissä, on return-lause muotoa:

```
return <w> {$polku1/../text()} </w>
```

Kysely `| => l | l` palauttaisi kaikki mahdolliset peräkkäiset sanaparit sanojen lemmamuodoissa. Ensimmäinen for-lause asettaa polku1-muuttujaan polut kaikkiin mahdollisiin lemma-nimisiin elementteihin, jotka ovat dokumentin (doc), kappaleen (p), lauseen (s) ja sanan (w) sisässä eli näiden jälkeläisiä. Koska muita ehtoja ei ole, ehdoksi asetetaan ainoastaan se, että sana ei voi olla lauseen viimeinen eli lemma-elementin vanhemman seuraavalla sisaruksella on lapsena lemma-elementti. Toinen for-lause sijoittaa polku2-muuttujaan ensimmäisenä haetun polun vanhemman (`parent::node()`) seuraavista sisaruksista (following-sibling) ensimmäisen (`*[1]`), jonka nimi on w ja palauttaa polun tämän lapseen, jonka elementin nimenä on lemma. Kysely olisi alkuosaltaan:

```
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[parent::w/following-sibling::*[1]
[name()="w"]/child:: lemma]
for $polku2 in $polku1/parent::node()/following-
sibling::*[1][name()="w"]/lemma
```

Kahden peräkkäisen elementin tulostaminen lemma-muodossa return-lauseilla olisi:

```
return <pari>
{if($polku1/child::segm) then $polku1/segm/text()
else $polku1/text()}
{if($polku2/child::segm) then $polku2/segm/text()
else $polku2/text()} </pari>
```

Vastaavasti jokainen uusi elementti lisäisi uuden osan ja uuden if-lauseen return-osaan, kun halutaan tulos lemmamuodossa. Kysely `A | A => l | l` olisi kokonaisuudessaan:

```
let $space = “ ”
for $polku1 in collection("kontaineri.dbxml")/doc/
p/s/w/ lemma[parent::w/following-sibling::*[1]
[name()="w"]/ child::lemma and contains(@tags,
"|A|")]
for $polku2 in $polku1/parent::node()/following-
sibling::*[1][name()="w"]/lemma[contains(@tags,
"|A|")]
return <yhdistelma> if($polku1/child::segm) then
$polku1/segm/text()
else $polku1/text()}{$space}
{if($polku2/child::segm)
then $polku2/segm/text()
else $polku2/text()}</yhdistelma>
```

Tällöin tulos olisi esimerkiksi:

```
<yhdistelma> erittäin vihainen </yhdistelma>
<yhdistelma> hyvin varakas </yhdistelma>
```

Kyselyssä voidaan määritellä palautettavaksi sekä sanojen lemma-muodot että sanojen esiintymismuodot. Tämä merkitään XTerm-kyselyn loppuosassa `w,l`. Tällöin asetetaan XQueryn return-lause palauttamaan nämä molemmat. Ensin palautetaan polkumuuttujan vanhemman tekstisisältö funktiolla text eli sanan esiintymismuoto. Tämän lisäksi palautetaan edellä esitetysti, joko lemma-elementin tekstisisältö tai

lemma-elementin segm-lapsielementtien tekstisisältö. Elementin kaikki jälkeläiset saadaan haettua merkinnällä *. Koska lemma-elementillä ei ole muita kuin segm-nimisiä lapsia, voidaan käyttää *-merkintää:

```
return {$polku1/../text()}
{if($polku1/child::segm) then $polku1/*/text()
else $polku1/text()}
```

### 4.2.3 Toisto- ja valinnaisoperaattoreiden toteuttaminen

XTerm-kyselykielessä käyttäjä voi esittää kaksi erilaista kyselyä ja liittää ne toisiinsa OR-operaattorilla. Tällöin ohjelma muodostaa kaksi kyselyä yllämainituilla tavoilla ja tuloksena palautetaan sanat, jotka täyttävät jonkin OR-operaattorilla yhdistetyistä ehdoista. OR-operaattoria ei suorita XQuery-kyselyssä.

Jos käyttäjä käyttää XTerm-kyselyssä valinnaisuutta merkitsevää ?-operaattoria, vastaa tämä käytännössä OR-operaattorin käyttöä. Joko ?-operaattorilla merkitty osuus esiintyy tai ei esiinny kyselyn tuloksissa. Jos käyttäjä syöttäisi kyselyn `?A | A | A => l | l | l`, muodostaa ohjelma kaksi kyselyä, `A | A => l | l` ja `A | A | A => l | l | l`. Molemmat kyselyt suoritetaan ja tuloksessa palautetaan kaikki yhdistelmät, jotka kumpi tahansa kyselyistä palauttaa.

XTerm-kyselykielen toisto-operaattori, esimerkiksi [2], toimii vastaavalla tavalla kuin valinnaisuusoperaattori ?. Myös toisto-operaatiot muodostetaan niin kuin kysely koostuisi useasta vaihtoehtoisesta kyselystä. Kysely `[2] A | A => l | l` vastaa täysin edellä esitettyä kyselyä `?A | A | A => l | l | l`. Ensimmäinen osa esiintyy joko kerran tai kaksi, joten tässäkin tapauksessa suoritetaan sekä kysely `A | A => l | l` että kysely `A | A | A => l | l | l`.

### 4.2.4 Frekvenssien palauttaminen

Kyselyssä voidaan määritellä haettavaksi sanan tai lemman frekvenssi tulosjoukossa tai koko kokoelmassa. Nämä merkitään kyselyn loppuosaan liittämällä tulostusmuotojen eli w ja l -merkintöjen perään pilkulla erotettuna frr eli frekvenssi tulosjoukossa tai frc eli frekvenssi koko tekstikokoelmassa. Myös frekvenssimääreet molemmat voidaan liittää samaan kyselyyn. Frekvenssilaskelmia ei kuitenkaan tehdä XQuery-kyselyssä vaan ne lasketaan PHP-ohjelmassa, joka suorittaa kyselymuunnoksen ja ajaa kyselyn tietokantaan sekä palauttaa tuloksen.

Vastaavasti voidaan kyselyssä määritellä, että halutaan koko haetun yhdistelmän frekvenssi. Tällöin merkitään FR kyselyn kolmanneksi osaksi. Myös koko yhdistelmän frekvenssi lasketaan ohjelmassa, joka suorittaa kyselyn ja palauttaa tuloksen.

### 4.3 Ohjelman kuvaus

PHP-ohjelma, joka muuntaa käyttäjän syöttämän kyselyn XQuery-kyselyksi, suorittaa kyselyn ja näyttää tulokset, on toteutettu proseduraalisesti. Se koostuu pääohjelmasta, joka kutsuu aliohjelmia. Jokainen aliohjelma suorittaa jonkin kokonaisuuden kyselyn muuntamisessa XQuery-kyselyksi. Ensimmäiseksi pääohjelma kutsuu tarkistaKysely-funktiota, joka tarkistaa, että käyttäjän syöttämä kysely on validi. Tämän jälkeen ohjelma kutsuu kasittele-funktiota. Kasittele-funktio kutsuu kasitteleKysymysmerkki- ja kasitteleNumerot -funktioita, jotka muuttavat kysymysmerkit ja numerot or-operaattorein yhdistetyiksi kyselyiksi. Tämän jälkeen tämä pääohjelma kutsuu muodostaKysely-funktiota, joka muodostaa kyselystä XQuery-kyselyn. XQuery-kysely suoritetaan suoritaKysely-funktiossa, joka lisää kyselyn tai osakyselyn tulokset taulukkoon. Lopuksi tulostaTulokset-funktio tulostaa taulukon näytölle ja laskee tarvittavat frekvenssit tulosjoukossa.

# 5 Vertailtavat korpuskyselykielet

## 5.1 Emdros ja MQL

### 5.1.1 Tausta ja tavoitteet

Emdros on avoimen lähdekoodin tekstihakujärjestelmä, joka on tarkoitettu kielitieteellisiin analyyseihin ja tekstin merkitsemiseen. Sitä voidaan käyttää erityisesti korpuskielitieteessä kielitieteellisten analyysien tekemiseen ja tallentamiseen. Emdros liitetään tiettyyn sovellusohjelmaan ja sitä voidaan käyttää erilaisten korpusten kanssa. Avoimen lähdekoodin ohjelmana Emdros on ollut lokakuusta 2001. [Petersen, 2004.] Lähdekoodia voidaan muokata omiin tarpeisiin sopivaksi ja käytettäväksi yhdessä erilaisten korpusten kanssa.

Abeillén [2003] mukaan olemassa olevien korpuskielten ongelmana on se, ettei niillä voi muodostaa kyselyjä liittyen subjektien käänteiseen sanajärjestykseen eikä agentittomiin passiiveihin. Emdrosin tarkoituksena on tarjota apua näihin ongelmiin [Petersen, 2004]. Petersenin [2004] mukaan Emdros mahdollistaa monimutkaisempien kielellisten kyselyiden tekemisen kuin suurin osa järjestelmistä nykyään.

### 5.1.2 Tietorakenne

Emdros lähestyy korpuskyselyitä hieman eri tavalla kuin muut kyselykielet. EMdF-malli on laajennos Christ-Jan Doedensin vuonna 1994 väitöskirjassaan kehittämästä MdF-mallista (Monads-dot-Features), johon kuului QL-kyselykieli. Emdrosin EMdF-malli perustuu neljään käsitteeseen, jotka ovat monadi, objekti, objektityyppi ja ominaisuus. Monadit ovat kokonaislukuja ja niiden sarja vastaa tiettyä tekstijonoa. Monadit eivät määrää lukusuuntaa vaan pelkästään loogisen tekstijärjestyksen. Objektit ovat sarja monadeja ja kuuluvat tiettyyn objektityyppiin. Objektit muodostavat lohkoja tekstistä ja jokaisella objektilla on yksilöivä numero. [Petersen, 2004.]

Objektityyppejä ovat esimerkiksi sana, fraasi, lause, sivu ja kappale. Objektityyppi määrittelee objektin attribuutit eli ominaisuudet. Esimerkiksi sanalla voi olla ominaisuuksina sanaluokka ja lemma. [Petersen, 2007.] Ominaisuus voi olla tyypiltään merkkijono, kokonaisluku, lista tai objektitunnus. Objektitunnusten avulla voidaan määritellä monimutkaisia suhteita objektien välillä, kun tietyn objektin ominaisuus voi saada arvokseen toisen objektin tunnuksen. Listaominaisuus tarkoittaa sitä, että ominaisuus voi saada arvokseen listan. Suunniteltaessa Emdros-tietokantaa valitaan monadirakeisuus, joka määrää mikä on tietokannan pienin objekti. Tämä vastaa yhtä monadia. Yleensä pienin objekti on sana, joten esimerkiksi tekstin ensimmäistä sanaa vastaa objektisarja {1} ja vastaavasti tekstin ensimmäinen kolmen sanan fraasi olisi objektisarja {1,2,3}. [Petersen, 2004.] Lähestymistapa on siis tyypillinen oliolähestymistapa. Monadi vastaa oliotietokannan litteraalia.

Taulukko 2. Esimerkikki EMdF-tietokannasta [Petersen, 2004]

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **word** | **w: 10001**<br>surface: The<br>psp: article | **w: 10002**<br>surface: door<br>psp: noun | **w: 10003**<br>surface: was<br>psp: verb | **w: 10004**<br>surface: blue<br>psp: adjective |
| **phrase** | **p: 10005**<br>phr_type: NP | | **p: 10006**<br>phr_type: VP | **p: 10007**<br>phr_type: AP |
| **clause** | **c: 10008** | | | |

Taulukko 2 havainnollista EMdF-tietokannan rakennetta. Taulukossa on kolme objektityyppiä sana, fraasi ja lause. Sanan ominaisuudet ovat sanan esiintymismuoto (surface) ja sanaluokka (psp). Fraasin ominaisuus on fraasityyppi (phr_type). Monadirakeisuus on tässä tapauksessa yksi sana. Esimerkiksi sanan w:1002 esiintymismuoto on door ja sanaluokka on substantiivi. Ensimmäistä sanaa vastaisi monadisarja {1} ja koko lausetta ’The door was blue’ vastaisi monadisarja {1,2,3,4}. Objekteille merkittävät ominaisuudet riippuvat siitä, miten tietokanta halutaan määritellä, ja mitkä ovat tutkijan tavoitteet [Petersen, 2004].

### 5.1.3 Ominaisuudet

MQL-kyselykieli on EMdF-tietokannan kyselykieli ja se perustuu kahteen tekstin ominaisuuteen, peräkkäisyyteen ja sisäkkäisyyteen. MQL:n vahvuus on sen kyvyssä ilmaista monimutkaisia peräkkäisiä ja sisäkkäisiä hakurakenteita sekä rakenne- että objektitasolla [Petersen, 2004]. Kyselykielen elementeillä kuvataan tekstin rakennetta.

Jokaisen kyselyn alkuun on liitettävä teksti ”`SELECT ALL OBJECTS WHERE`”, jotta Emdros tunnistaa, että kyseessä on kielitieteellinen kysely. Kyselyt muodostuvat lohkoista, jotka ympäröidään hakasulkein. Lohko voi olla objektilohko (esimerkiksi `[word]`), voimalohko (`..`) tai välilohko (`[gap]`). Lohkoilla kuvataan sitä, miten kyselyn palauttamien objektien tulee sijoittua tekstissä. Lohkot voidaan sijoittaa kyselyssä peräkkäin tai sisäkkäin. Kyseisten lohkojen kuvaamien elementtien tulee esiintyä vastaavalla tavalla tekstissä joko peräkkäin tai sisäkkäin. Objektilohkoihin merkitään, mitä objektia haetaan, esimerkiksi lausetta tai sanaa, ja liitetään objektille halutut ehdot. Voimalohkoilla kuvataan sitä, että voimalohkoa ympäröivien lohkojen ei tarvitse olla vierekkäisiä. Lisäksi voimalohkoihin voidaan liittää numeroin ja vertailuoperaattorein lohkon pituus monadeina. Esimerkiksi kysely `[Phrase] ..<= 5 [Phrase]` palauttaisi kaikki sellaiset kaksi fraasia (phrase), jotka esiintyvät enintään viiden monadin päässä toisistaan. Välilohkojen avulla on tarkoitus etsiä tietynlaisia välejä ympäröivästä kontekstista. Välilohko voidaan merkitä valinnaiseksi

kysymysmerkillä. [Petersen, 2007.] Esimerkiksi virkkeitä, joiden keskellä on relatiivilause, voidaan hakea määrittelemällä ensin lohko, joka sisältää virkkeen, ja jonka sisässä on välilohko, joka taas sisältää relatiivilauseen.

Objektien ominaisuusrajoitukset merkitään lohkojen sisään, esimerkiksi `[Word lemma="law"]` palauttaisi kaikki sanat, joiden lemma on 'law'. Ominaisuuksien arvot asetetaan kuten muissakin kyselykielissä rakenteella 'ominaisuus operaattori arvo'. Lohkojen sisällä voidaan käyttää Boolen operaattoreita ja sulkeita yhdistämään erilaisia ehtoja. Vertailuoperaattoreista käytössä on <>, <, ja >=. Merkintä ~ tarkoittaa säännöllistä lauseketta, ja merkintä !~ tarkoittaa säännöllisen lausekkeen negaatiota. Vertailuoperaattoreiden lisäksi voidaan tässä yhteydessä käyttää rajoitusta `BETWEEN X AND Y`, jonka avulla voidaan määritellä millä välillä tilan tulee monadeissa olla. Näiden lisäksi on käytössä muista vertailluista kielistä poiketen operaattorit IN ja HAS. IN-operaattorin oikealla puolella sulkeiden sisällä määritellään arvot, jotka operaattorin vasemmalla puolella oleva ominaisuus voi saada. HAS-operaattori on käänteinen IN-operaattorille. Sen avulla voidaan etsiä yhtä arvoa objektien listaominaisuuksista eli asetetaan arvo, jonka halutaan listassa esiintyvän. [Petersen, 2007.]

Kyselyissä voidaan käyttää säännöllisiä lausekkeita ja niihin liittyviä toisto-operaattoreita (*, +, ?). Toisto-operaattorit voidaan liittää yhteen lohkoon tai lohkojen yhdistelmään. Toisto-operaatioiden määrä voidaan myös rajata tietylle välille. Samalla tavoin kuin muissakin korpuskyselykielissä, toisto-operaatioiden määrä asetetaan aaltosulkeiden sisään ja vähimmäismäärä ja enimmäismäärä erotellaan toisistaan pilkulla tai viivalla. Ennen aaltosuljemerkintää merkitään toisto-operaation symboli *. Kaksi merkkiä, merkkiluokkaa tai ryhmää voidaan liittää toisiinsa tai-operaattorilla. Jos kaksi merkkijonoa halutaan asettaa tai-operaattorilla vaihtoehdoiksi, tulee merkkijonot merkitä sulkeiden sisään. [Petersen, 2007.] Pisteellä voidaan viitata kyselyssä mihin tahansa merkkiin. Myös pisteeseen voidaan yhdistää toisto-operaattorit.

Kyselyissä on mahdollista antaa objekteille nimi ja viitata niihin myöhemmin kyselyssä. Nimi annetaan AS-operaattorin avulla. Myöhemmin kyselyssä nimettyyn objektiin voi viitata pistenotaatiolla. Nimettyjä objekteja voidaan käyttää muuttujina ja muodostaa rakenteita ja ehtoja, jotka ilman muuttujia eivät olisi mahdollisia. [Petersen, 2007.]

MQL-kielessä voidaan määritellä FOCUS-avainsanalla tietty osa kyselyä niin sanottuun fokukseen. Riippuu sovelluksesta, miten fokuksessa olevat tulokset esitetään. Osa kyselystä voidaan määritellä tuloksen ulkopuolelle NORETRIEVE-sanalla. Avainsanojen FIRST ja LAST avulla voidaan määritellä tietyn objektilohkon sijainti sitä ympäröivän lohkon sisällä. Yhdistelmällä FIRST AND LAST määritellään lohko ainoaksi kontekstissaan eli kyseisten hakasulkeiden sisällä. NOTEXIST-avainsanalla voidaan määritellä, ettei tietty objektilohko saa esiintyä annetussa kontekstissa.

Huomioitava on, että NOTEXIST ja <> -operaattorit toimivat eri tavoin. Esimerkiksi kysely

```
[Sentence
   NOTEXIST [Word surface = 'see']
]
```

palauttaa kaikki lauseet, joissa sana 'see' ei esiinny.

Sen sijaan kysely

```
[Sentence
   [Word surface <> 'see']
]
```

palauttaa kaikki lauseet, joissa on yksikin sana, joka ei ole 'see'. Jälkimmäinen ei todennäköisesti ole yleensä haluttu kysely, joten erisuuruusoperaattorin sijasta tulee käyttää NOTEXISTS-avainsanaa. [Petersen, 2007.]

Myös lohkoja voidaan yhdistellä Boolen operaattorein. OR-operaattorilla voidaan muodostaa vaihtoehtoja erilaisista lohkoista tai lohkoyhdistelmistä. Hakasulkeiden avulla lohkoja voidaan yhdistellä kokonaisuuksiksi ja näin tehostaa OR-lauseiden käyttöä.

### 5.1.4 Puutteet ja tärkeimmät ominaisuudet

Erilaiset lohkot (objektilohko, voimalohko ja välilohko) sekä sisäkkäisyyden ja peräkkäisyyden kuvaaminen mahdollistavat erittäin monimutkaisten hakurakenteiden kuvaamisen. Tätä voidaan vielä monipuolistaa yhdistämällä lohkoihin Boolen operaattoreita, säännöllisiä lausekkeita ja toisto-operaatioita. Kieli soveltuukin erityisen hyvin pidempien tekstikokonaisuuksien ja lauserakenteiden tarkasteluun. Muut tässä työssä vertaillut kielet eivät mahdollista yhtä monipuolisia lauserakennehakuja.

MQL-kyselykieli muistuttaa vertailluista kyselykielistä eniten ohjelmointikieltä tai perinteistä SQL-kyselykieltä SELECT ja WHERE -määreineen. Kielessä on ominaisuuksia, jotka muista kielistä puuttuvat täysin. Muista kielistä ei löydy esimerkiksi IN-, HAS- ja NOTRERIEVE -operaatioita. Kielen käyttö saattaa olla hidasta oppia sen monipuolisuuden ja monimutkaisuuden takia.

Petersen [2004] toteaa MQL-kyselykielen voiman olevan sen kyvyssä ilmaista monimutkaisia hakurajoituksia sekä rakenne- että objektitasolla. Rakennetasoon kuuluu peräkkäisyys ja sisäkkäisyys ja objektitasoon viittaaminen erilaisiin tekstin objekteihin kuten lauseisiin, sivulauseisiin, fraaseihin ja sanoihin. Hän myös toteaa, että Emdros tarjoaa vakaan alustan, jolle voidaan rakentaa korpuskielitieteellisiä sovelluksia, jotka

voivat vastata monimutkaisempiin kielitieteellisiin kysymyksiin kuin useimmat tarjolla olevat järjestelmät tällä hetkellä.

## 5.2 NITE XML Toolkit ja kyselykieli NXT

### 5.2.1 Tausta ja tavoitteet

NITE-projekti on Euroopan komission rahoittama vuosina 2001 – 2003 käynnissä ollut Human Language Technology -projekti. Lyhenne NITE tulee sanoista Natural Interactivity Tools Engineering. Projektin tarkoituksena on kehittää työkaluja monitasoisten, multimodaalisten merkintöjen hakuun ja hyödyntämiseen luonnollisessa monien osapuolten välisessä interaktiivisessa kommunikointidatassa. Kommunikaatio voi olla ihmisten välistä tai ihmisten ja koneiden välistä. Tällä hetkellä projektiin liittyvän ohjelmiston kehittäminen jatkuu avoimen lähdekoodin ohjelmana. [NITE, 2003.]

NITE-projektissa on kehitetty työkaluja luonnollisen vuorovaikutteisen viestinnän merkitsemiseen ja analysointiin. Työkalujen tarkoitus on auttaa kehittämään monimutkaisten luonnollisten vuorovaikutustilanteiden ymmärtämistä mahdollistamalla datan tai korpusten merkitseminen ja analysointi. NITE-työkaluille on asetettu erityisiä vaatimuksia johtuen siitä, että data on pääsääntöisesti akustista ja graafista. Lisäksi NITE-työkalujen tulee olla käyttökelpoisia niin asiantuntijoille kuin opiskelijoillekin. [NITE, 2003.]

NITE XML Toolkit (NXT) sisältää kyselykielen NXT Query Language (NQL). Se perustuu MATEn (Multilevel Annotation Tools Engineering) kyselykieleen Q4M. NITE XML Toolkit sisältää graafisen käyttöliittymän ja on saatavilla osoitteesta http://sourceforge.net/projects/nite/files/. NQL-kyselykieli on tarkoitettu erityisesti ääni- tai videomuodossa olevan datan käsittelyyn. Näin ollen se eroaa selvästi muista vertailtavista kyselykielistä, mutta samalla tuo arvokkaan lisän tässä työssä tehtävään vertailuun. Tulevaisuudessa korpuksia saatetaan tallentaa yhä enemmän ääni- ja videomuodossa, minkä takia on syytä perehtyä hieman tähänkin alueeseen.

### 5.2.2 Tietorakenne

NQL-kyselykieli perustuu NXT-objektimalliin. Malli on järjestettyjen puiden formalismi, jota on laajennettu leikkaavilla hierarkioilla niin, että se pyrkii ylläpitämään hierarkkista ja peräkkäistä rakennetta. Kuten XML-objektimallissa myös NXT-objektimallissa objektina on puun solmu. NXT-objektimalli soveltuu hyvin merkityn multimodaalisen datan kuvaamiseen. Se keskittyy objektien ja niiden suhteiden kuvaamiseen sekä objektien ja niiden suhteiden rakenteiden monimutkaisuuden rajoittamiseen. Näin se helpottaa datan hallittavuutta ja haettavuutta. [Evert et al., 2003, 2.] NXT perustuu hierarkkiseen tietomalliin ja on osittain logiikkapohjaista. Objektit järjestetään hierarkioihin käyttäen vanhempi-lapsi -suhteita. Objektit voidaan järjestää

myös monimutkaisempiin rakenteisiin hyödyntäen osoittimia, joilla on nimetty rooli [Carletta et al., 2005].

NOM-korpuksen komponentit ovat:

1. Attribuutit, joihin on tallennettu tietoa merkkijonoina, ja elementit, joihin attribuutit ryhmitellään. Erikoisattribuutit yhdistävät elementit yhteiselle aikajanalle.
2. Rakenteiset suhteet elementtien välillä.
3. Rajoittamattomat osoittimet elementtien välillä.
4. Kerrokset, joihin korpus jaetaan, ja jotka määrittelevät nimetyt hierarkiat.
5. Metadata, joka tarjoaa informaatiota muun muassa elementtityypeistä, attribuuttien arvoalueista, osoittimien rooleista, kerroksista ja nimetyistä hierarkioista. [Evert et al., 2003, 4.]

NXT-objektimalli sisältää korpukseen koodattua rakenteellista informaatiota. Tätä rakenteellista informaatiota on:

1. dominanssijärjestys (dominance ordering) ja vertikaalinen etäisyys (vertical distance),
2. edeltävyysjärjestys (precedence ordering) ja sarjallinen etäisyys (sequential distance) jokaisessa hierarkiassa,
3. horisontaalinen etäisyys (horizontal distance), joka saavutetaan horisontaalisen akselin projektiona,
4. monet rajoittamattomat osoitingraafit (pointer graphs) ja
5. erilaiset ajalliset järjestykset (temporal orderings), jotka on johdettu ajoitetusta informaatiosta. [Evert et al., 2003].

Koska järjestetyt puut laajentavat hierarkkisen puurakenteen peräkkäisjärjestyksellä, niistä on tullut keskeisiä monissa sovelluksissa, esimerkiksi kielitieteessä ja luonnollisen kielen prosessoinnissa. Hierarkkinen dimensio kuvataan yleensä pystysuoralla akselilla ja siihen viitataan dominanssilla. Täydentävä peräkkäisjärjestysdimensio kuvataan vaakasuoralla akselilla ja siihen viitataan edeltävyydellä. Jokaisella järjestetyn puun solmuparilla on olemassa joko dominanssi- tai edeltävyyssuhde, mutta ei molempia. XML:ssä tämä dominanssisuhde kertoisi esivanhemman ja jälkeläisen suhteesta kun taas edeltävyyttä samassa merkityksessä ei XML:ssä ole. [Evert et al., 2003.]

### 5.2.3 Ominaisuudet

NQL-kysely koostuu muuttujien määrittelyistä ja ehto-osasta, jotka erotetaan toisistaan kaksoispisteellä. Muuttujat edustavat tekstissä olevia elementtejä. Ehto-osa muodostetaan Boolen lausekkein yhdistetyistä attribuuttitesteistä, sekä rakenteellisista että ajallisista relaatioista. Kyselyn tuloksena palautetaan lista tieto-objektien n-monikkoja, jotka täyttävät kyselyssä annetut ehdot. [Voormann et al., 2003; Carletta et al., 2005.]

Määrittelyosassa jokainen kyselyn muuttuja määritellään ja merkitään sulkeiden sisään. Määrittely koostuu muuttujan nimestä, joka alkaa $-symbolilla, sekä mahdollisesta muuttujalle asetettavasta tyypistä tai tyyppivaihtoehdoista. Vaihtoehdot erotellaan toisistaan |-operaattorilla. Määrittelyssä, `$a word`, muuttuja a sidotaan elementteihin, jotka ovat tyyppiä 'word'. Elementtimuuttujan tyyppi voidaan jättää myös merkitsemättä, jolloin kyseinen muuttuja voi vastata mitä tahansa elementtiä. [Voormann et al., 2003.]

Ehto-osassa voidaan käyttää operaattoreina negaatiota (!), konjunktiota (&), disjunktiota (|) sekä implikaatiota (→). Sulkeilla voidaan vaikuttaa operaattoreiden käsittelyjärjestykseen. Vertailuoperaattoreina ovat kaikki yleiset vertailuoperaattorit ==, !=, <, >, <=, =>. Muiden kielten tavoin voidaan käyttää myös säännöllisiä lausekkeita asetettaessa ehtoja. Attribuuttitesteillä verrataan kahta lauseketta. Lauseke voi olla attribuutin arvo, vakio tai funktion tulos. Valmiita funktioita ovat esimerkiksi aikaan liittyvät funktiot sekä id, joka palauttaa yksilöivän attribuutin arvon ja text, joka palauttaa elementin sisältämän tekstin. [Voormann et al., 2003.] Esimerkiksi kysely `($w word) : ($w@pos = ”N”) && (TEXT($w) = ”laki”))` palauttaa listan sanojen 1-monikkoja, joiden pos-attribuuttien arvo on N ja sanan tekstisisältö on laki.

NQL-kieli sisältää operaattoreita, joiden avulla päästään käsiksi informaation rakenteeseen eli edellä mainittuihin objektimallin viiteen rakennekohtaan. Yksinkertaisin rakenteellinen relaatio on kahden elementin yhtäsuuruuden vertailu. Elementti a dominoi elementtiä b, jos a on b:n esivanhempi. Dominointi merkitään symbolilla ^. Dominanssille voidaan määrätä myös vertikaalinen etäisyys eli polun pituus elementistä x elementtiin y kokonaislukuina. Etäisyyden lisäksi voidaan asettaa jälkimmäisen muuttujan horisontaalinen sijainti annetulla pystyakselilla, jonka etäisyys on määrännyt. Kysely `$p ^1 $c` asettaisi muuttujan c arvoksi mahdolliset $p:n lapset, koska etäisyys on yksi. Lisäksi voidaan määrätä, monesko kyseisistä elementeistä halutaan palauttaa. Negatiivinen luku aloittaa laskemisen akselin loppupäästä. Näin ollen kysely `$p ^1[1] $c` asettaisi muuttujaan c, $p:n ensimmäisen lapsen ja vastaavasti kysely `$p^1[-1] $c` asettaisi $c:ksi $p:n viimeisen lapsen. [Evert and Voormann, 2003.]

Edeltävyys määritellään joko nimetyn hierarkian eli tietyt ehdot täyttävän alikorpuksen tai puun juurielementin suhteen [Evert et al., 2003]. Kahdella elementillä on edeltävyyssuhde, jos niillä on yhteinen esivanhempi. Esivanhempi voi olla myös juurielementti. Edeltävyys merkitään symboleilla < >. Elementti x edeltää elementtiä y, jos jokin x:n esivanhempi on edeltäjä jollekin y:n esivanhemmalle tai y:lle itselleen. Edeltäminen merkitään kyselyissä muodossa `($a word) ($b word) : $a < > $b`, jolloin palautuu kaksi sanaa, joista $a edeltää $b:tä. [Voormann et al., 2003.] Myös edeltävyydelle voidaan merkitä etäisyys eli elementtien välimatka vaaka-akselilla. Elementtien sarjallinen etäisyys määritellään maksimaalisena elementtien lukumääränä kahden elementin välillä. Esimerkiksi `< > 1` hakee välittömät naapurit ja `1 < > 1` viereiset sisarukset. Negatiivisia arvoja voidaan käyttää edeltävyyssuhteissakin merkitsemään liikkumista horisontaalisella akselilla taaksepäin. Edeltävyydelle voidaan kyselyssä nimetä hierarkia, joka määrää sen, missä puun osassa edeltävyysvertailun kohteena olevien elementtien tulee sijaita. Esimerkiksi `$x <$a> $y` tarkoittaa sitä, että sekä $a dominoi $x:ää että $a dominoi $y:tä ovat tosia ja $x edeltää $y:tä. [Evert and Voormann, 2003.]

Horisontaalinen etäisyys esitetään sisäänrakennetuilla funktioilla D ja DA ja se lasketaan lineaarikerroksen projektiona. Muodossa `D(kerros, muuttuja1, muuttuja2)` kysely palauttaa muuttujien välisen horisontaalisen välimatkan, joka saavutetaan annetun kerroksen projektiona. Jos välimatkaa ei voida määritellä, palautetaan epätosi. `DA(L, $x, $y)` on absoluuttinen horisontaalinen välimatka ja se on validi, kun joko `D(L,$x,$y)` tai `D(L,$y,$x)` on määritelty ja palauttaa vastaavan arvon. [Evert and Voormann, 2003.]

NXT-objektimallissa osoittimilla voidaan ilmaista rajoittamattomia yhteyksiä elementtien välillä. Jokainen osoitin on linkki lähtöelementistä kohde-elementtiin ja osoittimella on rooli. Osoitingraafeja ei voida kuitenkaan yhdistää korpuksen hierarkkiseen ja peräkkäiseen rakenteeseen. Sovelluksilla, kuten kyselynkäsittelijällä, ei ole suoraa pääsyä paikallisten rakenteiden sulkeumiin ja niiden tulee käsitellä osoittimia askel kerrallaan. Nämä osoittimien rajoitukset heijastuvat NQL-kyselykieleen, joka tarjoaa monimutkaisia operaattoreita hierarkkisten ja peräkkäisten rakenteiden käsittelyyn, mutta osoittimille tarjotaan vain perustuki. [Evert et al., 2003.] Osoitin-operaattori merkitään kyselykielessä >-symbolilla. Osoitin-alipuuhun -operaattori, joka merkitään >^, yhdistää > ja ^ -operaattorit. Se palauttaa toden, jos ja vain jos, on olemassa elementti w kuuluu E:hen niin, että $x_1 \rightarrow_r$ `w ja` $x_2 \uparrow^*$ `w`. Kysely `$x >^ $y` vastaa kyselyä `{(exists $w): $x > $w & $y ^ $w)}`. [Evert and Voormann, 2003.]

Ajoitetuiksi elementeiksi (timed element) kutsutaan elementtejä, joilla on aikaleima. Aikaleimojen avulla voidaan johtaa lukuisia järjestyksiä ajoitetuille elementeille. Ajoitettujen elementtien kanssa voidaan käyttää erilaisia operaattoreita tai

sisäänrakennettuja funktioita kuten start, end, center ja duration. Taulukossa 3 esitellään NQL-kyselykielen ajalliset operaatiot ja niiden merkintätavat. [Voormann et al., 2003.]

Taulukko 3. NQL-kielessä käytetyt elementtien ajalliset suhteet [Voormann et al., 2003]

| Lyhennös | Operaatio | Määritelmä |
|---|---|---|
| % | overlaps.left | (start($a) <= start($b)) and (end($a) > start($b)) and (end($a) <= end($b) |
| [[ | left.aligned.with | start($a) = = start($b) |
| ]] | right.aligned.with | end($a) = = end($b) |
| @ | includes inclusion | (start($a) <= start($b)) and (end($a) = = end($b)) |
| [] | same.extent.as | (start($a) = = start($b) ) and (end($a) = = end($b)) |
| # | overlaps.eith | (end($a) > start($b)) and (end($b) > start($a)) |
| ][ | contact.with | end($a) = = start($b) |
| << | precedes | end($a) <= start($b) |
| | starts.earlier.than | start($a) <= start($b) |
| | starts.later.than | start($a) >= start($b) |
| | end.earlier.than | end($a) <= end($b) |
| | ends.later.than | end($a) >= end($b) |

NQL-kyselykieli sisältää myös eksistenssi- ja universaalikvanttorit, joiden avulla voidaan ilmaista monimutkaisia rakenteellisia suhteita. Yksinkertaistettuna universaalikvanttori eli kaikkikvanttori määrittelee, että joukon A kaikilla alkioilla on ominaisuus p. Vastaavasti eksistenssikvanttori eli olemassaolokvanttori määrittelee, että joukossa A on olemassa ainakin yksi sellainen alkio, jolla on ominaisuus p. [Merikoski et al., 2004.]

Kvanttorimuuttujat eivät näy kyselyn tuloksessa. Nämä kvanttorit sitovat muuttujat loogisiin lausekkeisiin ja samalla saadaan aikaan lauseke vähemmillä muuttujilla. Sekä eksistenssi- että universaalikvanttori voidaan liittää minkä tahansa muuttujan määrittelyyn. Eksistenssikvanttori kuvataan kyselyissä muodossa `exists(muuttuja, tyyppi)` ja vastaavasti universaalikvanttori muodossa

`forall(muuttuja, tyyppi)`. Muuttujia voidaan luetella useampia ja tyypeille voidaan antaa vaihtoehtoja. Tyyppimäärittely voidaan jättää myös kokonaan pois. Eksistentiaalikvanttorilla varustetun kyselyn suorittamisen jälkeen muuttujat poistetaan kyselyn tuloksesta ja kaksinkertaiset osumat yhdistetään yhdeksi osumaksi. [Evert and Voormann, 2003.]

Evert ja Voormann [2003] toteavat, että yhdistettäessä eksistenssi- ja universaalimäärittelijöitä, tulos riippuu ratkaisevasti nimenomaisesta muuttujien määrittelyjen järjestyksestä. He suosittelevatkin, että tällaisia kyselyjä tulisi muodostaa vain niiden käyttäjien, jotka hallitsevat propositiologiikan perusmerkinnät. Ei voida kuitenkaan olettaa, että kaikki kieltä käyttävät hallitsevat propositiologiikan. Kysely `($a) (exists $b) : $a ^1 $b` palauttaa elementit, joilla on lapsia ja kysely `($root) (forall $null) : !$null ^1 $root` palauttaa juurielementit. [Voormann et al., 2003, Evert and Voormann, 2003.] Eksistenssi- ja universaalikvanttoreiden avulla voidaan kuitenkin muodostaa ilmaisuvoimaisia kyselyjä.

NXT tukee kyselyiden sarjallistamista. NQL-kyselykielessä voidaan muodostaa moniosaisia kyselyjä, jotka koostuvat sarjasta tavallisia yksinkertaisia kyselyjä, jotka erotellaan toisistaan ::-symbolein. Moniosaisista kyselyistä suoritetaan ensin vasemmanpuolimmaisin kysely. Jokainen sarjan kysely suoritetaan edellisen kyselyn tuloksiin. [Voormann et al., 2003.] Moniosaisen kyselyn tulos on puurakenne. Kun kyselyitä on n kappaletta, on tulospuu n+1 tasoa syvä. Esimerkiksi moniosainen kysely `($wa word) : ($wa@agent = ”A”) :: ($wb word) : ($wb@agent = ”B”) && ($wa # $wb)` palauttaisi puun, joka osoittaa päällekkäisyydet kahden puhujan, A ja B, kesken. Kyselyn jälkimmäinen osa kohdistuu vain ensimmäisen osan palauttamaan tulosjoukkoon. [Carletta et al., 2005.]

Kyselyn tulokset ovat lista elementtien n-monikkoja, jotka täyttävät annetut ehdot. Tällöin n on ilman kvanttoreita määriteltyjen muuttujien lukumäärä. Kyselyn tulos palautetaan XML-muodossa, joka sisältää matchlist-elementin ja jokainen kyselyn osuma vastaa match-elementtiä, jonka osoittimet edustavat muuttujien siteitä ja osoittimen rooli antaa muuttujille nimen.

Esimerkki kyselystä, jonka muuttujat ovat $w ja $p.

```
<matchlist size= ″2″>
   <match n=″1″>
      <nite:pointer role=″w″ xlink:href=″…″/>
      <nite:pointer role=″p″ xlink:href=″…″/>
   </match>
<matchlist size= ″2″>
   <match n=″2″>
      <nite:pointer role=″w″ xlink:href=″…″/>
      <nite:pointer role=″p″ xlink:href=″…″/>
   </match>
</matchlist>
```

[Voormann et al., 2003.]

### 5.2.4 Puutteet ja tärkeimmät ominaisuudet

NQL-kysekielen syntaksi on selvästi yksi monimutkaisimmista tässä tutkimuksessa vertailluista kyselykielistä. NQL tarjoaa monipuolisimmat rakenteelliset relaatiot elementtien välillä. Lisäksi se on ainoa kieli, jossa voidaan hyödyntää ajallisia relaatioita esimerkiksi vertailemaan tiettyjen asioiden alku- ja loppuaikoja. NQL on myös ainoa vertailluista kielistä, jossa on käytössä universaali- ja eksistenssikvanttorit. Nämä kielen ominaisuudet mahdollistavat erittäin monipuolisten ja tarkkojen kyselyjen toteuttamisen. Ne vaativat kuitenkin sen, että käyttäjän tulee käyttää aikaa kielen syntaksin opiskeluun. Kielen rakenteet voivat olla vaikeita omaksua, jos esimerkiksi puurakenteet ja kvanttorit ovat ennestään vieraita. Tällaista kieltä ei välttämättä voi valita yksittäisen kielitieteellisen tutkimuksen työkaluksi, koska kielen opettelu vaatii aikaa.

## 5.3 IMS Corpus Workbench ja Corpus Query Processor Language

### 5.3.1 Tausta ja tavoitteet

Tukeakseen sanasto- ja terminologiatyötä, Stuttgartin yliopiston IMS-laitos (Institut für Maschinelle Sprachverarbeitung) kehitti järjestelmän suuriin tekstilähteisiin kohdistuvia kokotekstihakuja (full-text retrieval) varten. Työ aloitettiin Text Corpora and Tools for Their Exploitation -projektissa vuonna 1993. IMS Corpus Workbench (CWB) soveltuu monenlaisiin kielitieteen tarkoituksiin. Datasuuntautuneessa kielitieteessä sen avulla voidaan louhia kielitieteellistä tietämystä tekstilähteistä ja ristiintarkastaa kielitieteellisiä oletuksia suurten tekstikokoelmien pohjalta. Sanastotyössä CWB:n avulla voidaan löytää korpukseen perustuvia todisteita kielellisiin kuvauksiin. Terminologiassa sitä voidaan käyttää termien louhintaan ja ryhmittelyyn. [IMS, 2009.] Tällä hetkellä CWB:n kehittäminen jatkuu avoimen lähdekoodin ohjelmana ja se on saatavilla www-osoitteesta http://sourceforge.net/projects/cwb/. CWB tarjoaa työkaluja

korpusten koodaamiseen, indeksointiin, tiivistämiseen, purkamiseen ja frekvenssijakaumien muodostamiseen.

CWB:n kielitieteellisiin hakuihin tarkoitettu Corpus Query Processor (CQP) -hakukone on sellaisenaan ainoastaan komentorivityökalu. Se vaatii, että korpus on rekisteröity ja koodattu määrätyllä tavalla. IMS:ssä suurin korpus, jota on käsitelty Corpus Workbenchilla, on saksalainen sanomalehtikorpus, joka sisältää noin 200 miljoonaa sanaa tai välimerkkiä. Tämä korpus on merkitty lemmoilla, kahdella sanaluokkatunnistesarjalla ja lauserajoilla. [Christ, 1999.]

IMS:n Corpus Query Processor on todennäköisesti suosituin ja tunnetuin korpuskyselyjärjestelmä ja sitä hyödynnetään myös muissa korpushakujärjestelmissä. Esimerkiksi korpushakukone ja konkordoija Poliqarpin kyselysyntaksi perustuu Corpus Query Processoriin. Poliqarp on kehitetty morfosyntaktisesti merkittyä puolankielistä IPI PAN korpusta varten. IPI PAN korpus on kehitetty Linguistic Engineering -ryhmässä tietojenkäsittelytieteen laitoksella Polish Academy of Sciencesissä (ICS PAS). [IPI PAN, 2008] Myös esimerkiksi Oslon yliopistossa kehitetty bosniankielisiä tekstejä sisältävä Oslo Corpus on koodattu IMS Corpus Workbenchillä ja sen kanssa voidaan käyttää lähes kaikkea CQP-kyselykielen ominaisuuksista [Oslo Corpus, 2002].

### 5.3.2 Tietorakenne

Kuten kappaleessa 2.4. todettiin, Christ et al. [1999] jakavat korpukseen liitetyt merkinnät kolmeen tasoon, sijainnillisiin merkintöihin, rakenteellisiin merkintöihin sekä monitasoisiin merkintöihin. Sijainnillisiin merkintöihin kuuluvat yksittäisiin korpussijainteihin liitetyt merkinnät, joiden määrää ei ole rajattu. Näitä ovat esimerkiksi sanaluokkamerkinnät, sanojen lemmat sekä morfosyntaktinen informaatio. Rakenteelliseen informaatioon kuuluvat kappaleiden, lauseiden ja fraasien rajoihin liittyvät merkinnät. Monitasoiset merkinnät liittyvät korpuksen merkkaamiseen yhteneväisesti toisen korpuksen kanssa, jotta nämä korpukset on mahdollista yhdistää. Nämä merkinnät voidaan nähdä yleistyksenä rakenteellisesta informaatiosta. IMS Corpus Workbenchin tietorakenne perustuu näihin kolmeen tasoon.

Taulukko 4. IMS Corpus Workbenchin esimerkkikoodaus [Evert, 2005]

| **Korpussijainti** | **Sanan ulkoasu** | **ID** | **POS-merkintä** | **ID** | **Lemma** | **ID** |
|---|---|---|---|---|---|---|
| (0) | <text> value = ”id = 42 lang = ’English’” | | | | | |
| (0) | <text_id> value = ”42” | | | | | |
| (0) | <text_lang> value = ”English” | | | | | |
| (0) | <s> | | | | | |
| 0 | An | 0 | DET | 0 | a | 0 |
| 1 | easy | 1 | ADJ | 1 | easy | 1 |
| 2 | example | 2 | NN | 2 | example | 2 |
| 3 | . | 3 | PUN | 3 | . | 3 |
| (3) | </s> | | | | | |
| (4) | <s> | | | | | |
| 4 | Another | 4 | DET | 0 | another | 4 |
| 5 | very | 5 | ADV | 4 | very | 5 |
| 6 | easy | 1 | ADJ | 1 | easy | 1 |
| 7 | example | 2 | NN | 2 | example | 2 |
| 8 | . | 3 | PUN | 3 | . | 3 |
| (8) | </s> | | | | | |

CWB käyttää patentoitua tekstiyksiköihin (token) perustuvaa muotoa korpusten tallentamiseen. Tässä tekstiyksikkö tarkoittaa yhtä yksittäistä sanaa tai välimerkkiä. Taulukon 4 jokainen rivi vastaa yhtä tekstiyksikköä tekstissä. Rivien numerot yksilöivät jokaisen tekstiyksikköön ja viittaavat niiden sijaintiin korpuksessa. Tekstiyksikkötason merkinnät ovat sijainnillisia attribuutteja ja vastaavat saraketta taulukossa. [Evert, 2005.] Esimerkiksi British National Corpuksen BNCweb-ohjelmalla päästään käsiksi CQP-kyselyn kautta seuraaviin tekstiyksikkötason attribuutteihin, word, pos, hw, class, lemma ja type [Hoffmann, 2008]. Samannimiset elementit, jotka rajaavat alueen, kootaan rakenteellisiksi attribuuteiksi, joihin voidaan viitata. Nämä merkitään korpuksessa XML-elementtien avulla. Alueet eivät saa olla limittäisiä eivätkä rekursiivisia. XML-alkutunnisteisiin liitetyt attribuutti-arvo -parit tallennetaan vastaavien rakenteellisten attribuuttien merkinnöiksi. [Evert, 2005.]

### 5.3.3 Ominaisuudet

CQP-kyselykielen haut voidaan jakaa kohdistuviksi 1) yksittäisiin korpussijainteihin eli sijainnillisiin attribuutteihin, 2) korpussijaintien sarjoihin, 3) rakenteelliseen informaatioon ja 4) monitasoiseen informaatioon. CQP-kyselykielessä yksinkertaisin kysely muodostetaan kirjoittamalla haettava sana lainausmerkkeihin. Oletuksena tässä lyhennetyssä muodossa, eli lainausmerkkien sisässä, olevaa sanaa haetaan word-muuttujista. Haku `"sana"` on lyhenne hausta `[word = "sana"]`. DefaultNonbrackAttr-määreellä on kuitenkin mahdollista asettaa lainausmerkkilyhenne toimivaksi jonkin toisen muuttujan kanssa. Sanan merkitsemisessä voidaan käyttää säännöllisiä lausekkeita esimerkiksi etsittäessä sanan taivutusmuotoja tai erilaisia kirjoitusasuja. Käytössä on myös disjunktio-operaatio sekä toisto-operaattorit. Erikoismerkit pitää suojata kenoviivalla (\). Merkinnällä %c voidaan jättää haussa huomiotta isot ja pienet kirjaimet ja merkinnällä %d jätetään huomioimatta diakriittimerkit kuten aksenttimerkki. [Evert, 2005, Christ et al., 1999.]

Sijainnillisiin attribuutteihin päästään käsiksi asettamalla halutut attribuutti-arvo -parit hakasulkeiden sisään, esimerkiksi muodossa `[pos = "NN"]`. Arvojen vertailussa voidaan käyttää yhtäsuuruus- ja erisuuruusoperaattoreita. Tyhjät hakasulkeet täsmäävät kaikkiin tekstiyksiköihin. Koska attribuuteille asetetut arvot tulkitaan kyselyissä säännöllisiksi lausekkeiksi, voidaan merkinnällä %l asettaa arvo tulkittavaksi kirjaimellisesti, esimerkiksi `[word= "?" %l]`. Attribuuttirajauksia voidaan yhdistellä. Rajoituksia asetettaessa voidaan käyttää Boolen lausekkeita ja säännöllisiä lausekkeita. Operaattoreista käytössä ovat konjunktio (&), disjunktio (|), negaatio (!) ja implikaatio (→). [Evert, 2005.]

Muuttujiin voidaan tallentaa sanalistoja ja käyttää näitä ehtojen arvoina. Esimerkiksi `define $colors = "sininen punainen keltainen valkoinen"` asettaa $colors-nimiseen muuttujaan listan sanoja, joihin voidaan viitata

kyselyssä esimerkiksi muodossa `[lemma = $colors]`. Listamuuttujiin voidaan lisätä ja poistaa sanoja += ja -= -operaattoreilla. [Evert, 2005.]

Korpussijaintien sarjoihin päästään käsiksi Christin ja muiden [1999] kutsumilla sarjamalleilla (sequence patterns). Näissä malleissa yhdistetään edellä mainitut attribuutti-ilmaukset säännöllisiin lausekkeisiin ja Boolen lausekkeisiin. Yksinkertaisimmillaan mallissa luetellaan halutut sanat lainausmerkeissä. Esimerkiksi `"Suomi" "laki"` palauttaisi kaikki sanaparit, joissa esiintyy sanat Suomi ja laki peräkkäin. Kyselyssä asetettujen mallien eli tekstiyksikkötason ehtojen sarjaa sovitetaan vastaavaan sarjaan korpuksessa. Lainausmerkein tai hakasulkein rajattu alue vastaa aina yhtä tekstiyksikköä korpuksessa.

Sarjoja muodostettaessa käytössä on toisto-operaattorit ?, * ja +. Kyselyssä {n}-määreellä asetetaan toistojen lukumäärä ja vastaavasti {n,m}-määreen avulla asetetaan toistojen määrä tietylle välille. Käytössä on myös disjunktio-operaattori (|). Toisto-operaattoreiden ja disjunktion vaikutusalueet ja suoritusjärjestys voidaan tarvittaessa määrätä kaarisulkein. Kysely `"vasen" []? "oikea"` palauttaa kaikki sellaiset sarjat korpuksesta, joissa sanojen vasen ja oikea välissä on mikä tahansa yksi sana tai ei yhtään sanaa. [Evert, 2005.]

Rakenteellinen informaatio voidaan jakaa esimääriteltyihin rakenteisiin, eli rakenteellisiin merkintöihin, sekä ad hoc -rakenteisiin eli aikaisempien kyselyjen tuloksiin [Christ et al., 1999]. Rakenteelliset attribuutit eli XML-tagit voidaan suoraan kirjoittaa CQP-kielen kyselylausekkeisiin. Kysely `<s> [pos="ADJ"]` palauttaa lauseiden alussa olevat adjektiivit. Kysely `<np> []* ([pos="JJ.*"] []*) {3,} </np>` palauttaa kaikki np-elementit, jotka sisältävät vähintään kolme adverbiä. Kyselyissä voidaan käyttää tarvittaessa useita XML-tageja. CQP-kielen makro `/region[]` vastaa kokonaista aluetta. Komennolla within voidaan rajata kysely koskemaan vain tietyn elementin sisältöä. Esimerkiksi kysely `"helppo tehtävä" within s` rajaa tulokseen mukaan vain saman lauseen (s) sisällä esiintyvät ”helppo tehtävä” -esiintymät. Tämä rajoitus on usein tarpeen, koska CQP ei automaattisesti rajaa peräkkäisiä sanoja vain saman lauseen sisällä esiintyviin. [Evert, 2005.]

XML-elementtien attribuutteihin päästään käsiksi tagin nimi_attribuutin nimi -yhdistelmällä. Kysely `<np> a:[] []* </np> :: a.np_h = "bank"` palauttaa np-elementit, joilla on päälemma-attribuutin h arvona bank. Vastaavasti rajoitus voidaan liittää kyselyssä myös alkutunnisteen yhteyteen muodossa `<np_h = "bank"> []* </np_h>`. [Evert, 2005.]

Haku /region[s] vastaa hakua <s> []* </s> eli palauttaa lauseet millä tahansa sisällöllä. Rakenteellisen attribuutin nimen käyttö kyselyn mallissa saa arvon tosi, jos ja vain jos vastaava tekstiyksikkö kuuluu kyseisen attribuutin alueelle. Esimerkiksi kysely `[(pos = "N?") & !np]` palauttaa substantiivit (N), jotka eivät sisälly

substantiivifraasiin (np). Sisäänrakennetuilla funktioilla lbound ja rbound voidaan määritellä alueen alku ja loppu. Esimerkiksi kysely `[(pos=”NN”) & lbound(s)]` palauttaa kyselyt, joissa substantiivi on lauseen (s) alussa. Kysely `[(pos =”N”) & lbound(s)] []* [(pos = ”N” & rbound(s)]` palauttaa lauseet (s), jotka sekä alkavat että loppuvat substantiivilla. Kyselyn osumat voidaan laajentaa kattamaan koko alue eli esimerkiksi kokonainen lause. Laajennos voidaan haluttaessa rajata vain osuman oikealle tai vasemmalle puolen right ja left -avainsanojen avulla. Esimerkiksi aiemmin nimetty haku A voidaan laajentaa koskemaan koko lausetta seuraavasti, `B = A expand to s` tai vain lauseen loppuun `B = A expand to right s`. [Evert, 2005.]

CQP-kyselykieli tarjoaa operaattoreita myös kyselyn tuloksen lajitteluun ja laskentaan. Kyselyn perään liitetty määre `sort by word` järjestää tuloksen aakkosjärjestykseen sanan mukaan. Järjestäminen voidaan tehdä minkä tahansa sijainnillisen attribuutin mukaan. Sort-komentoon voidaan liittää komento descending, jolloin järjestys tehdään laskevasti, sekä reverse, jolloin osumat järjestetään loppuliitteiden perustella. Count-komennolla saadaan tulokseen mukaan frekvenssijakauma. Esimerkiksi `count by lemma` laskee frekvenssin lemman perusteella. Cut-määreellä voidaan frekvensseille asettaa raja. Esimerkiksi `count by word cut 10`, palauttaa vain ne sanat, joiden frekvenssi on yli 10. Kun tulosjoukko on suuri, voidaan valita sattumanvarainen osajoukko nopeaa tarkastelua varten. Reduce-komento pienentää tuloksen haluttuun osaan alkuperäisestä. Esimerkiksi `reduce A to 15%` pienentää A:n tulosjoukon 15 %:iin. [Evert, 2005.]

Kyselyn tuloksia voidaan lajittelun ja järjestämisen lisäksi käsitellä monilla tavoin. Kyselyn tulokset voidaan tallentaa antamalla kyselylle nimi ja asettamalla sen arvoksi haluttu kysely. Esimerkiksi `Laki = [lemma = ”laki”] []` tallentaa nimellä Laki kyselyn, joka palauttaa kaikki sanaparit, joista ensimmäisen sanan lemma on laki. Nimetyille kyselyille voidaan suorittaa joukko-opin leikkaus-, yhdiste- ja erotusoperaatioita. Esimerkiksi muuttujaan C voidaan tallentaa A:n ja B:n leikkaus C = intersection A B. [Evert, 2005.]

Kyselyn mallit voidaan merkitä nimiöin. Nimiöiden avulla voidaan muodostaa monimutkaisia kyselyjä ja pitkän välimatkan riippuvuuksia. Nimiöiden avulla saadaan aikaan rajoituksia, joita säännöllisillä lausekkeilla ei voida muodostaa. Esimerkiksi `adj:[pos = ”JJ.*”]` määrää nimiön adj viittaamaan määrättyyn tekstiyksikköön. Nimiöihin liitetään yleensä globaaleja rajoituksia, jotka esitellään symbolilla ::. Esimerkiksi `adj:[pos= ”ADJ.”] :: adj < 500` rajoittaa adjektiivit ensimmäiseen 500:an. Nimiöidyn elementin merkintöihin päästään pistenotaatiolla, esimerkiksi adj.word ja adj.lemma. Esimerkiksi kysely `a: [pos=”PP” [] {0,5} b:[pos = ”VB.*”] :: b.pos = ”VBZ” → a.lemma = ”he|she|it”`

varmistaa, että pronomini, joka edeltää 3-persoonan yksikkönä verbiä, on he, she tai it. [Evert, 2005.]

Käyttäjä voi asettaa hakustrategian (matching strategy), joka vaikuttaa siihen, miten kyselyssä tulkitaan toisto-operaattorit. Vaihtoehdot ovat shortest, standard ja longest. Shortest-tilanteessa ?, * ja + -operaattorit täsmäävät pienimpään mahdolliseen määrään tekstiyksiköitä. Tämä tarkoittaa sitä, että kysely löytää lyhimmän kyselyä vastaavan tuloksen. Valinnaiset elementit kyselyn alussa ja lopussa eivät tule mukaan tulokseen. Longest-tilanteessa kyseiset operaattorit täsmäävät niin moneen tekstiyksikköön kuin mahdollista. Oletuksena olevassa standard-tilanteessa valinnaiset elementit kyselyn alussa otetaan mukaan, kun taas lopussa olevia valinnaisia elementtejä ei oteta. [Evert, 2005.]

Mainittujen ominaisuuksien lisäksi CQP-kyselykielessä on ominaisuuksia, joita kaikkia ei ole yllä käsitelty. Esimerkiksi kyselyjä voidaan tallentaa makroiksi. Viimeisimmän kyselyn tuloksen saa esiin komennolla Last. Tulostietueiden lukumäärä voidaan tulostaa komennolla `size kyselynNimi`. Lisäksi kieli sisältää esimerkiksi kyselyjen turvallisuuteen ja esittämismuotoon liittyviä ominaisuuksia. Kyselyihin voidaan liittää Perl-skriptejä ja tätä varten on olemassa CWB/Perl-käyttöliittymä. Koska monissa sovelluksissa tarvitaan erilaisia frekvenssitauluja, tarjoaa CQP näiden tulostamiseen ja käsittelyyn erilaisia komentoja. CQP sisältää myös säännöllisten lausekkeiden optimoijan, joka havaitsee yksinkertaiset ilmaukset, joita on käytetty etu-, jälki- ja sisäliitteinä ja korvaa säännöllisten lausekkeiden regexp-moottorin tehokkaammalla Boyer-Moore -hakualgoritmilla. [Evert, 2005.]

### 5.3.4 Puutteet ja tärkeimmät ominaisuudet

CQP:n kyselykieli on selvästi yksi monipuolisimmista korpuskyselykielistä. Se tarjoaa hakulausekkeiden käsittelyyn hyvin monipuoliset operaattorit. Kyselykieli ei sisällä mahdollisuuksia liikkua XML-rakenteessa solmujen välillä. CQP onkin selvästi korpuskyselykieli, jonka ominaisuuksia tarkasteltaessa esiin nousevat kielitieteellisiin tarkoituksiin soveltuvat ominaisuudet. Ominaisuudet mahdollistavat hyvin monenlaiset kielitieteelliset kyselyt, mutta CQP-kielen vahvuutena ei ole erilaisten rakenteellisten suhteiden kuvaaminen. Varsinaisten kyselyyn liittyvien ominaisuuksien lisäksi CQP sisältää monenlaisia komentoja, jotka auttavat esimerkiksi tallentamaan ja käsittelemään kyselyjä. CQP-kielen mahdollisuus näyttää frekvenssit ja rajata frekvenssin mukaan palvelevat erityisesti termien louhinnassa.

## 5.4 BNCweb ja Simple Query Syntax

### 5.4.1 Tausta ja tavoitteet

BNCweb on www-pohjainen käyttöliittymä kielitieteellisten hakujen tekemiseen. Alkuperäinen BNCweb kehitettiin Zürichin yliopistossa vain yliopiston

käyttötarpeisiin. Vuonna 2002 siitä julkaistiin ensimmäinen julkinen versio. Simple Query Syntax on BNCweb-ohjelman helppokäyttöinen kyselykieli, jolla voidaan tehdä hakuja British National Corpukseen. Kyselykieli on suunniteltu niin, että se on helppo oppia. Syntaksin on haluttu olevan selkeä ja samaan aikaan riittävän tehokas yleisimpiin BNCwebin käyttäjien tarpeisiin. BNCwebin käyttöliittymä on monipuolinen ja se antaa käyttäjälle mahdollisuuden muun muassa rajata kysely kohdistumaan tiettyyn korpuksen osaan, järjestää tuloksia, tarkastella kyselyhistoriaa ja tarkastella kyselyjä eri tavoilla. [BNCweb, 2010.]

Koska BNCweb on kehitetty erityisesti British National Corpusta ajatellen, sitä ei voi helposti soveltaa käytettäväksi muiden korpusten kanssa. Alkuperäisen BNCwebin toiminnallisuus perustui SARA-ohjelmistoon, kun taas uusin versio perustuu IMS Corpus Workbenchin Corpus Query Processoriin. Nykyisen version käyttöliittymästä ovat luoneet Stefan Evert Osnabrückin yliopistosta ja Sebastian Hoffmann Lancasterin yliopistosta. [BNCweb, 2010.]

### 5.4.2 Tietorakenne

BNCweb-sovellusohjelma kääntää käyttäjän tekemät kyselyt Corpus Query Processorin käyttämälle kielelle. Taustalla oleva tietorakenne on siis sama kuin Corpus Query Processorissa. Simple Queryssa Syntaxissa käytetyt rakenteet on koodattu sovellusspesifisti.

### 5.4.3 Ominaisuudet

Yksittäisiä sanoja ja fraaseja voidaan Simple Query Syntaxissa hakea yksinkertaisesti kirjoittamalla sana tai fraasi hakulausekkeeksi. Fraasin sanat erotetaan toisistaan välilyönnillä. Erikoismerkit tulee suojata kenoviivalla. Erikoismerkkejä ovat ?, *, +, ,, :, /, (, ), [, ], {, }, _, ~, < ja >. Kyselyt ovat oletuksena merkkikokoriippumattomia (case-insensitive). Määre :c sanan perässä muuttaa kyselyn merkkikokoriippuvaiseksi (case-sensitive). Esimerkiksi kysely `Bath:c` palauttaa vain isolla kirjaimella alkavat Bath-sanat eli kaupungit nimeltä Bath. Määre :d kyselyssä jättää huomiotta diakriittimerkit. [Hoffmann et al., 2008.]

Kyselyihin voidaan asettaa sanaluokkaehtoja, jotka liitetään sanaan _-operaattorilla. Sanaluokista on käytössä tarkat lyhenteet sekä karkeammalla tasolla olevat lyhenteet. Karkeamman tason lyhenteet kuten A (adjektiivi), N (substanstiivi) ja PREP (prepositio) merkitään {}-sulkeiden sisään, kun taas tarkemmat lyhenteet, kuten AJS, NN1, PNP ja VVD, liitetään kyselyyn sellaisenaan. Myös sanaluokkamerkinnöissä voidaan käyttää korvausmerkkejä ja vaihtoehtoja. Esimerkiksi `_[AJC,AJS]` tarkoittaa samaa kuin `_AJ[C,S]`. Vaihtoehdot merkitään hakasulkeissa pilkulla eroteltuina. Sanaluokkamäärettä ei tarvitse liittää sanaan vaan sitä voidaan käyttää myös yksinään. Kysely `_N*` palauttaisi kaikki substantiivit ja vastaavasti `_V*` kaikki verbit. [Hoffmann

et al., 2008.] *-merkki toimii katkaisumerkkinä ja sen avulla kyselyyn saadaan mukaan esimerkiksi kaikki N-kirjaimella alkavat merkinnät.

BNCweb mahdollistaa myös lemmoihin (headwords) kohdistuvat haut. Lemma kirjoitetaan aaltosulkeiden sisään. Kysely `{show}` palauttaisi kaikki show-sanan esiintymismuodot eli show, shows, showed, shown ja showing. Myös lemmoihin voidaan liittää kyselyssä sanaluokkamääreet. Kysely `{show}_{V}` palauttaisi vain verbeiksi tulkitut show-sanat. Sama kysely voidaan esittää muodossa `{show/V}`. Kirjainten koon ja diakriittimerkit huomiotta jättävät määreet :d ja :c tulee liittää kyselyyn ennen sanaluokkamerkintöjä, esimerkiksi `{sautee*}:d_V*`. [Hoffmann et al., 2008.]

Simple Query Syntaxissa voidaan käyttää jokerimerkkejä (wild cards) eli tehdä sanankatkaisuja, joiden avulla voidaan hakea esimerkiksi sanoja, joiden taivutuspäätettä ei haluta määrätä. Korvausmerkkeinä ovat käytössä ?, * ja +. Kieli sisältää myös lyhenteitä, joita voidaan käyttää korvaamaan erilaisia asioita. Näitä lyhenteitä ovat:

- \a vastaa yksittäistä kirjainta, mutta ei välimerkkejä tai muita erikoismerkkejä
- \A vastaa yhtä tai useampaa kirjainta
- \l, \L pieniä kirjaimia (kun on asetettu kirjainkokoriippuvaisuus määreellä :C)
- \u, \U isoja kirjaimia (kun on asetettu kirjainkokoriippuvaisuus määreellä :C)
- \d, \D numeroita
- \w, \W sanamerkkejä

Kysely `s?ng` palauttaa esimerkiksi sanat sing, sang, song ja sung. Kysely `\a-team` palauttaa esimerkiksi A-team ja c-team -sanat. Kysely `post\W` voisi palauttaa esimerkiksi sanat postcard tai postgraduate. Korvausmerkit voidaan asettaa sanan alkuun, keskelle tai loppuun. Korvausmerkeillä ei voida hakea yli sanarajojen. Esimerkiksi kysely `black*white` palauttaa osumanaan black-and-whiten, mutta ei muotoa black and white. [Hoffmann et al., 2008.]

Peräkkäisiä sanoja haetaan asettamalla yksittäisten sanojen rajausehdot kyselyssä peräkkäin. Kysely `_{PREP} _{ART} _{A} _{N}` voisi palauttaa esimerkiksi tekstin ’in the cosy room’. Peräkkäisten sanojen kuvaamisessa voidaan käyttää säännöllisiä lausekkeita. Käytössä ovat toisto-operaattorit sekä Boolen operaattoreista tai-operaattori. Sulkeiden avulla voidaan rajata ja ryhmitellä ehtoja. Lisäksi voidaan käyttää toistojen määrää rajaavia {n,m} –merkintöjä. Säännöllisiä lausekkeita voidaan asettaa sisäkkäin, näin voidaan muodostaa hyvin monimutkaisiakin kyselyrakenteita. [Hoffmann et al., 2008.]

Kyselyissä ei ole käytössä erityistä tapaa kuvata sanaa tai tekstiyksikköä, jolle ei haluta antaa ehtoja. Merkki + symboloi mitä tahansa tekstiyksikköä ja merkki *

symboloi mitä tahansa tekstiyksikköä, joka voi esiintyä tai olla esiintymättä. Esimerkiksi kysely `(black*white | black * white)` palauttaisi sekä tekstin ’black-and-white’ että tekstin ’black and white’. [Hoffmann et al., 2008.]

Simple Query Syntax sallii myös XML-tagien käytön kyselymalleissa. XML-tageja voidaan käyttää missä tahansa kyselyn kohdassa. Esimerkiksi kysely `<s> but` palauttaa lauseet, jotka alkavat but-sanalla. Jos halutaan etsiä tiettyjä XML-elementtejä riippumatta niiden sisällöstä, voidaan kysely esittää muodossa `<quote> (+)+ </quote>`, jossa quote on halutun elementin nimi. Hyödyllisiä XML-elementtejä ovat esimerkiksi lause (s), kappale (paragraph), otsikko (head), lainaus (quote) ja listaelementti (item) sekä monisanayksikkö (mw). [Hoffmann et al., 2008.]

BNCweb käyttää kyselyissä lyhimmän osuman strategiaa (shortest match strategy), mikä tarkoittaa sitä, että kun ensimmäinen malliin sopiva osuma on löydetty, ei tätä pidempiä enää etsitä. Esimerkiksi kun kysely on `(_{ART})? (_{A})* (_{N})+` ja, jos substantiivifraasi käsittää kaksi tai useampia substantiiveja, kuten ’a warm autumn evening’, vain ensimmäinen näistä otetaan mukaan kyselyn osumaan. [Hoffmann et al., 2008.]

Simple Query Syntax sallii myös kyselyt, joissa halutaan hakea sanoja, jotka esiintyvät toistensa läheisyydessä. Näitä kutsutaan lähekkäisyyskyselyiksi (proximity queries). Läheisyysoperaattoria käytetään muodossa `<<konteksti>>`, jossa konteksti viittaa tekstin tasoon. Näitä voivat olla esimerkiksi lause (s-unit), kappale (paragraph) ja dokumentti (text). Esimerkiksi kysely `kukka <<s>> aurinko` palauttaisi tulokset, joissa sanan kukka kanssa samassa lauseessa esiintyy sana aurinko. Kontekstiin voidaan merkitä myös sanojen välimatka, esimerkiksi `pata <<3>> kattila` -haku etsisi kaikki sellaiset esiintymät, joissa sana kattila esiintyy korkeintaan kolme tekstiyksikköä oikealle tai vasemmalle sanasta pata. Tällöin haku ottaa huomioon sanat myös lauserajojen yli. Haku voidaan rajata vain toiseen suuntaan, esimerkiksi `<<5<<` hakisi kyseisestä sanasta korkeintaan viisi tekstiyksikköä vasemmalle ja vastaavasti `>>3>>` hakisi korkeintaan kolme tekstiyksikköä oikealle. [Hoffmann et al., 2008.]

Sisäkkäiset alikyselyt antavat käyttäjälle mahdollisuuden kirjoittaa monimutkaisia lähekkäisyyskyselyjä. Kyselyissä ei ole kuitenkaan mahdollista yhdistää lähekkäisyyskyselyjä fraasikyselysyntaksiin. Valinnaisia tekstiyksiköitä, toisto-operaattoreita tai XML-tageja ei ole mahdollista yhdistää lähekkäisyyskyselyjen alikyselyrajoituksiin. Esimerkiksi kysely `{kick/V} <<s>> (the (_{A})* bucket)` aiheuttaa syntaksivirheen. Myöskään läheisyysrajoituksia ei voi liittää normaaliin fraasikyselyyn. [Hoffmann et al., 2008]

### 5.4.4 Puutteet ja tärkeimmät ominaisuudet

Simple Query Syntax on nimensä ja tavoitteensa mukaisesti yksinkertainen ja helppokäyttöinen korpuskyselykieli. Käytössä olevat sanaluokkamerkintöjen lyhenteet helpottavat kielen käyttöä ja yksityiskohtaiset sanaluokkatunnisteet mahdollistavat tarkat sanaluokkiin kohdistuvat haut. Kieli sallii sekä sanojen esiintymismuotojen, lemmojen että sanaluokkainformaation hyödyntämisen kyselyissä. Säännöllisten lausekkeiden avulla sanojen ehdoista on mahdollista tehdä hyvin yksityiskohtaisia. Boolen operaattoreista käytössä ei ole konjunktio-operaattoria eikä negaatio-operaattoria. Kyselykielen syntaksi muodostetaan kuitenkin niin, ettei konjunktio-operaattorille oikeastaan ole tarvetta. Myös yksityiskohtaiset sanaluokkatunnisteet vähentävät konjunktio-operaattorin käyttötarvetta. Sen sijaan negaatio-operaattori olisi hyödyllinen, koska sen avulla voitaisiin rajata tuloksista pois haluttuja asioita. CQP-kielen tavoin myös Simple Query Syntax selviytyy monista kielitieteellisistä kyselyistä ja se mahdollistaa hyvinkin tarkat rajaukset sanojen sisältöihin.

Tavoitteeltaan sama kysely voidaan Simple Query Syntaxissa toteuttaa monilla tavoin. Esimerkiksi käyttäjä jos haluaa hakea kaikki can- ja might-sanojen esiintymät voi kyselyn tehdä muodossa `[can,might]_{N}` tai muodossa `(can_{N} | might_{N})`. Tämä voi olla käyttäjän kannalta joko kyselykielen käyttöä helpottavaa tai saada kyselykielen vaikuttamaan monimutkaiselta erilaisine vaihtoehtoineen.

Kyselykieleen kuuluvat lähekkäisyysoperaatiot antavat käyttäjille mahdollisuuden tehokkaisiin kyselyihin, joilla voidaan rajata vain tietyn elementin sisällä olevat tai tietyn välimatkan päässä toisistaan olevat sanat. Lyhyimmän osuman strategia vaikuttaa tuloksiin melko paljon ja olisikin hyvä, jos käyttäjällä olisi mahdollisuus valita hakustrategia.

# 6 Korpuskyselykielten vertailu

Kaikki korpuskyselyjärjestelmät eivät tarjoa käyttäjille mahdollisuutta käyttää hakujen kohteena itse annettua tekstiä tai tekstikokoelmaa. Esimerkiksi BNCwebin kyselyt kohdistuvat aina British National Corpukseen. Tässä tutkimuksessa ei siis voida yksiselitteisesti vertailla kielten palauttamia tuloksia tai kielten ja järjestelmien tehokkuutta ja nopeutta. Tässä luvussa vertaillaan kyselykieliä kymmenen esimerkkikyselyn pohjalta ja pyritään selvittämään, millaisiin tarpeisiin kielet vastaavat ja kuinka helppoa kyselyiden muodostaminen on. Esimerkkikyselyitä muodostettaessa oletetaan, että korpukseen on merkitty sanaluokat, sanojen perusmuodot, sanojen esiintymismuodot sekä lauseet. Jos kielten lauseet on mahdollista muodostaa useammalla tavalla, on esimerkeissä pyritty valitsemaan lyhyin ja yksinkertaisin vaihtoehto.

Tämän työn ulkopuolelle jätetään erilaisten hakujärjestelmien ominaisuudet ja keskitytään korpuskyselykieliin ja niiden komentoihin ja ominaisuuksiin. Tutkimuksen ulkopuolelle jäävät esimerkiksi Järvelinin [1995] mainitsemat hakujärjestelmien ominaisuudet kuten hakuhistorian esittäminen, käyttäjän opastaminen ja hakulausekkeiden tallennus. Vaikka nämäkin ominaisuudet voivat joissain järjestelmissä sisältyä korpuskyselykieleen, keskitytään tässä työssä hakujen rajaamiseen ja tulosten valintaan liittyviin kielten ominaisuuksiin.

## 6.1 Vertailussa käytetyt esimerkkikyselyt

Seuraavaksi vertaillaan olemassa olevia korpuskyselykieliä sekä tutkimuksessa toteutettua korpuskyselykieltä. Kielten vertailussa on pyritty kohdistamaan huomio kielten mahdollisuuksiin täyttää erilaisia tarpeita, joita kielitieteen tutkijoilla voi olla. Vertailu tapahtuu muodostamalla kymmenen esimerkkilausetta, joilla pyritään havainnollistamaan monipuolisesti erilaisia kielentutkimukseen liittyviä tilanteita. Tämän jälkeen selvitetään, voidaanko nämä esimerkkilauseet muodostaa kullakin kyselykielellä.

Esimerkkikyselyt edustavat erilaisia tilanteita ja tarpeita, joihin sekä korpusten että niihin liitettävien korpuskyselykielten tulisi vastata. Esimerkkikyselyt on jaettu viiteen ryhmään. Kaksi ensimmäistä esimerkkikyselyä edustavat kyselyjä, joissa on kyse mallin sovittamisesta ja säännöllisten lausekkeiden hyödyntämisestä korpushauissa. Seuraavat kaksi esimerkkikyselyä havainnollistavat termien louhintaa korpuskyselykielten avulla. Tilastotietojen yhdistämistä korpushakuihin havainnollistetaan myös kahdella kyselyllä. Kaksi seuraavaa esimerkkikyselyä havainnollistavat mahdollisuuksia vaikuttaa kyselyn tulostuksen muotoon. Tilastotietojen hyödyntäminen ja tulostuksen muotoon vaikuttaminen voivat vähentää tulosten jatkokäsittelytarvetta ja parantavat tulosten käytettävyyttä. Viimeiset kaksi

kyselyä pyrkivät osoittamaan, kuinka hyvin valituilla korpuskyselykielillä voidaan hyödyntää tekstin rakennetta ja esimerkiksi etäisyyksiä haettavien elementtien välillä.

Ensimmäiset kaksi esimerkkikyselyä havainnollistavat mallin sovittamista ja säännöllisten lausekkeiden hyödyntämistä. Näistä ominaisuuksista on hyötyä silloin, kun hakusanaa ei pystytä antamaan yksiselitteisesti. Sanojen katkaisua voidaan pitää olennaisena ja tarpeellisena piirteenä korpuskyselykielissä erityisesti silloin, kun haut halutaan kohdistaa sanojen esiintymismuotoihin, jotka voivat olla taivutettuja. Tämä on tärkeää erityisesti agglunatiivisissa kielissä eli kielissä, joissa sanan vartaloon liittyy useita päätteitä ja muita affikseja. Jos taas korpus ja kyselykieli mahdollistavat hakujen kohdistamisen sanojen perusmuotoihin, ei sanojen katkaisulle ole niin suurta tarvetta.

1. Sanat, jotka alkavat kirjaimilla ’sovel’
2. Sanaparit, joissa jälkimmäinen sana on joko ’color’ tai ’colour’

Seuraavat kaksi esimerkkikyselyä pyrkivät edustamaan tilanteita, joissa kyselykieltä käytetään termien louhintaan korpustekstistä. Tällöin on olennaista voida määritellä palautettavien sanojen sanaluokat sekä saada mahdollisuus rajata pois jokin tai joitakin sanoja, joista ei olla tässä tapauksessa kiinnostuneita.

3. Sanaparit, joissa ensimmäinen sana on substantiivi, ja sen lemma ei ole ’Suomi’ ja toisen sanan lemma on ‘laki’
4. Substantiivi – substantiivi -sanaparit tai substantiivi – prepositio – substantiivi –sanayhdistelmät

Seuraavat kaksi esimerkkikyselyä pyrkivät havainnollistamaan sen, miten kyselykielet mahdollistavat tilastotietojen hyödyntämisen. Erilaisten kielitieteessä hyödynnettävien kertoimien laskeminen ja esittäminen tulosten yhteydessä tuo käyttäjille lisäarvoa. Tällöin heidän ei tarvitse jatkokäsitellä tuloksia saadakseen käyttöönsä sanojen frekvenssejä tai muita tilastollisia lukuja. Kyselykieli palauttaa tällöin käyttäjälle valmiin tutkittavan tuloksen, josta voidaan sellaisenaan tehdä erilaisia johtopäätöksiä.

5. Kaikki adjektiivi – substantiivi -sanaparit ja sanojen frekvenssit tulosjoukossa
6. Kaikki sanaparit, joissa ensimmäinen sana on adjektiivi, ja toinen sana on ’laki’ sekä parien prosentuaaliset frekvenssit tulosjoukosta tai jokin sanojen sidoksisuuteen liittyvä kerroin

Seuraavilla kahdella esimerkkikyselyllä havainnollistetaan kyselykielten mahdollisuuksia vaikuttaa muotoon, jossa kyselyn tulos palautetaan. Kun haun

kohteena on laaja tekstikokoelma tai haku on muuten sellainen, että se palauttaa suuren joukon tuloksia, tulosten rajaaminen tai järjestäminen esimerkiksi frekvenssin mukaan vähentää käyttäjän työtä huomattavasti.

7. Kolme peräkkäistä sanaa, joista keskimmäinen on ’she’, ’he’ tai ’it’, tulos esiintymismuodossa

8. Sanaparit, joissa ensimmäinen sana on mikä tahansa, ja toinen sana on ’säädös’, tulokseen mukaan vain ne sanaparit, joiden frekvenssi tekstissä on yli kymmenen

Viimeiset kaksi esimerkkikyselyä liittyvät korpuksen rakenteeseen ja elementtien etäisyyteen. Sanojen etäisyyteen kohdistuvat haut ovat tärkeitä esimerkiksi silloin, kun tutkitaan kahden tai useamman sanan esiintymistä yhdessä, mutta ei voida olettaa, että sanat esiintyisivät peräkkäin.

9. Lause, jossa on sanat ’lain’ ja ’soveltaminen’ enintään kolmen sanan päässä toisistaan

10. Lauseet, jotka sisältävät tekstin ’eduskunnan päätöksen mukaisesti säädetään’

## 6.2 Korpuskyselykielten vertailu esimerkkikyselyiden avulla

Tässä luvussa esitellään kaikki kymmenen esimerkkikyselyä jokaisella vertailtavalla korpuskyselykielellä. Jos kielellä ei pysty muodostamaan täydellisesti esimerkkikyselyä, mutta kuitenkin osan siitä, esitetään vertailtavan kielen kyselylauseke kursiivilla.

**Kysely 1:** Sanat, jotka alkavat kirjaimilla ’sovel’, haku kohdistuu sanojen esiintymismuotoihin

- XTerm:
- MQL: [Word surface = ~’[Ss]ovel.*’]
- Simple Query Syntax: sovel+
- CQP: ”sovel.+” %c;
- NQL: ($a word): TEXT($a)=”~/sovel.*/”

XTerm-kyselykieli ei mahdollista sanojen katkaisua kun taas kaikki muut kielet antavat mahdollisuuden sanojen katkaisuun kyselylauseissaan. MQL-kyselykielen kyselylauseiden tulee aina alkaa tekstillä ’Select all objects where’, jonka jälkeen kirjoitetaan itse kysely kuten tässä `[Word surface = ~’sovel’?]`. Lisäksi tulee

määritellä sekä elementti (word) että sen attribuutti (surface), joille ehto annetaan. Kokonaisuudessa MQL-kielen kyselylause olisi:

```
SELECT ALL OBJECTS
WHERE
[Verse GET verse
   [Word surface = ~'[Ss]ovel.*']
]
GO
```

Esimerkkilauseessa esitetään kuitenkin vain haun kannalta olennaisin osa, jossa asetetaan ehdot haulle.

Simple Query Syntaxin ja CQP-kyselykielen kyselyt ovat merkittävästi MQL- ja NQL-kyselykielten kyselyitä yksinkertaisempia muodostaa. Simple Query Syntaxissa ja CQP-kielessä hakujen ehdot kohdistetaan sanojen esiintymismuotoihin, jos muuta ei ole määrätty. CQP-kielessä isot ja pienet kirjaimet jätetään huomiotta määreellä %c. Lisäksi kaikki CQP-kyselylauseet tulee päättää puolipisteeseen. NQL-kielen kyselyssä määritellään ensin muuttuja, joka sidotaan tietty elementtiin, esimerkissä muuttuja $a sidotaan elementtiin word. Tämän jälkeen voidaan antaa ehtoja kyseisen elementin ominaisuuksille. Määrittelyosa erotetaan ehto-osasta kaksoispisteellä. Muuttujien käsittelyyn ja ohjelmointiin tottumattomalle tällainen kyselyn muodostaminen saattaa ainakin aluksi tuntua hankalalta.

**Kysely 2:** Sanaparit, joissa jälkimmäisen sanan esiintymismuoto on joko 'color' tai 'colour'

- *XTerm: | 'color' =>w | w OR |'colour' =>w | w*
- MQL: [Word] [Word surface='colo[u]?r']
- Simple Query Syntax: [+]+ colo[u]?r
- CQP: [] "colo[u]?r" %c;
- NQL: ($a word) ($b word): TEXT($b)="colo[u]+r"

Kaikki kyselykielet mahdollistavat sanaparien etsimisen ja vaihtoehtojen antamisen hauissa. XTerm-kyselykieli ei mahdollista hakujen kohdistamista sanojen esiintymismuotoihin eikä yksittäisten sanojen käsittelyä. Jos kysely voitaisiin kohdistaa sanojen esiintymismuotoihin, olisi XTerm-kyselykielessä haku siltikin toteutettava kahtena vaihtoehtoisena kyselynä. Kaksi peräkkäistä sanaa erotellaan toisistaan |-merkillä. MQL-kyselykielessä peräkkäiset elementit merkitään kumpikin omien hakasulkeidensa sisään ja attribuutti-arvo -parilla muodostettu ehto asetetaan halutulle elementille. Simple Query Syntaxissa mikä tahansa sana merkitään esittämällä ensin symbolilla + mitä tahansa kirjainta ja tämän jälkeen ilmaisemalla merkillä +, että

kirjaimia voi olla yksi tai useampi eli muodossa (+)+. CQP-kielessä sanaan kohdistuvat haut merkitään lainausmerkkeihin, mutta muuten kysely on hyvin samankaltainen Simple Query Syntaxin kyselylauseen kanssa. Ensimmäisen esimerkin tavoin NQL-kyselylause on tässäkin monimutkaisin, koska ensin määritellään muuttujat ja sitten asetetaan niille ehtoja.

Tässä esimerkissä mikään kyselykieli ei erotu merkittävästi edukseen verrattuna muihin kieliin. Vaikka CQP-kielellä ja Simple Query Syntaxilla on mahdollista tehdä lyhyt ja tiivis kysely, vaativat ne kuitenkin erilaisten katkaisu-, toisto- ja vaihtoehtomerkkien muistamista. Kun oppii nämä operaattorit, on kyselylauseiden muodostaminen helppoa ja nopeaa.

**Kysely 3:** Sanaparit, joissa ensimmäinen sana on substantiivi ja sen lemma ei ole ’Suomi’ ja toisen sanan lemma on ‘laki’

- XTerm: N -’suomi’ | ’laki’ => w | w
- MQL: [Word psp=N AND lemma<>’Suomi’] [Word lemma=’laki’]
- Simple Query Syntax:{!Suomi/N} {laki}
- CQP: [(pos="N") & (lemma!=‘Suomi')] [lemma="laki"];
- NQL: ($a word) ($b word): ($a@pos=”N” & $a@lemma!=’Suomi’) & $b@lemma=’laki’

Kaikki vertailtavat kyselykielet mahdollistavat sekä sanaluokkien että sanojen perusmuotojen eli lekseemien hyödyntämisen hauissa. XTerm-kyselykielessä sanaluokat merkitään sellaisenaan ja lemmat merkitään yksinkertaisten lainausmerkkien sisään. Kuten edellä esitetyissä kahdessa lauseessa, myös tässäkin kyselylauseessa MQL- ja NQL-kyselylauseissa valitaan ensin elementit ja liitetään sitten näihin ehdot attribuutti-arvo -parien avulla. Näiden lisäksi myös CQP-kielessä sanaluokat ja sanojen perusmuodot tulee määrätä attribuutti-arvo -parein. Esimerkkilauseista nähdään kuitenkin selvästi, että esimerkiksi NQL-kyselykielen lauseista muodostuu hyvin pitkiä ja monimutkaisia. Tämä kysely on yksinkertaisin muodostaa XTerm-kyselykielellä sekä Simple Query Syntaxilla. XTerm-kyselykieli ja Simple Query Syntax erottuvat edukseen siinä, että haluttuja ehtoja ei tarvitse kohdistaa erikseen sanaluokkaan, perusmuotoon tai taivutettuun muotoon vaan hakuohjelma osaa tehdä tämän automaattisesti käyttäjän puolesta. Kaikki muut kielet vaativat näiden kohdistavien määrittelyjen muistamista. XTerm-kyselykieli ei myöskään vaadi lainkaan ja-operaattorin käyttöä vaan kaksi hakuehtoa yhdistetään automaattisesti ja-operaatioksi hakua suoritettaessa.

**Kysely 4:** Substantiivi – substantiivi –sanaparit tai substantiivi – prepositio – substantiivi –sanayhdistelmät

- XTerm: N | ? PREP | N => w | w | w
- MQL: [Word psp=N] [Word psp=PREP]*{0,1} [Word psp=N]
- Simple Query Syntax: _N (_PREP)? _N
- CQP: [pos="N"] ([pos="PREP"])? [pos="N"];
- NQL:

NQL-kyselykieltä lukuunottamatta kaikki kielet mahdollistavat haut, joissa haun tuloksen pituus voi vaihdella. Tämä kysely on selvästi yksinkertaisin muodostaa XTerm-kielellä sekä Simple Query Syntaxilla. Tässäkin esimerkkikyselyssä huomataan, että MQL- ja CQP-kyselyt muodostuvat pitkiksi ja monimutkaisiksi, koska niissä haun ehdot pitää kohdistaa ja merkitä attribuutti-arvo -parein.

**Kysely 5:** Kaikki adjektiivi – substantiivi -sanaparit ja sanojen frekvenssit tulosjoukossa

- XTerm: A | N => w, frr | w, frr
- *MQL: [Word psp=adjective] [Word psp=noun]*
- *Simple Query Syntax: _A _N*
- CQP: n1:[class=”ADJ”] n2:[class=”SUBST”] f(n1.lemma) f(n2.lemma);
- *NQL: ($a word) ($b word): $a@pos=”ADJ” & $b@pos=”NN”*

Vain XTerm- ja CQP-kyselykielet tarjoavat mahdollisuuden palauttaa sanojen frekvenssit kyselyn tuloksessa. MQL, Simple Query Syntax ja NQL eivät mahdollista frekvenssien hakuja. Kaikilla kyselykielillä voidaan muodostaa adjektiivi-substantiivi –sanaparit hakeva kysely. Pelkkä kahden peräkkäisen sanan ja niiden sanaluokkien määrittely on XTerm-kyselykielessä mahdollista esittää todella tiiviisti vain merkitsemällä sanaluokkien lyhenteet ja erottamalla nämä |-merkillä. Jotta voidaan palauttaa myös sanojen frekvenssit, vaatii CQP-kieli elementtien nimeämisen. Tällöin kyselyn alkuosa muodostuu monimutkaiseksi. Ensin elementit tulee nimetä ja sitten merkitä niille hakuehdot hakasulkeisiin ja sulkeissa on vielä merkittävä hakuehdot attribuutti-arvopareina eikä pelkästään antamalla haluttu arvo kuten XTerm-kyselykielessä. Tämän jälkeen kyselyn loppuosassa viitataan nimettyihin elementteihin ja haetaan näiden perusmuotojen frekvenssit. Tämä kysely on selvästi helpoin ja yksinkertaisin muodostaa XTerm-kyselykielellä.

**Kysely 6:** Kaikki sanaparit, joissa ensimmäinen sana on adjektiivi ja toinen sana on ’laki’ sekä parien prosentuaaliset frekvenssit tulosjoukosta tai jokin sanojen sidoksisuuteen liittyvä kerroin

- *XTerm: A | ’laki’ => w | w => FR*
- *MQL: [Word psp=ADJ] [Word surface=’laki’]*
- *Simple Query Syntax: _ADJ laki*
- *CQP: [pos="ADJ"] "laki" %c;*
- *NQL: ($a word) ($b word): $a@pos=”ADJ” & $b@lemma=’laki’*

Kaikilla kyselykielillä voidaan hakea sanaparit, joissa ensimmäinen sana on adjektiivi ja toinen sana on laki. Prosentuaalisia frekvenssejä hyödyntää vain CQP-kyselykieli, mutta silläkin prosentuaalinen frekvenssi voidaan palauttaa vain yhdelle sanalle, ei koko haun tulokselle. XTerm-kyselykieli on ainoa kieli, jolla voidaan palauttaa koko tulostietueen frekvenssi, ei kuitenkaan prosentuaalisena. Mikään kyselykielistä ei mahdollista sanojen sidoksisuuteen liittyvien kertoimien palauttamista eikä mahdollista tätä esimerkkihakua kokonaisuudessaan.

**Kysely 7:** Kolme peräkkäistä sanaa, joista keskimmäinen on ’she’, ’he’ tai ’it’, tulos sanojen perusmuodoissa

- XTerm: |’she’| => l | l | l OR => |’he’| => l | l | l OR |’it’| => l | l | l
- MQL:[ ] [Word lemma IN (’she’, ’he’, ’it’)] [ )]
- Simple Query Syntax: (+)+ ({she}|{he}|{it}) (+)+
- CQP:[] [lemma="she|he|it"] [];
- ($a word) ($b word) ($c word): $b@lemma=”she” | ”he” | ”it”

Kaikilla kyselykielillä voidaan muodostaa haku, joka palauttaa kolme peräkkäistä sanaa, joista keskimmäinen on she, he tai it. XTerm-kielessä se tulee tehdä hieman monimutkaisesti kolmella kyselyllä, jotka asetetaan vaihtoehtoisiksi. MQL-kieli tarjoaa IN-operaattorin, jonka avulla voidaan luetella mahdolliset vaihtoehdot. Tämä on hyödyllinen ominaisuus, jos vaihtoehtoja on runsaasti.

Kyselyn palautusmuotoon voidaan vaikuttaa vain XTerm-kielessä. Se onkin siis ainoa kieli, joka selviytyy tästä esimerkkikyselystä. Esimerkkilauseen kuusi tavoin myös tämä kyselyn tulostusmuotoon vaikuttaminen helpottaisi tulosten jatkokäsittelyä käyttäjän kannalta. Erityisesti voimakkaasti taipuvien kielten teksteihin kohdistuvissa hauissa olisi erittäin tärkeää voida palauttaa tulos haluttaessa joko esiintymismuodossaan tai perusmuodossaan.

**Kysely 8:** Sanaparit, joissa ensimmäinen sana on mikä tahansa ja toinen sana on ’säädös’, tulokseen mukaan vain ne sanaparit, joiden frekvenssi tekstissä on yli kymmenen

- *XTerm: | ’säädös’ => w | w => FR*
- *MQL: [Word] [Word lemma=’säädös’]*
- *Simple Query Syntax: (+)+ {säädös}*
- *CQP: Query= [] “säädös” %c; group Query target lemma cut 10;*
- *NQL: ($a word) ($b word): $b@lemma=”säädös”*

Kaikki kyselykielet mahdollistavat sanaparien, joissa jälkimmäinen sana on säädös, hakemisen. Yksikään kyselykieli ei kuitenkaan mahdollista tuloksen rajaamista frekvenssin perusteella. XTerm-kyselykieli on ainoa, jolla voidaan palauttaa, ei pelkästään yksittäisten sanojen, vaan koko tulosrivin frekvenssi. Tämä on merkittävä etu muihin kieliin verrattuna ja vähentää tulosten jatkokäsittelytarvetta.

**Kysely 9:** Lause, jossa on sanat ’lain’ ja ’soveltaminen’ enintään kolmen sanan päässä toisistaan eli väliin jää enintään kaksi sanaa

- *XTerm: ’laki’| [2] |’soveltaminen => w|w|w|w OR ’laki’|’soveltaminen => w|w*
- MQL: [Sentence .. [Word surface=’[Ll]ain’] .. <=3 [Word surface =’soveltaminen’] ..]
- Simple Query Syntax: lain <<3>> soveltaminen
- CQP: ("laki" [word!=”\.”]{0,2} "soveltaminen") %c within s
- NQL: ($a word) ($b word) ($c sentence): TEXT($a)=”lain” & text($b)=”soveltaminen” & $a <>3 $b & $c ^ $a & $c ^ $b

Kaikki kyselykielet mahdollistavat elementtien etäisyyden määrittelemisen jollain tavalla kyselyissään. Nämä esimerkkilauseet havainnollistavat kuitenkin erittäin hyvin kielten eroavaisuudet. XTerm-kyselykielen kyselylauseesta tulee pitkä ja siinä tulee käyttää erilaisia operaattoreita. Sanojen laki ja soveltaminen välille määritellään ensin joko yksi tai kaksi sanaa merkinnällä [2] ja tämän jälkeen määritellään haku, jossa sanat laki ja soveltaminen ovat peräkkäin. Nämä vaihtoehdot yhdistetään OR-operaattorilla. Tämä on kuitenkin keinotekoinen tapa määritellä sanojen etäisyys. XTerm-kyselykielessä ei sanaa voida hakulausekkeessa antaa esiintymismuodossaan, joten tuloksiin voisi tulla myös esimerkiksi ’laki soveltamisesta’. MQL-kyselykielessä tulee rajata haku lauseeseen ja tämän jälkeen liittää etäisyysoperaattorilla sanojen

esiintymismuotojen määrittelyt toisiinsa. Tällainen rakenne ei varmastikaan ole helppo muistaa, jos kyselykieltä ei käytä usein. Simple Query Syntax -kysely on tässä tarkoituksessa selvästi yksinkertaisin ja helpoin. Siinä tulee ainoastaan muistaa, miten etäisyys määritellään molempiin suuntiin. Myös CQP-kieli vaatii monenlaisten merkintöjen muistamista. NQL-kieli vaatii ensin sanojen ja lauseen määrittelemistä, tämän jälkeen määritellään sanojen sisältö ja lopuksi määritellään sanojen etäisyys toisistaan sekä sanojen kuuluminen haluttuun lauseeseen. Voidaankin todeta NQL:n olevan selvästi monimutkaisin. Koska kyselylauseke muodostuu pitkäksi, se on hankala oppia ja muistaa eikä näin sovellu harvoin käytettäväksi kieleksi. XTerm-kyselykielessä hakua ei voida kohdistaa sanojen esiintymismuotoihin, joten se on ainoa kieli, jolla tätä esimerkkikyselyä ei voida muodostaa.

**Kysely 10:** Lauseet, jotka sisältävät tekstin ’eduskunnan päätöksen mukaisesti säädetään’

- *XTerm: eduskunta | päätös | mukainen | säätää => w | w | w | w*
- MQL: [Sentence ..

  [Word surface=’[Ee]duskunnan’]

  [Word surface=’päätöksen’]

  [Word surface=’mukaisesti’]

  [Word surface=’säädetään’]

  ..]
- Simple Query Syntax: eduskunnan päätöksen mukaisesti säädetään
- CQP:”eduskunnan” ”päätöksen” ”mukaisesti” ”säädetään” %c expand to s
- NQL: ($a sentence): TEXT($a) =~/.* eduskunnan päätöksen mukaisesti säädetään .*”/

XTerm-kyselykieli ei mahdollista hakujen kohdistamista sanojen esiintymismuotoihin, mutta kyselykielessä on mahdollista hakea peräkkäiset sanat perusmuodoillaan ’eduskunta päätös mukainen säätää’. Tähän voitaisiin tietysti yhdistää kielitieteelliset merkinnät, joten esimerkiksi `’eduskunta’ GEN`, hakisi vain genetiivimuodossa olevia eduskunta-sanoja. XTerm-kyselykieli rajaa haut automaattisesti kohdistumaan vain lauseisiin eikä palauta tuloksenaan yhdistelmiä, jotka menisivät lauserajojen yli. Kokonaisia lauseita ei XTerm-kielessä ole mahdollista palauttaa. MQL-kielessä hakulausekkeesta muodostuu pitkä. Sen rakenne on kuitenkin looginen, mutta monimutkaisiin hakulausekkeisiin ja ohjelmointikieliin tottumattomalle käyttäjälle rakenne voi olla vaikea oppia ja muistaa. Lauseen sisältämät mahdolliset

ylimääräiset sanat merkitään kahdella pisteellä ennen ja jälkeen halutun tekstin. Nämä kaksi pistettä kuvastavat voimalohkoa, jolle ei anneta ehtoja.

Simple Query Syntax ja CQP-kieli ovat selvästi MQL- ja NQL –kyselykieliä yksinkertaisempia ja helpompia, koska halutut sanat voidaan vain luetella peräkkäin. NQL-kyselykielessä määritellään muuttuja, joka on tyyppiä lause, ja haetaan tämän tekstuaalinen sisältö ennen ja jälkeen sanojen ’eduskunnan päätöksen mukaisesti säädetään’. Pisteellä merkitään mitä tahansa merkkiä ja *-merkillä sitä, että merkki voi toistua nollasta moneen kertaa. Näin saadaan määriteltyä se, että ennen tai jälkeen haluttua tekstiä voi olla myös muuta tekstiä.

### 6.3 Yhteenveto esimerkkilauseiden pohjalta

XTerm-kyselykieli on ainoa kyselykieli, joka vaatii hakulausekkeessa tuloksen palautusmuodon määrittelemisen. Jokaisessa kyselyssä on oltava projektio-osa, jossa määritellään, halutaanko kyselyn tulos perusmuodossa vai esiintymismuodossa. Tämä tekee kyselyistä pidempiä ja vaatii tietysti käyttäjältä myös tämän projektio-osan määrittelyn muistamista.

Muut vertailtavat kielet vaativat sen sijaan erilaisten määrittelyjen tarkempaa muistamista. Esimerkiksi NQL-kyselyt ovat lähes aina pitkiä, koska ne koostuvat kahdesta osasta, joissa toisessa määritellään, mitä elementtejä haetaan ja toisessa osassa annetaan elementeille ehdot. Tuloksen palautusmuodon määrittely XTerm-kielessä on merkittävä etu muihin kyselykieliin verrattuna. Tulostusmuoto ja näytettävien lisätietojen määrittely vähentävät tulosten jatkokäsittelytarvetta. Käyttäjä saa heti käyttöönsä hakemiensa tietojen frekvenssit. Ainoastaan XTerm-kieli ja CQP-kyselykieli mahdollistavat tulosmuodon määrittelyn, muut kyselykielet vaativat hakutulosten jatkokäsittelyä.

Esimerkit osoittavat, että kyselykielet soveltuvat hyvin mallin sovittamiseen ja säännöllisiä lausekkeita hyödyntäviin kyselyihin. Termien louhintaan liittyviin tarpeisiin kyselykielet vastaavat melko heikosti. Ne mahdollistavat peräkkäisten sanojen etsimisen ja monipuolisten sana- ja sanaluokkaehtojen asettamisen yksittäisille ja peräkkäisille sanoille. Sen sijaan tulostusmuotoon vaikuttaminen onnistuu kunnolla ainoastaan CQP-kyselykielessä. Myös tilastotietoja kyselykielet voisivat hyödyntää nykyistä paremmin, koska tämä vähentäisi tekstin jatkokäsittelytarvetta. Hakujen tuloksia ei tällöin tarvitsisi siirtää erillisiin ohjelmiin, joilla tekstistä saataisiin laskettua erilaisia tilastotietoja. Myös tulostusmuodon vaikuttamiseen kyselykielet soveltuvat huonosti. Tämänkin toiminnallisuuden parantaminen auttaisi jatkokäsittelyssä ja tulosten analysoinnissa. Tekstin rakennetta kielet hyödyntävät melko hyvin. Useimmissa kielissä haut voidaan kohdistaa sanoihin ja lauseisiin ja näiden elementtien etäisyys voidaan kyselyissä määritellä.

## 6.4 Korpuskyselykielten ominaisuuksien vertailu

Taulukossa 5 esitellään kielten ominaisuuksia. Tarkoitus on havainnollistaa sellaisia ominaisuuksia, joita kaikissa kielissä ei ole. Taulukosta on jätetty pois sellaiset kielten perusominaisuudet, jotka sisältyvät lähes jokaiseen kieleen. Tällainen ominaisuus on esimerkiksi ja- ja tai-operaattoreiden käyttö annettaessa haettaville sanoille ehtoja.

Taulukko 5. Korpuskyselykielten ominaisuudet

| | XTerm | MQL | Simple Query Syntax | CQL | NQL |
|---|---|---|---|---|---|
| Säännölliset lausekkeet | - | x | x | x | x |
| Etäisyys | - | x | x | x | x |
| Sisäkkäisyys | - | x | x | x | x |
| Muuttujien käyttö | - | x | - | x | x |
| Sisäkkäiset kyselyt | - | - | - | x | x |
| Joukko-operaatiot | - | - | - | x | - |
| XML-elementtien hyödyntäminen | - | - | x | x | - |
| Arvon antaminen listana | - | x | - | x | - |
| Vaihtelevan pituiset haut samassa kyselyssä | x | x | x | x | - |
| Tulos esiintymismuodossa | x | x | x | x | - |
| Tulos perusmuodossa | x | - | - | - | - |
| Tuloksen rajaus (esim. koko lause) | - | x | x | x | x |
| Frekvenssi tai jokin kerroin | x | - | - | x | - |
| Määrän rajaaminen | - | - | - | x | - |
| Tuloksen järjestäminen | - | - | - | x | - |

XTerm-kieltä lukuun ottamatta kaikki kielet tarjoavat mahdollisuuden hyödyntää kyselyissä säännöllisiä lausekkeita ja määritellä haettavien elementtien etäisyyksiä ja sisäkkäisyyttä. Muuttujien käyttö kyselyn rakentamisessa sisältyy kaikkiin muihin paitsi XTerm-kieleen ja Simple Query Syntaxiin. Sekä muuttujien käyttö että sisäkkäiset kyselyt antavat kumpikin mahdollisuuden monimutkaisten kyselyiden rakentamiseen ja tekevät tällä tavoin kyselykielistä monipuolisempia. Sisäkkäisiä kyselyjä voi muodostaa CQP- ja NQL-kyselykielissä. Säännöllisten lausekkeiden lisääminen XTerm-kieleen on

helposti toteutettavissa. Muuttujat ja sisäkkäiset kyselyt lisäävät luonnollisesti kielen ilmaisuvoimaa, mutta tekevät kielestä hankalamman käyttää.

Joukko-operaatiot ovat mahdollisia ainoastaan CQP-kyselykielellä. Joukko-operaatiot vaativat kuitenkin erityisiä taitoja kielen käyttäjältä eikä niiden voida olettaa kuuluvan kielten ensisijaisiin ja tärkeimpiin ominaisuuksiin. Haun rajaamisen XML-elementtien avulla mahdollistavat Simple Query Syntax ja CQP. XML-elementtien hyödyntämisen voi kuitenkin korvata määrittelemällä hakuehtoja ja -rajauksia tekstin rakenteellisille elementeille, kuten lauseelle tai fraasille. Tämä on mahdollista MQL- ja NQL-kyselykielissä. Oikeastaan onkin loogisempaa muodostaa haut kohdistaen ehdot elementteihin ilman että näihin viitataan XML-merkintätavalla. XML-merkintätavan ja tavallisten elementteihin kohdistuvien viittausten yhdistäminen samassa kyselyssä voi olla vaativaa ja tekee kyselystä helposti vaikeasti ymmärrettävän käyttäjien kannalta.

Tietyn attribuutin tai elementin arvo voidaan antaa listana MQL- ja CQP-kyselykielissä. Lista voidaan kuitenkin muissa paitsi XTerm-kielessä muodostaa myös disjunktio-operaattorilla. Kaikilla muilla paitsi NQL-kyselykielellä voidaan muodostaa kyselyjä, joiden tuloksen pituus voi vaihdella. Tämä on tärkeä ominaisuus, koska se antaa käyttäjälle vapauden määritellä kyselyjä, joissa tuloksen pituutta ei tarvitse ennalta tietää eikä määritellä.

Kaikki muut kyselykielet paitsi NQL palauttavat kyselyiden tulokset esiintymismuodossa. XTerm-kyselykieli on ainoa kieli, jossa tulos on mahdollista saada myös perusmuodossaan. Tämä on selvästi merkittävä etu verrattuna muihin kyselykieliin. NQL-kyselykieli palauttaa tuloksen XML-muodossa, jossa jokaista kyselylausekkeen muuttujaa vastaa osoitin, joka sisältää linkin kyseiseen elementtiin. XTerm-kieltä lukuun ottamatta kaikki kielet antavat mahdollisuuden rajata haun tulokset esimerkiksi kokonaisiksi lauseiksi.

Vertailluista kyselykielistä CQP oli ainoa, joka sisältää merkittävästi erilaisia tuloksen esittämiseen liittyviä ominaisuuksia. Vain CQP-kielessä ja XTerm-kyselykielessä on tulokseen mahdollista saada mukaan haettujen sanojen tai muiden elementtien frekvenssit. CQP-kyselykieli on kielistä ainoa, jossa tuloksen määrää voidaan rajata ja tulos voidaan järjestää. Nämä ominaisuudet tekevät tuloksista helpommin tulkittavia ja vähemmän jatkokäsittelyä vaativia.

### 6.5 Yhteenveto korpuskyselykielistä ja niiden vertailusta

Kuten korpuskyselykielten kuvauksesta, esimerkkilauseista ja korpuskyselykieliä vertailevasta taulukosta 5 voidaan huomata, ovat vertaillut kyselykielet painottuneet eri tavoilla. Kullakin kyselykielellä on selvästi omat vahvuutensa ja heikkoutensa. Sekä XTerm-kyselykieli että Simple Query Syntax ovat helppokäyttöisiä ja yksinkertaisia kyselyjä on vaivaton muodostaa. Samalla kuitenkin molemmista kyselykielistä puuttuu ominaisuuksia, joita monimutkaisemmista kyselykielistä löytyy. Yhteenvetona voidaan

todeta, että kielet, jotka tarjoavat eniten ominaisuuksia ja mahdollisuuksia kyselyiden muodostamiseen, ovat samalla myös monimutkaisempia ja hankalammin opittavia. Se, mikä ominaisuuksissa saavutetaan, menetetään käytettävyydessä.

Koska XTerm-kyselykielen kehittäminen lähti liikkeelle tarpeesta löytää tekstistä erilaisia termiehdokkaita, on kyselykielen vahvuus selvästi peräkkäisten sanojen löytäminen yksinkertaisella ja lyhyellä kyselyllä. Voidaan varmasti todeta, että XTerm-kyselykieli täyttää tarpeet, joita termien löytämiseen liittyy, vaikka kielestä puuttuukin muita ominaisuuksia.

Jos käyttäjän on mahdollista valita korpuskyselykieli, jota hän käyttää, kannattaa kielten ominaisuuksia vertailla ennen päätöksen tekemistä. Jokaisen kyselykielen taustalla on toisista poikkeava lähtökohta siitä, mihin käsittelytarpeisiin kieli on suunnattu. XTerm-kielen lähtökohta on termien muodostamisen helppous ja tulostuksen mahdollistaminen sekä esiintymismuodossa että perusmuodossa. Simple Query Syntaxilla se on kielen käytön helppous ja nopeus yksinkertaisissa kyselyissä. CQP-kyselykielen vahvuus on erittäin monipuoliset ominaisuudet erilaisten kielitieteellisten ongelmien ratkaisuun ja tekstin jatkokäsittelytarpeen vähentäminen. MQL- ja NQL-kyselykielten vahvuus on elementtien suhteiden kuvaaminen esimerkiksi sisäkkäisyyden, edeltävyyden, etäisyyksien ja dominanssin avulla.

# 7 Johtopäätökset

## 7.1 Tutkimusmenetelmät

Tässä työssä esiteltiin neljä aikaisemmin toteutettua korpuskyselykieltä, Emdros-järjestelmän MQL-kyselykieli, NITE XML Toolkit -järjestelmän NXT-kyselykieli sekä IMS Corpus Workbench -ohjelmiston päällä toimivat korpuskyselykielet CQP-kieli sekä Simple Query Syntax. Näitä korpuskyselykieliä verrattiin tämän tutkimuksen yhteydessä toteutettuun termien louhintaan suunniteltuun XTerm-kyselykieleen. Kustakin kyselykielestä koostettiin lyhyt kuvaus, joka esittelee kielen tärkeimmät ominaisuudet ja kuvaa rakenteen, johon kielen määrittely perustuu. Kuvaukset ilmentävät kielten erilaisia piirteitä ja soveltuvuutta erilaisiin tilanteisiin. Pelkästään kielten kuvauksista voidaan päätellä, kuinka monipuolisia ne ovat, ja saada viitteitä siitä, millaisiin käyttötarkoituksiin kielet soveltuvat.

Kielten soveltuvuutta erilaisiin käyttötarkoituksiin havainnollistettiin kymmenellä esimerkkikyselyllä. Esimerkit pyrittiin valitsemaan niin, että ne edustaisivat mahdollisimman monipuolisesti kielentutkijoiden tärkeimpiä tarpeita ja yleisempiä korpuksen käyttöön liittyviä tilanteita. Esimerkkikyselyiden avulla voidaan osoittaa, mihin käyttötarkoituksiin ja -tilanteisiin kielet soveltuvat. Jotkin kielet eivät selviydy tietyistä tilanteista lainkaan ja joissakin tapauksissa kyselyistä tulee erittäin pitkiä ja monimutkaisia. Korpustutkimukseen liittyy oletettavasti myös tilanteita, joita esimerkkikyselyissä ei ole huomioitu, ja esimerkkejä olisikin voinut olla enemmän. Kymmenen erilaista kyselyä riittävät kuitenkin melko hyvin osoittamaan kielten erot sekä merkittävimmät vahvuudet ja heikkoudet.

## 7.2 Korpuskyselykielten arviointi

Korpuskyselykieliä ja -hakujärjestelmiä vertailtaessa voidaan arviointi jakaa kahteen pääkohtaan. Ensimmäiseksi voidaan vertailla kielten ominaisuuksia ja sitä, kuinka ne vastaavat kieltentutkijoiden tarpeisiin. Toinen huomioitava asia korpuskyselykielten vertailussa on kyselykielten käytettävyys.

### 7.2.1 Ominaisuudet

Tässä työssä on tutkittu korpuskyselykielten ominaisuuksia esimerkkikyselyjen avulla sekä listattu kielten ominaisuuksia taulukkoon 5. Esimerkkikyselyt havainnollistavat selvästi millaisiin kielentutkijan tai muun korpuksen käyttäjän ongelmiin kielet voivat vastata. Esimerkkikyselyjen perusteella kuitenkaan yksikään korpuskyselykielistä ei erotu selvästi muita paremmaksi tai huonommaksi. Ominaisuuksien taulukoinnin perusteella huomataan, että esimerkiksi CQP-kyselykieli sisältää lähes kaikki esitellyt ominaisuudet. Sen sijaan XTerm- ja NQL-kyselykielet sisältävät paljon vähemmän

ominaisuuksia. Ominaisuuksien suuri määrä ei kuitenkaan kerro sitä, pystyykö korpuskyselykieli vastaamaan kielen tutkijan tarpeisiin. Moniin ongelmiin voidaan saada vastaus eri tavoilla ja erilaisilla hauilla. Esimerkkikyselyt ovatkin siksi luotettavampi keino vertailla todellista soveltuvuutta erilaisiin tilanteisiin.

### 7.2.2 Käytettävyys

ISO 9241-11 -standardi määrittelee käytettävyyden mittaavan sitä, missä määrin tietyt käyttäjät tietyssä tilanteessa voivat käyttää tuotetta tiettyyn tarkoitukseen tuloksekkaasti, tehokkaasti ja tyytyväisenä. Käytettävyyden lähtökohtana on aina käyttäjä. Käytettävyys on sitä, että suunniteltu asia vastaa käyttäjän kykyjä ja taitoja sekä sopii käyttötarkoitukseensa. [Ovaska, 2004.]

Nielsen [1993] on jakanut käytettävyyden viiteen osaan, jotka ovat opittavuus, tehokkuus, muistettavuus, virheettömyys ja miellyttävyys. Ainakin opittavuutta, tehokkuutta ja muistettavuutta voidaan arvioida myös korpuskyselykieliä vertailtaessa. Opittavuus korpuskyselykielten käytön kannalta voi tarkoittaa esimerkiksi sitä, miten nopeasti käyttäjä pystyy oppimaan kielen ja muodostamaan sillä tarvitsemansa haut. Esimerkkikyselyt osoittavat selvästi, että opittavuuden kannalta Simple Query Syntax on kaikkein yksinkertaisin ja käytettävin. Perushaut onnistuvat pelkästään kirjoittamalla haetut sanat peräkkäin eikä haluttuja arvoja tarvitse sitoa attribuutteihin. Myös XTerm-kielen alkuosa on käyttäjälle helppo, peräkkäiset sanat vain erotellaan toisistaan |-merkillä. Sen sijaan loppuosan merkintätapa käyttäjän tulee opetella. Kaikki muut kielet vaativat käyttäjältä jonkin verran kielen opiskelua. MQL-kielen kyselyiden peräkkäinen ja sisäkkäinen rakenne vastaa kuitenkin tekstin peräkkäistä ja sisäkkäistä rakennetta ja voi näin olla käyttäjälle looginen ja intuitiivinen. NQL-kielen opiskelu on varmasti käyttäjille vaativampaa kuin muiden vertailtujen kyselykielten opiskelu. Elementit on ensin sidottava muuttujiin ja vasta kyselyn toisessa osassa voidaan näille muuttujille asettaa ehtoja.

Tehokkuus korpuskyselykieliä tutkittaessa voi tarkoittaa monenlaisia asioita. Tehokkuus voi olla sitä, kuinka nopeasti ja vaivattomasti haun muodostaminen käyttäjältä onnistuu. Se voi tarkoittaa myös monimutkaisen haun muodostamista suhteellisen yksinkertaisena tai ehtojen asettamista jollain käyttäjän kannalta yksinkertaisella ja nopealla tavalla. Erilaiset kielen tarjoamat oikopolut ja lyhyet tavat ilmaista monimutkaisia asioita lisäävät korpuskyselykielen tehokkuutta. Esimerkiksi CQP-kielessä voidaan käyttää listoja, joiden avulla halutun muuttujan arvoksi voidaan antaa ennalta määrätty lista. Jokaista vaihtoehtoa ei tarvitse luetella erikseen ja liittää näitä ehtoja yhteen tai-operaatiolla.

Yksi keino lisätä kyselykielen tehokkuutta on XTerm-kielessä ja Simple Query Syntaxissa oleva ominaisuus, joka sitoo oletuksena annetut hakuehdot tiettyyn elementtiin tai attribuuttiin. Yksinkertaisissa hauissa käyttäjän ei tarvitse asettaa ehtoja

attribuutti-arvo -parien avulla vaan järjestelmä tekee tämän käyttäjän puolesta. XTerm-kielessä haussa annetut sanat sidotaan sanojen lemmoihin ja Simple Query Syntaxissa sanojen esiintymismuotoihin.

Tehokkuuteen voidaan laskea myös se, miten paljon käyttäjä voi vaikuttaa kyselykielen tulostusmuotoon. Olisi hyvä, että käyttäjä voisi määritellä, haluaako nähdä tulokset niiden esiintymismuodoissa vai onko olennaisempaa nähdä sanojen lemmamuodot tai jotakin lisäinformaatiota. Jos tulos on mahdollista lajitella, ja siihen voidaan liittää esimerkiksi sanojen frekvenssit, pääsee käyttäjä huomattavasti helpommalla tulosten jatkokäsittelyn kannalta. Kyselykielen muistettavuutta voidaan parantaa sillä, että se antaa käyttäjälle loogisen ja intuitiivisen tavan muodostaa kyselyjä, vaikka tekstin rakenne olisi monimutkainen. Esimerkiksi XTerm-kyselykielessä kielitieteelliset merkinnät on tallennettu XML-elementtien attribuutteihin ja sanojen lemmat on tallennettu XML-elementtien sisällöiksi. Kuitenkin käyttäjä voi asettaa kyselyyn kummatkin ehdot eikä hänen tarvitse välittää siitä, miten tieto on tallennettu. Esimerkiksi NQL-kyselykieli vaatii käyttäjältä paljon muistamista. Sanan kielitieteelliset merkinnät liittyvät pos-attribuuttiin ja lemma on tallennettu lemma-nimiseen attribuuttiin. NQL-kyselykielen hauissa ehdot on liitettävä muuttujiin ja muuttujien attribuutit on nimettävä.

Kyselykielissä on ominaisuuksia, jotka selvästi lisäävät käyttäjän muistikuormaa. CQP-kyselykielessä on runsaasti ominaisuuksia, mutta monet näistä vaativat käyttäjältä tarkkojen komentoja muistamista. MQL- ja CQP-kielet taas tarjoavat käyttäjälle erittäin monipuoliset mahdollisuudet määritellä haettavan tekstin rakennetta, mutta käyttäjän on muistettava tarkasti, miten esimerkiksi sisäkkäisyys, dominanssi tai etäisyys määritellään. Jopa yksinkertaisimmissa ja selkeimmissä kyselykielissä, XTermissä ja Simple Query Syntaxissa, käyttäjältä vaaditaan monenlaisten pienten määrittelyjen muistamista.

Nielsenin [2005] esittelemiä kymmentä käytettävyysheuristiikkaa ei voi kaikkia soveltaa kyselykielten arvioitiin, koska heuristiikat soveltuvat parhaiten graafisten käyttöliittymien arviointiin. Koko korpusjärjestelmää suunniteltaessa ja graafista käyttöliittymää arvioitaessa voidaan arviointia tehdä yleisten käytettävyysarviointimenetelmien avulla. Korpushakujärjestelmien käyttöliittymät on jätetty tämän tutkimuksen ulkopuolelle. Yksi Nielsenin heuristiikoista on palvelun ja tosielämän vastaavuus. Tämä tarkoittaa muun muassa käyttäjälle tuttujen käsitteiden käyttämistä. Lähes kaikissa tutkittavissa kyselykielissä käytetyt attribuuttien ja elementtien nimet ovat kuvaavia ja todennäköisesti käyttäjille tuttuja. Esimerkiksi hauissa voidaan käyttää samoja nimiä, joilla korpus on loogisesti koostettu: word, sentence, lemma ja surface.

Yksi Nielsenin [2005] heuristiikoista on yhteneväisyys ja standardien käyttö. Monet korpuskyselykielistä sisältävät yleisiä tietojenkäsittelyyn ja tiedonhakuun liittyviä

toiminnallisuuksia. Näitä ovat sanojen katkaisu, säännölliset lausekkeet, Boolen lausekkeet sekä muuttujien käyttö. Koska sanojen katkaisua ja Boolen operaattoreita käytetään jo laajalti ja ne ovat tuttuja lähes kaikille, jotka ovat tehneet tietokoneavusteista tiedonhakua, näiden käyttö korpuskyselykielissä on käytettävyyttä parantavaa. Myös säännölliset lausekkeet ja muuttujien käyttö tehostavat korpuskyselykieliä ja tekevät niistä joustavampia. Nämä vaativat kuitenkin sitä, että käyttäjä on joko perehtynyt aiemmin tai perehtyy korpuskyselykieltä opiskellessaan näihin asioihin. Kieltentutkijan ei voida kuitenkaan olettaa osaavan etukäteen käyttää esimerkiksi säännöllisiä lausekkeita.

# 8 Yhteenveto

Korpusten käyttö kielitieteessä tarjoaa monenlaisia mahdollisuuksia kielten tutkijoille. Suurten tietomassojen tallentaminen ja nopeat kyselyt tekevät isoihin tekstimassoihin kohdistettavista kyselyistä mahdollisia. Jos olemassa olevista korpuskyselykielistä ja korpushakujärjestelmistä ei löydy itselle sopivaa, voi järjestelmän kehittää itsekin kuten tässä tutkimuksessa tehtiin. Oman korpuksen muodostaminen ja kyselyjen tekeminen korpukseen ei vaadi suunnattomia resursseja tai opiskelua. Tutkimuksessa kehitettyä kyselykieltä ja tietokantaa on myös mahdollista laajentaa ja kehittää. Uusien tekstien lisääminen tietokantaan on helppoa, jos teksteihin voidaan liittää tarvittavat kielitieteelliset merkinnät ja rakenne voidaan määritellä tarvittavina XML-elementteinä. Työssä toteutettu hakujärjestelmä ja korpuskyselykieli sallivat myös uusien kielitieteellisten tunnisteiden lisäämisen ilman, että järjestelmään tai kyselykieleen tarvitsee tehdä muutoksia. Korpushakujärjestelmiä suunniteltaessa ja kehitettäessä kieltä tärkeämpää onkin korpuksen merkitseminen. Sellaista tietoa, jota korpukseen ei ole merkattu, ei voida myöskään kyselykielellä palauttaa.

Kuten luvussa 4.4. ja sen aliluvuissa huomataan, XTerm-korpuskyselykielen kyselylauseet ovat huomattavasti selkeämpiä ja lyhyempiä kuin tietokannan ymmärtämän XQuery-kyselykielen lauseet. Vaikka korpuskyselykielen ominaisuudet eivät aina riitä kaikkien haluttujen asioiden tutkimiseen, ne tarjoavat kielentutkijoille helppokäyttöisen työkalun, jolla tehdä kyselyjä korpukseen tallennettuun dataan. Osa korpuskyselykielistä on monimutkaisia, mutta kyselyiden muodostaminen on niillä kuitenkin helpompaa kuin perinteisillä tietokantakyselykielillä. Korpuskyselykielet ovat pienen ammattiryhmän työkaluja, mutta niiden merkitystä ei silti tulisi väheksyä. Huomioon otettavaa on myös se, että korpuskyselykieliä voitaisiin hyödyntää myös muunlaisissa tutkimuksissa, ei pelkästään sanastotyössä tai kielitieteen tutkimuksissa. Esimerkiksi tiedonhaun tutkimuksessa ja kehittämisessä voidaan hyödyntää termien louhintaa.

Emdros-järjestelmän MQL-kyselykieli soveltuu vertailluista kielistä parhaiten pitkien tekstiosuuksien ja lauserakenteiden kuvaamiseen. Sen vahvuus on rakenteiden kuvaamisessa. NITE XML Toolkit -järjestelmän NQL-kyselykielen käyttö vaatii opiskelua, mutta tarjoaa hyvin monenlaisia tapoja liikkua elementtien välillä. Näitä suhteita elementtien välillä ovat muun muassa dominanssijärjestys, vertikaalinen etäisyys, horisontaalinen etäisyys sekä edeltävyysjärjestys. IMS Corpus Workbenchin CQP-kielen vahvuus on siinä, että sen käyttö on helppo aloittaa, mutta se tarjoaa myös monipuolisia lisäominaisuuksia, jotka vähentävät tekstin jatkokäsittelytarvetta. Simple Query Syntax on selvästi yksinkertaisin ja helpoin, mutta sen avulla pystyy silti tehokkaasti asettamaan erilaisia ehtoja haettaville teksteille. XTerm-kyselykieli

soveltuu hyvin tarkoitukseensa. Sillä on nopeaa ja vaivatonta etsiä termiehdokkaita korpuksesta ja palauttaa näiden frekvenssit mukana tuloksessa.

Tässä tutkimuksessa vertaillut korpuskyselykielet eroavat selvästi taustaltaan ja tarkoitukseltaan. Kaikki kielet eivät sovellu kaikkiin tilanteisiin. Valittaessa korpuskyselykieltä ja korpushakujärjestelmää tulee arvioida omia tarpeita. Monipuolisin kyselykieli ei välttämättä ole kaikkiin tilanteisiin soveltuvin. Kuten tutkimuksessa aiemmin todettiin, se, mikä käyttötarkoituksissa saavutetaan, menetetään usein käytettävyydessä. On turha valita korpuskyselykieltä tai -järjestelmää, jonka käyttö vaatii runsaasti opiskelua, jos kuitenkaan kaikkia kielen ominaisuuksia ei tulla tarvitsemaan.

XTerm-kieli eroaa muista vertailluista kielistä melko paljon. Taulukon 5 avulla voitaisiin tehdä johtopäätös, että XTerm ei sovellu moniin tilanteisiin siksi, että se ei sisällä kaikkia niitä ominaisuuksia, joita muissa vertailluissa kielissä on. Esimerkiksi sanan katkaisu ei ole mahdollista XTerm-kielessä. Koska XTerm-kielessä haut kohdistetaan sanojen lemmoihin, ei sanojen katkaisulle ole välttämättä tarvetta. Jos XTerm-kielestä halutaan tehdä monipuolisempi ja paremmin soveltuva erilaisiin kielitieteellisiin tarpeisiin, voitaisiin kieleen lisätä esimerkiksi mahdollisuus asettaa tuloksen palautusmuodoksi koko lause tai yhdyssanan osat. Esimerkiksi lauseen (s) palauttaminen voitaisiin merkitä kyselyn kolmanteen osaan, `’eduskunta’ | ’päätös’ => w | w => s`. Kysely palauttaisi kaikki lauseet, joissa esiintyy peräkkäin sanat eduskunta ja päätös.

Jos XTerm-kieli halutaan säilyttää nimenomaan termien louhintaan tarkoitettuna kielenä ja parantaa näitä ominaisuuksia, olisi hyvä lisätä kieleen ainakin mahdollisuus rajata frekvenssin perusteella. Tämä voitaisiin merkitä esimerkiksi liittämällä rajaava luku frekvenssimääreen perään `A | N => w | w => FR 10`. Kysely palauttaisi vain sellaiset adjektiivi-substantiivi -parit, joiden frekvenssi on 10 tai yli. Frekvenssirajauksen avulla tuloksesta saataisiin jo kyselyvaiheessa valikoitua pois harvoin esiintyvät termiehdokkaat, jotka eivät todennäköisesti ole termejä. Myös mahdollisuus tulosten järjestämiseen tulostietueen frekvenssin mukaan voisi tehostaa termien louhintaa. Järjestäminen voisi tapahtuma komennolla sort tai jollakin tähän tarkoitukseen varatulla yksittäisellä symbolilla.

## Viiteluettelo

[Abeillé, 2003] Anne Abeillé, *Treebanks – Building and Using Parsed Corpora*. Kluwer Academic Publishers, 2003.

[BNCweb, 2010] BNCweb (CQP-Edition) A web based interface to the British National Corpus. 2010. Available as *http://www.bncweb.info/*. (Viitattu 4.1.2012)

[Carletta et al., 2005] Jean Carletta, Stefan Evert, Ulrich Heid, Jonathan Kilgour, *The NITE XML Toolkit: data model and query language*. 2006. Language Resources and Evaluation. Springer Netherlands, **39**, 4 (Dec. 2005), 315-334.

[Christ et al., 1999] Oliver Christ, Bruno Schulze, Anja Hofmann, Esther König, *The IMS Corpus Workbench: Corpus Query Processor (CQP) User's Manual*. University of Stuttgart, Institute for Natural Language Processing, 1999. Available as *http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQPUserManual/PDF/cqpman.pdf.* (Viitattu 4.12.2011)

[Connolly and Begg, 2002] Thomas Connolly, Carolyn Begg, *Database systems – A practical Approach to Design, Implementation, and Management*. Addison Wesley, 2002.

[Cooper and Weekes, 1983] R.A. Cooper, A. J. Weekes, *Data, Models and Statistical Analysis*. Barnes & Noble Books, 1983.

[Daille, 1994] Daille, Béatrice. *Combined approach for terminology extraction: lexical statistics and linguistic filtering*. TALANA, University Paris 7 (March 1994). Available as *http://ucrel.lancs.ac.uk/papers/techpaper/vol5.pdf.* (Viitattu 4.1.2012)

[Elmasri and Navathe, 2009] Ramez Elmasri, Shamkant B Navathe, *Fundamentals of Database Systems*. Pearson, 2009.

[Evert and Voormann, 2003] Stefan Evert, Holger Voormann, *NQL – A Query Language For Multi-Modal Language Data*. Technical report, IMS, University of Stuttgart. Version 2.1. (March 2003). Available as *http://www.ims.uni-stuttgart.de/projekte/corplex/paper/evert/NITE-NQL.pdf.* (Viitattu 12.4.2012)

[Evert et al., 2003] Stefan Evert, Jean Carletta, Timothy J. O'Donnell, Jonathan Kilgour, Andreas Vögele, Holger Voormann, *The NXT Object Model*. Technical report, IMS, University of Stuttgart. Version 2.1. 2003. Available as *www.ims.uni-stuttgart.de/projekte/corplex/paper/evert/NITE-NOM.pdf.* (Viitattu 12.4.2012)

[Evert, 2005] Stefan Evert, *The CQP Query Language Tutorial (CWB Version 2.2.b90)* Available as *http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQPTutorial/cqp-tutorial.pdf.* (Viitattu 12.4.2012)

[Hoffmann et al., 2008] Sebastian Hoffmann, Stefan Evert, Nicholas Smith, David Lee, Ylva Berglund Prytz, *Corpus Linguistics with BNCweb – a Practical Guide*. Peter Lang, 2008.

[IMS, 2009] IMS Corpus Workbench. University of Stuttgart, Institute for Natural Language Processing, 2009. Available as *http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/*. (Viitattu 18.4.2012)

[ISO/IEC 14977, 1996] *International Standard ISO/IEC 14977:1996*. Extended BNF, 1996. (Viitattu 1.6.2012)

[Järvelin, 1995] Kalervo, Järvelin, *Tekstitiedonhaku tietokannoista. Johdatus periaatteisiin ja menetelmiin*. Suomen ATK-Kustannus, 1995.

[Katz, 2004] Howard Katz (ed.), *XQuery from the Experts*. Addison-Wesley, 2004.

[Lehtinen et al., 1995] Marja Lehtinen, Pirjo Karvonen, Tarmo Rahikainen, *Tekstikorpukset*. Kotimaisten kielten tutkimuskeskuksen julkaisuja 81, Raportti tekstikorpusten koostamisperiaatteista ja nykysuomen tekstiaineistojen tarpeellisuudesta Kotimaisten kielten tutkimuskeskuksessa. Kotimaisten kielten tutkimuskeskus, 1995.

[Manning and Schütze, 1999] Christophe D. Manning, Hinrich Schütze, *Foundations of statistical natural language processing*. MIT Press, 1999.

[Mason, 2008] Oliver Mason, *Developing software for corpus research. IJES*, **vol. 8 (1)**, (2008), 141-156. Also available as *http://revistas.um.es/ijes/article/view/49141/47011*.

[McEnery and Wilson, 2001] Tony McEnery, Andrew Wilson, *Corpus Linguistics, An Introduction*. Edinburgh University Press, 2001.

[Merikoski et al., 2004] Jorma Merikoski, Ari Virtanen, Pertti Koivisto, *Johdatus diskreettiin matematiikkaan*. Werner Söderström Osakeyhtiö, 2004.

[Nielsen, 1993] Jakob Nielsen, *Usability engineering.* Academic Press, 1993.

[Nielsen, 2005] Jakob Nielsen, *Ten Usability Heuristics*. 2005 Available as *http://www.useit.com/papers/heuristic/heuristic_list.html*. (Viitattu 15.4.2012)

[NITE, 2003] *NITE - Project goals, background and status June 2003*. Available as *http://nite.nis.sdu.dk/aboutNite/*. (Viitattu 7.12.2009)

[Nykänen, 2001] Ossi Nykänen, *XML*. Docendo, 2001.

[Oakes, 1998] Oakes, Michael P. *Statistics for Corpus Linguistics*. Edinburgh University Press 1998.

[Oracle, 2009a] Berkeley *DB XML Reference Guide, Version 2.5.16*, Oracle. Available as *http://docs.oracle.com/cd/E17276_01/html/ref_xml/toc.html.* (Viitattu 23.11.2011)

[Oslo Corpus, 2002] *The Oslo Corpus of Bosnian Texts*. 2002. Available as *http://www.tekstlab.uio.no/Bosnian/Corpus.html.* (Viitattu 23.4.2012)

[Oracle, 2009b] *Oracle Berkeley XML – Introduction to Berkeley DB XML*. Release 2.5. Oracle Berkeley DB. Available as *docs.oracle.com/cd/E17276_01/html/intro_xml/BerkeleyDBXML-Intro.pdf.* (Viitattu 23.11.2011)

[Ovaska, 2004] Saila Ovaska, *Johdatus vuorovaikutteiseen teknologiaan –kurssin luentomateriaali*. Tampereen yliopisto, 2004.

[Pearson, 1998] Jennifer, Person, *Terms in Context*. Benjamins, 1998.

[Petersen, 2004] Ulrik, Petersen, *Emdros – a text database engine for analysed or annotated text. 2004. In: Proceeding COLING '04 Proceedings of the 20th international conference on Computational Linguistics* Article No. 1190. (2004) Also available as *http://emdros.org/petersen-emdros-COLING-2004.pdf.*

[Petersen, 2007] Ulrik Petersen, *Emdros Query Guide*. 2007. Available as *http://emdros.org/MQL-Query-Guide.pdf.* (Viitattu 23.031.2012)

[Salton, 1989] Gerard, Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[Sinclair, 1991] John Sinclair, *Corpus Concordance Collocation*. Oxford University Press, 1991.

[Sivistyssanakirja, 2003] *Sivistyssanakirja*, Werner Söderström Osakeyhtiö, 2003.

[Ullman, 1998] Jeffrey D. Ullman, *Principles of Database and Knowledge-base Systems*, Volume I, Stanford University, Computer Science Press, 1998.

[Voormann et al., 2003] Holger Voormann, Stefan Evert, Jonathan Kilgour, Jean Carletta, *NXT Search – User's manual (Draft)*. IMS University of Stuttgart and HCRC Edinburgh, 2003. Available as *http://www.ltg.ed.ac.uk/NITE/ documents/search-manual.pdf*. (Viitattu 3.1.2011)

[Walmsley, 2007] Priscilla Walmsley, *Xquery*, O'Reilly, 2007.

[IPI PAN, 2008] *Welcome to the IPI PAN Corpus website!* Available as *http://korpus.pl/index.php?lang=en.* (Viitattu 15.4.2012)

[W3C, 1999] James Clark, Steve DeRose, *XML Path Language (XPath) Version 1.0*. W3C Recommendation, 1999. Available as *http://www.w3.org/TR/xpath/*.

[W3C, 2006] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C Recommendation, 2006. Available as *http://www.w3.org/TR/2006/REC-xml11-20060816/*. (Viitattu 5.4.2012)

[W3C, 2008] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, *Extensible Markup Language (XML)* Available as *http://www.w3.org/TR/2008/REC-xml-20081126/*. (Viitattu 15.4.2012)

[W3C, 2010a] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, *XQuery 1.0: An XML Query Language (Second Edition)*. W3C Recommendation, 2010. Available as *http://www.w3.org/TR/xquery/*. (Viitattu 15.4.2012)

[W3C, 2010b] Ashok Malhotra, Jim Melton, Norman Walsh, Michael Kay, *XQuery and XPath Functions and Operators*. Second Edition, W3C Recommendation 2010. Available as *http://www.w3.org/TR/xquery-operators/*. (Viitattu 15.4.2012)

**Liitteet**

LIITE 1

# Käyttöohjeet -
## termien louhinnassa käytettävään kieleen

### KYSELYN PERUSRAKENNE

Kyselykieli on tarkoitettu termien tunnistamiseen ja laajentamiseen. Kysely muodostaa mallin, jonka perusteella ohjelma palauttaa tekstikokoelmasta malliin sopivat termi-kandidaatit.

Kysely koostuu kolmesta osasta. Alkuosa sisältää mallin, jonka perusteella tulosjoukko muodostetaan. Toinen osa sisältää projektio-osan, jossa määritellään, miten lemmat halutaan esittää. Kolmas osa on vapaaehtoinen ja koskee koko mallia. Siinä asetetaan tieto siitä, halutaanko tulokseen kyseisen lemmayhdistelmän frekvenssi. Osat erotetaan toisistaan => -merkinnällä.

### KYSELYOSA

Kyselyosa koostuu osista, jotka on eroteltu |-merkein. Kukin osa vastaa yhtä sanaa tai lemmaa kohdetekstissä. Osien määrää ei ole rajoitettu.

#### Perustoiminnot

Kysely muodostetaan kyselyyn kuuluvien merkkien ja tagien avulla. Suomenkieliset tagit perustuvat Lingsoftin listaan. Lista löytyy osoitteesta:
http://www2.lingsoft.fi/doc/fintwol/intro/tags.html

**Esim** kysely A | N palauttaa kaikki adjektiivi-substantiiviparit.

Osa voidaan jättää myös tyhjäksi, tällöin kyseiselle lemmalle ei aseteta mitään ehtoja. Tyhjiä osia voi olla useampia ja ne voivat sijaita missä tahansa kohdassa.

**Esim** kysely A | | N palauttaa kaikki kolmen lemman yhdistelmät, joissa ensimmäinen lemma on adjektiivi, toinen mikä tahansa ja kolmas substantiivi.

Kyselyjä on mahdollista tehdä myös hakemalla tiettyä lemmaa, joka annetaan kyselyssä.

**Esim** kysely A | ’laki’ hakee kaikki sellaiset lemmaparit, joiden ensimmäinen lemma on adjektiivi ja toinen lemma on laki.

Yksi väli voi myös sisältää useamman tagin.

**Esim** kysely N GEN | N palauttaa kaikki sellaiset lemmaparit, joiden ensimmäinen lemma on substantiivi, joka on genetiivimuodossa ja toinen lemma on substantiivi missä tahansa muodossa.

**Unaarioperaatiot**

Jos halutaan määrätä, ettei tietty tagi saa esiintyä kyseisessä lemmassa voidaan tagiin liittää merkintä –.

**Esim** kysely –A | N | N palauttaa sellaiset kolmen lemman yhdistelmät, joissa ensimmäinen lemma ei ole adjektiivi ja kaksi jälkimmäistä lemmaa ovat substantiiveja.

Myös tietty lemma voidaan rajata pois tuloksesta.

**Esim** kysely N –’asetus’ –’laki’ | N palauttaa kaikki sellaiset substantiivi-substantiiviparit, joiden ensimmäinen lemma ei ole asetus eikä laki.

Yksi tekstissä esiintyvä sana voi sisältää useita lemmoja. Tulosjoukosta voidaan rajata pois kaikki sellaiset lemmat, joilla yksikään lemmaan liittyvän sanan lemmoista ei saa sisältää tiettyä tagia. Tällöin käytetään merkintää – –.

**Esim** kysely A – –ADV |A – –ADV palauttaa kaikki sellaiset adjektiivi-adjektiivilemmaparit, joissa lemmoja vastaavien sanojen mikään lemmoista ei saa olla adverbi.

Vastaavasti voidaan rajata pois jokin tietty lemma.

> **Esim** kysely A | --’kuu’ N palauttaa kaikki sellaiset lemmaparit, joiden ensimmäinen lemma on adjektiivi ja toinen substantiivi, mutta mikään sen lemmoista ei saa olla kuu.

Valinnaisuus merkitään ?-operaattorilla. Tällöin kyseinen tagi voi esiintyä tai olla esiintymättä eli esiintyä 0 – 1 kertaa.

> **Esim** kysely ?A | A | N palauttaa kaikki sellaiset lemmayhdistelmät, jotka koostuvat kahdesta adjektiivista ja näitä seuraavasta substantiivista tai yhdestä adjektiivista ja sitä seuraavasta substantiivista.

Jos halutaan jonkin tagin toistuvan voidaan käyttää tagin edessä numeroa. Numero määrää, kuinka monta kyseisen tagin sisältäviä lemmoja saa esiintyä. Esimerkiksi 4 tarkoittaisi sitä, että kyseisiä lemmoja voi esiintyä 1 – 4.

> **Esim** kysely [3] A | N palauttaa kaikki sellaiset lemmayhdistelmät, joissa on ensin yksi, kaksi tai kolme adjektiivia ja tämän jälkeen substantiivi.

**Binäärioperaatiot**

Kyselyjä on mahdollista yhdistää toisiinsa, jolloin tuloksessa palautetaan kaikkien kyselyjen palauttamat lemmayhdistelmät. OR-operaatiolla toisiinsa liitettyjen kyselyn tulee olla osiltaan yhtä pitkiä, eli palauttaa tuloksenaan yhtä monen lemman yhdistelmiä.

> **Esim** kysely A | ’laki’ OR ADV | ’laki’ palauttaa kaikki sellaiset lemmaparit, joiden ensimmäinen lemma on adjektiivi tai adverbi ja toinen lemma on laki.

OR-operaatiolla voidaan korvata ?-operaatio ja päinvastoin.

> **Esim** kysely A | A | N OR A | N palauttaa saman tuloksen kuin
> ?A | A | N.

**PROJEKTIO-OSA**

Projektio-osassa voidaan määrätä halutaanko lemmat nähdä lemmamuodossa vai tekstissä esiintyvässä muodossa. Lisäksi voidaan määrätä, halutaanko nähdä lemman

frekvenssi koko kokoelmassa tai lemman frekvenssi tulosjoukossa. Kunkin lemman määreet erotellaan |-merkillä kuten kyselyosassa noudattaen samaa järjestystä kuin kyselyosassa. Useampi määre erotellaan toisistaan pilkuilla.

l = lemmamuodossa
w = esiintymismuodossa
frc = lemman frekvenssi kokoelmassa
frr = lemman frekvenssi tulosjoukossa

> **Esim** kysely A|N => l | l palauttaa kaikki adjektiivi-substantiivilemmaparit lemmamuodossa.
>
> **Esim** kysely N|N => w, l | w, l palauttaa kaikki substantiivi-substantiivilemmaparit sekä esiintymismuodossa että lemmamuodossa.
>
> **Esim** kysely PRON => l, frr, frc palauttaa kaikki pronomit lemmamuodossa ja kaikkien näiden frekvensin tulosjoukossa ja frekvenssin kokoelmassa.

Projektio-osaan voidaan liittää myös toinen =>-operaattori.

> **Esim** kysely A | N => l | l => FR palauttaa kaikki adjektiivi-substantiiviparit lemmamuodossa ja näiden parien frekvenssit.

**Ohjelman käyttö**

Tagit on kirjoitettava juuri niin kuin ne listassa ovat. Ohjelma ei löydä oikeita tuloksia, jos tagin PRON kirjoittaa pienillä kirjaimilla ’pron’. Ohjelma ei muutenkaan tarkasta tagien oikeinkirjoitusta.

Käytettäessä unaarioperaatioita ? ja [2] tulee ne kirjoittaa jokaisen osan ensimmäiseksi. Esimerkiksi kyselyä A [2] => l ei ohjelma hyväksy. Numerot voivat olla vain väliltä 1 – 9. ?-operaattoria tulee käyttää vain haettaessa useamman lemman yhdistelmiä. Esimerkiksi hakua ?A => l ei tule käyttää.

Haettaessa lemmalla esimerkiksi ’asetus’ => w, hakee ohjelma lemmaa ’asetus’ nimenomaan sanan lemmasta, vaikka kyselyssä tulostetaankin kyseisen lemman tekstissä esiintynyt muoto (esimerkiksi asetuksen).

LIITE 2

**XTerm-korpukseen merkatut kielitieteelliset tunnisteet**

| Suomi | Venäjä | | Tärkeä yksikiel. analyysissä | Tärkeä kielten välisessä vert. | Vähempi tärkeä säädösteksteissä |
|---|---|---|---|---|---|
| **Part of speech** | | | | | |
| A | Adj | adjective | x | x | |
| ABBR | ? | abbreviation | | x | |
| AD-A | | ad-adjective | | | |
| ADV | Adv | adverb | | | |
| ART | | foreign article | | | |
| C | Conj | conjunction | | x | |
| INTJ | Interj | interjection | | | x |
| N | Noun | noun | x | x | |
| NUM | Numeral | numeral | | | |
| PP | Preposition (huom. ei postposition) | post- or preposition | | x | |
| PREP | | foreign preposition | | | |
| PRON | Pron | pronoun | | | |
| PSP | | postposition | | | |
| Q | merkkaus : Adj | quantifier | | | |
| V | Verb | verb | | | |
| mm PP, PSP, Clitics | Particle | particle (частицы) | | | x |
| kts. Infinitives, Participles | Gerund | gerund (дееприч.) | | | |
| kts. Participles | Part | participles (причастия) | x | | |
| | Parenthesis | parenthesis (вводное слово) | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **Comparation** | | | | | |
| POS | **?** | positive | | | |
| CMP | Compar | comparative | | | |
| SUP | **?** | superlative | | | |
| | | | | | |
| **Case** | | | | | |
| NOM | Nom | nominative (koira) | x | | |
| GEN | Gen | genitive (koiran) | x | | |
| PTV | | partitive (koiraa) | x | | |
| ESS | | essive (koirana) | x | | |
| TRA | | translative (koiraksi) | x | | |
| INE | | inessive (koirassa) | x | | |
| ELA | | elative (koirasta) | x | | |
| ILL | | illative (koiraan) | x | | |
| ADE | | adessive (koiralla) | x | | |
| ABL | | ablative (koiralta) | x | | |
| ALL | | allative (koiralle) | x | | |
| ABE | | abessive (koiratta) | x | | |
| CMT | | comitative (koirineen) | x | | |
| INS | | instructive (koirin) | x | | |
| | Dat | datiivi | x | | |
| | Acc | accusative | x | | |
| | Abl | instrumentaali | x | | |
| | Prep | prepositionaali | x | | |

| | | | | | |
|---|---|---|---|---|---|
| **Number** | | | | | |
| SG | Sg | singular | x | | |
| PL | Pl | plural | x | | |
| | | | | | |
| **Gender** | | | | | |
| | Masc | masculine | x | | |
| | Fem | feminine | x | | |
| | Neutr | neuter | x | | |
| | | | | | |
| | Anim | animate | | | x |
| | Unanim | inanimate | | | x |
| | | | | | |
| **Possessive suffixes** | | | | | |
| 1SG | | 1st person singular | | | |
| 2SG | | 2nd person singular | | | |
| 3 | | 3rd person singular or plural | | | |
| 1PL | | 1st person plural | | | |
| 2PL | | 2st person plural | | | |
| | reflex | reflex suffixes (verbeillä) | | | |
| | | | | | |
| **Mood** | | | | | |
| | Indic | indicative | | | |
| IMPV | Imper | imperative (lue!, mene!) | | | |
| COND | | conditional (lukisi, menisi) | | | |
| POTN | | potential (lukenee, mennee) | | | x |

| | | | | | |
|---|---|---|---|---|---|
| | | There is no feature for indicative forms (lukee, menee). | | | |
| **Tense** | | | | | |
| PRES | Pres | present tense (haluan) | | | |
| PAST | | past tense (halusin) | | | x |
| | Past | Perfect and pluperfect tenses are interpreted as participle forms. | | | x |
| | Fut | future | | | |
| | | | | | |
| **Voice** | | | | | |
| ACT | Active | Active | | | |
| PSS | Passive | Passive | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| **Form of the adjective** | | | | | |
| | Full | full | | | |
| | Brief | short | | | |
| | | | | | |
| **Person** | | | | | |
| SG1 | | 1st person singular (menen) | | | |
| PL1 | 1.p | 1st person plural (menemme) | | | |
| SG2 | 2.p | 2nd person singular (menet) | | | |

| | | | | | |
|---|---|---|---|---|---|
| PL2 | | 2nd person plural (menette) | | | |
| SG3 | | 3rd person singular (menee) | | | |
| PL3 | 3.p | 3rd person plural (menevät) | | | |
| PE4 | | passive ending (mennään) | | | |
| | | | | | |
| **Negative** | | | | | |
| NEGV | kts. Particle | negative verb (en, et, ei) | | | |
| NEG | | negative form (en tehnyt) | | | |
| | | | | | |
| **Infinitives** | | | | | |
| INF1 | inf | 1st infinitive (tulla, tullakseni) | | | |
| INF2 | kts. Gerund (Part of speech) | 2nd infinitive (tullessaan, tullessa) | | | |
| INF3 | | 3rd infinitive (tulemaan) | | | |
| INF5 | | 5th infinitive (tulemaisillaan) | | | |
| | | The 4th infinitive (tuleminen) is interpreted as a noun. | | | |
| | | | | | |
| **Participles** | | | | | |
| PCP1 | kts. (Participles) | 1st participle (lentävä, | x | | |

| | | | | | |
|---|---|---|---|---|---|
| | Part of speech | lennettävä) | | | |
| PCP2 | | 2nd participle (lentänyt, lennetty) | x | | |
| | | | | | |
| **Clitics** | | | | | |
| -han/-hän (poikahan) | | | | | x |
| -ka/-kä (eikä) | | | | | x |
| -kaan/-kään (poikakaan) | | | | | x |
| -kin (poikakin) | | | | | x |
| -ko/-kö (oletko) | | | | | x |
| -pa/-pä (oletpa) | | | | | x |
| -s (onpas) | | | | | x |
| **Other** | | | | | x |
| **FORGN** | | | | | x |
| **PROP** | | | | | x |
| pi | | | | | x |

LIITE 3

**Esimerkki kohteena olevan korpuksen sisältämästä XML-tiedostosta**

```
<?xml version="1.0"?>
<doc n="AlkoholVeroLi">
<p>
<s>
<w>
Laki
<lemma tags="|N|NOM|SG|">laki</lemma>
<lemma tags="|N|NOM|SG|">laki</lemma>
</w>
<w>
alkoholi-
<lemma tags="|N|NOM|SG|">alkoholi-</lemma>
</w>
<w>
ja
<lemma tags="|COORD|C|">ja</lemma>
</w>
<w>
alkoholijuomaverosta
<lemma
tags="|N|ELA|SG|"><segm>alkoholi</segm><segm>juoma</segm><segm>vero</s
egm></lemma>
</w>
<w>
29
</w>
<punct>.</punct>
</s>
<s>
<w>
12
</w>
<punct>.</punct>
</s>
<s>
<w>
1994/1471
</w>
<w>
1994/1471
</w>
</s>
</p>
<p>
<s>
<w>
Eduskunnan
<lemma tags="|N|GEN|SG|"><segm>edus</segm><segm>kunta</segm></lemma>
</w>
<w>
päätöksen
<lemma tags="|N|GEN|SG|">päätös</lemma>
</w>
<w>
mukaisesti
<lemma tags="|ADV|POS|MAN|">mukainen</lemma>
</w>
<w>
```

```
säädetään
<lemma tags="|V|PRES|PSS|PE4|">säätää</lemma>
</w>
<punct>:</punct>
</s>
</p>
<p>
<s>
<w>
Yleiset
<lemma tags="|A|POS|NOM|PL|">yleinen</lemma>
</w>
<w>
säännökset
<lemma tags="|N|NOM|PL|">säännös</lemma>
</w>
</s>
</p>
<p>
<s>
<w>
1
</w>
<w>
§
</w>
</s>
</p>
<p>
<s>
<w>
Etyylialkoholista
<lemma
tags="|N|ELA|SG|"><segm>etyyli</segm><segm>alkoholi</segm></lemma>
</w>
<w>
ja
<lemma tags="|COORD|C|">ja</lemma>
</w>
<w>
alkoholijuomista
<lemma
tags="|N|ELA|PL|"><segm>alkoholi</segm><segm>juoma</segm></lemma>
<lemma tags="|DV-
MINEN|N|PTV|SG|"><segm>alkoholi</segm><segm>juominen</segm></lemma>
</w>
<w>
on
<lemma tags="|COP|V|PRES|ACT|SG3|">olla</lemma>
</w>
<w>
suoritettava
<lemma tags="|PCP1|PSS|POS|NOM|SG|">suorittaa</lemma>
<lemma tags="|A|POS|NOM|SG|">suoritettava</lemma>
</w>
<w>
valtiolle
<lemma tags="|N|ALL|SG|">valtio</lemma>
</w>
<w>
alkoholi-
<lemma tags="|N|NOM|SG|">alkoholi-</lemma>
```

```
</w>
<w>
ja
<lemma tags="|COORD|C|">ja</lemma>
</w>
<w>
alkoholijuomaveroa
<lemma
tags="|N|PTV|SG|"><segm>alkoholi</segm><segm>juoma</segm><segm>vero</s
egm></lemma>
</w>
<w>
sen
<lemma tags="|DEM|PRON|GEN|SG|">se</lemma>
</w>
<w>
mukaan
<lemma tags="|ADV|">mukaan</lemma>
<lemma tags="|PSP|ILL|">mukaan</lemma>
</w>
<w>
kuin
<lemma tags="|N|INS|PL|">kuu</lemma>
<lemma tags="|CMPR|C|">kuin</lemma>
</w>
<w>
tässä
<lemma tags="|DEM|PRON|INE|SG|">tämä</lemma>
</w>
<w>
laissa
<lemma tags="|N|INE|SG|">laki</lemma>
<lemma tags="|N|INE|PL|">laki</lemma>
</w>
<w>
säädetään
<lemma tags="|V|PRES|PSS|PE4|">säätää</lemma>
</w>
<punct>.</punct>
</s>
</p>
<p>
<s>
<w>
2
</w>
<w>
momentti
<lemma tags="|N|NOM|SG|">momentti</lemma>
</w>
<w>
on
<lemma tags="|COP|V|PRES|ACT|SG3|">olla</lemma>
</w>
<w>
kumottu
<lemma tags="|V|PAST|PSS|NEG|">kumota</lemma>
<lemma tags="|PCP2|PSS|POS|NOM|SG|">kumota</lemma>
<lemma tags="|PCP2|PSS|A|POS|NOM|SG|">kumottu</lemma>
</w>
```