Ohjelmistoagenttiteknologian hyödyntäminen tiedonhaussa

Jiri Uitto

Pro Gradu –tutkielma
TAMPEREEN YLIOPISTO
Informaatiotutkimuksen ja
interaktiivisen median laitos
Tampereen yliopisto

TAMPEREEN YLIOPISTO

Informaatiotutkimuksen ja interaktiivisen median laitos
UITTO, JIRI: Ohjelmistoagenttiteknologian hyödyntäminen tiedonhaussa
Pro Gradu –tutkielma, 87 s., 11 liites.
Informaatiotutkimus

Toukokuu 2010

Tämän opinnäytetyön tarkoituksena on tutkia ohjelmistoagenttien mahdollisuuksia toimia tiedonhaun välittäjäjärjestelminä. Agenttiteknologiaan pohjautuvat tiedonhaun välittäjäjärjestelmät toimivat itsenäisesti ja auttavat tiedontarvitsijaa tiedonhankinnan ongelmissa. Opinnäytteessä selvitetään kuinka luoda tiedonhakuagentille mahdollisuus autonomiseen toimintaan, oppimiseen ja päättelyyn suhteessa sille annettuun tiedonhakutehtävään.

Työ alkaa ohjelmistoagenttien määrittelymallien esittelyllä ja jatkuu tiedonhakuagentin suunnittelun ja tiedonhakumallien käytön kuvaamisella. Ajatuksellisena lähtökohtana työssä on tekoälytutkimus ja erityisesti John Searlen ajatusmallit tekoälytutkimuksen alueella. Agentin suunnitteluteoria pohjautuu Wooldridgen & Jenningsin ja Caglayanin & Harrisonin esittelemiin ohjelmistoagenttien suunnitteluperiaatteisiin.

Työ yhdistää informaatiotutkimuksen, filosofian ja tietojenkäsittelytieteen teorioita muodostaen suunnittelumallin, jonka avulla voidaan luoda tiedonhaun tietämyspohjainen välittäjäjärjestelmä, tiedonhakuagentti. Tämä tiedonhakuagentti on autonominen ja älykäs agentti, joka toimii tiedontarvitsijan tekstinkäsittelyohjelman yhteydessä ja osaa poimia hakutermejä kirjoitettavasta tekstistä. Aika ajoin hakuagentti suorittaa hakuja tiedonlähteisiin, arvioi tuloksia, laajentaa hakua ja kommunikoi tuloksen tiedontarvitsijalle.

Tutkimus pyrkii vastaamaan seuraaviin kysymyksiin:

- Minkälaisia haasteita nousee esiin suunniteltaessa agenttiteknologialla toimivaa, tekstinkirjoittajaa avustavaa reaaliaikaista tiedonhaun välittäjäjärjestelmää?
- Mitä vaatimuksia tiedonhakutehtävä asettaa agenttiteknologialle?
- Mitä tiedonhakututkimuksessa kehitettyjä menetelmiä voidaan soveltaa agenttipohjaiseen tiedonhakuun?

Tiedonhakumetodien valinnassa on pyritty yksinkertaisesti toteutettavaan, kieliriippumattomaan kokonaisuuteen. Työ esittää agentin rakenteen koko tiedonhaun prosessin osalta, alkaen käyttäjän dokumentista ja päättyen palautteen vastaan ottamiseen ja haun laajentamiseen.

Abstract

The idea of this thesis is to evaluate information retrieval agents that work autonomously and help users in problems of information seeking. This thesis concentrates on the defining, planning, evaluating and selecting information retrieval methods to be used with agent technology. The starting point is artificial intelligence research in general and John Searle's work in that area in particular. The main challenge that arises from the combination of software agent technology and artificial intelligence research is how to create an autonomous, socially aware and intelligent information retrieval agent.

In this thesis, the theory behind the design of the software agent is based on the studies of Wooldridge & Jennings and Caglayan & Harrison. The approach of this thesis is different from the previous information retrieval mediator systems design, but the goals are still the same: provide the user with documents, while hiding the complex information retrieval process.

Based on information studies, philosophy and computer science, this thesis combines theories to form design principles that would be most beneficial when creating user-centric mediator systems with the aid of software agent technology. The agent described in this thesis is an autonomous agent built inline with a text editor and able to search new reference documents to the user. The assumption is that such an agent can be created, and when given source data from the user's document, it should be able to create an understanding on the given text, form a query and learn from the query results.

The questions guiding this thesis are as follows:

- What are the challenges when designing a real-time information retrieval application, which is based on agent technology?
- What are the requirements set by the information retrieval task to agent technology?
- What methods of information retrieval science can be applied to agent technology based information retrieval?

The selection of the information retrieval methods aimed for an efficient, language independent implementation the agent as the thesis tries to represent the creation of a mediator system end-to-end, from user input to the query expansion.

Sisällysluettelo

1 Johdanto

Tutkimuksen tavoite on arvioida agenttiteknologialla toteutettujen autonomisten tiedonhakuvälineiden määrittely-, suunnittelu- ja toteutusmahdollisuuksia. Työn ensimmäisessä osassa esitellään agenttien luokittelu ominaisuuksien mukaan ja taustalla olevia teorioita. Toisessa osassa keskitytään tiedonhakuagentin erityisominaisuuksiin. Lopuksi esitellään koko työn pääajatus eli miten yhdistää informaatiotutkimuksen tutkimustuloksia agenttiohjelmistoon. Esiteltyjä informaatiotutkimuksen keinoja ovat esimerkiksi Zipfin laki sanojen esiintymisfrekvenssistä luonnollisessa kielessä (Salton & McGill, 1983), tf.idf -kaava termien esiintymisestä dokumentissa suhteutettuna yleisyyteen dokumenttikokoelmassa (Salton, 1989), Kwokin teoria hakutermien painottamisesta niiden suhteellisen yleisyyden mukaan dokumentissa ja dokumenttikokoelmassa (Kwok, 1996), Kwokin teoriaa vastaava RATF-kaava ja sen soveltaminen kieltenvälisessä tiedonhaussa,(Pirkola et al., 2002) ja haun laajentamiseen käytettävä F4-kaava ja siitä kehitetyt versiot (Efthimiadis 1993).

Työn kantavana ajatuksena on, kuinka tekstinkäsittelyohjelman yhteyteen rakennettu tiedonhakuagentti toteutettaisiin yhdistellen näitä teorioita. Lopputuloksen tulisi olla käyttäjää tukeva, tekstinkäsittelyohjelmassa automaattisesti toimiva agentti, joka osaa ehdottaa kirjoittajalle mahdollisia uusia lähdedokumentteja. Agentti luo itsenäisesti hakulauseen, kehittää sitä, esittää haun lopputuloksen käyttäjälle ja kehittää hakua edelleen käyttäjän palautteen perusteella. Hakuagentin tulee olla mahdollisuuksien mukaan kieliriippumaton, mutta samalla sen tulee myös ymmärtää lähdetekstin rakenne tarpeeksi tarkasti hyvien hakulauseiden luomiseksi.

Tutkimus pyrkii vastaamaan seuraaviin kysymyksiin:

- Minkälaisia haasteita nousee esiin suunniteltaessa agenttiteknologialla toimivaa, tekstinkirjoittajaa avustavaa reaaliaikaista tiedonhaun välittäjäjärjestelmää?
- Mitä vaatimuksia tiedonhakutehtävä asettaa agenttiteknologialle?
- Mitä tiedonhakututkimuksessa kehitettyjä menetelmiä voidaan soveltaa agenttipohjaiseen tiedonhakuun?

Nykyisten tiedonhakuagenttien toiminta toteuttaa harvoin agenttien suunnitteluperiaatteita. Vielä harvempi toimii informaatiotutkimuksen kautta löydettyjen tiedonhakuteorioiden mukaan. Käyttäjän kanssa interaktiossa toimivia aitoja tiedonhakuagentteja ei myöskään ole juuri löytynyt, vaikka Internet-tiedonhakuvälineiden yhteydessä puhutaan paljon agenteista, jotka väsymättä keräävät tietoa hakukoneen tietokantoihin. On totta, että tässä on kyse eräänlaisista tiedonhakuagenteista, mutta nämä agentit on kuitenkin usein suunniteltu tiedonkeräämiseen, eivätkä ne ota kantaa itse tiedonhakuprosessiin. Nämä agentit siis vain keräävät tietoa sisällöstä riippumatta tietokantoihin ja jättävät hakuprosessin käyttäjälle.

Tässä tutkimuksessa tiedonhakuagentti pyritään suunnittelemaan toisesta lähtökohdasta: agentin tulee kerääjän sijasta olla noutaja. Kokonaisuutena työn kantava idea on etsiä mahdollisuuksia uusille tiedonhankintaa helpottaville työvälineille, mallintaa tiedonhaun teoriat ohjelmalliseen muotoon ja tehostaa agentin ja tiedontarvitsijan vuorovaikutusta. Tiedon kerääminen kantoihin jääköön tietojenkäsittelytieteen huoleksi; tämä tutkimus suunnittelee itse tiedonhaunprosessin agenttiohjelmistoksi. Lyhyesti sanottuna, tiedontarvitsijan pitäisi pystyä hakemaan tietoa agentin avulla, käyttäen normaaleja ihmisen ymmärtämiä tiedonesitystapoja. Kun työssä esitetään suunnittelumalli, otan lopussa kantaa myös agenttien kommunikaatioprotokollan suunnitteluun. Omasta näkökulmastani työ yhdistää siis osittain käyttäytymispsykologiaa, tietojenkäsittelytiedettä ja informaatiotutkimusta.

Agentin suunnittelun ideologinen lähtökohta löytyy John Searlen esittämästä kiinalaisen huoneen ongelmasta (Searle 1980, 417-418). Searlen teoria on mallina tekoälylle, jonka pyrin agenttiin luomaan käyttäen tiedonhakututkimuksen teorioita. Agentin toteutus perustuu paljolti valmiiden tietoteknisten ratkaisujen yhdistämiseen. Agenttiteknologian tutkimus on tähän asti selkeästi perustunut tietojenkäsittelytieteeseen, joten tiedonhaun tutkimuksella tuntuu olevan selkeä tilaus. Tiedonhakututkimuksellisena tämä tutkimus ei esittele agentin lähdekoodin toteutusta tai valmista toimivaa agenttia, vaan pyrkii esittämään tavan, jolla informaatiotutkimuksen tulokset voidaan toteuttaa agenttiteknologiassa. Työn ohessa on toteutettu sekä termien käsittelyyn että käyttöliittymään sopivaa ohjelmakoodia, mutta toteutus on silti tässä työssä toisarvoisessa asemassa: se on vain väline, jolla pyritään todistamaan teknologian mahdollisuudet.

2 Agenttiteknologian ja välittäjäjärjestelmien tutkimuksesta

Tiedonhaun välittäjäjärjestelmiä on tutkittu jo 1980-luvulla. Silloin näitä järjestelmiä kutsuttiin tiedonhaun älykkäiksi välittäjäjärjestelmiksi ja myöhemmin tietämyspohjaisiksi välittäjäjärjestelmiksi. Tietämyspohjainen välittäjäjärjestelmä kuvaakin erinomaisesti agenttiteknologialla toteutetun järjestelmän toimintaa ja viittaa tarpeeseen luoda välittäjäjärjestelmälle ensin tietoisuus tehtävästä, ei ainoastaan tekniset edellytykset tehtävän suorittamiseen. Välittäjäjärjestelmät voivat toimia tiedonlähteen tai aihepiirin mukaisena tai yleiskäyttöisenä. Tämän työn agentti pyrkii toimimaan aihepiiriltään yleiskäyttöisenä kieliriippumattomana agenttina, jonka tehtävä on tukea tiedonhakua tekstinkäsittelyohjelman yhteydessä.

Työssä esittelemäni agenttiteknologiateoriat käsittelevät samoja tietämyksen ongelmia. Agenttiohjelmisto voidaankin nähdä tekoälyntutkimuksena alana, jolle on asetettu usein välittäjänä toimiva rooli. Tästä syystä agenttiteknologia sopii hyvin tiedonhaun välittäjäjärjestelmien toteutukseen.

Välittäjäjärjestelmät ovat haasteellinen tutkimusalue, sillä on hyvin monimutkaista suunnitella kaikki tilanteet huomioon ottava tekoälyjärjestelmä ihmisen ja koneen välille. Kansainvälisesti yksi tunnetuimpia välittäjäjärjestelmien tutkijoita on Peter Ingwersen, joka esitteli Mediator-mallin kuvaamaan kuinka suunnitella tiedonhaun välittäjäjärjestelmä (1992, 8.1). Vaikka tämä työ ei seuraakaan Ingwersenin mallia, voidaan kuvatusta prosessista huomata yhtäläisyyksiä Mediator-malliin. Tämän työn esittelemä hakustrategian suunnittelu (ks. luku 4.2) on hyvin samantyyppinen kuin Ingwersenin (1992):

- Lähteen valinta
- Termien valinta
- Hakustrategian valinta
- Kyselyn rakentaminen
- Haun suorittaminen

Agenttiteknologian teoriat, joita työssä esitellään periytyvät 1990-luvulta. Tarkasteltuani aihetta nykyisten tutkimusten valossa havaitsin, että taustalla olevat jaottelu ja suunnitteluteoriat ovat pitkälti samoja. Esimerkiksi Wooldridge on sittemmin soveltanut alkuperäisiä teorioitaan agenttien ominaisuuksista (Wooldridge & Jennings 1995) ja kuvaa kuinka semanttisen verkon yhteydessä voidaan käyttää proaktiivisesti toimivia, sosiaalisia ja autonomisia agentteja (Dickinson & Wooldridge 2003). Tutkimuksessa agenttien suunnitteluperusteiden taustalla oleva teoria oli kuitenkin sama kuin vuonna 1995 esitelty.

Agenttien määrittelyn kulta-aika sijoittuikin 1990-luvulle ja esimerkiksi Nwanan (1996) artikkelia kutsutaan tieteenalalla jo klassikoksi. Uutta tutkimusta tullut myös 2000-luvulla, mutta ehkä johtuen tietojenkäsittelytieteen luonteesta tämä materiaali on kuitenkin suurelta osin agenttien toiminta-alustoja kuvaavaa, eikä niinkään agenttien tekoälyn luonnetta kuvaavaa. Esimerkiksi UMBC:n (The University of Maryland, Baltimore County) agenttisivut (UMBC 2005) ja FIPA (Foundation for Intelligent Physical Agents) esittelevät edelleen joko näitä 1990-luvun lopun klassikoita tai uutena materiaalina teknisiä kuvauksia agenttien alustoista. Myös alan oppilaitokset (mm. Tampereen yliopiston tietojenkäsittelytieteen laitos) viittaavat yhtenä pääteoksena jo 1997 ilmestyneeseen ”Agent sourcebook”-teokseen (Caglayan & Harrison 1997), joka on yksi tämänkin työn pää lähteistä.

Tämän työn aihepiiriä vastaavaa tutkimusta löytyy esimerkiksi 1980-luvulta (Sormunen 1989). Toisin kuin agenttien suunnitteluperusteiden tutkimus, tiedonhaun tietämyspohjaisten välittäjäjärjestelmien tutkimus on edennyt myös 2000-luvulla. Uudempaa tiedonhaun välittäjäjärjestelmätutkimusta edustaa esimerkiksi Järvelinin, Kekäläisen & Niemen ExpansionTool (2001). Sormusen (1989) tutkimus tuo mielenkiintoisesti esiin, miten tiedonlähteiden käyttöliittymät olivat vielä 1980–1990 lukujen taitteessa hankalia ja vaativat teknistä osaamista ja avustajaa suoritettaessa tiedonhakua. Vaikka maailma on sittemmin muuttunut ja internet Googleineen tehnyt tiedonhausta arkipäivästä, on tiedonhaun välittäjäjärjestelmille silti oma paikkansa. Mielestäni nyt välittäjäjärjestelmien ei tulisi pyrkiä poistamaan ainoastaan käyttöliittymien monimutkaisuutta, vaan tuottamaan lisäarvoa myös itse hakuprosessiin. Kuten agenttiohjelmistojen tutkimuksessa ovat tietyt perusasiat säilyneet myös tiedonhaussa. Tiedonhaun prosessi etenee myös tässä työssä Ingewersenin (1992, 8.1) tai Sormusen (1989, 11) kuvaamalla tavalla. Tiedonlähteiden valinnan jälkeen tulee valita hakustrategia ja hakutermit, luoda hakulogiikka, suorittaa haku ja kommunikoida tulos. Hakuketjun aikana jonkun, joko tiedonhakijan

tai välittäjäjärjestelmän tulee tehdä päätöksiä perustuen tietämykseen ja tavoitteeseen (Sormunen 1989, 10), tässä työssä kuvatussa järjestelmässä nämä päätökset ja arvioinnit pyritään toteuttamaan agentin tekoälyyn. Huomattavaa on myös miten tarvittavien eri ominaisuuksien kuvaus on samantapaista välittäjäjärjestelmien ja agenttien puolella. Kommunikaatiotarpeet, tietämys kielenrakenteesta ja käytettävyysvaatimukset (Sormunen 1989, 25, 56-59), eivät sinänsä ole suoraan agenttien ominaisuuksia korkealla tasolla, mutta kylläkin suoraan tiedonhakuagentin ominaisuuksia.

ExpansionTool (Järvelin et al. 2001) kuvaa kuinka tiedonhakijaa voidaan tukea luomalla ja laajentamalla automaattisesti hakuja konseptitasolta lingvistiselle tasolle ja aina hakutermien tasolle asti. Siinä missä ExpansionTool menee hyvin syvälle itse tiedonhakuun ja tiedonhaun prosessiin, tässä työssä keskitytään enemmän käyttöliittymätason problematiikkaan ja tiedontarvitsijan (kirjoittajan) tukemisen kuvaamiseen alusta loppuun agenttiteknologian keinoin. ExpansionToolin kaltaisen järjestelmän integroiminen osaksi kuvattua agenttia on silti varsin kiinnostava ajatus ja ehkä aiheellinen jatkotutkimusalue.

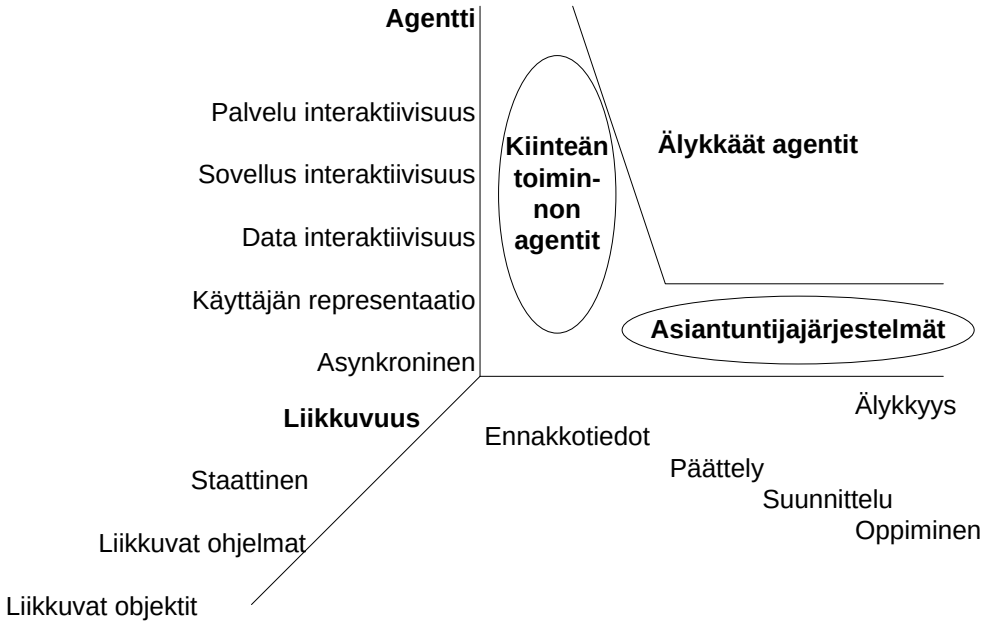
Tässä työssä suunnittelun teoriapohjaksi valittiin yleinen ohjelmistoagenttiteoria, jolla tiedonhaun tietämyspohjainen välittäjäjärjestelmä pyritään toteuttamaan siten, että sen hakutoiminnot käyttävät hyväkseen tiedonhakututkimuksen tuloksia.

3 Agenteista

Tarve agenttiteknologialle lähtee tietojenkäsittelyn historiasta. Alun perin tietokoneet suunniteltiin suorittamaan eräajotyyppisiä toimintoja, käytännössä reikäkorttien lukua. Reikäkorttien lukuun on vaikea keksiä tapoja, joissa tietokone voisi toimia autonomisesti, tehden valintoja teoissaan ja oppien virheistään. Klassinen ja nykyinenkin tietojenkäsittely perustuu pohjimmiltaan lineaarisuuteen. On olemassa vain suorittavia käskyjä, jotka jo valmiiksi sitovat suorituksen tiettyyn kaavaan. On siis olemassa yksiselitteinen komento: ”Lue kaikki reiät kortista ja siirry seuraavaan korttiin.” Sama yksiselitteinen komento voitaisiin esittää toisessa muodossa seuraavasti: ”Lue reikiä korteista kunnes niissä oleva tieto on selvinnyt.” Jälkimmäinen komento antaa tulkinnan varaa, sillä kaikkia kortteja ei tarvitse lukea, vaan tiedon selvittäminen riittää. Jälkimmäinen komento sisältää myös mahdollisuuden tehokkaampaan tai ainakin nopeampaan käsittelyyn, kun koko korttimäärää ei tarvitse lukea. Agenttiteknologian tavoite perustuu tietojenkäsittelyn tehostukseen: kun päätelmät ja ratkaisut tehdään johdonmukaisemmin, on lähtökohdan ja tuloksen välinen aikajana lyhyempi (Wooldridge & Jennings 1995, 3.1.1.2). Jälkimmäinen komento esittelee myös pääongelman, johon nykyisessä tietojenkäsittelyssä törmätään: kun käsitellään nimenomaan informaatiota, ei useinkaan tiedetä kuinka käsittelyssä tulisi edetä. Oikeastaan tarkkaan tiedetään vain tavoite. Tarkemmin sanottuna sovellukselle halutaan antaa vain tehtävä, suoritustavalla ei ole merkitystä lopputuloksen kannalta. Tämäntyyppisen toiminnallisuuden omaavaa ohjelmistoa sanotaan agentiksi. (Wooldridge & Jennings 1995, 4-6)

Agentti on ohjelmisto, joka on sidottu tiettyyn ympäristöön, kuten internetiin. Tässä ympäristössä agentti kykenee autonomiseen toimintaan, esimerkiksi tiedonhakuun, sille määritetyn ympäristön rajoissa. Agentti voi oppia toimintamalleja ympäristöstä, mutta se ei ole mikään vaatimus agentin toiminnalle. Samoin olen esitellyt aikaisemmin, että agentin on oltava autonomisesti toimiva. Se ei ole kuitenkaan välttämätöntä kaikissa tilanteissa. Luettuani Kluschin (1999) ja Wooldridgen & Jenningsin (1995) artikkelit tulin siihen tulokseen, että melkein mikä vain voi olla agentti. Agenttiterminologian käyttöä tulee evaluoida terveellä järjellä. Esimerkiksi valokatkaisijan voidaan nähdä olevan hyvin yhteistyökykyinen agentti, joka välittää virtaa aina lamppuun parhaalla katsomallaan tavalla, kun se uskoo, että me haluamme tehdä niin. Koska kuitenkin tiedämme valokatkaisijan rakenteen voidaksemme antaa sille ja sen käyttäytymiselle yksinkertaisemman mekaanisen selityksen, ei agenttiterminologiasta ole hyötyä. (Shoham 1993, 53) Näkemyksenä

työssäni on, että jotkin agenttiin liittyvistä ominaisuuksista ovat tärkeämpiä kuin toiset. Eräät ominaisuudet, esimerkiksi autonomisuus, ovat niin tiukasti sidoksissa agentin käsitteeseen, että agentin kuin agentin tulee jollain tasolla toteuttaa nämä ominaisuudet, jotta sitä voitaisiin kutsua agentiksi. Seuraavassa kuvassa on esitelty agentin perustoiminnallisuudet.



Kuva 1. Agentin perusominaisuuksien jakaminen akseleille (ks. Bradshaw 1997, 9 ja Intelligent Information Agents SIGin www-sivut http://www.dbgroup.unimo.it/IIA/softagents.html).

Agentin perusominaisuudet voidaan jakaa kolmeen haaraan: älykkyys, sosiaaliset kyvyt ja liikkuvuus. Älykkyyteen kuuluu toiminnassa esiintyvä oppiminen, suunnitelmallisuus ja reagointikyky. Toiseen perusominaisuuteen, sosiaalisuuteen, kuuluu kyky kommunikointiin käyttöliittymän kautta, kyky kommunikointiin ja kyky yhteistoiminnallisuuteen jonkin muun olion, kuten ihmisen, kanssa. Kolmas perusominaisuus liittyy paikkasidonnaisuuteen. Mielestäni paikkasidonnaisuus on lähinnä verkkoagenttien ominaisuus, ja liittyy niiden kykyyn siirtyä verkon sisällä paikasta toiseen. Agentin agenttimaisuus kasvaa, mitä pidemmälle kuvan 1 akseleilla liikutaan.

Yhteistoiminnallinen oppiva agentti, joka pystyy liikkumaan paikasta toiseen ympäristönsä rajoilla, lienee tavoitteellisin agentti ainakin tekoälytutkimuksessa. Agentilla voi olla myös mahdollisuus

sulautua muihin agentteihin ja muodostaa näin yhä paremmin toiminnallisia kokonaisuuksia. Tässä tutkimuksessa kyseistä ominaisuutta sivutaan, mutta sitä ei ole toteutettu tämän työn esittämässä ratkaisussa.

On tärkeätä huomata, että agenttien ominaisuudet eivät ole representaatioita ihmisten toiminnoista, vaikka agentit pyrkivät tietyissä tehtävissä toimimaan ihmisten tavoin tietojenkäsittelytieteen välinein. Vaikka agentin ominaisuuksien nimet ovat samoja kuin ihmisen ominaisuuksien nimet, niiden takana oleva toiminta perustuu kuitenkin pohjimmiltaan lineaarisuuteen (Magedanz 1997, 27-29). Tiedonhakuagentin suunnittelun kannalta agentin ominaisuuksien ymmärtäminen on tärkeää. Ominaisuudet määrittävät agentin kokonaisuutena.

3.1 Agenttien luokittelu toiminnan mukaan

Agentteja luokitellaan paitsi ominaisuuksien myös toiminnan ja toimintaympäristön perusteella. Esittelen seuraavassa erilaiset agenttityypit luokiteltuna kolmeen perustyyppiin toiminnan ja toimintatyypin mukaan, yhdistäen Caglayanin, Nwanan ja Haverkamp & Gauch teorioita (Caglayan & Harrison 1997, 7-12, 47-71, Nwana 1996, 5 ja Haverkamp & Gauch 1998, 306-310). Luokittelu ei tarkoita ettei agentti voisi kuulua useampaankin ryhmään, mutta se erottaa selkeästi osa-alueet, jotka asettavat vaatimukset agentin toiminnalle. Suunnittelun kannalta jako on tärkeä. Voisi sanoa että agentin suunnittelu on mahdotonta tai ainakin epäonnistuu, ellei suunnittelijalla ole mitään käsitystä ympäristöstä, jossa agentti tulee työskentelemään.

1. Työpöytäagentit (käyttöliittymäagentit)
   Työpöytäagentteihin kuuluvat kaikki käyttäjän kanssa interaktiossa toimivat agentit, jotka on toteutettu käyttöjärjestelmän tai ohjelman yhteyteen. Näillä agenteilla on ulospäin näkyvä käyttöliittymä ja ne toimivat suoraan käyttäjän kanssa yhteistyössä. Esimerkiksi Microsoft Wordin yhteydessä oleva avustava hahmo (usein paperiliitin) on eräänlainen käyttöliittymäagentti. Se tekee havaintoja käyttäjän työstä ja ehdottaa mahdollisia ratkaisuja. Samalla se oppii myös jos johonkin ratkaisuun ei haluta kuulla sen mielipiteitä. Tutkimuksen kohteena oleva tiedonhakuagentti olisi toteutettavissa parhaiten nimenomaan käyttöliittymäagenttina ja sen kanssa yhteistyötä tekevänä tiedonhakuagenttina.

2. Intranet-agentit

   Intranet-agentit ovat agenttiohjelmistoja, jotka automatisoivat toimintoja yrityksen tai muun suljetun alueen verkossa. Myös yritysten sisäiset asianhallintajärjestelmät sisältävät paljon toimintoja, joita voidaan kutsua agenteiksi. Tällaiset agentit palvelevat usein ohjelmistoja rajapintoina tietokantoihin tai välittävät töitä ihmiseltä toiselle sovelluksen tai sovellusympäristön sisällä. Intranet-agentteja on hyvin erilaisia, eivätkä kaikki niistä täytä agenttiohjelmiston määritteitä. Silti voi hyvällä syyllä sanoa, että intranet-agentit ovat yksi tehokas ja yleinen agenttiteknologian sovellusmuoto. Tiedonhakuagentteja ei sen sijaan tässä ympäristössä usein käytetä.

3. Internet-agentit

   Internetin agentit ovat agentteja, joista suurta osaa edustaa nimenomaan tiedonhakuagentit. Internet-agentit voidaan luokitella omiin alalajeihinsa seuraavasti: palveluagentit, tiedonsuodatusagentit, tiedonhakuagentit, liikkuvat agentit ja tiedottaja-agentit.

3.1. Palveluagentit

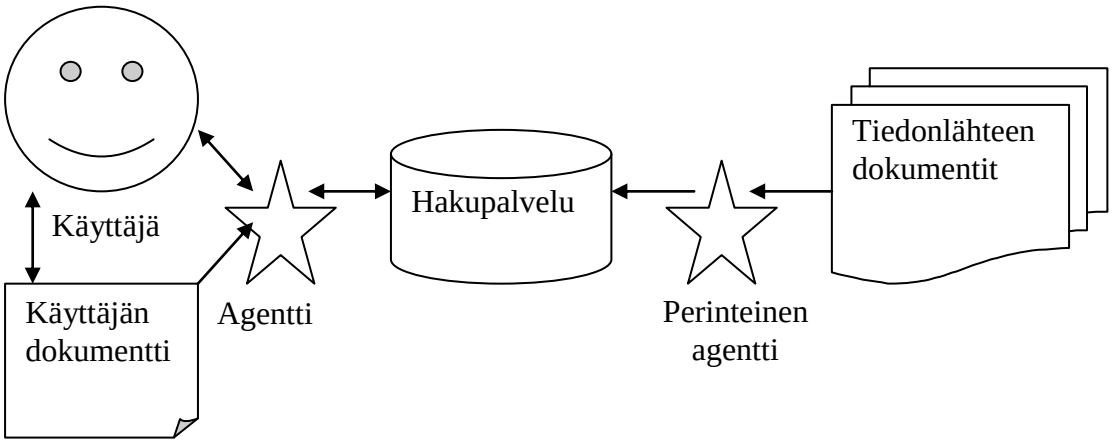
   Palveluagenttien päätarkoitus on tarjota jokin Internetissä sijaitseva palvelu käyttäjälle. Palveluagentit ovat usein myös käyttöliittymäagentteja eli ne ovat suoraan vuorovaikutuksessa käyttäjän kanssa. Tällainen agentti on esimerkiksi Googlen työkalupalkki, vaikka sen päätoiminnallisuus pohjautuukin Googlen hakukoneeseen ja Internetiin. Tutkimuksen kohteena oleva agentti toteuttaa joiltakin osin palveluagentin määrityksen. Tutkimuksen agentti nimittäin toimii linkkinä tiedonlähteen ja käyttäjän välillä ilman, että käyttäjän tarvitsee suoraan käyttää tiedonlähteenä olevaa palvelua.

3.2. Tiedonsuodatusagentit

   Joillakin agenteilla on mahdollisuus suodattaa tietoa käyttäjän antamien määritysten mukaan. Esimerkkejä tällaisista tiedonsuodatusagenteista ovat osaketietoja internetistä noutava agentti tai Amazon-verkkokaupan sivuille toteutettu hakutoiminto, jonka toiminta perustuu käyttäjän edellisiin ostoksiin. Tällöin agenttina toimiva ohjelmisto pyrkii tarjoamaan käyttäjälle vain ne tiedot, jotka käyttäjä haluaa (tai oletettavasti haluaa). Tällaisen agentin ohjelmisto on suunniteltu siten, että se tutkii tiedonlähteen tarjoaman tiedon sisältöä tai metadataa. Tämä erottaa tiedonsuodatusagentin muista tiedonhakuagenteista, jotka tarjoavat pääasiassa kokonaisia dokumentteja tai linkkejä dokumentteihin.

3.3. Tiedonhakuagentit

Tiedonhakuagentit hakevat käyttäjälle tietoa käyttäjän antamien määritysten mukaan. Käyttäjällä tarkoitetaan tässä toista ohjelmistoa, joka voi olla toinen agentti, osa toista agenttia tai hakupalvelu. Internet-tiedonhakuagentti on tunnetuin ja yleisin verkossa toimivan agentin muoto. Tiedonhakuagentteja ei kuitenkaan nimestään huolimatta sovi yleistää kaikkeen tiedonhakuun. Esimerkiksi tutkimuksen kohteena olevista agenteista toinen ei kuulu puhtaasti tähän kategoriaan vaan se on myös tiedonhakuagenttien käyttäjä ja käyttöliittymäagentti (Kuva 2.).



Kuva 2. Perinteisen tiedonhakuagentin, ja tutkimuksessa suunnitellun tiedonhakuagentin ero.

3.4. Liikkuvat agentit

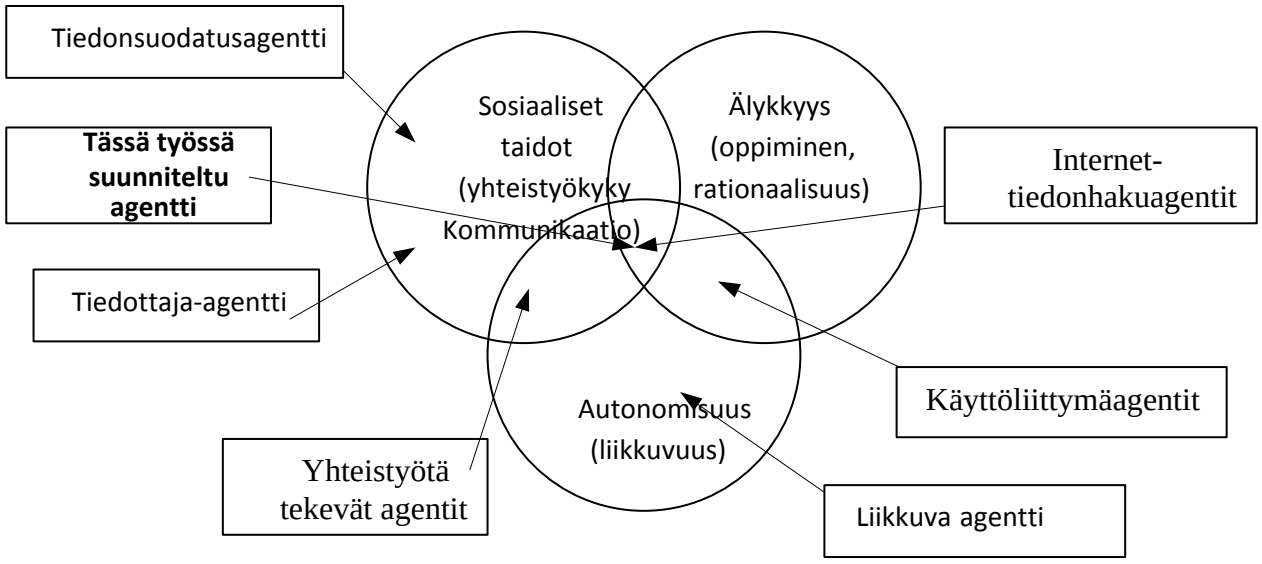
Liikkuvat agentit ovat määrityksen mukaan agentteja, jotka matkustavat paikasta toiseen suorittamassa käyttäjän antamia tehtäviä. Liikkuvan agentin käsite voidaan nähdä myös muiden agenttien ominaisuutena, joten sen esitys omana agenttityyppinään on sinänsä ongelmallista. Liikkuvuus on agenttiteknologissa hankala termi, sillä sitä käytetään sekä ominaisuuden että toiminnan kuvaamiseen. Hakupalveluille tietoa keräävät agentit ovat liikkuvia ominaisuuksiltaan.

3.5. Tiedottaja-agentit

Tiedottaja-agentit ovat yksi tiedonhakuagentin muoto. Useimmiten nämä agentit toimivat vahtipalveluna esimerkiksi asunnonhakupalvelussa ja ilmoittavat käyttäjälle, kun hänen määrityksiään vastaavia kohteita tulee myyntiin. Ero tiedonhaku ja tiedonsuodatusagentteihin on erittäin pieni. Tunnusomaista on kuitenkin usein liikkumattomuus ja yksinkertaisuus.

3.2 Agenttien luokittelu ominaisuuksien mukaan

Agentteja voidaan luokitella myös niiden ominaisuuksien mukaan. Luokitteleminen ominaisuuksien mukaan on kuitenkin varsin hankalaa, sillä harvaa agenttia voidaan luokittaa puhtaasti liikkuvaksi tai oppivaksi. Useimmiten agentit ovat yhdistelmä useita ominaisuuksia, joista sitten muodostuu kokonaisuus. Nwana luokittelee agentin seuraavan kuvan mukaisesti. Kuvan perusteella voidaan asettaa agentteja tiettyihin perusluokkiin, mutta kuten kuvasta käy ilmi, esimerkkinä käytetyt agentit asettuvat usein vähintään kahden perusominaisuuden väliin (Nwana 1996, 4.1)



Kuva 3. Agenttien jako perusominaisuuksien mukaan

Kuvasta on pääteltävissä, että mitä enemmän kuvatuista kolmesta ominaisuudesta agentti omaa sitä monimutkaisempi se on. Koska kuvan malli ei kuitenkaan ota kantaa siihen ympäristöön missä agentti toimii, kuvassa muun muassa Internet-tiedonhakuagentti ja tutkimustyössä konstruoitu agentti näkyvät samalla kohtaa ominaisuuksien suhteen.

Agenttien luokittelu ei siis ole ongelmatonta, mutta se on tarpeellista agentin suunnitteluvaiheessa, jotta voidaan tarpeeksi tarkkaan erotella eri agentille asetettavat vaatimuksen ja sitä kautta toteutukselle ja suunnittelulle asetettavat vaatimukset.

3.3 Yhteenveto

Luvussa 3 keskityttiin agentin luonteeseen ohjelmistoteknisenä oliona ja agentin luokitteluun niiden ominaisuuksien perusteella. Tunnusomaisia ominaisuuksia agenteille ovat kommunikaatio, älykkyys ja liikkuvuus. Samoja ominaisuuksia tai agenttien tehtäviä voidaan käyttää niiden luokittelussa. Agenttiteknologialla pyritään tiedonkäsittelyn tehostamiseen ja hajauttamiseen; tämän lisäksi agentit toteuttavat myös tekoälyyn liittyviä piirteitä. Seuraavassa luvussa esitetään mitä agentin suunnittelussa tulee ottaa huomioon ja mistä lähtökohdista tässä työssä esitelty agentti suunnitellaan.

4 Ohjelmistoagentin suunnitteluperiaatteet

Autonomisen agentin suunnitteluun liittyy paljon ongelmia, jotka eivät ole ratkaistavissa perinteisen tietojenkäsittelyn menetelmien avulla. Tiedonhakuagentin toteutuksessa kyse on tiedonhaun prosessin mallintamisesta ohjelmiston toiminnallisuuteen. Tiedonhaun prosessin mallintaminen on se varsinainen ongelma, sillä tietojenkäsittelyn maailma ei tunne käsitettä *sosiaalinen*. Tietojenkäsittelytieteen edustajat todennäköisesti pitäisivät näkökantaa lähinnä filosofisena, mutta käytännössä he ovat joutuneet kuitenkin toteamaan, että ohjelmointikielellä on helpompi toteuttaa lineaarista aikakäsitystä ja toimintaa kuin epälineaarista sosiaalista toimintaa. Jotta problematiikka tulisi selvemmin esille, esittelen John Searlen esittämän kiinalaisen huoneen nimellä kulkevan argumentin (1980, 417-418). Argumenttiin liittyvä ajatuskoe kulkee seuraavasti:

> Kuvitellaan englanninkielinen henkilö, joka ei osaa puhua eikä ymmärrä kiinaa. Hän on suljetussa huoneessa. Huoneessa on paperiliuskoja, joihin on painettu kiinankielisiä symboleja. Huoneen seinässä on kaksi luukkua. Toisesta luukusta henkilölle toimitetaan kiinankielisiä liuskoja ja toisesta otetaan vastaan liuskoja huoneessa olevalta henkilöltä. Lisäksi huoneessa on englanninkielinen ohjekirja.
>
> Henkilön tehtävänä on ottaa vastaan symboleita luukusta ja ohjekirjassa olevien neuvojen mukaisesti etsiä huoneesta toisia symboleja ja palauttaa ne luukusta. Luukusta tulevat symbolit ovat kiinankielisiä kysymyksiä ja henkilön palauttamat symbolit ovat vastauksia esitettyihin kysymyksiin.
>
> Olennaista on, että henkilö ei tiedä sisään tulevien liuskojen olevan kysymyksiä eikä ulos menevien olevan vastauksia. Ja vaikka tietäisikin, niin hän ei ymmärtäisi mitä niissä sanotaan. Ohjekirjassa olevat neuvot nimittäin eivät perustu liuskoissa olevien symbolien merkityksiin, vaan niiden muotoihin. Henkilö yhdistelee symboleja toisiinsa puhtaasti geometrisin kriteerein eli sen perusteella miltä symbolit näyttävät. (Searle 1980, 417-418)

Tässä ajatuskokeessa ohjekirjaan kirjoitetut säännöt vastaavat tietokoneohjelmaa. Ihminen ja huone paperiliuskoineen vastaavat tietokonetta. Luukuista huoneeseen tulevien liuskojen informaatio vastaa tietokoneelle annettavaa syötettä ja huoneesta ulos lähtevien liuskojen informaatiotulostetta. Näin henkilö toimii puhtaasti sääntöjen eli syntaksin perusteella. Vaikka hän vastaisi oikein

liuskoissa oleviin kysymyksiin ja näin läpäisisi niin sanotun Turingin testin, ei hänen kuitenkaan voida sanoa ymmärtävän hänelle esitettyjä kysymyksiä (Turing 1950). Turingin testin tarkoituksena on selvittää, onko tietokone älykäs vai ei. Älykkyyden kriteerinä on ”huijata” tietokoneen kanssa keskustelevaa ihmistä luulemaan, että hän keskustelee toisen ihmisen kanssa. Jos tietokone onnistuu huijauksessaan, läpäisee se testin ja on näiden kriteerien puitteissa älykäs. (Turing 1950, 433-460)

Ajatuskokeen tavoitteena on valottaa sitä, että puhtaasti syntaksin perusteella toimivat systeemit eivät pysty tuottamaan semantiikkaa. Henkilö huoneessa simuloi syntaktisesti toimivaa tietokoneohjelmaa; funktionaalisesti kuvattuna tietokoneella ohjelmineen ovat samat osat kuin ajatuskokeen kuvaamalla asetelmalla. Koska henkilö ei siis ymmärrä kiinaa, ei ole syytä olettaa, että tietokonekaan ymmärtäisi. Jos huoneessa sen sijaan olisi kiinaa puhuva henkilö, joka liuskoihin painettujen symbolien merkitysten perusteella hakisi vastauksia kysymyksiin, toimisi hän olennaisesti eri tavalla kuin ohjekirjaan tukeutuva esimerkiksi englanninkielinen henkilö tai tietokoneohjelma. Kiinalainen henkilö toimisi semantiikan perusteella. Toiminnan perusteissa on selkeä ero, sillä semantiikan perusteella toimiminen on tarkoituksenmukaista, kun taas syntaksin perusteella toimiminen ei. Syntaksin perusteella toimivista tietokoneista voidaan sanoa, että ”they don’t give a damn”, kun taas ihmiset toimivat tarkoitusten ja merkitysten perusteella. (Haugeland 1996, 660.)

Miten tämä kaikki liittyy sitten tiedonhakuun ja tiedonhaun agentteihin? Tiedonhakuagentin kohtaama ongelma on kouriintuntuvasti sama kuin Searlen henkilön. Agentti on kuin tuo henkilö kiinalaisessa huoneessa. Jotta agentti selviäisi tästä testistä, annan sille seuraavat ohjeet:

1. Nämä ovat yleisimmät sanat käytetyssä kielessä (painotettu sanakirja)
2. Nämä ovat tiedonhaun kannalta merkityksettömiä sanoja käytetyssä kielessä (esimerkiksi suomen kielessä *ja, sekä, eli, tai...)*

Sisään tulevat liuskat sisältävät kaikki lähdedokumentissa olevat sanat. Ulosmenevissä liuskoissa ovat kaikki merkitykselliset sanat järjestyksessä harvinaisimmasta yleisimpään. Lisäksi tulee ottaa huomioon lähdetekstin semanttinen rakenne. Semanttista rakennetta voi arvioida esimerkiksi

sanojen läheisyydellä. Sanojen läheisyyden analysoinnin jälkeen voidaan löytää merkityksellisiä sanapareja, joiden paino haussa on suurempi kuin yksittäisen termin.

Vielä lopuksi varsinainen argumentti, jota yllä oleva ajatuskoe tukee (Searle 1990, 27):

1. Programs are formal (syntactical)
2. Minds have contents (semantics)
3. Syntax is not sufficient for semantics

Searlea lainatakseni: “From these three propositions the conclusion logically follows: programs are not minds.” (Searle 1994, 546-547). Toisin sanoen, agenttia ei voi luoda ihmismäiseksi, se ei saa omaa ajatteluaan vaan seuraa aina edeltä ohjelmoituja toimintoja.

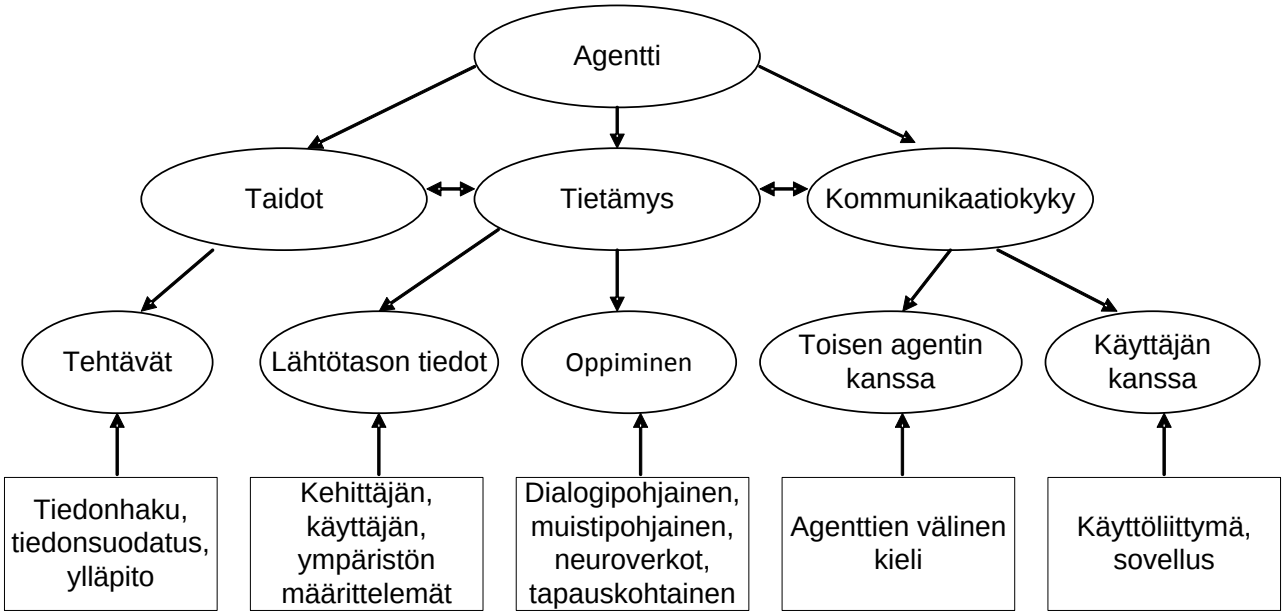
Tässä työssä esiteltävä, tavallaan dokumenttia lähteenä käyttävä agentti, on tiedonhaun kannalta varsin omituinen olio. Tekeehän se ensin analyysin lähdedokumenteista, sen jälkeen hakujärjestelmä tekee vastaavan analyysin haettavien aineistojen joukosta. Lopulta näitä tuloksia verrataan keskenään. Onko kyse oikeastaan siitä, että tiedonhaku kohdistetaankin käyttäjän dokumenttiin? Kuinka paljon tulokseen silloin vaikuttaa järjestelmien analyysimenetelmien samankaltaisuus tai erilaisuus?

4.1 Tiedonhakuagentin suunnittelu

Agentin suunnittelussa tulee ottaa huomioon:

- Ympäristö jossa agentti tulee toimimaan
- Tehtävät, joita agenttia vaaditaan suorittamaan

Nämä vaatimukset määrittävät ne ominaisuudet, joita agentilla tulee olla. Agentin määrittelystä on esittänyt mallin Caglayan & Harrison (1997, 3-12). Lisäksi suunnittelu pohjautuu ominaisuuksien puolesta Nwanan tutkimukseen (Nwana 1996, 4.1).



Kuva 4. Agentin määrittely sen ominaisuuksien mukaan (Caglayan & Harrison 1997, 7).

Mallissa agentin ominaisuudet on jaettu ensin kolmeen pääominaisuuteen: taidot, tietämys ja kommunikointikyky. Ominaisuudet ovat vuorovaikutuksessa toisiinsa. Taidot lisäävät tietämystä ja tietämys taitoja, samoin kuin kommunikointikyky lisää tietämystä ja tietämyksen lisääntyminen lisää kommunikointikykyä. Malli tarkentaa ominaisuudet vielä käytännön toiminnoiksi. Toimiakseen käytännössä, agentilla tulee olla tehtävä, lähtötasontiedot, opittutieto ja kommunikaatiokyky eri vastinpareja kohtaan.

Seuraavaksi esittelen agentin ominaisuudet, jotka liittyvät suoraan tiedonhaussa käytettäviin ominaisuuksiin. Tiedonhakuagentti on älykäs autonomisesti toimiva agentti, jolla on pääsy yhteen tai useampaan tiedonlähteeseen. Tiedonlähteet voivat olla heterogeenisiä ja sijaita missä tahansa agentin toimintaympäristön sisällä. Tiedonhakuagentti hankkii lähteistä tietoa, evaluoi löydettyjä dokumentteja, etsii mahdollisesti uusia tiedonlähteitä ja tarjoilee löydettyä aineistoa joko käyttäjälle tai toiselle agentille. Hyvin toteutettuna tiedonhakuagentti tutkii tietoa syntaktisella periaatteella, mutta käyttäen hyväksi tietoa luonnollisesta kielestä. Agentin tehtävänä on analysoida sille annettu dokumentti ja tarjota vastaavia dokumentteja omista lähteistään. Klusch (1999, 5) jakaa tiedonhakuagentin tehtävät seuraavasti:

1. Providing a pro-active resource discovery
2. Resolving the information impedance of information consumers and providers
3. Offering value-added information services and products to the user or other agents

Agentin tulee siis hallita monia eri tiedonlähteitä (1), saada selville tiedontarpeet (2) ja tarjota relevanttia tietoa asiakkaalleen (3).

Tiedonhakuagentti voidaan nähdä myös perinteisen tietopalveluhenkilön mallinnuksena. Esimerkiksi Nardi ja O'Day (1996, 86) päätyivät tähän näkemykseen tutkimuksessaan, jossa tietopalvelussa työskentelevä henkilö (tutkimus käyttää termiä librarian) pystyy parempaan tietopalvelutyöhön, koska hän pystyi personoimaan hakuja käyttäjien tarpeita vastaavaksi ja ottamaan huomioon myös sosiaalisen tilanteen. Tietopalvelutyöhön sisältyy siis yhtenä osana käyttäjän ymmärtäminen, jota agentilla on hyvin vaikea toteuttaa, mutta toisaalta myös Nardin ja O'Dayn tutkimuksen argumentti oli ”Jos agentti on suunniteltu oikein, se voi personoida teknologian käyttöä, luoden vallankumouksen käyttäjäkokemuksessa”. Juuri tähän samaan pyritään tämän työn agentilla, eli luomaan uusia ajatuksia tiedonhakuagenttien mahdollisuuksista.

Nykyisellä teknologialla tiedonhakuagentin on vielä mahdoton tehdä seuraavaa (Nardi & O’Day 1996, 80-83):

1. Puhua ja ymmärtää. Tietopalvelussa voidaan hakea aitoja hakuavaimia keskustelulla ja myös tietopalvelutyöntekijän oman kokemuksen avulla.
2. Lukea ja ymmärtää sisältöä. Agentti ei voi tietoisesti ymmärtää sisältöä, mutta kuten edellä mainitussa kiinalaisen huoneen teoriassa, sille voidaan opettaa tiettyjä malleja tekstin rakenteesta ja syntaksista. Sen sijaan tietopalvelussa työntekijä voi hyvinkin arvioida lähteen relevanssia semanttisin perustein vaikka ei olisikaan asiantuntija aihealueella.
3. Löytää yhteyksiä erilaisten tiedonlähteiden välillä. Semanttisen analyysin puute ja kykenemättömyys tekstien sisällöllisten yhteyksien arviointiin estää agentteja yhdistämästä esimerkiksi eri alojen tietoa. Vaikka agentilla olisi tietämystä, se ei ole ’sivistystä’.
4. Paperilähteet ja niiden analysointi. Esimerkiksi arkistot.
5. Pääsy rajoitettuun materiaaliin. Osaan materiaalista, esimerkiksi henkilötietoihin, ei ole koskaan turvallista sallia automaattista yhteyttä.

6. Arvioida automaattisesti tiedonlähteen laadukkuutta. Käyttäjän palautteen ja lähdeanalyysin perusteella agentti voi tehdä päätelmiä lähteen laadusta suhteessa aiheeseen, mutta ei perustuen kokemukseen.
7. Inhimillinen kosketus.

Mitä sitten saman tutkimuksen mukaan agentin pitäisi toteuttaa? Nardin ja O'Dayn tutkimuksen mukaan tiedonhakuagenttien tulisi ensinnäkin käyttää haussa tietoja käyttäjän kiinnostuksen alueesta, taustoista, tehtävästä ja saatavissa olevista resursseista (1). Mahdollisuuksien mukaan agentin tulisi työskennellä käyttäjän kanssa, eikä olettaa, että käyttäjä osoittaa suoraan mitä hän tarvitsee (2). Ohjelmistoagenttien tulisi myös kyetä etenemään tehtävässään, vaikka itse tehtävän kuvaus olisi vielä hatara. Esimerkiksi tämän työn agentin pitää toimia, vaikka lähdedokumentti olisi vielä alkutekijöissään (3). Ohjelmistoagentin pitää myös varautua siihen, että jossakin vaiheessa käyttäjä voi tehdä virheen ja ohjata agenttia haussa väärään suuntaan. Tällöin agentin tulisi kyetä arvioimaan uuden suunnan poikkeavan liikaa vanhasta ja jatkaa muuttamatta toimintaansa (4). (1996, 74-75)

Kun suunnitellaan ohjelmistoagenttia, Nardi ja O'Day (soveltaen 1996, 74-75) toivat tutkimuksessaan seuraavat yleiset suunnitteluperiaatteet esiin:

- Muista ns. hiljainen tieto, jota on niillä ihmisillä, joiden toimintaa mallinnetaan.
- Käytä hyväksesi kaikki ohjelmistoagentin mahdollisuudet, jotta voit kompensoida sen puutteita suhteessa kilpailijaan eli ihmiseen. Yksi ohjelmistoagentin vahvuus on agentin kyky käsitellä nopeasti valtavia tietomääriä.
- Älä oleta agentin suoriutuvan kaikesta, vaan paloittele toiminta pieniin osakokonaisuuksiin, joihin myös käyttäjä voi vaikuttaa.
- Huomioi, että agentin toiminta tulee suhteuttaa myös käyttäjän odotuksiin. Näin esimerkiksi haku ei voi kestää loputtoman kauan, vaan relevanteiksi oletetut lähteet on esitettävä käyttäjälle, vaikka haku ei olisikaan vielä lopussa ja eikä tulosjoukko olisi täydellinen.

Työssä suunniteltu agentti pyrkii noudattamaan yleisten agenttien suunnitteluperiaatteiden lisäksi myös näitä periaatteita. Seuraavaksi esittelen tiedonhakuagenttien ominaisuuksia perustuen edellisessä luvussa esittämiini agenttien yleisiin ominaisuuksiin.

4.1.1 Sosiaaliset taidot: Yhteistyökyky ja kommunikointi (collaborative, communicative)

Agentit voivat olla joko yhteistoiminnallisia toisten agenttien kanssa tai työskennellä itsenäisesti. Yhteistyö voi perustua toimintojen, tiedonlähteiden tai tiedon jakamiseen. Tavallisinta on tehtävien jako, jossa tiedonlähde jaetaan useampaan osajoukkoon. Jokaisesta osajoukosta vastaa oma agenttinsa. (Magedanz 1997, 27-29). Esimerkiksi internet hakukoneiden taustalla toimivat agentit kykenevät jakamaan Internetin sivustoalueisiin, joista jokaisesta vastaa oma agenttinsa. Yhteistyökyky vaatii agentilta kommunikointikykyä, sillä agentit, jotka eivät pysty kommunikoimaan keskenään, eivät voi tehdä yhteistyötä.

Sosiaalisia kykyjä pidetään usein ohjelmistoagenttien keskeisinä ominaisuuksina, juuri niinä jotka tekevät ohjelmistosta agentin. Kuitenkaan kommunikointikyky tai yhteistyökyky ei sinänsä luo itsessään agenttia. Toisin sanoen, vaikka ohjelmisto voi vaihtaa tietoa toisen ohjelmiston kanssa, ei se välttämättä ole sen enempää agentti kuin mikään muukaan ohjelmisto. Lähtökohtana tällaiseen ajatteluun lienee se, että agentit nähdään usein toimimassa nimenomaan avoimissa verkkoympäristöissä, kuten internetissä (Jain, Chen & Ichalkaranje 1999, 62). Laajoissa avoimissa ympäristöissä on sisäänrakennettu vaatimus hajautetusta toiminnasta. Tällöin agentin kommunikaatiokyky on yksi agentin tärkeimmistä ominaisuuksista, jotta työnjako olisi mahdollinen.

Tiedonhakuagentin tulee kyetä kommunikointiin. Tutkimuksessa suunniteltu tiedonhakuagentti kommunikoi kolmen tahon kanssa: tiedonlähteen (toisen agentin avustuksella), käyttäjän ja hakuavainten lähteenä olevan dokumentin kanssa. Kaikkien näiden kanssa agentin on kommunikoitava omalla tavallaan. Kommunikointia käyttäjän eli ihmisen kanssa voidaan pitää luonnollisimpana sosiaalisen kommunikaation muotona, kun taas kommunikointi dokumentin sisältävän ohjelmiston kanssa on kaikkea muuta kuin luonnollista kieltä, ainakin ihmisnäkökulmasta.

Tehokkaimmillaan, ja tietoverkoissa toimiessaan, tiedonhakuagenttien tulisi kommunikoida keskenään ja vaihtaa tietoa keskenään. Tätä varten tulisi olla olemassa yksinkertainen, mutta globaalisti yhteiseksi sovittu kommunikaatioprotokolla. Yhtä mahdollisuutta tällaiseen protokollaan

esitellään myöhemmässä luvussa 5.4, koska kyseinen kommunikaatiokyky voidaan nähdä yhtenä tärkeimmistä hakuagentin ominaisuuksista.

4.1.2 Älykkyys: Oppimiskyky

Oppivat agentit pystyvät havainnoimaan tiedonlähteissä tapahtuvia muutoksia ja muuttamaan toimintojaan muutosten perusteella (Magedanz 1997, 27-29). Yksinkertainen esimerkki tällaisesta toiminnasta on www-sivun saatavuus. Internetissä toimivien hakuagenttien tulee mukautua verkossa tapahtuviin muutoksiin: yhteydet katkeilevat, sivustot muuttuvat ja katoilevat. Voidaan sanoa, että agentin tulee kyetä reagoimaan tiettyihin tilanteisiin ja toisaalta tekemään johtopäätöksiä keräämiensä tietojen ja kokemusten perusteella.

Tiedonhakuagentin tulisi oppia, millaisista lähteistä käyttäjä on kiinnostunut. Tämä voi tapahtua joko käyttäjän toimintaa seuraamalla tai suoraa palautetta vastaanottamalla. Käyttäjän toimintaa agentti voi seurata esimerkiksi tallentamalla tietoja siitä, mitä hakutuloksessa olevia lähteitä käyttäjä katselee. Tähän sisältyy kuitenkin myös ongelmia, sillä käyttäjä saattaa jättää avaamatta lähteet, jotka ovat jo tuttuja, mutta silti relevantteja, ja toisaalta avata tuntemattomia mutta epärelevantteja lähteitä. Näistä syistä agentin autonominen oppiminen on hyvä rajata tekstin analyysiin. Käyttäjän palautetta sen sijaan voidaan käyttää hyvinkin tehokkaasti oppimisen lähteenä. Käyttöliittymä täytyy suunnitella huolella sellaiseksi, että palautteenanto ei häiritse käyttäjää ja että kysymykset kysytään käyttäjän omalla kielellä ja käyttäjän ymmärrettävin käsitemallein. Lisäksi haun tehokkuuden lisäämiseksi hakuagentin tulisi voida kohdistaa haku sellaisiin tiedonlähteisiin, jotka näyttävät sisältävän relevanttia tietoa, ei epärelevanttia informaatiota sisältäviin. Esimerkiksi haettaessa lääketieteellistä materiaalia hakujen kohdistaminen vain lääketieteen tietokantoihin voisi rajoittaa hakua liikaa, mutta myöhemmin auttaa haun tarkentamisessa. Jos esimerkiksi halutaan tietoa urheilusta ja lääketieteestä, on hakulähteisiin valittava molempien aihealueiden tiedonlähteitä.

4.1.3 Älykkyys: Rationaalisuus

Agentin rationaalisuutta arvioidaan sen kyvyllä suorittaa tehtäviä suhteessa tavoitteeseen. Rationaalinen agentti toimii siten, että se arvioi toimintansa tuloksellisuutta hyötynäkökulmasta. Tällä tarkoitetaan esimerkiksi kykyä arvioida maksullista tiedonlähdettä edellisten kokemusten perusteella. Agentilla voi olla käytettävissään kaksi ilmaista tiedonlähdettä ja yksi maksullinen. Agentin tulee kyetä arvioimaan mahdollisimman kustannustehokas tapa hankkia tietoa, mutta

kuitenkin siten, että tiedonlähteistä saatava hyöty on maksimaalinen. (Klusch, 1999, 10) On olemassa myös hakuagentteja, jotka taltioivat maksullista tietoa ja vaihtavat sitä toisten agenttien kanssa. Tässä tapauksessa voi todeta agentin suorastaan myyvän tietoa hyvään hintaan.

Tutkimuksen kohteena olevan hakuagentin älykkyys perustuu sen kykyyn arvioida lähdedokumenttia ja siinä olevien termien yleisyyttä matemaattisin perustein. Agentti ei siis omaa kehittyvää päättelykykyä, mutta osaa kuitenkin analysoida sille annettujen sanalistojen perusteella mikä dokumentissa on tärkeää ja mikä ei. Agentin älykkyys on kuitenkin hyvinkin toteutettavissa antamalla agentille mahdollisuus oppia uusia termejä. Vaikka agentti toteuttaisi sanojen katkaisun ja niiden käyttämisen hakutermeinä, on agentin myös hyvä oppia uusia termejä, kuten eri alojen erityistermejä. Käyttöliittymätasolla näitä termejä voisi listata käyttäjälle listaan hyväksyttävistä termeistä. Lyhykäisyydessään agentti siis voisi rakentaa omaa tesaurustaan muun toiminnan ohella. Tätä mahdollisuutta on esitelty enemmän luvussa 5.3.

4.1.4 Liikkuvuus

Tiedonhakuagentti voi olla täysin järjestelmäsidonnainen tai sitten autonomisesti verkossa liikkuva. Autonomisesti liikkuvat agentit ovat yleisiä internetin kaltaisissa verkoissa, paikkasidonnaiset tämän työn esimerkkijärjestelmän kaltaisessa ympäristössä (Magedanz 1997, 27-29). Liikkuvuuteen liittyy myös kyky vaihtaa tietoa muiden agenttien kanssa. Agentin ei tarvitse itse liikkua tiedonlähteen luo, jos se pystyy menestyksekkäästi kommunikoimaan jo siellä käyneiden agenttien kanssa. Tässä tutkimuksessa esitelty agentti ei ole liikkuva, mutta se voi käyttää hyväkseen liikkuvia agentteja hakupalvelujen kautta.

Liikkuvuus on terminä hankala, ja syytä avata. Useimmiten agenttiteknologiassa pyritään fyysiseen liikkuvuuteen (vrt. tietokonevirukset), ja liikkuvuudella tarkoitetaan kykyä siirtää agentin sijaintia lähemmäksi kohdetta automaattisesti. Teknisesti tämä tarkoittaisi esimerkiksi toisen agentin toteutuksen siirtymistä verkossa aasialaisiin tiedonlähteisiin ja toisaalta taas toisen agentin siirtymistä palvelimelle, joka on lähellä eurooppalaisia lähteitä. Käytännössä osa agenteista on pakotettuja staattiseen toimintaan, sillä niiden toimintaympäristö ei välttämättä tue liikkuvuutta. Staattisen agentin toimintaympäristö voi olla esimerkiksi palvelin, jolle se on asetettu, mutta josta se ei voi poistua tai monistaa itseään toiselle palvelimelle.

Liikkuvien agenttien toimintaympäristönä on, joko internet tai sisäiset verkot. Liikkuva agentti ei nouda tarvitsemaansa tietoa verkosta vaan voi joko siirtyä tiedon luokse ja takaisin lähtöpisteeseensä tai prosessoida tieto tiedonlähteellä. Liikkuvuuden edellytyksenä on kuitenkin, että kohdejärjestelmä tukee agentin liikkuvuutta. Teknisesti ottaen tämä tarkoittaa, että agentin kohteena olevan ympäristön ja agentin tulee toimia samalla teknologialla.

Yksi mahdollinen teknologia on IBM:n kehittämä Aglets (IBM, 2002). Agletit ovat IBM:n kehittämä liikkuvien agenttien kehitysympäristö, joka perustuu avoimeen lähdekoodiin. Liikkuvat agentit on toteutettu Java-kielellä. Aglets on järjestelmänä suunniteltu siten, että liikkuvien agenttien toteutus on helppoa. Useat tiedonlähteet ovat kuitenkin vielä liikkuvien agenttien ulottumattomissa, johtuen arkkitehtuurivaatimuksista.

Kuten IBM:n www-sivuilla todetaan:

> "Q:Do I need special server software to use aglets?
>
> A:Yes, you will need to run the atp daemon (in the ASDK package) to receive incoming aglets over the network."

Ei ole kovin todennäköistä, että monikaan tiedonlähde tukisi Aglet- ympäristöä tai vaikka tukisikin, on epätodennäköistä, että kaikki ympäristöt tukisivat samaa palvelintoteutusta. Enemmän tietoa liikkuvien agenttien kehitysalustoista löytyy esimerkiksi Huntusin tutkimuksesta (Huntus, 1999).

Tämä tutkielma keskittyy staattisten agenttien suunnitteluun. Tämä valinta tuo mukanaan myös vaatimuksen hyvästä kommunikaatioprotokollasta, josta yksi esimerkki on esitelty myöhemmin luvussa 5.4.

4.2 Hakulogiikka ja agentti

Agentin sisältämän hakualgoritmien valinta pohjautuu seuraaviin olettamuksiin:

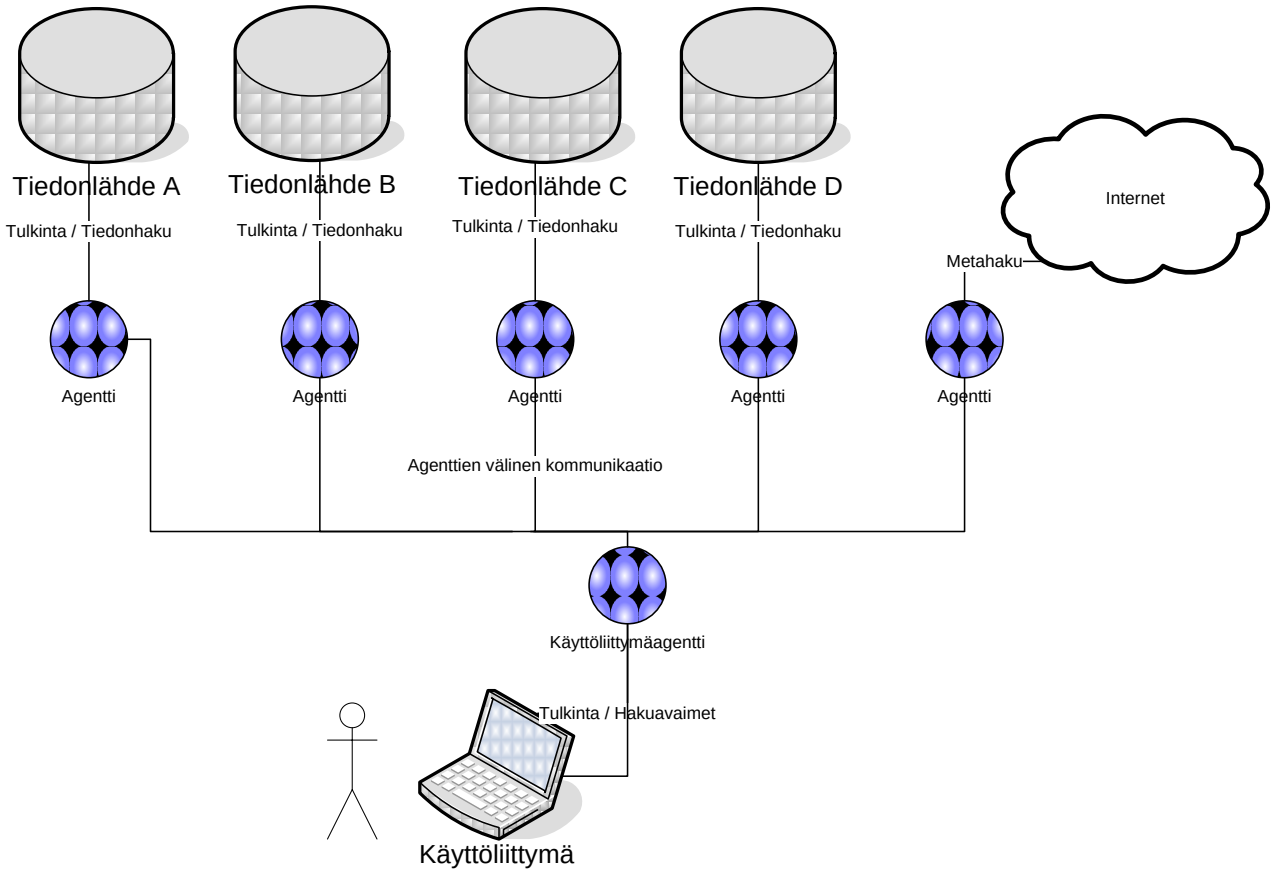
- Hakuja tulisi voida tehdä useammasta tietokannasta
- Hakuja tulisi voida tehdä käyttäjän valitsemilla kielillä
- Hakujen pitäisi tuottaa mahdollisimman relevantteja tuloksia
- Hakusanaston tulisi olla oppiva

Tiedonhakuagentit ovat tehokkaimmillaan tietoverkoissa, mutta niitä voidaan käyttää hyväksi myös useamman tiedonlähteen järjestelmässä. Useamman tiedonlähteen järjestelmässä agenttiteknologian valinnalla ei pyritä kuorman tasaamiseen, vaan enemmänkin erilaisten tiedonlähteiden integroimiseen yhdeksi. Tällöin agenttien liikkuvuus perustuu niiden sijaintiin lähellä tiedonlähdettä ja toisaalta kykyyn valita nopeasti saavutettavissa olevat tiedonlähteet. Joissakin tapauksissa useat agentit voivat hakea tietoja samoista tiedonlähteistä. Silloin kokoavan agentin, kuten tämän työn käyttöliittymäagentin, tulee kyetä poistamaan hakutuloksesta samat linkit. Tästä syystä myös käyttöliittymäagentin tulee tietää alkuperäinen lähde.

Jotta kyseinen integraatio toimisi mahdollisimman tehokkaasti, olisi agenttien toimittava ikään kuin tulkkeina tiedonlähteen käyttöliittymän ja tiedonhaun käyttöliittymän välillä. Sen sijaan, että haut keskitettäisiin yhden agentin tai agenttiryppään toimesta yhteen kohteeseen, kuten internetiin, haku kohdistettaisiin useampaan kohteeseen kuvan 5. mukaisesti, jossa agentit toimivat siis tulkkeina tiedonlähteen ja käyttäjän välillä, mutta myös toisinpäin. Agentteja on siis järjestelmässä kahdenlaisia:

1. Tiedonlähteeseen integroituvat agentit
2. Hakuavaimia toisille agenteille tuottavat agentit

Koska valittu rakenne eriyttää agentit toisistaan, on jokaiselle agentille luotavissa sen ominaispiirteet. Näitä ominaispiirteitä voivat olla tiedonlähteen käyttöliittymän ymmärrys, mutta myös tiedonlähteen sisältämien dokumenttien kieli.



Kuva 5. Agenttien sijainti ja kommunikaatio

Käyttöliittymäagentin on tarkoitus poimia käyttäjän syötteestä sopivia hakuavaimia hakua varten sekä kommunikoida käyttäjän kanssa, mutta myös valita hakuagenttien keräämästä aineistoista relevantimmat viitteet. Agenttien välillä tulee siis olla myös tapa kommunikoida sekä löydettyjen dokumenttien keskenään verrattavissa oleva dokumenttipaino että myös dokumenttien metatiedot. Koska jokainen tiedonhakuagentti on oma olionsa ja “ymmärtää” vain oman tiedonlähteensä rakenteen, juuri tämän agentin tulisi pystyä laajentamaan omaan synonyymisanastoaan edellisten hakujen perusteella. Myös tiedonhakuagentilla tulee siis olla tässä mielessä oppimiskykyä ja muistia.

Kun verrataan agentteja toisiinsa, on huomioitava, että hakulogiikkaa on syytä pyrkiä toteuttamaan mahdollisimman paljon käyttöliittymän läheisyydessä toimivassa agentissa ja tiedonlähteillä olevien agenttien on syytä keskittyä erityisesti niiden omaan tehtävään toimia tulkkeina tiedonlähteen ja käyttöliittymäagentin välillä. Muussa tapauksessa jokaisen tiedonlähteellä olevan hakuagentin toteuttaminen voi osoittautua varsin suuritöiseksi. Ohjelmistoagenttimalli perustuu paitsi tiedon

käsittelyn hajauttamiseen, myös aina siihen, että operaatio tehdään siellä missä se on järkevintä. Kuten monessa muussakin agenttiteknologiaan liittyvässä, myös tässä tapauksessa malli löytyy tosielämästä. Tiedonlähteillä olevat agentit tuottavat raaka-aineen hakuagentille, joka jalostaa tuon raaka-aineen käyttöliittymässä näytettäväksi lopputuotteeksi.

4.3 Yhteenveto

Agenttia suunnitellessa tulee huomioida ympäristö, jossa agentti tulee toimimaan ja tehtävät, joita agenttia vaaditaan suorittamaan. Luvussa esitettiin ne suunnitteluperiaatteet, joita työssä on käytetty ja yleisemmin hakuagentin erityiskysymyksiä, sekä aiheen tieteellinen tausta filosofian ja tekoälytutkimuksen piiristä. Tiedonhakuagentin tulee toteuttaa monia agentin ominaisuuksia ja kuten agentit yleensäkin, mitä tarkemmin se toteuttaa teoreettisen mallin, sitä tehokkaammin se pystyy toimimaan. Hakutyökalujen kannalta agenttiteknologia on erinomainen alusta. Se tarjoaa mahdollisuuksia laajojen kokonaisuuksien yhdistämiseen ja ongelmanratkaisun mallintamiseen agentin ohjelmakoodissa. Tiedonhakumetodien kannalta valintaan tulee vaikuttaa joustavuus, nopeus, kyky haun tarkkuuden arviointiin ja kyky haun laajentamiseen. Toisaalta myös käyttäjän palautekanavat tulee pystyä integroimaan tehokkaasti agentin toimintaan.

5 Tiedonhakuagentin suunnittelu

Tässä työssä on tähän mennessä käyty läpi agenttien taustaa, esitetty tieteellistä ajatusleikkiä mihin agentin tietokoneohjelmallinen toteutus voidaan perustaa, lueteltu suunnitteluperusteita ja sivuttu tiedonhakumalleja. Lienee syytä selkeyttää tilannetta kuvaamalla, mikä agentti itse asiassa toteuttaa mitä toiminnallisuuksia ennen tiedonhakumallien esittelyä ja agentin suunnittelua.

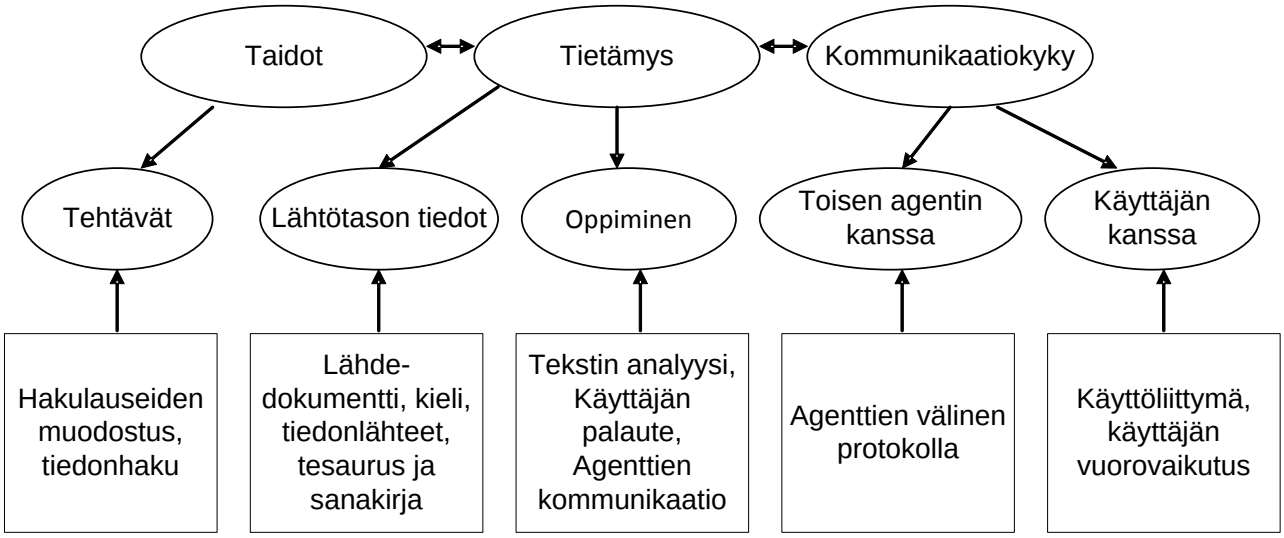
Agentteja on siis kaksi tai kaksi ja puoli, riippuen nähdäänkö käyttöliittymäominaisuudet erillisenä agenttina vai ei. Oikeastaan kyse on käyttöliittymäagentista, jolla on hakuominaisuuksia, vaikka hakuominaisuudet esittävätkin varsin isoa roolia agentin toiminnallisuudesta. Selkeyden vuoksi jaan agentit kuitenkin kahteen:

1. Käyttöliittymäagentti
2. Tiedonhakuagentti

Esittelen seuraavaksi molempien agenttien määrittelymallin mukaiset ominaisuudet ja sen jälkeen molempien tiedonhakuominaisuudet.

5.1 Määrittelymallin mukaisesti suunniteltu agentti

Luvussa 4 esiteltiin Caglayanin ja Harrisonin (1997, 7-12) agentin määrittelymalli. Esittelen nyt samaa kaaviota käyttäen tässä työssä suunniteltavien agenttien ominaisuudet ja kyvyt.



Kuva 6. Agentin määrittely sen ominaisuuksien mukaan, soveltaen Caglayanin ja Harrisonin mallia (1997, 7).

5.1.1 Käyttöliittymäagentti

Tehtävä

Käyttöliittymäagentin tehtävä on toimia käyttäjän ja tiedonhakuagenttien välisenä tulkkina. Käyttöliittymäagentin tehtävä on myös arvioida haun tuloksena saatava tulosjoukon koon riittävyyttä. Käyttöliittymäagentti pyrkii myös arvioimaan tulosjoukon tarkkuutta, vaikka se muotoileekin hakutermejä pääasiassa käyttäjän palautteen mukaan. Käyttöliittymäagentti pyrkii myös järjestämään dokumentteja oletettuun relevanssijärjestykseen käyttäjän puolesta ja laajentamaan hakua analysoimalla käyttäjän relevanteiksi toteamia dokumentteja.

Lähtötason tiedot

Lähtötason tietoina käyttöliittymäagentilla voi olla kielessä esiintyvät sanat yleisyysjärjestyksessä ja kielen sulkusanalistat. Lähtötason tietoina voidaan pitää myös ymmärrystä kielen rakenteesta ja hakualgoritmeista. Toteutettaessa agentti, nämä olisikin teknisesti hyvä erottaa itse agentin muusta

ohjelmakoodista päivityksien mahdollistamiseksi. Käyttöliittymäagentilla tulee myös olla tieto kaikista sille työskentelevistä agenteista ja niiden sijainnista, jotta se voisi kommunikoida näiden kanssa ja ottaa vastaan hakutuloksia.

Oppiminen

Käyttöliittymäagentin oppiminen perustuu käyttäjän palautteeseen: mikä hakutuloksesta tullut dokumentti oli käyttäjän mielestä relevantti ja mikä ei. Oppiminen perustuu myös tulosjoukossa olevien dokumenttien määrään eli oliko hakuavaimien valinta tai määrä onnistunut. Oppimisessa voidaan myös käyttää hakuavaimien synonyymejä. Käyttäjälle voidaan tarjota mahdollisuus antaa synonyymejä hakutermeille tai niitä voidaan päätellä tekstistä ja antaa käyttäjälle mahdollisuus valita niitä. Synonyymien automaattinen löytäminen tekstistä vaatii tekstin analyysiä ohjelmallisesti. Oppimista on myös haun laajentaminen löydettyjen dokumenttien perusteella, joista voidaan edelleen jalostaa lisää hakutermejä ja näin laajentaa hakua.

Kommunikaatio

Kommunikaatio toimii sekä muiden agenttien että käyttäjän välillä. Tässä käytetään hyväksi käyttöliittymän tai tekstinkäsittelyohjelman tarjoamia palveluita, sekä tässä työssä esiteltyä XML:ään pohjautuvaa mahdollisuutta kommunikaatioprotokollaksi agenttien välillä. Käyttäjän kanssa agentti kommunikoi sekä hakutuloksen että hakuavaimien avulla käyttöliittymän välityksellä.

5.1.2 Tiedonhakuagentti

Tehtävä

Tiedonhakuagentin tehtävä on käyttöliittymäagentin tehtävää yksinkertaisempi, mutta toteutuksena vaikeampi. Siinä missä käyttöliittymäagentti toimii tiedonhakijan (käyttäjän) ja tiedonhakuagentin välisenä tulkkina, toimii tiedonhakuagentti tiedonlähteen ja käyttöliittymäagentin välisenä tulkkina. Tiedonhakuagentin tehtävä on siis toteuttaa yksi yhteinen rajapinta jokaisen tiedonlähteen oman käyttöliittymän korvaavaksi.

Lähtötason tiedot

Tiedonhakuagentilla tulee olla lähtötason tiedot tiedonlähteen käyttöliittymästä, eli siitä miten se kytkeytyy tiedonlähteeseen ja muodostaa juuri sille tiedonlähteelle sopivia hakulauseita. Lisäksi sillä tulee olla tieto käyttöliittymäagentin sijainnista, jotta se voisi kommunikoida käyttöliittymäagentin kanssa.

Oppiminen

Tiedonhakuagentin oppiminen ei ole suuressa roolissa sen toiminnan kannalta, mutta tarkoitus on silti mahdollistaa, että tiedonhakuagentti voisi koota omaa tesaurusta käyttäjän ja käyttöliittymäagentin palautteen perusteella. Tesauruksen käyttökelpoisuus kuitenkin riippuu hyvin paljon käytetyn tiedonlähteen ominaisuuksista. Tiedonhakuagentin täytyy myös arvioida tulosjoukon kokoa ja osata sen mukaan reagoida hakulauseen rakenteen muutostarpeeseen. Tässä mielessä voidaan puhua reaktiivisesta oppimisesta.

Kommunikaatio

Tiedonhakuagentin tulee pystyä kommunikoimaan kahteen suuntaan, aivan kuten käyttöliittymäagentinkin. Tiedonhakuagentti keskustelee tiedonlähteen käyttöliittymän kanssa erityisesti kyseistä tiedonlähdettä varten toteutetun kommunikaatiorajapinnan kautta ja käyttöliittymäagentin kanssa tarkoitusta varten suunnitellun erityisen kommunikaatioprotokollan avulla.

5.1.3 Agentti välittäjäjärjestelmänä

Käytettyä suunnittelumallia voi verrata myös Ingwersenin kuvaamaan Mediator-malliin (1992). Mediator-mallissa suunnittelu perustuu tiedonhaunvälittäjän osien jakamiseen toimintojen mukaan.

Mallissa välittäjäjärjestelmän tietämys perustuu tehtävien (domain model), tiedonhakujärjestelmien (system model) ja käyttäjän tietoihin (user model), kuten tiedontarpeeseen ja tiedon tason tarpeeseen. Tätä voidaan hyvällä syyllä kutsua lähtötason tiedoiksi.

Ingwersenin malli jatkuu taustatiedon yhdistämisellä kokonaisuudeksi, jota voidaan käyttää tiedonhaussa hyväksi (system model adaptor ja user model adaptor), jossa välittäjäjärjestelmä myös oppii tietoja tiedonlähteistä ja käyttäjän sen hetkisestä tiedontarpeesta. Tätä oppimista ja tiedonyhdistämistä kutsutaan agenttiteknologiassa yksinkertaisesti oppimiseksi. Tässä työssä oppiminen on toiminta, mikä tapahtuu ennen hakua, haun aikana ja sen jälkeen.

Ingwersenin mallissa seuraavaksi tehdään itse toimintojen valinta ja toiminto (retrieval strategy) ja jatketaan kommunikaatioon. Näitä voi verrata Caglayan ja Harrisonin (1997) mallin mukaisiksi taidoiksi ja kommunikaatioksi. Kommunikaatio jatkuu mallissa vielä tietämyksen päivittämisella (mapping, explanation, transformer ja planner), jota taas tämän työn agentti tekee hakuagenttien ja käyttäjän palautteen avulla.

5.2 Tiedonhakumallien mukaisesti suunniteltu agentti

Koska agenttien päätehtävä on tiedonhaku ja tiedonhakumallien toteuttaminen, esittelen seuraavaksi molempiin agentteihin suunnitellut tiedonhakuominaisuudet tarkemmin.

Käyttöliittymäagentti

Käyttöliittymäagentti toteuttaa seuraavat tiedonhakuun liittyvät toiminnot:

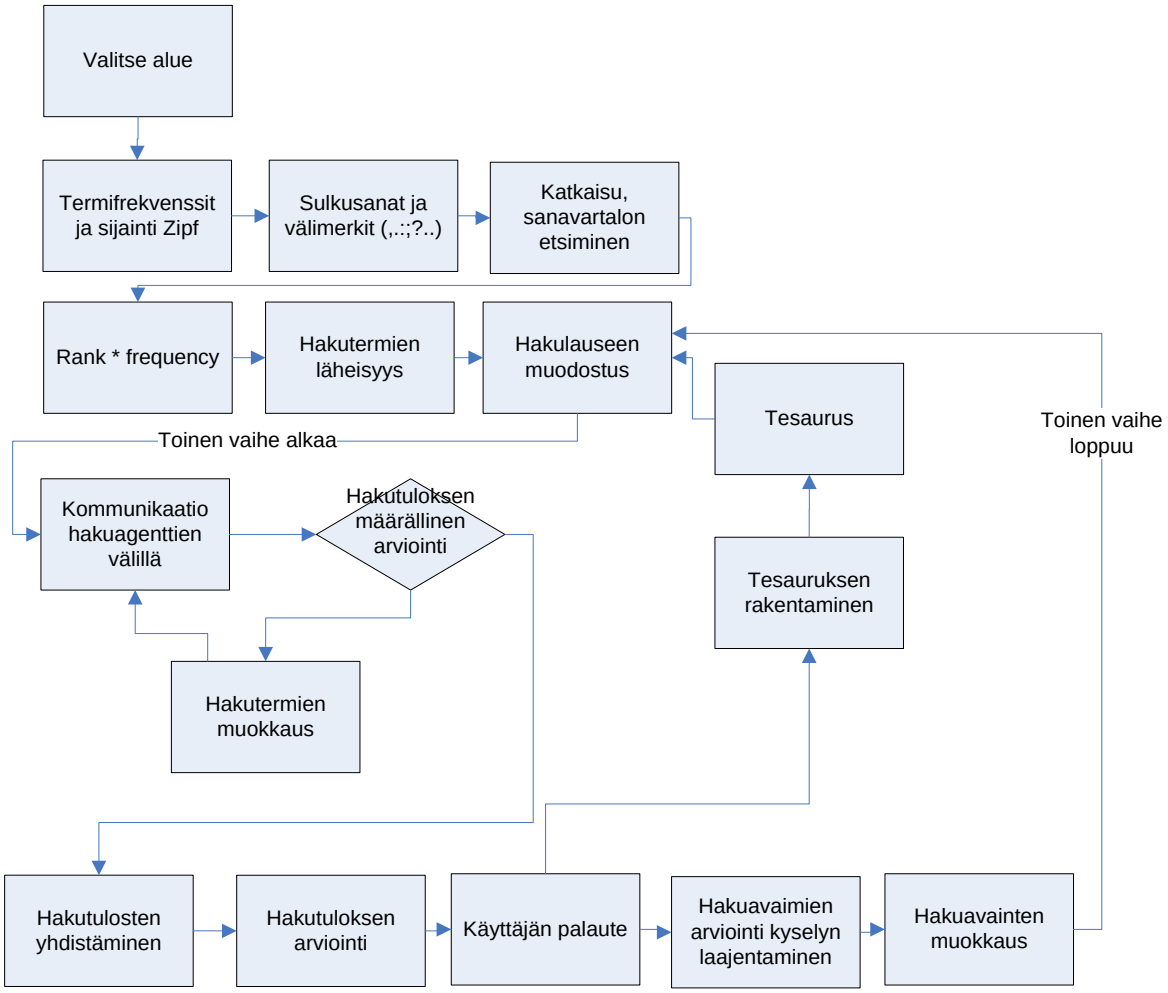
- Dokumentin analyysi hakutermien generointia varten
- Dokumentin sanojen frekvenssin laskenta
- Sulkusanojen poisto
- Sanojen katkaisu, sanavartalon etsiminen
- Sanojen ranking (asetus oletettuun paremmuusjärjestykseen)
- Läheisyysoperaattorien soveltaminen
- Hakuavainten muokkaus käyttäjän palautteen mukaan
- Haun laajentaminen
- Tesauruksen rakentaminen hakutuloksen ja käyttäjän palautteen perusteella

Hakuagentti

Hakuagentti toteuttaa seuraavat tiedonhakutoiminnot:

- Hakuavainten muokkaus tiedonlähteen mukaiseksi hakulauseeksi
- Tulosjoukon viitteiden määrän arviointi
- Haun muokkaaminen tulosjoukon perusteella

Kuvassa 7 molempien agenttien toiminnot tiedonhakijoina on kuvattu korkealla tasolla. Jokaisen osakomponentin toiminta kuvataan erikseen seuraavissa luvuissa 5.3.1 ja 5.3.2.



Kuva 7. Agenttien suorittama hakupolku.

Suunniteltua hakupolkua voi verrata myös Ingwersenin kuvaamaan Mediator-malliin (1992). Siinä hakuun liittyvät toiminnot hakustrategia, vastauksen generointi ja palautteen generointi voidaan nähdä sisältyvän edellä kuvattuun hakupolkuun.

Ingwersenin mallissa haku (retrieval strategy) etenee seuraavasti (1992, 8.1):

- Tiedonlähteiden valinta
    - Arvioidaan ja valitaan käytettävät tiedonlähteet
- Termien valinta
    - Valitaan termit, jotka vastaavat hakua; agenttimme tapauksessa tutkitaan käyttäjän dokumenttia
- Hakustrategian valinta
    - Hakustrategian valinta tiedonlähteen ominaisuuksien mukaan; tässä hakuagenttimme luodaan siten, että se voi toimia välittäjänä käyttöliittymäagentin ja tiedonlähteen välillä
- Kyselyn rakentaminen
    - Kyselyn rakentaa käyttöliittymäagentti, jonka hakuagentti edelleen tulkitsee tiedonlähteelle tiedonlähteen käyttöliittymän kautta
- Haun suorittaminen
    - Haun suorittaa hakuagentti tiedonlähteen omalla liittymällä
- Vastauksen generointi (response generator)
    - Vastauksen generointi käyttäjän luettavaan muotoon
    - Samalla käyttöliittymäagentti pyrkii laajentamaan hakua käyttäjän palautteen ja dokumenttien analyysin avulla.

5.2.1 Valitut tiedonhakumallit

Tässä työssä esitellyt agentit suorittavat tiedonhakuja monesta hyvin erityyppisestä lähteestä. Yhteistä kaikille lähteille on kuitenkin, että ne sisältävät tekstimuodossa olevia dokumentteja. Yksi kantavista ideoista on, että käyttäjällä ei ole tarvetta muodostaa itse hakulauseita, tai edes hakuavaimia. Käyttäjän käytössä olevaan yhdistettyyn käyttöliittymä- ja tiedonhakuagenttiin toteutettaisiin suunnitelman mukaan samat hakuominaisuudet kuin itse tiedonlähteillä oleviin hakuagentteihin. Voidaan siis sanoa, että siinä missä tiedonlähteiden mukaan toteutetut tiedonhakuagentit toteuttavat tiedonhaun tiedonlähteestä, käyttöliittymäagentti toteuttaa tiedonhaun käyttäjän dokumentista. Tässä tapauksessa käyttöliittymäagentti kuitenkin hakee dokumentista hakuavaimia, ei dokumentteja. Koska hakuavaimet haetaan automaattisesti, ja avaimien painoarvo

tulee pystyä arvioimaan samaan tapaan kuin tiedonlähteistä saatujen dokumenttien paino oli löydettävä tiedonhakumalli, joka sopisi toteutettavaksi agentin ohjelmakoodilla parhaiten ja lisäksi huomioisivat kieltenvälisen tiedonhaun erityistarpeet.

Jo tutkimuksen alkuvaiheessa selvisi, ettei mikään malli vastaisi suoraan vaatimuksia, vaan agentteihin olisi toteutettava soveltaen useita malleja. Tärkein malleista tulisi olemaan käyttäjän dokumentin analyysin malli toisella kierroksella (hakukierroksista enemmän luvuissa 5.3 ja 5.4). Haun laajennukseen käytettävän mallin tulisi hyväksyä palaute käyttäjältä ja toisaalta pystyä laajentamaan hakua tehokkaasti jo löydettyjen dokumenttien avulla.

Agentti mahdollistaa myös kieltenvälisen tiedonhaun. Tässä voidaan käyttää suoria sanakirjoja, mutta myös jonkinlaisen hakuavaimien analyysimallin käyttäminen on tarpeen. Hakualgoritmin valinnassa päädyttiin RATF-tiedonhakumalliin (relative average term frequency) (Pirkola et al 2002). Syitä valintaan oli useita. Ensinnäkin RATF-malli on tarpeeksi kevyt toteutettavaksi agenttiohjelmistoissa, joiden on tarkoitus olla hajautetun mallin mukaisesti massa- ja keskusmuistin sekä suoritintehon vaatimuksiltaan pieniä. Toisekseen RATF-mallilla voidaan toteuttaa kieltenvälinen tiedonhaku (Pirkola et al 2003, 3), sillä se sisältää tf.idf-mallista poikkeavasti myös avainten relevanssin arvioinnin (ks. esimerkiksi Salton 1989). Tf.idf-mallit ovat malleja, jotka keskittyvät termien löytämiseen puhtaasti niiden esiintymisen perusteella. Esittelen silti myös vaihtoehtoisia tapoja tehdä haun tarkennus, mutta periaatteeltaan nämä vastaavat RATF mallia.

Ensimmäisellä hakukierroksella tärkeintä on muodostaa käyttäjän dokumentista lähtökohta tuleville hakulauseille. Eri vaihtoehtojen analyysin jälkeen totesin, että pitäydyn yksinkertaisessa Zipf-laissa (Salton & McGill, 1983), joka lyhykäisyydessään määrittää:

Sanojen frekvenssi * sanojen järjestys = paino

Tämän lisäksi käytetään läheisyysoperaattoria, jonka perusteella oletetaan samassa kappaleessa esiintyvien sanojen liittyvän samaan aiheeseen (mikromalli dokumentista).

Mallien kritiikkinä sanottakoon Zipf-mallin nimenomaisen keskittymisen termifrekvensseihin ja termien relevanssin huomioimattomuuden, kun taas RATF tyyppisten mallien sopivuus vaatii kokoelman tuntemista, jolloin se ei sovellu käytettäväksi kaikkien tiedonlähteiden kanssa.

Dokumenttien analyysissä erityinen ongelmakenttä on hakutermien laajentaminen synonyymeillä tai sanojen katkaisu. Salton & McGill (1983, 74) esittävät yhtenä vaihtoehtona ns. –s katkaisun, joka soveltuu kuitenkin vain englannin kielelle. Agentit tekevät tiedonhakuja usealla kielellä, joten katkaisujen tekeminen voi osoittautua vaikeaksi tai mahdottomaksi.

Dokumenttien analyysiä varten tämän työn liitteeksi toteutin yksinkertaisen Perl-kielisen ohjelman, jolla ASCII-muotoista lähdedokumenttia voidaan analysoida. Lähteenä analyysiin on ollut tämä työ siinä vaiheessa, kun se on testianalyysejä tehtäessä ollut. Liitteessä 2 on sanalista, joka kirjoittamisvaiheessa on ollut käytettävissä. Samalla ohjelmakoodilla on ollut mahdollista todistaa myös käytännössä, miten erilaiset hakutoimenpiteet olisivat helposti toteutettavissa ohjelmallisesti. Työkalu osaa poistaa lähdedokumentista sulkusanat, poistaa alun termit halutun prosenttimäärän mukaan sekä laskea sanojen frekvenssin sekä Zipf -frekvenssin ja järjestellä termit frekvenssin mukaan. Työkalun käyttöä varten tarvitaan komentoriviltä toimiva Perl asennus ja Wordin yhteyteen hakuagentin asennus. Ohjelmaa voi käyttää myös komentoriviltä seuraavasti:

```
freq.pl {n} tiedosto {kyselynpituus} optiot [-s sulkusanalista, -m
[kuinka paljon listan alusta poistetaan]

Missä 'n':
-f = frekvenssin mukaan
-z = Zipf mukaan
```

Esimerkiksi:

```
perl freq.pl –z Teksti.txt 10 -s finstopword -m 0.5
```

Esimerkin komento lajittelisi Teksti.txt tiedoston termit Zipf-lain mukaan, poistaisi sulkusanat, poistaisi alusta 0,5 % termeistä ja tulostaisi ensimmäiset 10 termiä termipainoineen.

5.2.2 Automaattisen indeksoinnin ongelmat

Useat automaattisen indeksoinnin mallit, jotka sopisivat muuten hakuagentin toimintatapaan, mutta ne edellyttävät, että hakuavaimien esiintymistiheyttä voidaan arvioida sekä dokumentissa itsessään että dokumenttikokoelmassa (mm. Salton & McGill 1983, 73-75). Tämä muodostaa ongelman, koska hakuavaimet muodostetaan ensin automaattisesti yhdestä yksittäisestä dokumentista, ei usean dokumentin kokoelmasta. Tämä ongelma voidaan myös kuvata siten, että siinä missä suuri osa automaattisen indeksoinnin malleista kohdistaa indeksoinnin useaan dokumenttiin, on käyttöliittymäagentilla vain yksi varmasti relevantti dokumentti käytössään. Tämä sama ongelma ilmeni myös RATF-mallissa, kun olin valitsemassa sitä käyttöliittymäagentin ominaisuudeksi.

RATF perustuu samaan ideaan, mihin aikaisempi KWOK-malli (KWOK, 1996): hakuavaimet, joilla on korkea frekvenssi dokumenteissa, mutta matala frekvenssi dokumenttikokoelmassa, ovat todennäköisesti relevanteimpia.

Tiedonhakuagenttiemme tulee vastata kahteen kysymykseen:

1. Mitkä ovat hakuavaimet lähdedokumentissa?
2. Mitkä dokumentit vastaavat hakuavaimia tiedonlähteissä?

Valitun hakumallin tulisi olla sopiva, jotta molempiin kysymyksiin voidaan vastata ja se soveltuisi molempiin käyttötarkoituksiin.

Ensimmäisessä käyttötarkoituksessa ongelma muodostuu puuttuvasta vertailuaineistosta, toisessa käyttötarkoituksessa ongelma taas on se, että tiedonlähteellä on käytännössä aina jokin käyttöliittymä, joka rajoittaa hakua. RATF-mallia ei voitu siis valita pääsääntöiseksi hakumalliksi, mutta sitä voidaan kuitenkin käyttää avuksi haun tarkentamisessa toisessa vaiheessa ja esimerkiksi kielten välisen haun tehostamisessa. Lopulta sopivaksi malliksi yksinkertaiseen haun laajentamiseen löytyi ns. $w_t(p_t - q_t)$ kaava, joka on tarkemmin esitelty luvussa 5.3 (Efthimiadis 1993, 148-150). Seuraavassa luvussa esitellään tarkemmin, miten erilaisia tiedonhaunperiaatteita voi käyttää joustavasti toteutettaessa tiedonhakuagentteja.

5.3 Suunniteltujen tiedonhakuagenttien toiminta

Dokumentti analysoidaan kahdessa osassa, joita kutsun analyysikierroksiksi. Ensimmäinen analyysikierros keskittyy hakuavaimien löytämiseen sanojen harvinaisuuden perusteella Zipfin laissa määritellyllä kaavalla. Toinen analyysikierros taas laajentaa hakua sekä käyttäjän palautteen että tiedonlähteiden palautteen (saatujen dokumenttien) perusteella. Koska käyttöliittymä keskittyy dokumentin analysointiin, ei ensimmäisessä vaiheessa ole käytettävissä minkäänlaisia vertailudokumentteja, eikä tulosjoukon ominaisuuksiin perustuvien kyselyn laajennusmallien käyttäminen onnistu ensimmäisessä vaiheessa.

Dokumentin analyysikierrokset perustuvat Saltonin ja McGillin jo 1980-luvulla esittämään malliin yksinkertaisesta automaattisesta indeksoinnista (1983, 74):

1. Tunnista sanat, jotka esiintyvät dokumenttikokoelman dokumenteissa
2. Käytä sulkulistaa poistaaksesi sulkusanat, koska ne ovat tehottomia tiedonhauissa
3. Käytä automaattista sanojen katkaisua. Salton & McGill perustivat oman katkaisunsa englannin kielen ns. –s katkaisumenetelmään Operations -> Operation
4. Poista yleisimmät ja harvinaisimmat sanat. Tässä työssä korvaan tämän mallin kohdan omalla ratkaisullani eli lasken sanojen painot Zipf-lailla
5. Laske jokaiselle hakukokoelman dokumentille dokumentin paino

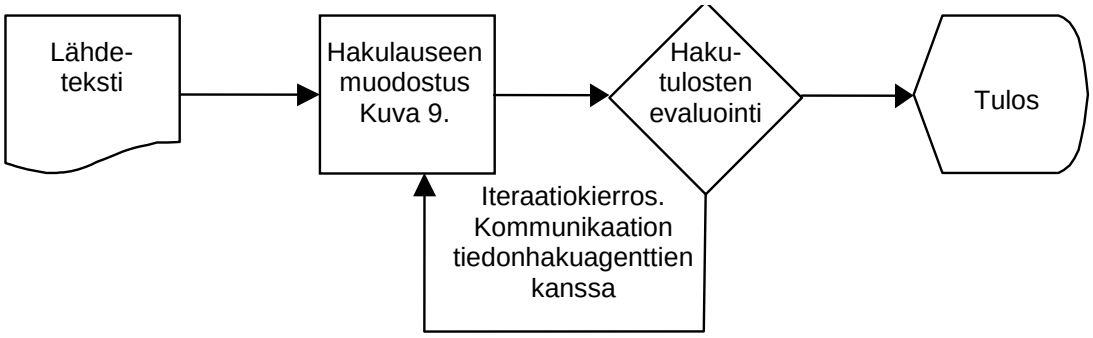
Koska käyttöliittymäagenttimme kohdistaa automaattisen indeksointinsa vain yhteen dokumenttiin, ei vaihe 5 ole suoraan toteutettavissa. Tämä pyritään ratkaisemaan toisella kierroksella käyttämällä RATF analyysiä tai vastaavaa analyysitapaa. Tässä vaiheessa tutkimusta rajaan pois käyttötapauksen, jossa useamman dokumentin kokoelmaa käytetään hakuavainten lähteenä. Tällainen kokoelma voisi olla vaikka käyttäjän dokumentit yhdessä hakemistossa tai käyttäjän samaan aiheeseen liittyväksi määrittelevät dokumentit. Tämä käyttötapaus on kuitenkin mielenkiintoinen, eikä sitä pidä jättää jatkossa huomiotta, jos tutkimusta alueella laajennetaan. Teknisesti haun laajentaminen käyttäjän dokumenttikokoelmaan olisi helppoa, mutta ongelmat tulevat termirelevanssin arvioinnissa, sillä käyttäjän ”dokumenttikanta” ja käyttäjän ajatus relevanteista dokumenteista saattaa erota. Toisinsanoin, onko oletettavaa, että käyttäjä hakee edelleen samoja dokumentteja vai kohdistuuko haku dokumentteihin, joita käyttäjällä ei vielä ole? Tutkimuksessa suunniteltu agentti tarjoaa tähän ratkaisun, keskittämällä haun käyttäjän aktiiviseen intressialueeseen dokumentissa.

5.3.1 Dokumentin ensimmäinen analyysikierros

Käyttöliittymäagentti koostuu siis kahdesta osasta, käyttöliittymästä ja itse toiminnot suorittavasta agentista. Käyttöliittymän rooli on esittää käyttäjälle hakutulos luettavassa muodossa ja toimia linkkinä sovelluksen ja agentin välillä. Se siis toteuttaa toisen käyttöliittymäagentin kommunikaatiotarpeesta, eli kommunikaation käyttäjän ja agentin välillä. Itse agentin rooli on kommunikoida muiden agenttien välillä ja muodostaa hakulauseita. Kokonaisuudessa

käyttöliittymäagentti toimii tietopalveluhenkilön teknisenä ilmentymänä, Nardi & O'Day (1996) kuvaamalla tavalla.

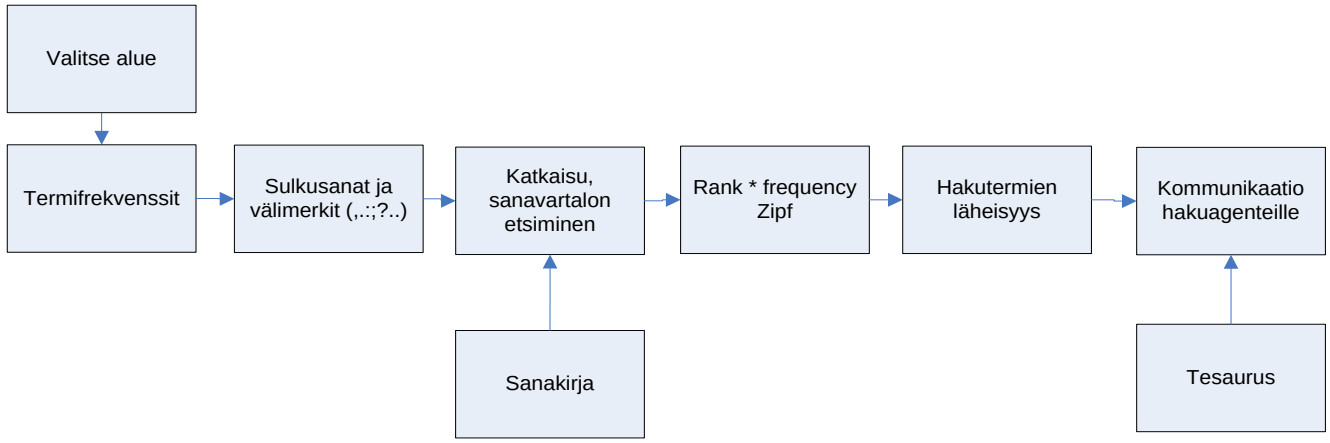
Toiminnallisuus tapahtuu iteratiivisesti sovelluksen ja agentin välillä:



Kuva 8. Käyttöliittymän toiminta

Käyttöliittymäagentin ensimmäinen dokumentin analyysikierros tehdään etsimällä ensin pelkän dokumentin, sanakirjan ja esiintymistiheyden perusteella oletetusti relevantit hakuavaimet. Avuksi voidaan käyttää myös käyttäjältä kerättyä tesaurusta, joka sisältää hakutermeille synonyymejä.

.



Kuva 9. Käyttöliittymäagentin ensimmäinen analyysikierros

Käyttöliittymäagentin toiminta ensimmäisen kierroksen aikana on esitetty seuraavassa esimerkissä:

1. Käyttäjällä on tekstinkäsittelyohjelmassaan avoinna käsiteltävä dokumentti
2. Käyttäjä klikkaa hakuagentin käynnistävää painiketta tekstinkäsittelyohjelmassa tai agentti toimii itsenäisesti käyttäjän kirjoittaessa dokumenttia
3. Käyttöliittymän tallennusosa valitsee käyttäjän dokumentista sivun verran tekstiä kursorin ympäriltä tai kohdistaa haun kappaleeseen, mikäli kappale sisältää jo tarpeeksi paljon tekstiä
4. Käyttöliittymäagentin hakuosuus laskee dokumentissa oleville sanoille painoarvot (termifrekvenssit)
5. Sulkusanat poistetaan listalta
6. Sanojen katkaisu, sanavartalon etsiminen
7. Löydetyt hakuavaimet järjestellään (Rank * Frequency)
8. Hakuagentti laskee termien etäisyydet toisistaan (sijainti, läheisyys)
9. Hakuagentti muodostaa viestin muille hakuagenteille ja lisää synonyymisanat tesauruksesta
10. Hakuagentti jää odottamaan vastausta muilta hakuagenteilta
11. Perustuen hakuagenttien löytämiin dokumentteihin, käyttöliittymäagenttiin liittyvä hakuagentti arvioi hakuavainten relevanssin uudelleen (katso toisen kierroksen kuvaus luvussa 5.3.2)

Seuraavaksi jokainen vaihe on esitelty tarkemmin:

1. **Käyttäjällä on tekstinkäsittelyohjelmassaan avoinna käsiteltävä dokumentti**

Hakuavaimien muodostuksessa käytetyn tekstin valinta perustuu oletukseen kahteen oletukseen:

- Dokumentti, joka käyttäjällä on sillä hetkellä avoinna tekstinkäsittelyohjelmassa, on dokumentti, josta käyttäjä on kiinnostunut
- Dokumentista, joka on avoinna käyttäjän tekstinkäsittelyohjelmassa, voidaan valita tietty osa tai koko dokumentti laajuudesta riippuen

Lähtötilanteessa, käyttäjällä on avoinna tekstidokumentti järjestelmässä, johon agentti integroidaan. Oletus on, että teksti, jota käyttäjä joko katselee tai kirjoittaa juuri sillä hetkellä on käyttäjän kiinnostuksen kohteena. Tätä kohdetta seurataan kursorin sijainnilla.

Tässä vaiheessa myös oletetaan, että käyttöliittymäagentti on rakennettu tekstinkäsittelyohjelman, esimerkiksi Microsoft Wordin yhteyteen. Esimerkki käyttöliittymän prototyypistä, joka on toteutettu tämän työn yhteydessä, löytyy luvussa 5.5.

2. **Käyttäjä klikkaa hakuagentin käynnistävää painiketta tekstinkäsittelyohjelmassa, tai agentti toimii itsenäisesti käyttäjän kirjoittaessa dokumenttia.**

Käyttäjälle annetaan kaksi vaihtoehtoa agentin käynnistykseen: agentti voi joko käynnistyä napin painalluksesta, mikä avaa käyttäjälle tiedonhakudialogin (katso 5.5) tai agentti voi toimia autonomisesti käyttäjän syötteestä riippumatta tausta-ajossa. Paras vaihtoehto on toki näiden kahden yhdistelmä ja se on myös tässä työssä oletuksena. Käyttäjän kirjoittaessa lisää tekstiä agentti aika-ajoin päivittää hakulauseita ja tekee täydentäviä tiedonhakuja, mutta vasta kun käyttäjä avaa agentin, pyydetään häneltä palautetta. Toinen mahdollisuus olisi esimerkiksi pieni ikkuna, johon uudet linkit ilmestyvät, mutta tämäkin tulee tehdä mahdollisimman vähän käyttäjää häiriten.

3. **Käyttöliittymän tallennusosa valitsee käyttäjän dokumentista sivun verran tekstiä kursorin ympäriltä, tai kohdistaa haun kappaleeseen mikäli kappale sisältää jo tarpeeksi paljon tekstiä.**

Normaalitilanteessa käyttöliittymäagentti pyrkii analysoimaan tekstin nimenomaan kursorin ympäriltä ja rajaamaan valinnan kappalerajoihin. Mikäli kuitenkin valmiiden lauseiden määrä on liian alhainen hakulauseen muodostamiseen, voidaan myös valita koko sivu tai jopa koko dokumentti, mikäli käyttäjä näin haluaa.

4. **Käyttöliittymäagentin hakuosuus laskee dokumentissa oleville sanoille painoarvot (Termifrekvenssit)**

Lähdeteksti josta hakuavaimet siis valitaan, on joko käyttäjän kokodokumentti tai sillä hetkellä oletetusti työskentelyn kohteena oleva kappale. Hakuavaimien lähdealuetta voidaan myöhemmin laajentaa käyttäjän tai hausta saadun palautteen perusteella.

Tässä vaiheessa agentilla olisi siis esimerkiksi lista sanoista tämän työn tekstistä:

Frekvenssi	**Termi**
217	on
168	ja
142	agent
91	agentin
61	ei
60	että
51	käyttäjän
50	tai
49	myös
46	agentti
43	agenttien
41	voidaan
37	se
36	agentit
35	tulee

Taulukko 1. Termit esiintymisen mukaan

… (Taulukossa ensimmäiset 15 sanaa)

Taulukko on toteutettu liitteen 2 ohjelmakoodilla.

Samalla kaikki sanat muutetaan pienillä kirjaimilla kirjoitetuksi ja välimerkit poistetaan. Myös numerot poistetaan. Välimerkit poistetaan, koska ne ovat merkityksettömiä termifrekvensseillä, joskaan ne eivät toki ole merkityksettömiä lauseen semanttiselle rakenteelle. Samasta syystä tehdään sanojen muuttaminen pieniin kirjaimiin, mikä on kielestä riippuen mahdollista. Suomen kielen ollessa kyseessä yleistän, että harvoille hakutuloksille tehdään näin haittaa vaikka esimerkiksi etunimi ”Liina” muuttuisikin sanaksi ”liina”.

5. **Sulkusanat poistetaan listalta**

Seuraavaksi agentti poistaa lähdedokumentista ensin sulkusanat, eli sanat, joilla ei ole merkitystä tekstin tietosisällön tai tiedonhaun kannalta. Tällaisia sanoja ovat suomen kielessä esimerkiksi ja, sekä, eli, tai, vai, mutta, vaan jne. Poistaminen tehdään valmiin listan perusteella, joka on joka kielelle omansa.

Esimerkin listalle käy seuraavasti:

Termifrekvenssi	**Termi**
142	agent
91	agentin
51	käyttäjän
46	agentti
43	agenttien
36	agentit
30	dokumentin
30	perusteella
29	a
28	käyttöliittymäagentin
28	avain
28	the
28	http
27	sanojen
27	information

Taulukko 2. Sulkusanat poistettu

… (listalla seuraavat 15 sanaa)

Listalta on siis poistettu yleisimmät suomenkielen sanat, samalla osa termeistä nousee 15 ensimmäisen termin listalle. Tässä työssä on valittu sulkulistojen käyttö, koska eri kielissä sulkusanojen määrät voivat poiketa toisistaan, jolloin määrällinen katkaisu ei ole välttämättä tarpeeksi tarkka toiminto. Aina sulkusanalistaa ei kuitenkaan ole tarjolla ja tällöin voidaan tehdä yleisyyteen pohjautuva leikkaus. Jos sulkulistaa ei ole saatavissa, voidaan poistaa dokumentin koosta riippuen tietty osa yleisimmistä termeistä (Salton & McGill, 1983, 62), tämä on kuitenkin tehtävä vasta myöhemmässä vaiheessa 7.

6. Sanojen katkaisu, sanavartalon etsiminen

Perusmuotoistus tai sanojen katkaisu on ohjelmallisesti hyvin ongelmallista suomen kielellä. Suomea on nimitetty jopa vaikeimmaksi kieleksi sen morfologisen rikkauden vuoksi (Kettunen 2007, 25). Hakutermien muokkaus tulee toteuttaa agenttiin ja valita sekä käyttötarkoitukseen että kieleen sopiva muokkausmenetelmä. Pääongelma kielen morfologisessa vaihtelussa tiedonhaulle on, että ajatusmalli "yksi hakuavain, yksi käsite, yksi täsmäytys", ei toimi tekstuaalisessa tiedonhaussa (Kettunen 2007, 26). Metodeja käsitellä sanoja tiedonhaulle sopiviksi on päätyypiltään kahdenlaisia, vähentäviä (reductive) ja laajentavia (generative). Vähentämistä edustaa sanojen muuttaminen perusmuotoon tai muuhun vastaavaan yleisempään muotoon. Hyvä puoli on, ettei tällaisessa tapauksessa tarvitse yrittää ohjelmallista katkaisua, mutta huonona puolena on termien valinta ja sen siitä seuraavat tulosjoukon tarkkuusongelmat (Kettunen 2007, 29). Laajentamisessa yksi vaihtoehto on lisätä hakuun hakutermin eri muodot, esimerkiksi: kissa -> kissan, kissalla, kissoilla... Toinen vaihtoehto on laajentaa hakutermejä katkaistuilla muodoilla: nainen, naise, naisi, naist. Tässä katkaistut termit ovat pidempiä kuin vähentävässä muokkauksessa ja haku on todennäköisesti tarkempi (Kettunen 2007, 31). Molemmat laajentavat metodit johtavat kuitenkin pitkiin hakulauseisiin ja voivat hidastaa hakua (Kettunen 2007, 32-33). Esitän seuraavaksi useamman mahdollisen menetelmän hakutermien käsittelyyn edellisiin luokituksiin pohjautuen ja myös suosituksen valittavasta menetelmästä.

Ensimmäisinä vaihtoehtoina on perusmuotoistus tai sanojen yksinkertainen katkaisu. Perusmuotoistuksessa listan sanat muutetaan ensin perusmuotoonsa, koska listalla on paljon samoja sanoja eri muodoissa. Se ei ole tiedonhaussa tehokasta, ellei käytössä ole tiedonlähteitä, joissa on käytössä perusmuotoinen suomenkielinen indeksi. Perusmuotoiset termit eivät ole luonnollisia normaalissa tekstissä, joten pelkkien perusmuotojen käyttämisellä kokotekstihaussa leikattaisiin tulosjoukon kokoa radikaalisti. Jotta monikäyttöisyys turvattaisiin, on suositeltavaa valita perusmuotoistuksen sijaan ensimmäiseksi vaihtoehdoksi yksinkertainen katkaisu. Yksinkertainen katkaisu tarkoittaa yleisten päätteiden poistamista, esimerkiksi -ksi. Englanninkielisen dokumentin ollessa kyseessä usein pelkkä monikon tunnuksen s:n poistaminen riittäisi. Tämä niin sanottu s-katkaisu ei ole kuitenkaan mikään täydellinen sanankatkaisutapa. Ongelmia tällaisesta perusmuotoistuksesta syntyy sekä yhdyssanojen että joidenkin taivutusmuotojen suhteen, kuten alla olevasta taulukosta voi huomata.

Katkaisua käyttäen listalle nousisi taas uusia sanoja ja lista muuttuisi seuraavasti:

Termifrekvenssi	**Termi**
358	**agent (agentin eri muodot)**
51	käyttäjä (käyttäjän)
33	esimerk (esimerkiksi)
30	dokument (dokumentin)
30	perusteella (peruste)
28	käyttöliittymäagent (käyttöliittymäagentin)
28	avain
28	the
28	http
27	sanojen (**katso selitys)**
27	information
23	and
22	org
22	soap
22	tiedonhakuagent (tiedonhakuagentin)

Taulukko 3. Katkaistut termit

Samalla agentti päivittää dokumentista sanojen frekvenssin eli perusmuotoistettujen sanojen esiintymistiheyden. Taulukossa se on esitetty numerolla ennen sanaa.

Huomioitavaa on, että nyt ”agentti”-sanan frekvenssi on noussut listalla olevien eri agentti-sanamuotojen vuoksi. Lisäksi listalla on taivutuksellisesti ongelmallinen sana: ”sanojen”, joka perusmuodossaan on ”sana”, mutta haun kannalta katkaisu pitäisi ennemmin olla ”san”, mikä taas johtaa automaattisesti epätarkkuuteen tulosjoukossa. Näiden sanojen osalta paras tapa olisi yhdistää eri muodot fasettiin, mutta laajan sanakirjan vaativana toimenpiteenä se ei välttämättä toimisi agentin osana hyvin.

Kun analyysi tehtiin alun perin pelkästä johdantoluvusta, arvot olivat pienempiä, mutta lista oli vastaavan kaltainen. Tämä osoittaa hyvin, kuinka ajatusmalli aktiivisen kappaleen tekstin poimimisesta hakutermien lähteeksi toimii. Esimerkissä ote oli johdanto luku, mutta siksi sitä onkin hyvä verrata koko dokumentin sisällöstä luotuun hakuavaimistoon. Esitän sen seuraavaksi osoittaakseni, että analyysi voidaan tehdä myös pienestä osasta dokumenttia, ja tuodakseni selvemmin esiin taivutusmuotojen ongelmaa:

Termifrekvenssi	Termi
5	informaatiotutkimukse (-n)
12	agent (-in, ien,(nousee myös listalla piilossa olleidein sanojen vuoksi)
4	tutkimu
3	tiedonha
3	työ
3	toteut
2	esitellä
2	keräävä
2	tiedonhakuväline
2	osa
2	mukaan
2	itse
2	pyritään
2	pyrkii
2	tiedonhakuagent
2	tietokantoihin

Taulukko 4. Johdanto-luvun termeistä käsiteltynä samaan tapaan.

Tälläkin listalla esiintyy hankalia sanoja: ”esitellä”, ”keräävä” ja ”pyritään, pyrkii” sekä yhdyssana ”tiedonhakuagentti”. Verbin ”esitellä” on jo perusmuodossaan, kun taas sanan ”keräävä” perusmuoto olisi ”kerätä”. Katkaisussa sanoista tulisi ”esit-” ja ”kerä-”. Haun kannalta nämä sanat olisivat ongelmallisia ja jopa haitallisia tarkkuuden kannalta. Sanat ”pyritään” ja ”pyrkii” lisäävät problematiikkaa entisestään. Katkaisu on hankalaa myös sulkusanojen suhteen. Mikäli sulkusanastossa on vain perusmuotoiset sanat (olla), voi osa sanoista jäädä poistamatta (on), kuten itse asiassa kävi testiaineistossani.

Toinen vaihtoehto ongelman ratkaisuun olisi esimerkiksi perusmuotoistettujen hakemistojen käyttäminen. Riitta Alkula (2000) käsittelee väitöskirjassaan ”Merkkijonoista suomen kielen sanoiksi” kuinka luoda perusmuotoisia termejä sisältävä hakemisto kielen morfologisten tulkintaohjelmien avulla. Tämä voi johtaa tulosjoukon pieneen kokoon, mikäli tiedonlähteessä ei ole tiedonlähteessä ei ole käytössä hakemistoa, vaan haku kohdistuu kokoteksteihin (kuten usein on) (Alkula 2000, mm. 271). Hakutermien laajentamiseksi suomenkielellä on käytössä morfologisia tulkintaohjelmia jotka tuottavat taivutusvartaloita tai perusmuotoja, toisaalta ulkoisten järjestelmien kytkeminen (varsinkin maksullisten) ei välttämättä ole toivottava ominaisuus agentin kannalta.

Alkulan tutkimuksen mukaan, ”jos haluttaisiin tarjota mahdollisimman monipuoliset hakumahdollisuudet, pitäisi tietokannan dokumenteista itse asiassa tuottaa kolme eri hakemistoa:

- Kaikki sananmuodot sellaisenaan sisältävä hakemisto
- Ositettu perusmuotohakemisto
- Tunnistamattomat sananmuodot sisältävä hakemisto”

(Alkula 2000, 273). Toisin sanoin tämä tarkoittaisi käyttöliittymäagentin tekemää analyysiä lähdetekstistä. Teknisesti tämä olisi mahdollista, mutta toteuttaisi taas vain yhden kielen ratkaisun.

On myös olemassa tekniikkoja, joilla voidaan välttää sekä hakusanojen katkaisu että hakemistosanojen perusmuotoistus. Kettunen tuli tutkimuksessaan tulokseen, että morfologisesti haastavien kielien (ainakin suomi, mutta myös saksa ja ruotsi) yhteydessä hakutermejä laajentavat metodit toimivat hyvin (2007, 52). Tämän pohjalta kehitettiin FCG-metodi (Frequent Case Generation), joka perustuu sanojen yleisyyteen kielessä suhteessa toisiin sanoihin. Kettusen työssä FCG-metodilla tutkittiin voitaisiinko termien yleisyyden perusteella hallita paremmin hakutermien laajentamista. Tutkimuksessa FCG tuottikin hyviä tuloksia ja osoitti, ettei hakulauseessa välttämättä olisi tarpeen laajentaa termien määrää kuin yleisimmillä sanoilla. Tämänkaltainen hakutermien laajentaminen olisi ehkä agentin kannalta suositeltavin. Termistöjä ei tarvitsisi olla kuin vaikeaksi tiedetystä kielistä ja muiden kielien osalta voitaisiin käyttää yksinkertaista katkaisua tai tutkia sananvartaloita.

Viimeinen vaihtoehto onkin tutkia sanavartaloita. Käytännössä käyttöliittymäagentti yrittäisi katkaista sanoja automaattisesti perussääntöjen mukaan. Esimerkiksi suomen kielessä katkaisu voidaan perustaa sanavartaloihin. Suomessa sanan taivutusmuoto koostuu vartalosta ja taivutustunnuksesta. Sanavartalo on siis se osa sananmuotoa, joka jää jäljelle kun taivutustunnukset erotetaan. Sanavartalo on tunnistettavissa, sillä kaikilla suomen sanoilla on vokaaliin loppuva vartalo; osalla sanoista on lisäksi myös konsonanttivartalo. (VISK § 54). Käytännössä agentti pyrkisi löytämään näitä sanavartaloita ja poistamaan niistä taivutustunnukset. Jäljelle jääneiden sanojen vartaloita verrattaisiin toisiinsa, ja sanoja käytettäisiin hakulauseessa yhdistettynä:

1. Etsitään termin 1 sanavartalo
2. Etsitään kuinka suuri osa sanoista alkaa samalla sanavartalolla

3. Esitetään käyttäjälle yleisimmät samoilla vartaloilla esiintyvät termit ja pyydetään palaute
4. Mikäli palaute oli, että sanat ovat samoja, voidaan samalla valinnalla jatkaa. Tämä tehdään vain muutamalle pää termille, jotka ovat ehdokkaita hakulauseeseen
5. Löydetyistä sanoista voidaan muodostaa fasetteja

Sanavartalo erotetaan taivutustunnuksien poistamisella. Yksinkertaisinta on kohdentaa poisto taivutustunnusten fonologiseen rakenteeseen eli poistaa seuraavia päätteitä:

C	V	CV	CCV	CVV	VC	CVC	MUUT
N	*i*	*nA*	*ssA*	*kAA*	*Vn*	*vAt*	*seen*
T	*V*	*TA*	*stA*			*den*	*siin*
J		*Ne*	*llA*			*ten*	*tten*
		Te	*ltA*			*hVn*	*isi*
		mA	*lle*			*NUT*	
		kO	*ksi*				
		ni	*ttA*				
		si	*nsA*				
		TU	*nne*				
			tte				
			mme				

Taulukko 5. Taivutustunnusten fonologinen rakenne (VISK § 58).

Vartalon ja tunnuksen rajan löytäminen täydellisesti ohjelmallisesti voi olla haasteellista, mutta tiedonhaun kannalta riittänee yleisimpien päätteiden poisto ja oletusten tekeminen sillä perusteella. Tasapainon löytäminen tulosjoukon koon laajentamisen ja tarkkuuden välillä on tärkeintä, mutta koska lähdeaineisto on sama dokumentti, voidaan yleistys mielestäni hyvinkin tehdä. Suomen kieli on esitetty tässä esimerkkinä. Muissa kielissä esiintyy samantyyppisiä säännönmukaisuuksia; esimerkiksi englanninkielisessä aineistossa säännöt ovat huomattavasti helpompia.

Agenttia toteutettaessa tulee pohtia agentin käyttötarkoitusta ja toimintaympäristöä. Todennäköisesti ei voida valita vain yhtä keinoa sanamuotojen käsittelyyn, vaan tekstin kielen tunnistuksen perusteella voidaan valita kielelle sopiva metodi. Tekstinkäsittelyohjelmat onneksi tukevat kielen tunnistamista jo varsin hyvin. Katkaisun sijaan morfologisesti monimutkaisissa

kielissä olisi suositeltavaa käyttää laajennusta. Morfologisesti yksinkertaisissa kielissä voidaan sen sijaan käyttää katkaisua. Sanamuotojen käsittely on kieliriippuvaista, ja vaatii agentilta lähtötason tietoa erilaisista kielistä.

Mikäli sen sijaan päädytään käyttämään perusmuotoistusta, esimerkiksi tiedonlähteen ominaisuuksien vuoksi, tulee kohta 5 ajaa uudelleen, jotta kaikki sulkusanat saadaan pois hausta. Mikäli taas päädytään laajentamiseen tai katkaisuun, on sulkusanojen poisto tehtävä vasta termien käsittelyn jälkeen.

7. Löydetyt hakuavaimet järjestellään (Rank * Frequency)

Hakuavaimien painotuksessa voidaan myös laskea koko dokumentissa olevien termien yleisyys ja käyttää tätä frekvenssilukua avuksi arvioitaessa sanojen mahdollista tehokkuutta hakuavaimina.

Zipf-laki määrittää kyseisen arviotavan seuraavasti:

$$\text{Frequence} * \text{rank} \approx \text{constant}$$

Missä termin esiintymistiheys kerrottuna sen järjestysluvulla dokumentissa on useimmiten sen relevanssi suhteessa dokumentin aiheeseen muiden hakuavainten joukossa (Salton & McGill, 1983, 60). Huomioitavaa on, että tämä malli toimii sitä paremmin mitä laajempi dokumentti on kyseessä, oletuksella, että se käsittää samaa aihealuetta kokonaisuudessaan.

Zipf-lakia käyttämällä testiaineistosta saataisiin seuraava taulukko, josta on sulkusanat poistettu:

Zipf	**Termi**
204	käyttäjän
230	agentti
258	agenttien
273	agentin
284	agent
288	agentit
420	dokumentin
450	perusteella
464	a
476	käyttöliittymäagentin
532	avain
560	the

588	http
594	sanojen
608	kieli
616	org
621	information
621	tietoa
624	and
624	jarjestysnro

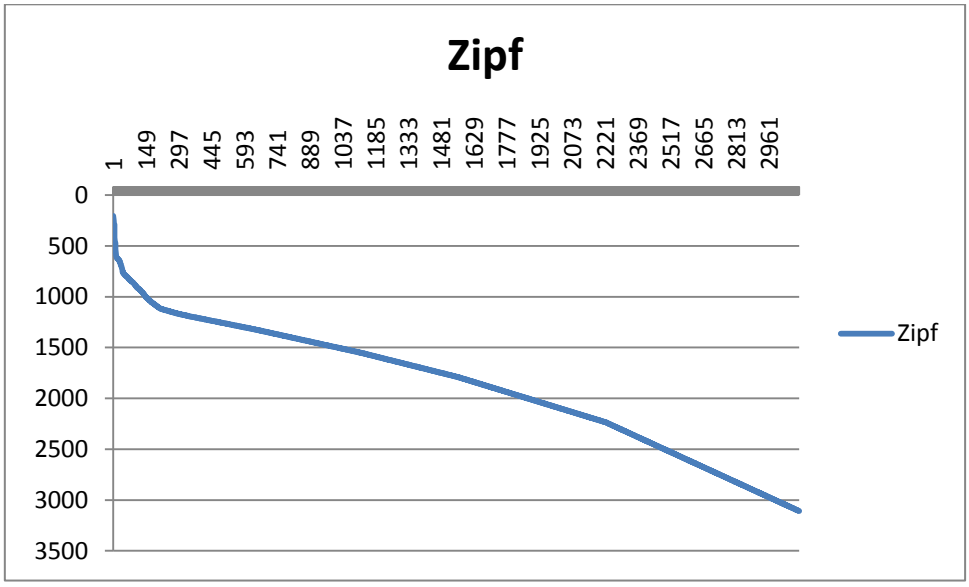
Taulukko 6. Zipf käsitelty lista.

Listalta nähdään kuinka termien järjestys muuttuu käytettäessä Zipf mukaista järjestystä relevanssijärjestyksenä. Koska sulkusanat oli poistettu jo aiemmin, ne eivät vaikuta laskentaan. Listalla on edelleen myös useita eri muotoja sanoista, johtuen nimenomaan katkaisun ongelmista. Itse näen parhaaksi katkaisutavaksi tällä listalla sananvartaloiden etsimisen ja niiden yhdistämisen faseteiksi.

Saltonin ja McGillin malli perustuu automaattiseen indeksointiin, ei hakuavaimien luomiseen, mitkä sinänsä voidaan kokea samantyyppisinä toimintoina. Termejä tarvitaan haussa vain vähän, useiden termien käyttö usein nostaa relevanssia, mutta laskee tulosjoukon kokoa niin paljon, että myös relevantit dokumentit jäävät ulkopuolelle. Salton & McGill (1983, 62,) käsittelevät myös tätä ongelmaa kirjoituksessaan. Yhtenä mahdollisuutena välttää termien hukkaaminen olisi myös kontekstin ja tesaurusten käyttäminen. Hakukierroksen viimeisessä kohdassa esitellään läheisyys operaattorien mahdollisuuksia hakuagentissa, mitä voidaan pitää myös yhtenä vaihtoehtona hakuavainten määrittämisessä. Toisin sanoen, faseteissa useammin esiintyvät termit voidaan laskea muita termejä relevantimmaksi dokumentin aiheen kannata. Läheisyysoperaatioista on enemmän tietoa seuraavassa luvussa.

Salton & McGill esittävät mallissaan perustellusti, että listalta poistetaan termejä lopusta ja että hakuun otetaan mukaan termejä listan alusta ensin 5-10 kappaletta, riippuen lähteen laajuudesta. Listan lopussa olevat termit ovat usein merkityksettömiä, ja tarpeen vaatiessa niitä voidaan poistaa(1983, 63). Esimerkiksi tämän työn analyysistä syntyi yli 3000 termiä, joista voidaan poistaa suuri osa, koska relevantit termit (olettaen sulkusanojen poiston) esiintyvät listan alussa tai hiukan alun jälkeen (ns. keskifrekvenssin termit). Termien mukaan ottoa laajennetaan tarpeen mukaan,

mikäli hakuagentit kommunikoivat tulosjoukon koon olevan riittämätön. Seuraavassa kaaviossa Zipf-testiaineistosta X-akselilla termien määrä, Y-akselilla Zipf-arvo



Kaavio 1. Zipf-luokiteltujen sanojen relevanssi

Kaaviosta nähdään kuinka termien oletettu relevanssi laskee. Saltonin & McGillin esittämän oletuksen mukaan relevantit hakutermit löytyvät keskeltä. Tästä testiaineistosta sulkusanat oli poistettu automaattisesti. Koska listassa käytettiin vain suomenkielistä sulkulistaa, jäivät sanat ”the”, ”of” ja ”and” listalle. Myös näiden sanojen poistaminen on tarpeen, mutta useiden sulkulistojen samanaikaisessa käytössä pitää olla varovainen. Englannin kieli ja suomen kieli ovat varsin turvallinen yhdistelmä, mutta tätä ei kuitenkaan pidä yleistää muihin kielipareihin.

Hakutermien valinnassa erityisesti Zipf- tai tf.idf-tyyppisen luokittelun jälkeen on huomioitavaa yleistys, joka voidaan tehdä termien frekvenssien ja termien arvon hakuavaimena välillä, kun valitaan hakutermejä (Efthimiadis, 1993, 146):

- Erittäin suuren frekvenssin termit eivät ole useinkaan käyttökelpoisia
- Termit, joiden yleisyys on lähellä keskiosaa (toisessa neljänneksessä) ovat usein hyvinkin käyttökelpoisia

- Harvinaiset termit ovat usein käyttökelpoisia, mutta eivät niin käyttökelpoisia kuin keskifrekvenssin termit
- Erittäin harvinaiset termit ovat hyviä hakutermejä siinä mielessä, että ne osoittavat löydetyn dokumentin relevantiksi, mutta haussa ne ovat tuloksien saannin kannalta tehottomia

Yleistys sisältää myös ongelman. Mistä kohtaa termien mukaan ottaminen aloitetaan? Esimerkiksi edellisen listan termit on otettu sanalistan alusta ja siinä esiintyy relevantteja termejä. On kuitenkin muistettava, että sulkusanat on jo poistettu ja että kyseessä on suomenkielinen teksti. Esimerkiksi Salton & McGill (1983, 62) mainitsevat tämän tavan olevan helposti liian rankka ja se saattaa johtaa relevanttien sanojen leikkaantumiseen pois tuloksesta. Salton & McGill poistivat omassa esimerkissään termejä 30 kpl 6000 osajoukosta eli 0,5 % termeistä käyttämällä sääntöä, että termit esiintyvät vähintään 25 % dokumenteista. Näin leikkaus ei olisi liian suuri. Käytännössä pyritään poistamaan kaaviossa näkyvä vasemman laidan piikki. (Salton & McGill, 1983, 74)

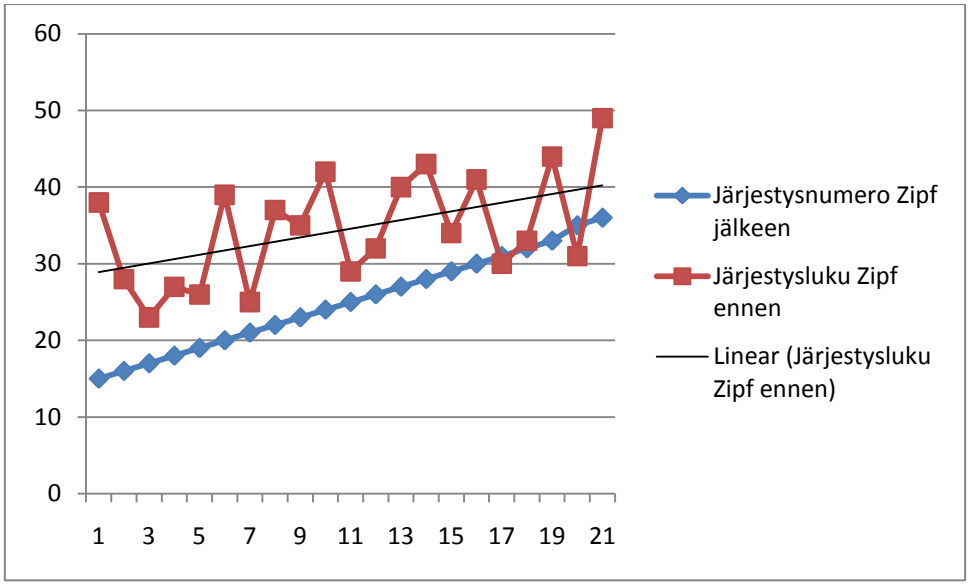
Mikäli edellisestä listasta poistettaisiin samalla säännöllä 0,5 % eli käytännössä 15 termiä, lista muuttuisi seuraavasti:

Zipf	**Termi**	**Alkuperäinen lista**
608	**kieli**	*käyttäjän*
616	Org	*agentti*
621	information	*agenttien*
621	Tietoa	*agentin*
624	And	*agent*
624	jarjestysnro	*agentit*
625	Of	*dokumentin*
629	Agents	*perusteella*
630	käyttäjälle	*A*
630	hakuagentti	*käyttöliittymäagentin*
638	Soap	*avain*
640	toteuttaa	*the*
640	käyttöliittymäagentti	*http*
645	Käyttäjä	*sanojen*
646	käyttöliittymä	**kieli**
656	tiedonhaun	org
660	tiedonhakuagentin	information
660	tiedonhakuagentti	tietoa
660	käyttöliittymän	and
682	tiedonlähteen	jarjestysnro

686	kommunikaatio	of

Taulukko 7. Taulukossa hakutermeistä poistettu sekä sulkulistalla olevat että 0,5 % ensimmäisistä termeistä. Toteutettu liitteenä olevalla työkalulla.

Zipfin käytön vaikutus termien järjestykseen näkyy paremmin seuraavassa kaaviossa:



Kaavio 2. Zipf-vaikutus järjestysnumeroon

Kaaviossa on alkuperäinen järjestysnumero (frekvenssin mukaan) ja siirtymä eli se missä järjestysnumero nyt on (X-akseli), nähdään siis alkuperäisten järjestysnumeroiden muutos, lähemmäs listan alkua (relevantimmaksi). Ote on 0,5 % listan alusta.

Alkuperäinen ensimmäisen kierroksen kysely on myös toisen kierroksen kyselyn lähtökohtana. Ensimmäisellä kierroksella näimme, kuinka kyselyä voidaan laajentaa jo lähdedokumentista laajentamalla hakulausetta termein. Tässä mielessä agentin luoma hakulause nimenomaan eroaa käyttäjän tekemistä hauista eli hakulause sisältää todennäköisesti useita termejä, eikä tyypillistä muutamaa hakuavainta. Kwokin mukaan (1996, 187) esimerkiksi Text REtrieval Conference- eli TREC-aineistossa kokeellisissa testeissä havaittiin, että kyselyiden hakuavaimien väheneminen keskimäärin 19 hakuavaimesta kuuteen hakuavaimeen johti 25 % laskuun tarkkuudessa. On siis suositeltavaa käyttää useita hakuavaimia haussa, eikä vain yhtä hakuavainta aihetta kohti. Yksi sana voi olla varsin monimerkityksellinen ja sitä ei sinällään pitäisi käyttää ainoana hakuavaimena.

Lisäksi kaksi sanaa, jotka liittyvät toisiinsa eivät sinällään kerro tiedonhakujärjestelmälle mitään, vaikka niillä olisikin merkitysyhteys käyttäjälle. Siksi useiden hakuavaimien käyttö ja näiden yhdistäminen hakulauseeksi tuottaa toivottavamman lopputuloksen (Kwok 1996, 187-188). Tämä sama pätee sekä useamman termin valitsemiseen Zipf-listalta että hakutermien katkaisun ongelmassa. Listalta voidaan jo valita ensimmäiset termit kommunikoitavaksi hakuagenteille.

8. Hakuagentti laskee termien etäisyydet toisistaan (sijainti, läheisyys)

Viimeisenä käyttöliittymäagentin haunmuodostusoperaationa agentti tutkii sanojen läheisyyksiä dokumentissa eli hakuagentti laskee termien etäisyydet toisistaan (sijainti, läheisyys). Sanojen esiintymistiheyden ja läheisyyden lisäksi agentin voi ottaa huomioon sanojen yleisyyden kohdekielessä, mikäli tähän vaadittava aineisto on olemassa. Jos siis sana *tiedonhaku* esiintyy lähdedokumentissa 100 kertaa, on sillä todennäköisesti suurempi merkitys kun samassa dokumentissa myös 100 kertaa esiintyvällä sanalla *kirja*. Mikäli painoarvoltaan suuret sanat toistuvat usein lähellä toisiaan lisätään niiden painoarvoa. Nämä sanat ovat hyviä ehdokkaita myös fasettien muodostukseen, mikäli hakujärjestelmä tukee niitä. Painoarvot (käyttöliittymässä pisteet) muodostuvat siis:

- Sanojen yleisyydestä kielessä
- Sanojen läheisyydestä dokumentissa (edellyttää relevanttien termien tunnistusta)
- Sanojen Zipf-frekvenssistä dokumentissa

Termejä on mahdollista arvioida tesauruksen avulla, jossa aihepiiriin liittyvät sanat kuten ”Tiedonhaku, saanti, tarkkuus” olisi luokiteltu samaan kategoriaan. Näin ollen vaikka sana ”tiedonhaku” esiintyisi lähdedokumentissa useasti, sitä ei tulisi poistaa hausta.

Koska agenttimme perusoletus on, että sitä voidaan käyttää aiheesta ja kielestä riippumatta mihin tahansa tiedonhakuun, tesauruksen käyttö ei olisi aina käytännöllistä. Lisäksi ohjelmallisesti on jopa helpommin toteutettavissa algoritmi, joka automaattisesti etsii edellisessä vaiheessa löydetyt termit ja analysoi niiden etäisyyksiä toisistaan. Lähtöoletus fasettien muodostamiseen on, että termit, jotka esiintyvät usein samassa lauseessa tai samassa kappaleessa liittyvät samaan aihealueeseen. Termien välinen etäisyys lasketaan ja termeistä muodostetaan fasetteja sen mukaan.

Esimerkiksi valittujen hakuavaimien listalta lasketaan ensin:

1. ”Agentti”-termin keskimääräinen etäisyys muihin termeihin
2. ”Käyttöliittymäagentti”-termin etäisyys muihin termeihin
3. Tätä jatketaan kunnes saadaan aikaan lista jokaisen termin keskimääräisestä etäisyydestä toiseen termiin dokumentissa tai hakuavaimien lähdetekstin poiminta-alueella. Kaikkiaan arvoja tulee listalle n! kpl
4. Seuraavaksi yhdistetään vahvimmat termit toisiinsa muodostaen fasetteja näistä. Fasettien muodostuksessa käytetään AND-operaattoria

Fasettien käyttö ei ole pakollista, mutta ne kommunikoidaan aina hakuagenteille. Hakuagentit voivat näin kokeilla fasettien käyttöä. Tällöin myös hakuagentille on tärkeää ohjelmoida ymmärrys fasettien vaikutuksesta hakutulokseen.

9. Hakuagentti muodostaa viestin muille hakuagenteille ja lisää synonyymisanat tesauruksesta

Löydettyjen termien perusteella käyttöliittymäagentti voi vielä lisätä termejä tesaurusta avuksi käyttäen. Tätä haun laajentamista voidaan tehdä joko ennakkoon määritetyllä tesauruksella tai sitten automaattisesti luotavalla, sen hetkiseen aiheeseen liittyvällä tesauruksella. Valmiit tesaurukset voivat olla esimerkiksi tieteenalaan liittyviä, kun taas automaattisesti luotava tesaurus liittyy käyttäjän dokumenttiin. Olisi hyvä jos tesaurus voitaisiin myös liittää dokumentin mukaan, jolloin seuraavalla hakukerralla sen käyttö voidaan aloittaa heti. Tesauruksen käytöstä ja luomisesta on enemmän luvussa 5.3.2 dokumentin toisesta analyysikierroksesta.

Mikäli tesaurus on jo olemassa, käyttöliittymäagentti etsii siitä hakutermeiksi valittujen sanojen synonyymit ja lähettää ne hakuagenteille luvun 5.7 protokollan mukaisesti. Tämän jälkeen käyttöliittymäagentti jää odottamaan vastausta hakuagenteilta ja siirtyy seuraavaan dokumentin analyysikierrokseen.

5.3.2 Dokumentin toinen analyysikierros, hakuavaimien kehitys

Dokumentin toisella analyysikierroksella käyttöliittymäagentti kehittää hakutermejä käyttäjän palautteen ja löydettyjen dokumenttien perusteella. Kantavana ajatuksena on haun laajentaminen ja siten mahdollisesti uusien dokumenttien löytäminen tiedonlähteistä. Prosessi on iteratiivinen ja voi toistua useita kertoja haun aikana, riippuen käyttäjäpalautteesta ja tulosjoukon koon muutoksesta.

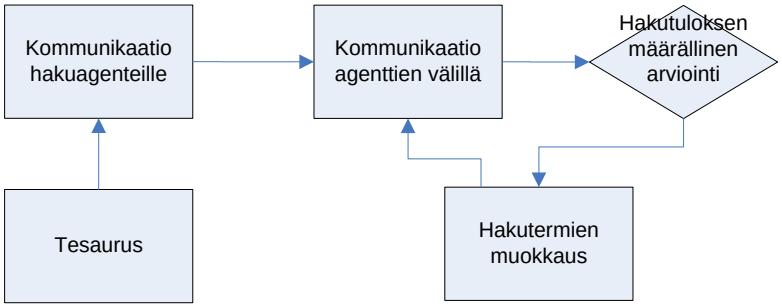
Lähteinä toiselle kierrokselle ovat:

- Käyttäjän palaute relevanteista dokumenteista
- Hakuagenttien raportoimat dokumentit
- Hakuagenttien antamat tiedot hakuavainten toimivuudesta tiedonlähteissä
- Käyttäjän palaute synonyymeistä

Toinen kierros käynnistetään aina jos

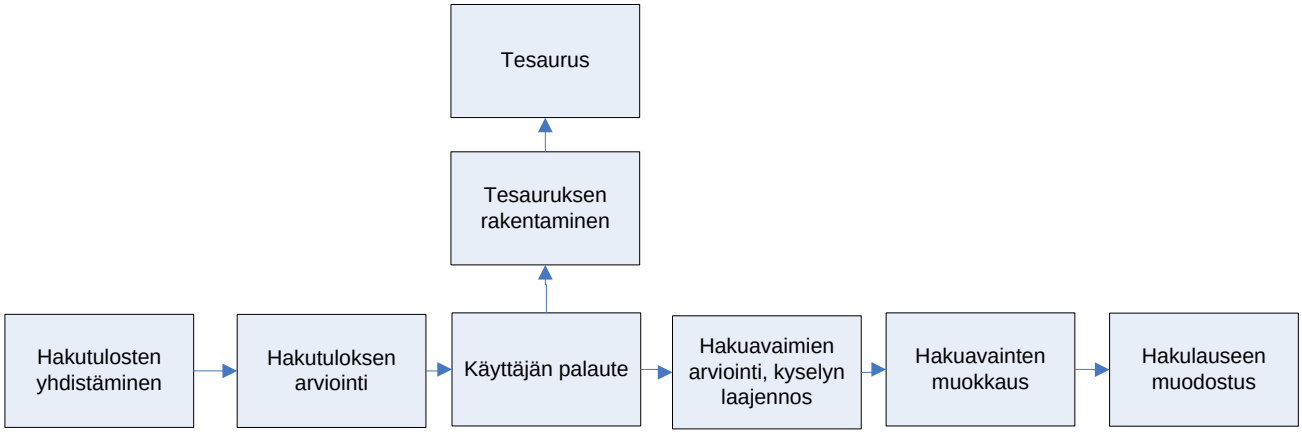
- Hakuagenttien haun perusteella saama tulosjoukon koko muuttui edelliseen hakuun verrattuna. Käyttöliittymäagentin tulee siis käyttää tässä muistiaan
- Käyttäjä antoi palautetta dokumenttien relevanssista tai muuttaa tesaurusta lisäämällä tai poistamalla sanoja)

Toinen hakukierros toimii iteratiivisesti:



Kuva 10. Haku käynnissä kommunikaatio agenttien välillä ja hakuagentin toiminta

Jokainen iteraatio suorittaa uuden hakukierroksen ja muokkaa hakua. Agentti tutkii hakuavaimia verraten hakuavaimien esiintymistä löydetyissä dokumenteissa, dokumenttikokoelmassa ja alkuperäisessä dokumentissa. Agentti voi myös laajentaa hakua eri kielille. Osa näistä eri toiminnoista onnistuu kuitenkin vain, jos dokumentit ovat saatavissa myös käyttöliittymäagentille; muussa tapauksessa hakuavaimien kehitys perustuu ainoastaan käyttäjän palautteeseen.



Kuva 11. Käyttöliittymäagentin toinen analyysikierros

Käyttöliittymäagentin toiminta toisen kierroksen aikana on esitetty seuraavassa esimerkissä:

1. Käyttöliittymäagentti lähettää hakuavaimet hakuagentille, joka prosessoi ne (kuva 10)
2. Hakuagentti suorittaa haun ja palauttaa vastauksen käyttöliittymäagentille
3. Käyttöliittymäagentti saa määräaikaan mennessä viestin kaikilta agenteilta ja yhdistää tuloksen (kuva 11)
4. Käyttöliittymäagentti arvioi tuloksen
5. Tulos esitetään käyttäjälle
6. Tesaurus luodaan löydettyjen dokumenttien ja käyttäjäpalautteen pohjalta
7. Haun laajentaminen, hakuavainten arviointi ja kieltenvälinen haku
8. Luodaan uusi hakulauseke ja se lähetetään hakuagenteille

1. **Käyttöliittymäagentti lähettää hakuavaimet hakuagentille joka prosessoi ne**

Käyttöliittymäagentti luo protokollan mukaisen hakupyynnön ja lähettää sen kaikille rekisterissään oleville hakuagenteille. Hakuagenteille annetaan tietty aika, aluksi esimerkiksi viisi sekuntia, toteuttaa haku järjestelmässään. Myöhemmin käyttöliittymäagentin tulee oppia hakuagenttien palautteesta ja lisätä tai vähentää aikaa tarpeen mukaan. Jos hakuagentti siis palauttaa aina viestin esimerkiksi sekunnin myöhässä, voidaan sen agentin aikaa lisätä. Käyttäjälle tuloksia tulisi kuitenkin esittää mahdollisimman nopeasti, vaikka osa agenteista olisikin vielä vastaamassa.

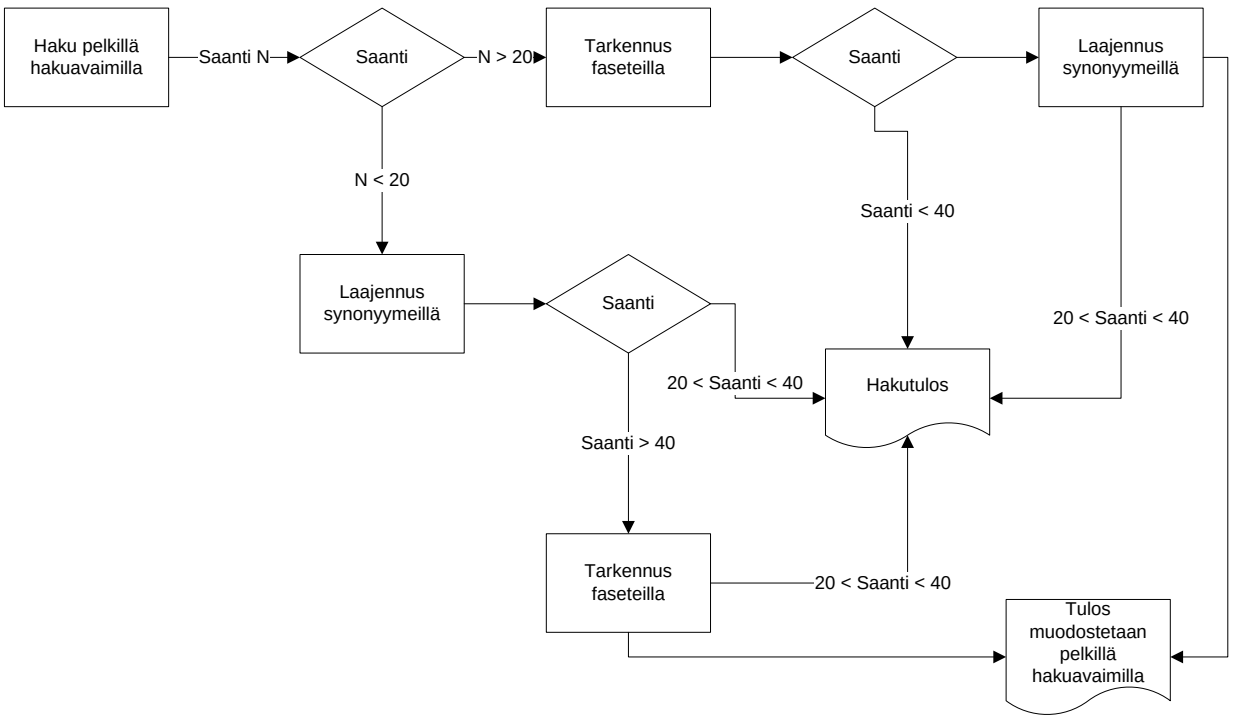
Hakuagentin saadessa viestin se purkaa viestin ja muodostaa juuri omalle hakujärjestelmälleen sopivan hakulausekkeen viestistä. Hakuagentti pitää siis ohjelmoida aina vastaamaan hakujärjestelmää. Helpoin tapa tähän on rakentaa itse agentti modulaarisesti siten, että se toteuttaa eri osat: kommunikaatiolle, tiedonkäsittelylle ja tiedonhaulle. Edellisistä vain tiedonhaku on muuttuva osa ja muissa käytetään sisäisesti abstrakteja muuttujia ja funktioita. Hakuagentti on tässä merkityksessä siis tulkkina hakujärjestelmän ja käyttöliittymäagentin välillä. Hakuagentti on syytä ohjelmoida siten, että se käyttää viestissä mukana tulleita tietoja mahdollisimman hyvin avuksi. Näitä tietoja ovat siis hakuavaimien lisäksi avainten painot, kieli, synonyymit ja fasetit.

2. **Hakuagentti suorittaa haun ja palauttaa vastauksen käyttöliittymäagentille**

Hakuagentti muodostaa hakulausekkeen käyttöliittymäagentin lähettämän tiedon perusteella ja suorittaa sitten kohdejärjestelmässä haun. Tämä on ainoa kohta, jossa hakuagentti muokkaa itse hakulausetta. Hakulauseen muodostus olisi syytä toteuttaa aina järjestelmästä riippuen siten, että ensimmäinen haku toteutetaan aina pelkillä hakuavaimilla ilman yhdistämistä tai synonyymejä. Mikäli haku tuottaa paljon lähteitä, voidaan hakua yrittää tarkentaa fasettien tai sanapainojen avulla. Mikäli tulosjoukko on suppea, hakua voidaan yrittää laajentaa synonyymeillä. Tämän jälkeen tehdään vielä toinen muutos riippuen haun tuloksesta. Tulosjoukon koon arviointi perustuu tässä vaiheessa pelkästään oletettuun minimiin. Myöhemmin arvoa voidaan muokata käyttäjän toiveiden mukaan.

1. Hakuavaimet
    a. Tulosjoukko suppea, laajennus synonyymeillä
    b. Tulosjoukko laaja, kavennus termien painoilla tai faseteilla

2. Hakutulos edellisestä
    a. Tulosjoukko suppea, laajennus synonyymeillä
    b. Tulosjoukko laaja, kavennus termien painoilla tai faseteilla
3. Tulos
    a. Tulosjoukko suppea tai liian laaja, haun toteuttaminen pelkästään hakutermein
    b. Tuloksen lähettäminen käyttöliittymäagentille



Kuva 12. Hakuagentin tuottama haun arviointi

Vastaus hakuagentilta käyttöliittymäagentille sisältää dokumenttiviitteiden lisäksi tiedot toimineista hakuavaimista ja faseteista, mikäli tiedonlähteestä tällainen tieto on saatavilla. Lisäksi analyysiä varten mukana on tieto tiedonlähteen dokumenttien määrästä ja kuinka monessa dokumentissa hakuavaimet esiintyivät. Tietoa käyttöliittymäagentti voi käyttää muodostaessaan seuraavaa kyselyä.

Kuvassa 12 näkyy tavoitteellisena tulosjoukkona agentille noin 20 dokumenttia, mutta tämän luvun tulisi olla joustava sen mukaan, miten hakujärjestelmästä saadaan dokumentteja. Toisin sanoen hakulogiikalle pitää asettaa muuttujaksi tavoite tulosjoukon koko n, jonka mukaan agentti toimii.

Arvo 20 voi hyvin toimia maksimiarvona järjestelmässä, mutta pienemmätkin tavoiteluvut ovat mahdollisia. Toimiessaan hakuagentti suorittaa siis automaattista haun laajennusta ja toisaalta kavennusta tuloksista riippuen ja toimii siten itsenäisesti ja oppivasti.

3. Käyttöliittymäagentti saa määräaikaan mennessä viestin kaikilta agenteilta ja yhdistää tuloksen

Käyttöliittymäagentti määrittää hakupyyntiä lähettäessään ajan, jonka se odottaa vastauksia hakuagenteilta. Jos käyttöliittymäagentti ei saa vastausta määräaikaan mennessä, se jättää hakutuloksen analysoimatta ja tuloksen ilmoittamatta käyttäjälle, mutta muuttaa odotusaikaa hakuagentilta pidemmäksi. Tällä pyritään siihen, että käyttäjä saisi tiedon hakutuloksista mahdollisimman nopeasti. Hakutuloksia voidaan päivittää myös jälkikäteen käyttäjän näkymässä sitä mukaan kuin tuloksia saapuu.

Joissain tapauksissa useat agentit voivat hakea tietoja samoista tiedonlähteistä. Silloin kokoavan agentin, kuten tämän työn käyttöliittymäagentin, tulee kyetä poistamaan hakutuloksesta samat linkit. Tästä syystä myös käyttöliittymäagentin tulee tietää alkuperäinen lähde. Hakuagentti kommunikoi lähteen käyttöliittymäagentille.

4. Käyttöliittymäagentti arvioi tuloksen

Käyttöliittymäagentti arvioi jokaisen hakuagentin lähettämän tuloksen tässä vaiheessa ainoastaan määrällisesti, ei siis vielä laadullisesti. Käyttöliittymäagentti pisteyttää sisäisesti tiedonlähteitä niiden antamien oletetusti relevanttien viitteiden perusteella ja myöhemmin käyttäjän palautteen perusteella. Tällä pyritään järjestämään linkkejä käyttäjän näkymässä siten, että paremmaksi osoittautuneista tiedonlähteistä tulevat dokumentit nousisivat ensimmäiseksi listalle. Järjestelyllä on merkitystä, mikäli tiedonlähteitä on useita ja saatu tulosjoukko on laaja. Myöhemmässä vaiheessa voidaan tiedonlähteitä arvottaa myös sen mukaan, kuinka hyvin hakuavaimet toimivat tiedonlähteessä käyttämällä esimerkiksi termien keskimääräisen esiintymistiheyden kaavoja.

5. **Tulos esitetään käyttäjälle**

Edellä kuvattu hakuagentin toiminta olettaa noin 20 viitteen tulosjoukkoa tiedonlähteestä. Järjestelyä tehdään siksi, että esimerkiksi viidestä tiedonlähteestä viitteiden määrä voi kivuta jo sataan viitteeseen.

Ensimmäisellä kerralla viitteet eri tiedonlähteisiin pyritään esittämään mahdollisimman heterogeenisesti, esimerkiksi seuraavalla tavalla:

Agentti A: 25 viitettä

Agentti B: 5 viitettä

Agentti C: 20 viitettä

Käyttäjälle tulos näytetään seuraavasti:

A, B, C, A, B, C, A, B, C, A, B, C, A, B, C, A, C, A, C, A, C ….

Lisäksi järjestelyssä voidaan käyttää hakuagenttien palautetta relevanssista. Riippuen kuitenkin tiedonlähteistä tämä ei välttämättä ole suositeltavaa, koska tiedonlähteet ilmoittavat (jos ilmoittavat) relevanssin suhteessa omaan dokumenttikokoelmaansa ja hakutermeihin, eikä arvio ole näin verrattavissa toisiin tiedonlähteisiin. Tämä ongelma on juuri se, minkä toinen tiedonhakukierros pyrkii ratkaisemaan.

Kun samaa dokumenttia ja samaa hakua käsitellään toisen kerran, voidaan käyttäjälle esitettyä näkymää muokata sen mukaan, kuinka paljon relevantteja dokumentteja missäkin tiedonlähteessä on käyttäjän palautteen mukaan ollut. Käyttäjä voi esimerkiksi kertoa käyttöliittymän kautta, että Agentti B:n viitteet ovat olleen relevanteimpia, joten niitä voidaan painottaa tuloslistassa suhteessa muihin. Käyttöliittymässä painotus voitaisiin esimerkiksi toteuttaa liukuvalinnalla, jonka oletusarvo on viitettä kohti 1, ja jota käyttäjä voi säätää välillä 0-2. Tiedonlähteen arvo kyseisellä hakukierroksella saadaan viitteiden arvojen tulolla. Tämä on kuitenkin vain yksi mahdollisuus käyttäjälle arvottaa dokumentteja. Yksi lisämääre, jota käyttöliittymäagentti voi käyttää järjestelyssä, on lähteen maksullisuus. Käyttäjä voi halutessaan pyytää käyttöliittymäagenttia painottamaan ilmaisia tai maksullisia tiedonlähteitä. Joka tapauksessa maksullisuus on syytä tuoda näkyviin jo tuloslistassa. Tuloksien järjestelyyn voi myöhemmin vaikuttaa myös haun laajennuksen ja dokumenttien relevanssipalautteen laskeminen (kohta 7).

6. Tesaurus luodaan löydettyjen dokumenttien ja käyttäjäpalautteen pohjalta

Hakuagenttien antamassa palautteessa on mukana myös tiedot toimineista hakuavaimista. Tämän tiedon perusteella hakuavaimia voidaan muokata. Toimimattomasta hakuavaimesta voidaan tehdä kolme oletusta:

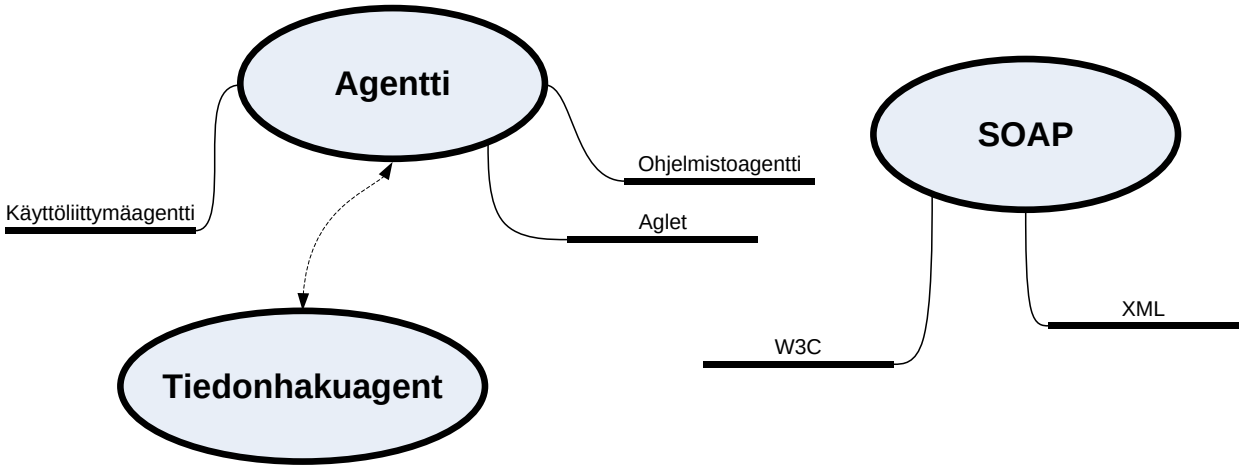
- Hakuavaimen perusmuotoistus ei ole onnistunut tai katkaisu on väärä
- Hakuavain on harvinainen tai kirjoitettu väärin
- Tiedonlähde on suhteessa hakuavaimeen väärä

Perusmuotoistus ja harvinaisuus voidaan ratkaista esittämällä käyttäjälle käytetyt hakuavaimet ja niiden hyvyys/huonous. Käyttäjälle tulisi antaa mahdollisuus synonyymien syöttämiseen ja hakuavaimien poistoon.

Esimerkkilistallamme olevat termit:

- agent
- tiedonhakuagentti
- soap
- information
- the
- http
- of
- avain
- käyttöliittymäagentti

Edellämainitut termit voidaan antaa käyttäjälle arvioitavaksi. Kirjoittajana osaan käyttöliittymän kautta poistaa termit ”of” ja ”the”, vaikka sulkulistaa ei olisi käytetty englannin kieleen ja laajentaa termejä seuraavasti:



Kuva 13. Esimerkki käyttäjän luomista assosiaatioista tesauruksessa.

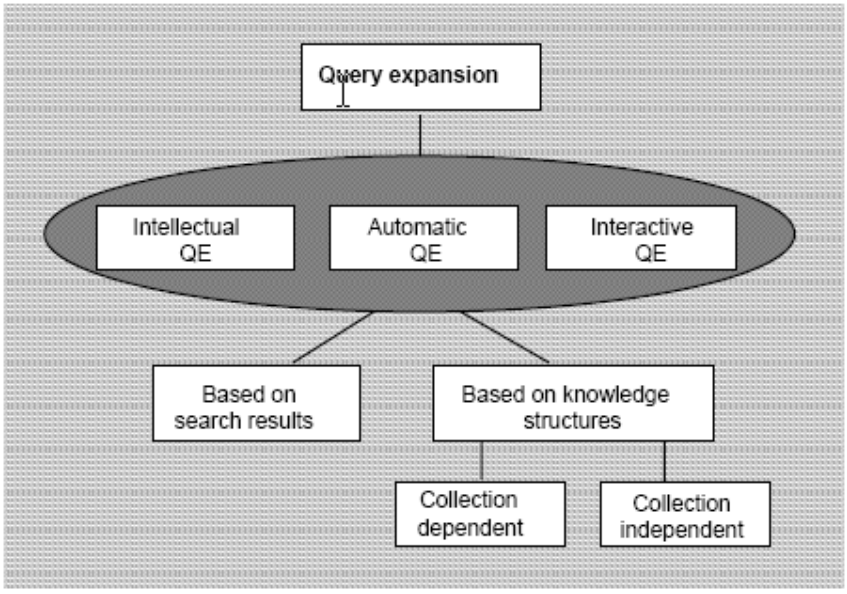
Huomattavaa on, että vaikka käyttäjältä kysytään synonyymejä, ei pelkkien synonyymien käyttäminen ole ratkaisevaa, vaan myös aihealueeseen liittyvien muiden sanojen kerääminen tesaurukseen. Tesaurusta voidaan laajentaa myös ohjelmallisesti. Tämä on mahdollista vain, jos käyttöliittymäagentilla on pääsy hakuagenttien tarjoamiin dokumentteihin kokonaisuudessaan ja mahdollisuus tehdä kyselyn laajennus viitattujen dokumenttien perusteella. Osa toiminnallisuudesta voitaisiin myös toteuttaa hakuagenteissa, mutta selvyyden vuoksi käsittelen nyt toteutusta keskitetysti käyttöliittymäagentissa.

7. Haun laajentaminen, hakuavainten arviointi ja kieltenvälinen haku

Ensimmäisellä kierroksella luotu hakulause voi olla jo sellaisenaan hyvä, mutta hakua voidaan silti laajentaa automaattisesti soveltaen tiedonhakututkimuksen kehittämiä metodeja. Kyselyn laajentamista on tutkittu paljon tiedonhauntutkimuksen piirissä: tiedonhaku on usein asiakkaiden palvelemista, jossa hakuavaimet muodostaa maallikko, jolloin ensimmäisten hakuavaimien määrä voi olla suppea ja jokainen aihealue on ilmaistu vain yhdellä termillä (Bates, Wilde & Sigfried 1993, 23-25.). Käyttöliittymäagentin luoma hakuavain voi myös olla samaan tapaan rajoittunut ja sen laajentaminen onkin siksi tarpeen.

Ensimmäinen tapa laajentaa kyselyä on termien lisääminen ilman hakuavainten uudelleen arviointia. Tätä teemme jo osittain käyttäjän palautteen perusteella tesauruksen avulla ja samaa tekevät myös hakuagentit omassa prosessissaan. Käyttöliittymäagentti toteuttaa siis sekä käyttäjän

aktiivisen että passiivisen roolin: käyttäjälle annetaan mahdollisuus lisätä hakutermejä (Intellectual QE), mutta niitä myös lisätään käyttäjälle näkymättömästi (Automatic QE). Lisäksi käyttäjällä tulisi olla mahdollisuus katsella automaattisesti luotua tesaurusta ja tehdä siihen muutoksia tarpeen mukaan (interactive QE). (Kekäläinen 1999, 32.).



Kuva 14. Kyselynlaajennuksen metodit (Kekäläinen 1999, 32)

Toinen tapa, jolla käyttöliittymäagentti voi tarkentaa ja laajentaa hakua, on termien hyvyyden arviointi. Edellä esitin kuinka löydämme hakutermejä dokumentista, ja kuinka niitä voidaan järjestellä painojen mukaan. Toinen lähestymistapa aiheeseen on, kuinka arvioida hakuavaimien relevanssi käyttäen keskimääräistä esiintymistiheyttä tunnetussa dokumenttikokoelmassa ja dokumenteissa. Jotta näin voisi tapahtua, käyttöliittymäagentin pitää saada lisää tietoa tiedonlähteestä ja tämän tiedon voi hakuagentti sille kommunikoida. Mikäli tieto on saatavilla, voidaan haussa käytettyjen termien esiintymistiheyden perusteella arvioida hakutermien relevanssia suhteessa tiedonlähteeseen ja näin päätellä termien hyvyyttä/huonoutta haun kannalta. Tämän laskelman avulla voidaan hakua muuttaa ja vaihtaa termejä. Toisaalta voidaan myös tehdä arvioita löydettyjen dokumenttien kautta siten, että ensimmäinen kierros kohdistetaankin löydettyihin dokumentteihin ja pyritään sitä kautta laajentamaan hakua. Hakua voidaan lisäksi laajentaa kieltenvälisellä tiedonhaulla, mikäli tarjolla on sanakirjoja ja erikielisiä tiedonlähteitä.

Kieltenvälinen haku

Termien keskimääräisen esiintymistiheyden teorian esittivät esimerkiksi Pirkola, Leppänen ja Järvelin (2002). Esitelty termien arviointi perustuu RATF-kaavaan, jolla voidaan arvioida termien hyvyyttä tiedonlähteessä, esimerkiksi dokumenttitietokannassa. RATF on kokoelmataseen laskentaa, kun taas Zipf- ja tf.idf-kaavat keskittyvät sanojen frekvenssiin dokumentissa. Tutkimuksen pääoletus oli, että RATF-metodeja voitaisiin käyttää myös kieltenvälisessä tiedonhaussa. Kieltenvälisessä tiedonhaussa on kyse nimenomaan siitä, että alkuperäiset hakuavaimet ovat eri kielellä kirjoitettuja kuin kohdejärjestelmän dokumentit. RATF laskee hakutermeille relatiivisen termifrekvenssin. Käytännössä termeille lasketaan niiden luokitus, jonka perusteena on termin yleisyys kokoelmassa (collection frequency/cf) suhteessa dokumenttien määrään, joissa termi esiintyy (document frequency/df).

RATF-kaava on seuraava:

$$RATF(k) = (cf_k / df_k) * 10^3 / ln(df_k + SP)^p$$

k on hakuavain

cf_k on hakuavaimen kokoelma frekvenssi

df_k dokumenttien määrä, joissa hakuavain esiintyy

SP ja p kokoelmasta riippuvaisia vakioita

Käyttöliittymäagentin tulisi siis saada palautteen ohessa tiedot:

- Kuinka usein termit esiintyvät kokoelmassa (collection frequency)
- Dokumenttien määrä, joissa termit esiintyvät (document frequency)

Nämä tiedot eivät ole välttämättä kaikissa tiedonlähteissä tarjolla, mutta niissä missä ne ovat, voidaan hakuavaimien relevanssi laskea kokoelmassa.

RATF-kaavan käytössä haasteena on oikeiden SP- ja p-arvojen valinta, koska hakujärjestelmää ei tunneta välttämättä ennalta. Siksi kieltenvälisen haun toteuttaminen olisi tehokkainta

hakujärjestelmälle tehdyssä hakuagentissa käyttöliittymäagentin sijaan. Tämäntyyppinen toteutus saattaisi hankaloittaa sanakirjojen saatavuutta, sillä kun sanakirjat ovat keskitettyjä, jokaisella agentilla tulee olla yhteys sanakirjaan, mikä saattaa hidastaa hakua ja on myös vastaan suunnitteluperiaatetta, että hakuagentti toimii tiedonlähteen luona.

Mikäli agentti käyttäisi RATF-arvoa hyväkseen Pirkolan, Leppäsen ja Järvelinin esittämään tapaan, agentin tulisi käydä ensin läpi ensimmäinen hakukierros ja arvioida alkuperäisten sanojen RATF-arvo (Pirkola et al. 2002). Agentin täytyy näin ollen päästä myös tuloksena tulleisiin kokoteksteihin tai saada hakuagenteilta palaute sanojen esiintymisestä dokumenteissa. Kyseinen ominaisuus on sisään rakennettu agenttien käyttämään protokollaan. Tämän jälkeen agentin tulisi kääntää sanakirjan avulla hakuavaimet, suorittaa haku ja verrata tämän haun RATF-arvoja alkuperäisiin RATF-arvoihin.

Kieltenvälisellä tiedonhaulla voidaan hakua laajentaa hyvinkin tehokkaasti kattamaan vieraskielisiä dokumentteja. Erityisesti, mikäli suunniteltu käyttöliittymäagentti on liitetty tekstinkäsittelyohjelmiston yhteyteen, voi haun laajentaminen toiseen kieleen olla hyvä ratkaisu. Esimerkiksi tätä työtä kirjoittaessa lähes kaikki materiaali on englanninkielistä, mutta työ itsessään suomenkielinen, joten kieltenväliselle tiedonhaulle on selkeä tarve. Termien kääntäminen voidaan agentissa yhdistää kohtaan, jossa tesaurusta luodaan. Selkeintä on kuitenkin pitää tesaurukset omina kieliversioinaan. Erityinen hyöty saadaan eri kieliversioista, mikäli hakuagentti osaa kertoa käyttöliittymäagentille, minkä kielinen hakujärjestelmä on kyseessä. Toisaalta, mikäli käyttöliittymäagentti osaa kertoa hakuagenteille myös minkä kielisistä termeistä on kyse, voi hakuagentti mahdollisesti käyttää tehokkaammin hakujärjestelmää.

Erikieliset hakuavaimet ovat myös hyvin ehdokkaita fasettien rakentamiseksi. Fasetti eli sanoista muodostettu joukko, jossa termit on yhdistetty loogisilla operaatioilla, voisi olla esimerkiksi ((Information AND retrieval) OR Tiedonhaku). Fasetteja voidaan kuitenkin käyttää vain, jos tiedonlähde tukee niiden käyttöä. Muussa tapauksessa tulee tiedonhakuagentin muokata fasettien käyttö tiedonlähteelle sopivaksi. Sanakirjojen avulla automaattisesti käännettyjen hakuavaimien hyvyyttä tulisi arvioida kriittisesti. Sanakirjojen ongelma olevan siinä, että ne antavat tyypillisesti useita eri käännöksiä samalle sanalle ja usein aiheeseen liittymättömien käännettyjen sanojen lukumäärä on iso. Pirkola, Leppänen ja Järvelin totesivat tämän artikkelissaan ja ehdottavat siksi

RATF-kaavaa käytettäväksi kieltenvälisten tiedonhakujen parantamiseksi (Pirkola et al. 2002). Jos agenttiteknologilla halutaan käyttää sanakirjoja automaattisesti tiedonhaun laajentamiseen, RATF-kaavan käyttö hakutermien arvioinnissa on välttämätöntä, koska agentti tekee päätelmiä hakutermien toimivuudesta täysin ilman käyttäjän apua.

Kieltenväliseen tiedonhakuun liittyvä yksi mielenkiintoinen havainto, joka tehtiin lähdeaineistoa analysoidessa, oli että Microsoft Word kykeni tunnistamaan tästä dokumentista ”Abstract” luvun automaattisesti englanninkieliseksi. Myös tätä voitaisiin käyttää haussa hyväksi siten, että käytettäessä kokotekstiä hakutermien lähteenä, voisi erikieliset osuudet erottaa omiksi alueikseen haussa, tai käyttää itse dokumenttia oikeiden käännösmuotojen etsimiseen sanakirjan tarjoamien vaihtoehtojen joukosta. Kun ensimmäisen kierroksen hakutermien luonti kohdistettiin pelkkään lyhyeen englanninkieliseen lukuun, hakutermit muodostuivat seuraavasti:

Termi, abstract (Zipf, stopword)	**Termi, Johdanto (Zipf, sulkusanat)**
agent	Agentin
information	Agenttien
work	Informaatiotutkimuksen
problems	Keräävät
searle’s	Toteutus
give	Työn
course	Agentit
tools	Tietoa
applications	Normaaleja
published	Toteutetun
seeking	Lähtökohdasta
technology	Esittämään
minds	Muotoon
source	Tiedonhakuprosessiin
evaluate	Sisällöstä

Taulukko 8. Johdantolukujen vertailu

Suomenkielisestä listasta on poistettu englanninkieliset termit käsin vertailun vuoksi. Taulukosta käy mielenkiintoisesti esiin, miten Abstract edustaa koko suomenkielistä versiota, ja kuinka samantyyppisiä termejä molemmista alueista löytyi.

Haun laajentaminen

Hakuavaimia voidaan ensin kehittää tiedonlähdekohtaisesti siten, että mikäli tiedonlähde antaa palautetta hakutermien toimimisesta (mm. esiintymistiheys), hakulause eroaa näissä järjestelmissä suhteessa tiedonlähteisiin, jotka eivät anna tarpeeksi informaatiota hakuavaimien toimimisesta kokoelmassaan. Lisäksi, mikäli hakuavainten hyvyys voitaisiin laskea esimerkiksi kahdessa järjestelmässä viidestä ja käyttäjä omalla palautteellaan suosisi näistä järjestelmistä saatuja dokumentteja haun tarkennuksen jälkeen, voidaan harkita hakuavaimien hyvyyden perusteella muokatun hakulauseen yleistämistä myös muihin järjestelmiin, josta palautetta ei ollut saatavilla. Yleistäminen vaatisi kuitenkin kokeellista tutkimusta, jotta sen toiminta voitaisiin todistaa ennen toteuttamista agenttiin.

Voidaanko sen sijaan kyselyä laajentaa muuta kautta kuin kieltenvälisen haun avulla? Efthimiadis vertaili erilaisia kyselyn laajentamiseen sopivia algoritmeja (Efthimiadis 1993, 147-150). Efthimiadisin mukaan F4-algoritmin pohjalta muokattu $w_t(p_t-q_t)$ oli koeasetelmassa yksi tehokkain kyselyn laajennuksessa, joten esittelen sen tässä vaihtoehtoisena tai RATFia täydentävänä. F4-algoritmi perustuu ideaan relevanssin arvioimisesta avaimien löytämiseksi. F4 lähtee oletuksesta termien riippumattomuudesta ja dokumenttien järjestyksestä.

Kaava relevanssien arvioimiseen eli binäärinen riippumattomuus (BIM Binary Independence retrieval Model) on

$$w_t = \log \frac{p_t(1-q_t)}{q_t(1-p_t)}$$

(BIM formula)

W_t = hakuavainen t, paino

P_t= todennäköisyys, että hakuavain esiintyy relevantissa dokumentissa

Q_t=todennäköisyys, että termi esiintyy epärelevantissa dokumentissa

(Efthimiadis 1993, 147)

Kaavan soveltaminen vaatii, että hakuavaimen relevanssista dokumenteissa on jotain tietoa (kuten vastaava RATF). Efthimiadis kuitenkin esittää, että todennäköisyys q ja p voidaan myös arvioida käyttäjän palautteesta. F4-kaava esitettynä on seuraava:

$$w_t = \log \frac{\frac{r}{R-r}}{\frac{n-r}{N-n-R+r}} = \log \frac{r(N-n-R+r)}{(n-r)(R-r)}$$

(F4 formula)

Tai

$$w_t = \log \frac{(r+.5)(N-n-R+r+.5)}{(n-r+.5)(R-r+.5)}$$

(F4 point-5 formula)

Missä äskeiset todennäköisyydet on ilmoitettu seuraavasti:

p_t = r/R ja q_t =n-r/N-R.

N on dokumenttien kokonaismäärä järjestelmässä

R on relevanttien dokumenttien määrä käyttäjän palautteen perusteella

n on dokumenttien määrä jotka on luokiteltu t:n mukaan

r on relevanttien dokumenttien määrä näytteestä R joissa hakuavain t esiintyy.

(Efthimiadis 1993, 148)

F4 ottaa siis huomioon käyttäjän palautteen arvioitaessa löydettyjä dokumentteja. Kyselyn tarkennus voidaan jo yrittää tätä kautta tunnistamalla relevantteja hakutermejä.

F4-algoritmi on alun perin Robertsonin ja Jonesin kehittämä (Efthimiadis 1993, 147). Robertsonin lisäykset algoritmiin (Robertson 1986, 186) tekevät F4-algoritmista käyttökelpoisemman kyselyn automaattisessa laajentamisessa. Uusi kaava (F4modified) ottaa nimittäin huomioon uusien termien lisäämisen kyselyyn (Efthimiadis 1993, 148).

$$w_t = \log \frac{(r+c)(N-n-R+r+1-c)}{(n-r+c)(R-r+1-c)}$$

(F4Modified)

c = n/N

r, R, n ja N ovat samoja kuin alkuperäisessä kaavassa F4, mutta tätä kaavaa voisi Robertsonin (1986, 186) mukaan käyttää automaattisessa kyselynlaajentamisessa siten, että jokainen termi käyttäjän relevantiksi merkitystä dokumentista painotettaisiin kaavalla:

c=n/N

ja suurimman relevanssi painon saaneet lisättäisiin hakulauseeseen. Käyttäjän itse valitsemille tai lisäämille hakuavaimille painojen laskenta sen sijaan voidaan tehdä F4point5-kaavalla.

Efthimiadis löysi kritiikkiä vielä F4modified ja F4point5 kaavoissakin, joista Robertson kehitti vielä version, ns. $w_t(p_t - q_t)$ kaavan, jolla päästäisiin edellisiä kaavoja parempaan tulokseen. Molemmat kaavat nimittäin tuottivat lähes samanlaisen tuloksen ja hypoteesi oli, että olisi olemassa parempi kaava laskea relevanssipainoja termeille, joita kyselyn laajentamisessa käytettäisiin. (Efthimiadis 1993, 149) Efthimiadis tutki tätä kaavaa empiirisesti työssään ja totesi sen tehokkaaksi. Siksi se on valittu vaihtoehtoiseksi tavaksi toteuttaa kyselyn laajennus tämän tutkimuksen agentissa. Algoritmi perustuu ajatukseen, että hakua laajennettaessa voidaan tehdä oletus termien riippumattomuudesta toisistaan. Alkuperäisen haun termien tulee olla riippumattomia uudesta termistä, jolloin uusi termi laajentaa mahdollisimman tehokkaasti hakua. Tulee erottaa kaksi osajoukkoa: relevantit dokumentit, joissa arvioitavaa termiä ei esiinny ja relevantit dokumentit, joissa se esiintyy. Kyseessä ovat siis uusi ja vanha tulosjoukko, jotka ovat omat osajoukkonsa. Mikäli tätä ei huomioida, hakuavaimen relevanssin arviointi voi vääristyä.

Hypoteesi on, että hakuavaimen t, jolla on paino w_t, lisää hausta saadun tulosjoukon relevanssia:

$a_t = w_t(p_t - q_t)$

missä paino w_t on F4point5 kaavalla laskettu paino termille

p_t on todennäköisyys, että hakuavain t esiintyy relevantissa dokumentissa

q_t ollessa vastakkaisesti todennäköisyys, että hakuavain t ei esiinny relevantissa dokumentissa

Termien vertailu keskenään tulisi siis perustua ennemmin arvoon a_t kuin pelkkään painoon w_t .

Kun kaava Wt (F4point5) sisällytetään kaavaan, saadaan lopuksi:

$$a_t = \log \frac{(r+.5)(N-n-R+r+.5)}{(n-r+.5)(R-r+.5)} \cdot (\frac{r}{R} - \frac{n-r}{N-R})$$

($w_t(p_t - q_t)$ avattuna)

(Efthimiadis 1993, 148-150).

Yllä olevaa kaavaa voidaan sellaisenaan käyttää arvioitaessa hakutermien relevanssia tiedonlähteillä, joille hakuagentti on asiakas. Edelleenkin arvio tulee tehdä tiedonlähdekohtaisesti, ei suhteessa suoraan käyttöliittymäagentin näkemään eri lähteistä yhdistettyyn tulosjoukkoon. Yksi mahdollinen tapa hyväksikäyttää sekä RATF-kaavaa että $w_t(p_t - q_t)$ kaavaa olisi yhdistelmä, jossa agentti käyttäisi RATF-kaavaa kieltenväliseen (katso luku 4.3.3.) tiedonhakuun ja sitä kautta haun laajentamiseen uusiin lähteisiin ja lisäksi $w_t(p_t - q_t)$ kaavaa haun laajentamiseen.

8. Luodaan uusi hakulauseke ja se lähetetään hakuagenteille

Haun laajentamisen perusteella käyttöliittymäagentti voi luoda nyt uuden kyselyn alkuperäisten hakuavaimien, haun laajennuksessa löytyneiden avaimien ja käyttäjän tesaurukseen syöttämien avaimien perusteella. Termit on useimmiten loogista yhdistää faseteiksi Boolen operaattorilla OR, jolloin hakuagentilla on tiedossa termien suhteet, ja se voi tehokkaammin käyttää hakuavaimia hyväkseen tiedonlähteen näin salliessa. Mikäli tiedonlähde ei tue suoraan boolen operaattoreita, tulee hakuagentin tulkita hakulause tiedonlähteelle sopivaksi.

Lopuksi hakulauseke luodaan protokollan mukaisesti ja toimitetaan hakuagenteille. Samalla haun ensimmäinen vaihe käynnistyy uudelleen ja käyttöliittymäagentti jää odottamaan vastauksia hakuagenteilta suorittaakseen toisen vaiheen uudelleen.

5.4 Agenttien välinen kommunikaatio

Tiedonhaun tehostamiseksi ja tulosten luokittelun mahdollistamiseksi agenttien tulee kyetä kommunikoimaan keskenään tavalla, joka on yksiselitteinen ja mahdollisimman kevyt tietoliikenteen kannalta. Tietoliikenteen kannalta on oleellista protokollan keveys, jotta määrällisesti usean tiedonlähteen käyttö on mahdollista ja viitteitä saadaan lähteestä nopeasti. Tehokkaimmillaan tiedonhakuagentti olisi esimerkiksi samassa järjestelmäarkkitehtuurissa tiedonlähteen kanssa. Mikäli tämä ei ole mahdollista, kannattaa sijainti toteuttaa verkkoteknisesti siten, että tiedonlähteelle on nopea verkkoyhteys. Hakuagentin ja käyttöliittymäagentin välinen kommunikaatio pitää taas suunnitella siten, että ei turhaan kuormiteta tiedontarvitsijan verkkoyhteyksiä.

Toteutuksen helpottamiseksi, jokaisen järjestelmässä olevan agentin tulee siis toteuttaa sama kommunikaatioprotokolla. Tässä tutkimuksessa käytetty protokolla perustuu SOAPiin (Simple Object Access Protocol) (W3C, SOAP Versio 1.2) . Muutakin kuljettavaa protokollaa voidaan käyttää; tärkeintä kuitenkin on yksiselitteiset syötteet ja vasteet. SOAP versio 1.2 on kevyt protokollatoteutus, jolla voidaan siirtää rakenteista tietoa hajautetussa ympäristössä, millainen myös suunniteltu tiedonhakuagentti ideaalisessa toteutuksessa on. SOAP perustuu tekstipohjaisen formaatin XML:än käyttöön ja se mahdollistaa näin myös hyvän laajennettavuuden protokollalle; ja toisaalta siirtoon voidaan käyttää monenlaisia alemman tason tiedonsiirtoprotokollia, kuten esimerkiksi http (HyperText Transfer Protocol). Etuna on, että http-liikenne on sallittu liikennöintiprotokolla lähes kaikissa tietoverkoissa, mukaan lukien yritysten ja yliopistojen verkot. Koska http-protokollaa käytetään kantavana protokollana, mutta sisältö ei ole tavallista xhtml/html-viestintää, on http-viestin sisältötyypin arvo (Content-type) asetettava arvoon application/soap+xml. Tässä työssä en kuitenkaan http-protokollaa käsittele enempää. Enemmän tietoa protokollasta löytyy IETF:n sivuilta (IETF RFC 2616). Viestintä tulee toteuttaa kokonaisuudessaan täyttämään SOAPin määritykset, ja itse tietosisältö on sijoitettava ns. envelope-viesteihin, joiden rakenteen esitän seuraavaksi (W3C, SOAP Versio 1.2, 1.).

5.4.1 XML/SOAP toteutus

Agenttien välinen kommunikaatio voidaan yleistää kahteen pääkategoriaan:

- Hakuavaimet
- Hakutulokset

Jotta hakuagenteilla olisi tarpeeksi lähtökohtia suorittaa hakuja, tulee käyttöliittymäagentin lähettää viesti, mikä sisältää seuraavat elementit:

```
Viesti(hakuavainten määrä, fasettien määrä, voimassaoloaika,
viestin tunniste,
Hakuavain(paino, kieli, järjestysnumero, synonyymitunniste,
hakuavain),
Fasetti(fasettitunniste(fasetti/avain tunniste OPERAATTORI
fasetti/avain tunniste)))
```

Viestien rakenteessa elementit tarkoittavat seuraavaa:

Header-envelope Viesti sisältää viestin perustiedot, joiden perusteella viesti voidaan tunnistaa ja sen voimassaoloa seurata.

Header-envelope sisältää seuraavat elementit:

Header-envelope, viittaa SOAP-protokollan (W3C, SOAP Versio 1.2, 1.) ensimmäiseen elementtiin, mikä kuvaa jäljessä tulevan viestin metatiedot. Kuvatussa ” **Header-envelope”** elementissä on sisällä seuraavat tiedot.

Hakuavainten määrä kertoo hakuagentille odotettavissa olevan hakuavaimien määrän. Tätä tietoa voidaan käyttää hyödyksi tarkistettaessa hakuavainten mahdollista käyttörajoitusta järjestelmässä.

Fasettien määrä kertoo tiedonhakuagentille mukana olevien fasettien määrän.

Voimassaoloaika kertoo kuinka kauan käyttöliittymäagentti odottaa tiedonhakuagentin vastausta. Arvo asetetaan käytännön syistä rajoittamaan ylimääräistä verkkokuormaa ja viivettä käyttäjän järjestelmässä. Voimassaoloaika asetetaan lähteestä riippuen, mutta käytännöllisintä on pitää se noin viiden sekunnin maksimiarvossa.

Arvo on muotoa: vuosi-kuukausi-päiväTtunnit:minuutit:sekunnit eli esimerkiksi 2010-05-01T12:00:05

Viestin tunniste on uniikki tunniste, koska hakuagentti voi saada viestejä usealta tiedonhakuagentilta ja useita kappaleita. Helpoin tapa muodostaa tunniste on käyttää ns. tiivistealgoritmeja. Esimerkissä käytetään md5- tiivistettä (RFC1321 - The MD5 Message-Digest Algorithm)

Esimerkiksi:

```
md5("2010-05-01T12:00:05 AgenttiA.uta.fi") =
"435656d71ac488c1295ccab4bbcc3136"
```

Esimerkin tiiviste muodostetaan vanhenemisajasta ja AgenttiA (käyttöliittymäagentti) domain-nimestä. Pitää huomioida, että MD5-algoritmi valittiin vain sen yleisyyden vuoksi ja, kirjoittaja on tietoinen algoritmiin liittyvistä tietoturvallisuusongelmista.

Body-envelope Avain sisältää viestin sisällön eli hakuavaimet ja fasetit. Tämä viesti sisältää myös kaikki synonyymit, joita käyttöliittymäagentti on termeille löytänyt. Jokainen synonyymi ja sen pääsana saa oman synonyymitunnisteensa.

”**Body-envelope”**, viittaa SOAP protokollan alun jälkeiseen elementtiin, mikä sisältää itse viestin tietosisällön (W3C, SOAP Versio 1.2, 1.).

Paino elementti kertoo hakuagentille käyttöliittymäagentin hakuavaimelle määrittämän järjestysnumeron suhteessa muihin toimitettaviin hakuavaimiin. Paino voi kuitenkin olla sama useallekin termille.

Kieli elementti kertoo hakuavaimien kielen. Tätä tietoa voidaan käyttää eri tiedonlähteissä hyväksi tai kieltenvälisessä tiedonhaussa.

Järjestysnumero. Jos hakuavaimia on esimerkiksi 15, jokainen avain saa juoksevan numeron välillä 1-15. Tällä hakuavain tunnistetaan viestissä, ja vältetään hakuavaimien siirtämistä useaan kertaan agenttien välillä

Synonyymitunniste on numero, joka synonyymiryhmälle annetaan, mikäli niitä on. Synonyymitunnisteen avulla hakuagentti tunnistaa kaksi samaa tarkoittavaa termiä. Sama asia voidaan ilmaista fasetein, mutta synonyymitunnisteiden käyttö on yksinkertaisempaa ja varaa fasetit muuhun käyttöön. Synonyymitunniste on kokonaisluku ja sama kaikille samaa tarkoittaville termeille mitä viestin mukana tulee.

Hakuavain on itse löydetty termi viestin mukaisella kielellä.

Body-envelope Fasetti on viimeinen elementti ja se sisältää hakulauseen fasettirakenteet, mikäli niitä on. Fasetti rakennetaan yksinkertaisesti kokoamalla fasettiin kuuluvien termien tunnisteet yhteen hakutermien joukoksi, sekä fasettien tunnisteet yhteen tai termien ja fasettien tunnisteet yhteen. Lopputuloksena on hakulause, jossa on boolen operaattoreilla kerrottu hakuagentille termien suhde toisiinsa hakulauseessa.

Fasetti tunniste on fasetin juokseva numero lisättynä prefiksillä F, esimerkiksi F7.

Fasetti sisältää tunnisteet, jotka yhdistetään yksinkertaisilla Boolen operaattoreilla, esimerkiksi 1 AND 43 OR 2 AND F7.

Esimerkki SOAP viesti (lähettäjänä käyttöliittymäagentti):

```
<?xml version="1.0"?>
<soap:Envelope xmlns:env=http://www.w3.org/2003/05/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <env:Header>
            <agent:viesti xmlns:agent="http://example.org/agent">
                    <agent:hakuavaintenmaara>5</agent:hakuavaintenmaara>
                    <agent:fasettienmaara>1</agent: fasettienmaara>
                    <agent:vanhenee>2010-05-01T12:00:00</agent:vanhenee>
                    <agent:tunniste>d56a40dc69a3c8e7fbea1ac6f10345f6
                    </agent:tunniste>
            </agent:viesti>
    </env:Header>
    <env:Body>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:paino>2</agent:paino>
                    <agent:kieli>FI</agent:kieli>
                    <agent:jarjestysnro>1</agent:jarjestysnro>
                    <agent:avain>information</agent:avain>
            </agent:avain>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:paino>1</agent:paino>
                    <agent:kieli>FI</agent:kieli>
                    <agent:jarjestysnro>2</agent:jarjestysnro>
                    <agent:synonyymitunniste>1</agent:synonyymitunniste>
                    <agent:avain>retrieval</agent:avain>
            </agent:avain>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:paino>1</agent:paino>
                    <agent:kieli>FI</agent:kieli>
                    <agent:jarjestysnro>3</agent:jarjestysnro>
                    <agent:avain>agent</agent:avain>
            </agent:avain>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:paino>1</agent:paino>
                    <agent:kieli>FI</agent:kieli>
                    <agent:jarjestysnro>4</agent:jarjestysnro>
                    <agent:synonyymitunniste>1</agent:synonyymitunniste>
                    <agent:avain>search</agent:avain>
            </agent:avain>
```

```
            <agent:fasetti xmlns:agent="http://example.org/agent">
                    <agent:fasettinro>F1</agent:fasettinro>
                    <agent:tunnisteet>1 OR 2</agent:tunnisteet>
                    <agent:fasettinro>F2</agent:fasettinro>
                    <agent:tunnisteet>F1 AND 3 AND 4</agent:tunnisteet>
            </agent:fasetti>
    </env:Body>
</env:Envelope>
```

Tulosten vertailun mahdollistamiseksi vastaus tulee olla muotoa:

```
Viesti(viitteiden määrä, viestin tunniste,
Avain(avain, rank, esiintyminen dokumenttikokoelmassa, dokumenttien
määrä joissa esiintyy),
Dokumentti(dokumentin_otsikko, viite, järjestysnumero, kieli,
lähde, tiivistelmä))
```

Header-envelope Viesti sisältää viestin tunnisteen lisäksi viestissä olevien viitteiden määrän.

Viitteiden määrä on kokonaisluvulla ilmaistuna kaikkien lähetettävien viitteiden lukumäärä.

Viestin tunniste on vastaava MD5 tiiviste kuin edellisessä viestissä.

Body-envelope Avain sisältää tiedonhakuagentin palautteen avaimien toimivuudesta. Viesti koostuu elementeistä järjestysnumero ja rank.

Järjestysnumero on käyttöliittymäagentin viestissään välittämän hakuavaimen yksiselitteinen tunniste (kokonaisluku).

Rank on tiedonhakuagentin hakuavaimelle antama hyötyarvio suhteessa tulosjoukon kokoon. Arvo voi olla 0-10, jossa numerolla kuvataan hakuavaimen vaikutusta tulosjoukon kokoon, mutta on huomattava että siinä ei kuvata hakuavaimen vaikutusta viitteiden relevanssiin tulosjoukossa. Asteikolla 10 on paras ja 1 huonoin. 0 on erikoistapaus, jolla viitataan, että dokumentteja ei saatu tai termi oli väärää muotoa.

Esiintyminen dokumenttikokoelmassa eli cf_k, mikäli hakujärjestelmä tämän tarjoaa.

Dokumenttien määrä joissa esiintyy eli df_k, mikäli hakujärjestelmä tämän tarjoaa.

Body-Envelope Viite sisältää itse tiedonhaun tuloksen kyseisessä lähteessä. Viite elementtejä on yhtä monta kuin löydettyjä lähteitä (0-n).

Otsikko on järjestelmästä kopioitu käyttäjälle esitettävä dokumentin otsikko kuten: “Remembrance Agent: A continuously running automated information retrieval”.

Viite on linkki, usein URI muotoinen (Uniform Resource Identifier) tai muu viite tiedonlähteeseen, josta dokumentti löytyy. Esimerkiksi ”https://www.aaai.org/Papers/Symposia/Spring/1996/SS-96-02/SS96-02-022.pdf”

Järjestysnumero on joko tiedonlähteen dokumentille antama relevanssiluku muunnettuna käyttöliittymäagentin ymmärtämään muotoon tai järjestysnumero tiedonlähteestä saadussa tulosjoukossa. Esimerkiksi 10 dokumentin tulosjoukossa dokumenteilla olisi arvot 1-10.

Kieli on viitteen kieli, mikäli arvo on saatu tiedonlähteenä olevasta järjestelmästä.

Lähde kertoo vielä erikseen tiedonlähteen viitteen mahdollisesti URI-muodossa.

Tiivistelmä on tiedonlähteen tarjoama lyhennelmä dokumentista, mikäli sellainen on tarjolla.

Esimerkki SOAP vastaus tiedonhakuagentilta:

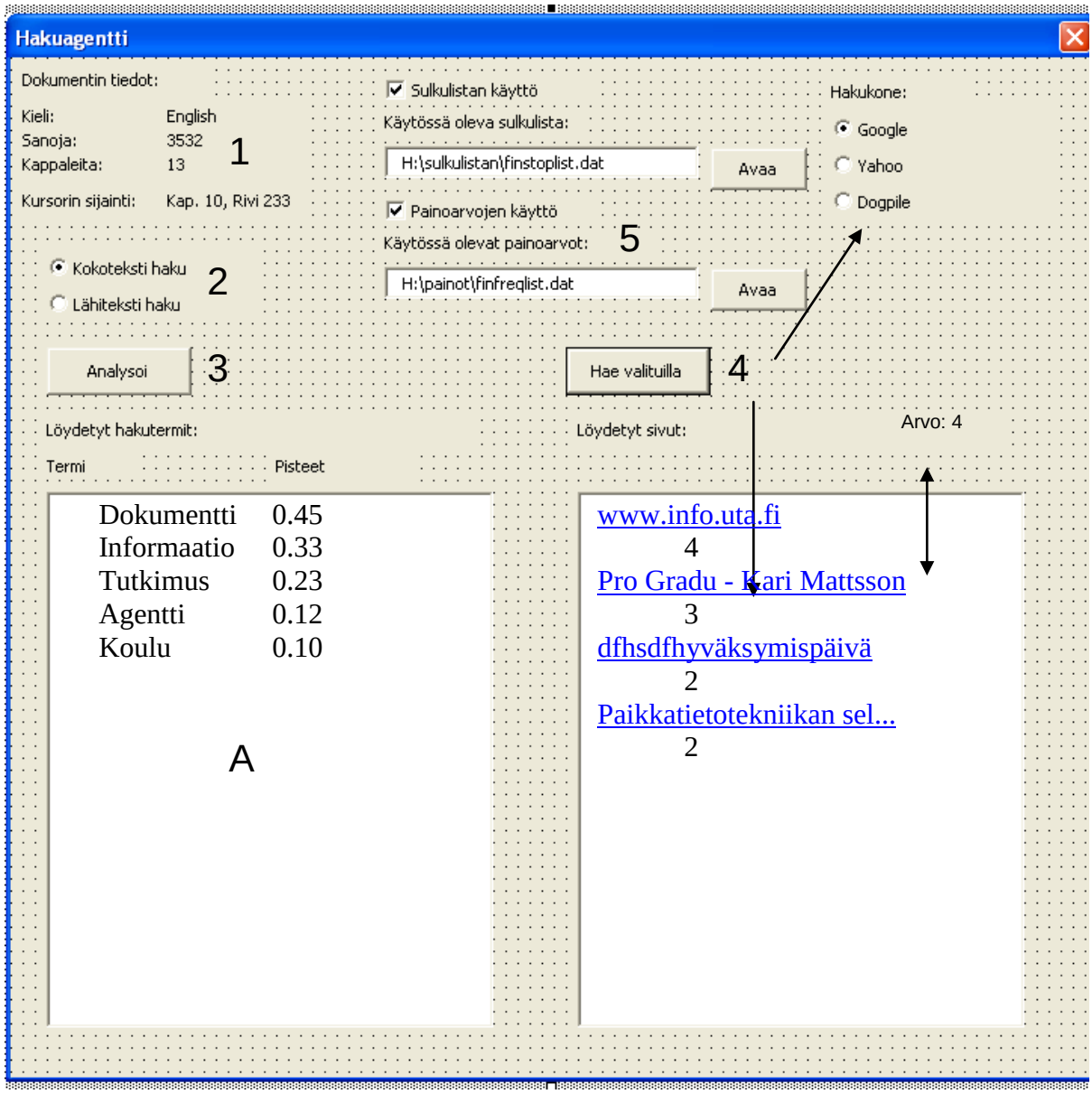
```
<?xml version="1.0"?>
<soap:Envelope xmlns:env=http://www.w3.org/2003/05/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <env:Header>
            <agent:viesti xmlns:agent="http://example.org/agent">
                    <agent:viitteidenmaara>12</agent:viitteidenmaara>
                    <agent:tunniste>d56a40dc69a3c8e7fbea1ac6f10345f6
                    </agent:tunniste>
            </agent:viesti>
    </env:Header>
    <env:Body>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:jarjestysnro>1</agent:jarjestysnro>
                    <agent:rank>1</agent:rank>
            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:jarjestysnro>2</agent:jarjestysnro>
                    <agent:rank>3</agent:rank>

            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:jarjestysnro>3</agent:jarjestysnro>
                    <agent:rank>2</agent:rank>

            <agent:avain xmlns:agent="http://example.org/agent">
                    <agent:jarjestysnro>4</agent:jarjestysnro>
                    <agent:rank>1</agent:rank>
            </agent:avain>

            <agent:viite xmlns:agent="http://example.org/agent">
                    <agent:otsikko> B Rhodes, T Starner - … Application Of
                    Intelligent Agents and Multi Agent …, 1996 -
                    aaai.org</agent:otsikko>
                    <agent:viite>https://www.aaai.org/Papers/Symposia/Spring/
                    1996/SS-96-02/SS96-02-022.pdf<agent:viite>
                    <agent:viitejarjestysnro>1</agent:viitejarjestysnro>
                    <agent:kieli>EN</agent:kieli>
                    <agent:lahde>http://scholar.google.com/</agent:lahde>
                    <agent:tiivistelma> Abstract The Remembrance Agent (RA)
                    is a program which augments human memory by displaying
                    a list of docu- ments which might be relevant to the
                    user's current context. Unlike most information
                    retrieval systems, the RA runs continuously without user
                    intervention. Its unobtrusive ... <agent:tiivistelma>
            </agent:viite>
    </env:Body>
</env:Envelope>
```

Huomattavaa on, että mikään viestien elementeistä ei ole pakollinen, mutta käytännössä ilman joitain elementtejä ei kommunikaatio toimi. Virhekommunikaatio ja muu SOAP-protokollaan liittyvä kommunikaatio tulisi toteuttaa W3C:n suositusten mukaisesti(W3C, SOAP Versio 1.2, 5.).

5.5 Esimerkki agentin käyttöliittymästä

Agentin käyttöliittymä on ainoa osa, jonka on tarkoitus näkyä käyttäjälle. Autonomisen tiedonhakuagentin ideaan kuuluu, että käyttäjän ei tarvitse olla minkäänlaisessa interaktiossa agentin kanssa, vaan agentti toimii taustalla ja antaa aika ajoin viitteitä. Koska projektissa tutkitaan agenttiteknologian toimintaa, olen suunnitellut käyttöliittymän, joka on siis käytännön pakosta olemassa. Käyttöliittymä näyttää käyttäjälle valitut avainsanat ja niiden painot. Käyttäjä voi itse vaikuttaa sanojen painotukseen. Käyttöliittymä on toteutettu esimerkiksi lukijalle eikä se täysin toteuta työssä kuvatun agentin toimintaa.

Agentin käyttöliittymä on toteutettu MS Wordiin ja se hoitaa myös Word-dokumentin tekstin siirtämisen käyttöjärjestelmätasolle tiedostoihin, kutsuu ydintä ja esittää käyttäjälle tulokset. Lisäksi käyttöliittymään on toteutettu rutiinit hakujen lähdetekstin poimimiseen käyttäjän dokumentista.



Kuva 15. Esimerkin käyttöliittymä

1. Dokumentin tiedot näkyvät käyttöliittymässä. Dokumentin kielen perusteella valitaan automaattisesti oletusarvot sulkulistalle ja painoarvoille. Lisäksi dokumentin tiedoista näkee kappaleen mitä käytetään lähitekstihaussa (kursorin sijainti)
2. Käyttäjä voi valita tehdäänkö haku koko dokumentin tekstin mukaan vai kursoria lähellä olevan tekstin mukaan.

Käyttäjä voi myös arvottaa dokumentteja oman näkemyksensä mukaan, jolloin hakuavaimisto luodaan uudelleen.

Lähitekstihaku. Lähitekstihaku on ominaisuus, jonka käyttäjä voi valita käyttöliittymästä. Se on agentin toiminnassa oletusarvo. Ominaisuus perustuu oletukseen, että käyttäjän kannalta oleellinen teksti löytyy avoinna olevasta kappaleesta kursorin läheisyydestä.

Kokotekstihaku. Kokotekstihaku on käyttäjän valittavissa oleva ominaisuus, jolla saadaan agentti käyttämään tiedonhaunlähteenä koko dokumentin tekstiä. Toteutan ominaisuuden siksi, että haluan vertailla kuinka paljon hakutulokseen vaikuttaa lähdetekstin rajaaminen, siis ero käytettäessä lähteenä avointa kappaletta tai koko tekstiä.

3. Painonappi ”Analysoi” käynnistää termien analysoinnin tekstistä. Tämän jälkeen käyttäjä voi hakea ruudussa A valitsemillaan sanoilla.
4. Painonappi ”Hae valituilla” käynnistää haun valitusta tiedonlähteestä. Haun tulokset esitetään käyttöliittymään upotetussa selaimessa. Käyttäjän painaessa linkkiä avautuu sivu omaan ikkunaansa.
5. Käyttäjä voi myös vaikuttaa käytetäänkö haussa sulkusanalistaa tai painoarvotaulukkoa. Mikäli järjestelmä ei löytänyt automaattisesti oikeita tiedostoja tai ne halutaan vaihtaa, käyttäjä voi tehdä sen myös käsin.

Yleisesti käyttöliittymän suunnittelussa tulisi pyrkiä mahdollisimman nopeaan interaktioon käyttäjän ja käyttöliittymän välillä sekä käyttöliittymän avaamiseen vain tarvittaessa. Tiedonhakutoiminnot tulisi toteuttaa automaattisesti tapahtuvaksi tausta-ajossa, jolloin viimeisimmät tulokset ovat aina käyttäjän saatavissa tarvittaessa.

5.6 Yhteenveto

Edellä esitettiin kahden agentin, tiedonhaun käyttöliittymäagentin ja tiedonhakuagentin mallit. Agentteja suunnitellessa, on tärkeää erottaa toisaalta mitä tietoa käsitellään missä ja toisaalta missä muodossa. Oikealla tiedonkäsittelyllä agentille saadaan rakennettua muisti, joka toimii tehtävässä aineistona (tiedot), jonka mukaan agentti voi tehdä päätelmiä (tehtävä). Suunnittelun agentissa muistia edustavat hakulauseet, tiedot lähdedokumentista, hakuagenttien toiminta edellisissä hauissa ja tesaurus. Oppimista edustaa käyttöliittymäagentin kyky muuttaa esimerkiksi hakuagenttien odotusaikaa, tiedonlähteiden relevanssijärjestystä tai termien katkaisun analyysi käytössä olevassa kielessä. Kommunikaatio on usein agentin tärkein ominaisuus ja tämä pitää paikkansa myös tämän työn agentissa. Agenttien välinen kommunikaatio tulee olla mahdollisimman tehokasta, mutta toisaalta protokollan tulee olla laajennettava ja yksinkertainen, jotta järjestelmä on mahdollisimman

laajennettava. Erityisesti laajennettavuuden vaatimus koskee erilaisten hakuagenttien luomista tiedonlähteestä riippuen.

Hakuagentin tekoälyn rakentaminen on agenttiteknologian suurin haaste. On liian helppoa lähteä ratkaisemaan ongelmaa lineaarisesti perinteisen ohjelmakoodin tapaan. Siinä missä lineaarista ohjelmistoa hallitsee lopussa oleva ”jos mikään muu edellä olevista ehdoista ei täyttynyt lähtötiedoilla tee näin (else ehto)”, agentti perustuu erilaiseen ideaan. Hyvin suunniteltu agenttiohjelmisto osaa muokata omaa lähdeaineistoaan. Agenttien suunnittelussa tulee hyväksyä, että lopputulos ei aina ole tiedossa (hakutulos), mutta käsittelyn tahtotila on (lista relevanteista viitteistä), jolloin hakutuloksesta tulee kyetä tekemään päätelmiä. Päätelmäkartta, jota tutkimuksen agentit käyttävät on kaikkien edellisen luvun toimintojen summa: se pyrkii ottamaan huomioon kaikki tiedonhaun prosessissa tehtävät valinnat ja mahdollisuudet. Päätelmissä kaikkein hankalinta on sanojen katkaisu, kuten yleensäkin suomenkielisessä tiedonhaussa. Tämä on kuitenkin ratkaistavissa, kunhan muistetaan, ettei pyritä täydellisyyteen vaan vähempikin haun laajennus riittää. Toinen ongelman on haun laajentaminen toisessa vaiheessa. Tarjolla on hyviä algoritmeja haun laajentamiseksi, mutta riippuvaisuus syntyy tiedonlähteiden ominaisuuksiin.

6 Johtopäätökset

Työn alkuperäiset ongelmat olivat:

- Minkälaisia haasteita nousee esiin suunnitellessa agenttiteknologialla toimivaa, tekstinkirjoittajaa avustavaa reaaliaikaista tiedonhaun välittäjäjärjestelmää?
- Mitä vaatimuksia tiedonhakutehtävä asettaa agenttiteknologialle?
- Mitä tiedonhakututkimuksessa kehitettyjä menetelmiä voidaan soveltaa agenttipohjaiseen tiedonhakuun?

Työssä pyrittiin ottamaan kantaa tiedonhakua avustavien agenttien suunnittelun ja toteutuksen problematiikkaan. Lähestymistavaksi valittiin ohjelmistoagenttiteknologia ja sitä kautta tiedonhaun tietämyspohjaisen välittäjäjärjestelmän, agentin, suunnittelu alun käyttöliittymästä aina tekniseen toteutukseen saakka. Parhaimmillaan tiedonhakuagentin tulisi voida ymmärtää lähdedokumentin semantiikka ja syntaksi ja pystyä siitä tuottamaan kyselyitä tiedonlähteisiin. Tiedonhakuprosessi ei kuitenkaan saisi päättyä tähän, vaan tiedonhakuagenttien pitäisi kyetä arvioimaan myös tulosjoukkoa kriittisesti ja laajentamaan hakua sen perusteella. Suljetuissa ympäristössä haun laajentaminen löytyneiden dokumenttien perusteella onkin mahdollista, mutta silti kaikki tiedonlähteet eivät välttämättä tarjoa pääsyä kokotekstiin tai tarpeeksi tarkkoja tietoja tiedonlähteen cf-ja df-arvoista, mikä rajoittaa agentin toimintavalmiuksia.

Tiedonhakumetodit, joita agentille löysin, olivat varsin perinteisiä ja yksinkertaisia, mutta varmasti sellaisenaan riittäviä ja tarpeeksi joustavia. Lähdedokumentin analyysiin sopii hyvin Zipf-tyyppinen kaava (Salton & McGill, 1983), kun taas haun laajentamiseen voi käyttää monipuolisempaa $w_t(p_t - q_t)$ kaavaa (Efthimiadis 1993, 148-150).

Yksi agentin vaatimuksista oli kieliriippumattomuus. Käytännössä termien katkaisun osalta vaatimus rajoittaa agentin toimintoja, ja joitain kielikohtaisia sanastoja ja katkaisutapoja on varmasti käytettävä. Katkaisu osoittautuikin haasteellisimmaksi toteutukseltaan agenttiohjelmistossa, eikä katkaisun problematiikkaan yhtä ratkaisua loppujen lopuksi löytynytkään.

Kieltenväliseen tiedonhakuun sen sijaan tuo tehokkaan työkalun RATF-kaava (Pirkola et al. 2002), jonka avulla voidaan arvioida tiedonhaun onnistumista automaattisesti käännetyin termein.

Yksinkertaisen haun lisäksi agentin on tärkeää osata laajentaa itse hakua myös ilman käyttäjää. Tämä vaatii agentilta useita iteraatioita ja myös agentin toteuttajalta huolellista arviointia miten agentin päättelylogiikka saadaan toimimaan. Riskinä on saada suppea tulosjoukko, jos hakua kehitetään liian rajaavaksi.

Itse agenttiteknologian kehitys on ollut hidasta. Pääideologiat ovat samoja ja periytynevät olio-ohjelmoinnista, missä pyritään erilaisten käsitteellisten mallien kautta kuvaamaan ohjelmistoja ihmismäisesti. Totta on, että esimerkiksi ohjelmistoagenteissa tämä hyödyttää suunnittelua paljonkin. Itse havaitsin poikkitieteelliselle tutkimukselle aihealueella tilauksen. Psykologia, lingvistiikka, käytettävyys- ja tietojenkäsittelytiede voivat tarjota alustan, joka yhdistäisi tiedonhakututkimuksen kehittämiä hakumenetelmiä käyttöliittymiin ja tiedonhakujärjestelmiin. On erikoista, että esimerkiksi tekstinkäsittelyohjelmat eivät vieläkään sisällä automaattista tiedonhakua kirjoitettavan dokumentin perusteella, vaikka esimerkiksi Microsoft Officeen sisältyy kaikki mahdolliset ominaisuudet tähän liittyen. Google on nykyään päässyt tässä pisimmälle Google Docs -palvelunsa kautta, mutta sitä ei voida pitää vielä kovinkaan tehokkaana työkaluna.

Agenttiteknologia antaa myös odottaa itseään. Sen suurin haaste on yhteisen palvelinalustan puute, minkä vuoksi teknologia ei pääse leviämään ja agentit ovat enemmän ja tai vähemmän staattisia. Epäilen, että myös tässä työssä esitetty agentti päätyisi toteutuksen jälkeen staattiseksi.

Työn pyrkimyksenä oli tuoda esiin yksi tapa toteuttaa tiedonhakuagentti. Työ onkin näin poikkitieteellinen “kertomus”, lähtien filosofisesta ongelmasta ja päättyen varsin tietotekniseen protokollatoteutukseen. Työ sivusi, mutta ei käsitellyt tarkasti tiettyjä aihealueita, kuten kielten välistä tiedonhakua ja sanojen katkaisua. Näihin on olemassa myös paljon aineistoa, eikä työn laajentaminen tätä kautta olisi mitenkään poissuljettua. Lisäksi itse toteutus olisi syytä altistaa formaaliin saanti/tarkkuus-analyysiin ja kehittää sitä kautta lisää tiedonhakualgoritmien toimintaa agenttiympäristössä. Toteutuksen pääpaino tulisi kuitenkin pitää ajatuksessa käyttäjää tukevasta agentista. Käyttäjä ajateltiin työssä tiedontarvitsijana, mutta ei tottuneena tiedonhakijana. Siksi käyttöliittymä ja tiedonhakuagentit toimivat varsin itsenäisesti. Keskustelusta käyttäjän kanssa

pyrittiin poistamaan täysin tiedonhakuprosessiin liittyvät termien valinnat. Ennemminkin kysyttiin asioita, jotka ovat käyttäjälle tuttuja reaalimaailmasta, kuten synonyymeja ja dokumentin huonoutta/hyvyyttä.

Tiedonhakuagentti on hyvinkin toteutettavissa käyttäjää tukevaksi järjestelmäksi, mutta sen tekninen suunnittelu vaatii aikaa. Kyse ei ole yksinkertaisesta ohjelmistosta, vaikka ajatus siltä saattaa aluksi tuntuakin. Aiheen kompleksisuudesta saa hyvin käsityksen, kun vertaa esimerkiksi Caglayanin ja Harrisonin (1995) agenttiohjelmistojen suunnitteluteoriaa Ingwersenin (1992) esittämiin teorioihin välittäjäjärjestelmien suunnittelusta. Agenttiteorian suoraviivaisuus ja yksinkertaisuus voi viedä helposti harhaan, sillä tiedonhaun teorioiden automaattinen toteuttaminen voi osoittautua kompleksiseksi.

Itse asiassa tämä on juuri haaste monessa nykyisessä hakujärjestelmässä. Pelkkä termifrekvenssien laskeminen kun ei riitä tyydyttävään saantiin.

Lähteet

Alkula, Riitta (2000). *Merkkijonoista suomen kielen sanoiksi.* Tampereen yliopisto. Acta Universitatis Tamperensis 763.

Bates, Marcia J., Deborah Wilde & Susan Siegfried (1993). An analysis of search terminology used by humanities scholars: The Getty Online Searching. Project report number 1. *Library Quarterly*, 63(1), 1-39.

Bradshaw, Jeffrey M. (1997). *Software agents*. Cambridge: MIT Press. ISBN: 0-262-52234-9.

Caglayan, Alper. & Colin Harrison (1997). *Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents*. New York: John Wiley & Sons. ISBN: 0-471-15327-3.

Dickinson, Ian & Michael Wooldridge (2003). Practical reasoning agents for the semantic web. *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, Melbourne, Australia. Saatavilla: http://www.csc.liv.ac.uk/~mjw/pubs/aamas2003c.pdf

Efthimiadis, Efthimis N (1993). A user-centred evaluation of ranking algorithms for interactive query expansion. Annual ACM Conference on Research and Development in Information Retrieval. *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. 146-159. ISBN:0-89791-605-0.

Guttenplan, Samuel (toim.) (1994): *Companion to the philosophy of mind*. Oxford: Blackwell Publishers Ltd.

Haugeland, John (1996). The prospects for artificial intelligence. Teoksessa William G. Lycan (toim.): *Mind and cognition*: *A reader.* Oxford: Blackwell. 660-670.

Haverkamp, Donna S. & Susan Gauch (1998). Intelligent information agents: Review and challenges for distributed information sources. *Journal of the American Society for Information Science*, 49 (4), 304-311.

Huntus, Harri (1999). *Elektronisen kauppapaikan toteuttaminen liikkuvilla agenteilla*. Pro Gradu – tutkielma. Tampereen yliopisto, Tietojenkäsittelyopin laitos.

IBM (2002). *IBM Aglets software development kit v2*. Saatavilla: http://www.trl.ibm.co.jp/aglets/. Viitattu 23.3.2010.

Ingwersen, Peter (1992). *Information Retrieval Interaction*. London: Taylor Graham. Saatavilla: http://vip.db.dk/pi/iri/index.htm

Intelligent Information Agents SIG (2002). *Law of Electronic Agents*. Saatavilla: http://www.dbgroup.unimo.it/IIA/softagents.html, Viitattu 23.3.2008.

Jain, Lakhmi, Zhengxin Chen & Nikhil Ichalkaranje (1999). *Intelligent agents and their applications*. Heidenlberg: Physica-Verl. ISBN: 3-7908-1469-5.

Järvelin, Kalervo, Jaana Kekäläinen & Timo Niemi (2001). ExpansionTool: Concept-based query expansion and construction. *Information Retrieval* 4(3/4), 231-255.

Kekäläinen, Jaana (1999). *The effects of query complexity, expansion and structure on retrieval performance in probalistic text retrieval*. Tampereen yliopisto. Acta Universitas Tamperensis 678. ISBN 951-33-4596-1.

Kettunen, Kimmo (2007). *Reductive and Generative Approaches to Morphological Variation of Keywords in Monolingual Information Retrieval.* Tampereen yliopisto. Acta Universitas Tamperensis 1261. ISBN 978-951-44-7087-5.

Klusch, Matthias (1999). *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet.* New York: Springer-Verlag.

Kwok, Kui-Lam. (1996). A new method of weighting query terms for ad-hoc retrieval. *Proceedings of the 19th Annual* I*nternational ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland. New York: Association for Computing Machinery. 187-195.

Lycan William G. (toim.) (1996): *Mind and cognition*: *A reader.* Oxford: Blackwell.

Magedanz , Thomas. (1997). *Mobile agents – An overview.* ACTS IS&N Conference. Cernobbio (Como). Italy, May 27-29.

Nardi, Bonnie A. & O'Day, Vicki (1996). Intelligent agents, What we learned at library. *Libri* 46. 59-88.

Nwana Hyacinth S. (1996), Software Agents: An Overview. *Knowledge Engineering Review* 11(3), 1-40. Saatavissa: http://agents.umbc.edu/introduction/ao/. Viitattu 10.4.2010.

Pirkola, Ari, Erkka Leppänen & Kalervo Järvelin (2002). The RATF formula (Kwok's formula): exploiting average term frequency in cross-language retrieval. *Information Research,* 7(2) Saatavilla: http://InformationR.net/ir/7-2/paper127. Viitattu 20.1.2008

[RFC 1321] *The MD5 Message-Digest Algorithm*. Rivest R. (1992). Saatavilla: http://www.ietf.org/rfc/rfc1321.txt. Viitattu 28.2.2010.

[RFC 2616] *Hypertext Transfer Protocol -- HTTP/1.1*. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk Nielsen, P. Leach, L. Masinter and T. Berners-Lee, Editors.(1999). Saatavilla: http://www.ietf.org/rfc/rfc2616.txt. Viitattu 10.4.2010.

Robertson, Stephen. (1986). On relevance weight estimation and query expansion. *Journal of Documentation* 42 (3). 182-188. ISSN: 0022-0418.

Salton, Gerard & Michael J. McGill. (1983). *Introduction to modern information retrieval*. Auckland: McGraw-Hill. ISBN 0-07.Y66526-5.

Salton, G. (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Reading, MA: Addison-Wesley.

Searle, John R. (1980). Minds, brains and programming. *Behavioral and brain sciences* 3, 417-457.

Searle, John R.. (1990). Is the brain's mind a computer program? *Scientific American* 262(1), 26-31.

Searle, John R. (1994). Searle, John R. Teoksessa Samuel Guttenplan (toim.): *Companion to the philosophy of mind*. Oxford: Blackwell Publishers Ltd. 544-550.

Shoham, Yoav (1993). Agent-oriented programming, *Artificial Intelligence*, Vol. 60, 51-92.

Sormunen, Eero (1989). *An analysis of online searching knowledge for intermediary systems*. Sivuaineen tutkielma. Tampereen yliopisto, Kirjastotieteen ja informatiikan laitos.

Turing, Alan (1950). Computing machinery and intelligence, *Mind*, 59(236). 433-460. Saatavilla myös: http://www.loebner.net/Prizef/TuringArticle.html

[UMBC AgentWeb], *Recommended Papers*. Saatavilla: http://agents.umbc.edu/Publications_and_presentations/Recommended_Papers/index.shtml.Viitattu 6.5.2010.

VISK = Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja Riitta Heinonen ja Irja Alho 2004: *Iso suomen kielioppi*. Helsinki: Suomalaisen Kirjallisuuden Seura. Verkkoversio, viitattu 1.11.2008. Saatavissa: http://scripta.kotus.fi/visk. ISBN:978-952-5446-35-7

[W3C], *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Saatavilla: http://www.w3.org/TR/soap12-part1/. Viitattu 13.1.2010.

Wooldridge, Michael. & Jennings Nick. R. (1995). Intelligent agents: Theory and practice. *Knowledge engineering review*. 10(2). Saatavilla: http://www.csc.liv.ac.uk/~mjw/pubs/ker95/ker95-html.html.

Liite 1: Hakuavainten etsiminen

```
#!/usr/bin/perl -w
#
# Tiedonhakuagentti
# Jiri Uitto
#
# Perustuu kiinalaisen huoneen ideaan
# Käyttää lähteenä parametriksi annettua ASCII
# tiedostoa ja tulostaa ruudulle kaikki tiedoston
# uniikit sanat, joko aakkosjärjestyksessä tai
# Esiintymisen perusteella. Esiintyminen lasketaan sekä frekvenssin, että Zipf algoritmin
mukaan.
# Työkalulla voi myös poistaa automaattisesti stopwordit.
#
# Käyttö:
# freq.pl {n} tiedosto {kyselynpituus} optiot [-s stopwordlista, -m % kuinka paljon
listan alusta poistetaan]
# Missä 'n':
# -f = frekvenssin mukaan
# -z = Zipf mukaan
#
# Esimerkiksi perl freq.pl –f Teksti.txt 10 -s finstopword -m 5
#
# Kyselynpituus 0 = kaikki termit

use locale;
use Switch;

my $filename;       # Tiedostonnimi
my $ext;      # Tiedoston lisämääre
my $querylenght;# Haluttu sanojen määrä kyselyssä, 0 kaikki termit
my $OLD;      # Tiedoston nimi
my $what;     # n optio
my $cnt2;     # Laskuri, hakutermien määrälle
my $stop;     # -s mikäli poistetaan stopwordit
my $stopfile; # Stopword tiedosto
my $stopword; # Stopwordi
my @stopwords;      # Stopword taulukko
my $word;     # Käsiteltävä sana
my @words;    # Kaikki sanat
my $cnt;      # Sanalaskuri
my @query;    # Kysely (sana+sana+sana)
my %freqorder;  # Hash jolla järjestellään frekvenssin mukaan tulos $freqorder{query}
my %zipforder;  # Hash jolla järjestellään Zipfin mukaan tulos $zipforder{query}
my $zipf;     # Paino
my $locatio;  # Järjestysluku
my @uniq;     # Uniikkisanalista
my $seen;     # Apumuuttuja dublikaattien poistoon
my $medium;   # Kysely muodostetaan Zipf listan keskeltä
my $mediumcnt;  # Relevanssialueen alku
my $temp1;  # Tilapäismuuttuja
my $temp2;  # Tilapäismuuttuja

# Tutkitaan ensin annettujen argumenttien lukumäärä
             if ($ARGV[0] eq "-f" or $ARGV[0] eq "-z"  and $#ARGV > 1)
             {
                    $what=$ARGV[0];
             $OLD=$ARGV[1];
             $cnt2=$ARGV[2];
             }


             if ($ARGV[3] eq "-s" and $#ARGV > 3) {
```

```
		$stop=$ARGV[3];
			$stopfile=$ARGV[4];

		}
		elsif ($ARGV[3] eq "-m" and $#ARGV eq 4) {
			$medium = $ARGV[3];
			$mediumcnt = $ARGV[4];
		}
		elsif ($#ARGV eq "3") {
			$what = "Optiot";
		}


		if ($ARGV[5] eq "-m" and $#ARGV eq "6") {
			$medium = $ARGV[5];
			$mediumcnt = $ARGV[6];
		}
		elsif ($#ARGV eq "5") {
			$what = "Optiot (-m ?)";
		}

if ($what ne "-f" and $what ne "-z") {

		print "Argumenttien määrä väärin! $what\n";
		print "Syntaksi: freq {n} tiedosto {kyselynpituus} [-s tiedosto]\n";
		print "    Missä 'n':\n\t";
		print "             -f = frekvenssin mukaan\n\t";
		print "             -z = Zipf mukaan järjestelty\n\t";
		print "\n\t";
		print "             -s <tiedosto> = poista stopwordit\n\t ";
		print "kyselynpituudella 0 saat kaikki termit\n\n";
		print "Esimerkiksi perl freq -f Teksti.txt 10\n";
		exit -1;

}


# Jos haluttiin stopword poisto, avataan stopword tiedosto
if ($stop eq "-s" and $stopfile ne "0") {
	open stopfile, "<$stopfile"		or die "ei voi avata $stopfile: $!";
	while (<stopfile>){
		# Stopword tiedostosta poistetaan erikoismerkit ja muutetaan sen sanat
pieniksi kirjaimiksi
		# Pyritään siis täsmälleen samaan muotoon kuin alkuperäinen teksti, lisäksi
stopwordlista voi olla csv tiedosto
		chomp;
		s/[\n.,"?><\_\-=+`~!\@#\$%^&*();:\/\\1234567890]/ /g;
		s/ '|' |^'|'$|\t'|'\t/ /g;
		$_= lc($_);
		push (@stopwords, $_);
	}
	# Viestitään seuraavalle funtiolle, että stopwordlist on käytettävissä
	$stopword = 1;
	close (stopfile) || die "tiedosto ei sulkeudu: $stopfile: $!";
}

($filename,$ext) = split(/\./, $OLD);
# Erota tiedostonnimi tyypistä

# Avataan lähdeteksti käsittelyä varten
open OLD, "<$OLD"		or die "ei voi avata $OLD: $!";


$cnt = 0;
$locatio = 1;

while (<OLD>) {
	chomp;
```

```
	s/[\n.,"?><\_\-=+`~!\@#\$%^&*();:\/\\1234567890]/ /g;
	# Tässä kohtaa kaikki merkit jotka eivät
	# ole kirjaimia korvataan välilyönnillä
	# tämä tehdään koska suomen kielessä ei
	# yleisesti käytetä välimerkkejä sanojen yhteydessä

	s/ '|' |^'|'$|\t'|'\t/ /g;
	$_= lc($_);
	# Kaikki termit muutetaan pienillä kirjaimilla kirjoitetuiksi
	# Tämä helpottaa laskentaa ja hakua joissain hakukoneissa

	@words = (split);
	foreach $word (@words) {
			push (@uniq, $word) unless $seen{$word}++;
			# Edellisellä etsitään samoja sanoja ja lasketaan samalla
			# sanojen määrä muuttujaan cnt
			$cnt++;
	}
}

close(OLD)  || die "tiedosto ei sulkeudu: $OLD: $!";

format =
@<<<<<<<<<<<<<<<< - @>>>>>
$word, $seen{$word}
.

# Lasken ensin mistä alkaa haluttu aloituskohta
$mediumcnt = $mediumcnt / 100 * $cnt;
$mediumcnt = ($mediumcnt == int($mediumcnt) ? $mediumcnt : int($mediumcnt + 1));

# Frekvenssi lajittele ja laske zipf

if ($what eq "-f" or $what eq "-z") {
	print "Järjestysluku\t Frekvenssi\t Zipf\t\ Termi\n";

	foreach $word (sort { $seen{$b} <=> $seen{$a} } keys %seen ) {
		$zipf = $locatio * $seen{$word};
		if ($stop eq "-s" and $stopfile ne "0") {
			$temp2 = 0;
			foreach $stopword (@stopwords){
				if ($word eq $stopword) {
					$temp2 = 1;
				}
			}
			if ($temp2 == 0) {
				$Temp1 = ("$locatio\t\t $seen{$word}\t\t $zipf\t
$word\n");
				if ($what eq "-z") {
					$zipforder{$zipf} = $Temp1;
				}
				else {
					$zipforder{$locatio} = $Temp1;
				}
				$locatio++;
			}
			$temp2 = 0;
			}
			else {
				$Temp1 = ("$locatio\t\t $seen{$word}\t\t $zipf\t
$word\n");
				if ($what eq "-z") {
					$zipforder{$zipf} = $Temp1;
				}
				else {
					$zipforder{$locatio} = $Temp1;
				}
```

```
                                $locatio++;

                        }
        }


$temp2 = 0;

# Mikäli pyydetty termien määrä oli 0, tulostetaan kaikki termit
# $cnt2 on haluttu määrä
# $mediumcnt on alusta poistettava määrä
# $cnt2 + $mediumcnt on kokonaismäärä

        # Tulostetaan lopputulos järjesteltynä
        for my $key ( sort {$a<=>$b} keys %zipforder) {
                if ($cnt2 eq 0) {
                        if ($temp2 >= $mediumcnt) {
                                print "$zipforder{$key}";
                        }
                }
                elsif ($temp2 >= $mediumcnt and $cnt2 + $mediumcnt > $temp2) {
                        print "$zipforder{$key}";
                }

                $temp2++;
        }
}

exit 1;
```

Liite 2: Käyttöliittymä

```
Sub startup()
‘ Aliohjelma käynnistetään kun käyttäjä kutsuu käyttöliittymää tai painaa päivitä nappia
    Dim RetVal
    Dim exefile As String
    exefile = "c:\Hakukikkula\query.cmd"
    Load Hakukikkula
    If (Hakukikkula.Lähiteksti.Value) Then
        Call paratallennus
    Else
        Call kokotallennus
    End If
    RetVal = Shell(exefile, vbMinimizedFocus)
    Hakukikkula.Show
End Sub


Private Sub generoitiedostot()
‘ Aliohjelma kutsuu ulkoista komentoa joka generoi lähdetiedostosta sanalistan
    Dim RetVal
    Dim exefile As String
    exefile = "c:\Hakukikkula\query.cmd"
    RetVal = Shell(exefile, 6)
End Sub


Private Sub navigointi()
‘ Aliohjelmaa kutsutaan kun halutaan lukea sanalista välitiedostosta (destination.txt) ja
laittaa sanat käyttäjän valintaruutuun

    Dim destinationfile As String
    destinationfile = "c:\Hakukikkula\destination.txt"

    Const ForReading = 1, ForWriting = 2, ForAppending = 3
    Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0
    Dim fs, f, ts, s

    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(destinationfile)
    Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
    While (s <> "QSTOPQ") ' Etsi kunnes lopetusmerkki tulee vastaan
        s = ts.ReadLine
    Wend
    s = ts.ReadLine
    WebBrowser1.Navigate s
    ts.Close
End Sub

Private Sub hae_valituilla()
’ Suorittaa haun käyttäjän valitsemilla sanoilla
        Dim cnt As Integer
        Dim query As String
        query = "http://www.google.com/search?q="
        cnt = 0
        While (cnt < ListBox1.ListCount)
            If (ListBox1.Selected(cnt)) Then
                query = query + ListBox1.List(cnt) + "+"
            End If
            cnt = cnt + 1
        Wend
```

```
        query = query + "+&btnG=Google+Search"
        WebBrowser1.Navigate query
End Sub

Private Sub Haeuudelleen_Click()
' Päivittää haun
    If (Hakukikkula.Lähiteksti.Value) Then
        Call paratallennus
    Else
        Call kokotallennus
    End If
    Call UserForm_Activate
    Call navigointi

End Sub

Private Sub UserForm_Activate()
    WebBrowser1.AddressBar = False
    WebBrowser1.ToolBar = False
    WebBrowser1.Visible = True
    Dim sourcefile As String
    Dim destinationfile As String
   Dim exefile As String
   sourcefile = "c:\Hakukikkula\source.txt"
   destinationfile = "c:\Hakukikkula\destination.txt"
   exefile = "c:\Hakukikkula\query.cmd"
    Call generoitiedostot ' Ajetaan ulkoinen komento query.cmd joka generoi tiedostot
    Call navigointi 'Ajetaan oletuskysely

End Sub

Private Sub Userform_Initialize()
' Ajetaan kun käyttöliittymä ajetaan ensimmäisen kerran
    Call navigointi
End Sub

Private Sub kokotallennus()
' Tallentaa koko tekstin lähteeksi
    Dim sourcefile As String
    sourcefile = "c:\Hakukikkula\source.txt"
    Selection.WholeStory
    Selection.Copy
    Documents.Add Template:="c:\hakukikkula\qblank.dot", NewTemplate:=False,
DocumentType:=0
    Selection.Paste
    ActiveDocument.SaveAs FileName:="c:\Hakukikkula\source.txt",
FileFormat:=wdFormatText, _
        LockComments:=False, Password:="", AddToRecentFiles:=True, WritePassword _
        :="", ReadOnlyRecommended:=False, EmbedTrueTypeFonts:=False, _
        SaveNativePictureFormat:=False, SaveFormsData:=False, SaveAsAOCELetter:= _
        False
    ActiveDocument.Close
End Sub

Private Sub paratallennus()
'
'Tallentaa osan tekstistä lähteeksi. Käytännössä siirrytään ensin yksi ruudullinen
ylöspäin ja tämän jälkeen maalataan kaksi ruudullista alaspäin. Maalattua aluetta
käytetään lähteenä. Ei hienostunein mahdollinen tapa…

Dim sourcefile As String
    sourcefile = "c:\Hakukikkula\source.txt"

    Selection.MoveUp Unit:=wdScreen, Count:=1
    Selection.MoveUp Unit:=wdScreen, Count:=2, Extend:=wdExtend

    Selection.Copy
    Selection.MoveRight Unit:=wdCharacter, Count:=1
```

```
    Documents.Add Template:="c:\hakukikkula\qblank.dot", NewTemplate:=False,
DocumentType:=0
    Selection.Paste
    ActiveDocument.SaveAs FileName:="c:\Hakukikkula\source.txt",
FileFormat:=wdFormatText, _
        LockComments:=False, Password:="", AddToRecentFiles:=True, WritePassword _
        :="", ReadOnlyRecommended:=False, EmbedTrueTypeFonts:=False, _
        SaveNativePictureFormat:=False, SaveFormsData:=False, SaveAsAOCELetter:= _
        False
    ActiveDocument.Close
 End Sub
```

Liite 3: Esimerkin hakuavaimet

Esitettynä ensimmäiset 100 termiä

Järjestysluku	Frekvenssi	Zipf	Termi
1	217	217	on
2	168	336	ja
3	142	426	agent
4	91	364	agentin
5	61	305	ei
6	60	360	että
7	51	357	käyttäjän
8	50	400	tai
9	49	441	myös
10	46	460	agentti
11	43	473	agenttien
12	41	492	voidaan
13	37	481	se
14	36	504	agentit
15	35	525	tulee
16	33	528	ovat
17	33	561	esimerkiksi
18	31	558	•
19	30	570	siis
20	30	600	voi
21	30	630	dokumentin
22	30	660	perusteella
23	29	667	a
24	29	696	sen
25	28	700	käyttöliittymäagentin
26	28	728	tässä
27	28	756	avain
28	28	784	the
29	28	812	http
30	28	840	mukaan
31	27	837	sanojen
32	27	864	information
33	27	891	mutta
34	25	850	ole
35	25	875	of
36	25	900	olla
37	24	888	and
38	23	874	tietoa
39	23	897	kanssa
40	22	880	org

41	22	902	soap
42	22	924	koska
43	22	946	kuitenkin
44	22	968	tiedonhakuagentin
45	22	990	tiedonlähteen
46	20	920	toteuttaa
47	20	940	tiedonhakuagentti
48	19	912	käyttöliittymä
49	18	882	käyttäjälle
50	18	900	kuin
51	17	867	eli
52	17	884	tämä
53	17	901	itse
54	17	918	agents
55	17	935	jotka
56	16	896	kieli
57	16	912	jarjestysnro
58	16	928	käyttöliittymäagentti
59	16	944	tiedonhaun
60	15	900	hakuagentti
61	15	915	välillä
62	15	930	joka
63	15	945	käyttäjä
64	15	960	sisältää
65	15	975	käyttöliittymän
66	15	990	tulisi
67	15	1005	käyttää
68	15	1020	www
69	15	1035	paino
70	14	980	kommunikaatio
71	14	994	perustuu
72	14	1008	rank
73	14	1022	envelope
74	14	1036	hakuavaimet
75	14	1050	tunniste
76	14	1064	xmlns
77	14	1078	sanat
78	14	1092	olisi
79	13	1027	kuva
80	13	1040	viestin
81	13	1053	sekä
82	13	1066	ne
83	13	1079	c
84	13	1092	kannalta
85	13	1105	viesti
86	12	1032	lisäksi

87	12	1044	retrieval
88	12	1056	mikäli
89	12	1068	env
90	12	1080	ominaisuuksien
91	12	1092	paljon
92	12	1104	dokumentissa
93	12	1116	dokumentti
94	12	1128	vain
95	12	1140	example
96	12	1152	ominaisuudet
97	11	1067	oleva
98	11	1078	yksi
99	11	1089	sillä
100	11	1100	fasetti

Liite 4: Johdantojen analyysit

Järjestys	Frekvenssi	Zipf	Termi	Järjestys	Frekvenssi	Zipf	Termi
1	7	7	agent	2	6	12	agentin
2	7	14	information	3	5	15	agenttien
3	5	15	work	4	5	20	informaatiotutkimuksen
8	2	16	problems	11	2	22	keräävät
17	1	17	searle’s	12	2	24	toteutus
18	1	18	give	5	5	25	työn
19	1	19	course	13	2	26	agentit
20	1	20	tools	9	3	27	tietoa
21	1	21	applications	29	1	29	normaaleja
22	1	22	published	30	1	30	toteutetun
23	1	23	seeking	32	1	32	lähtökohdasta
24	1	24	technology	33	1	33	esittämään
25	1	25	minds	34	1	34	muotoon
26	1	26	source	35	1	35	tiedonhakuprosessiin
27	1	27	evaluate	36	1	36	sisällöstä
28	1	28	rooms	37	1	37	löytynyt
29	1	29	references	38	1	38	omasta
30	1	30	programs	39	1	39	valmiiden
31	1	31	create	40	1	40	jättävät
32	1	32	semantics	41	1	41	tiedonhakuagentin
33	1	33	able	42	1	42	programs
34	1	34	exist	43	1	43	esittämästä
35	1	35	common	44	1	44	luokittelu
36	1	36	construction	45	1	45	esittävä
37	1	37	implementing	46	1	46	ominaisuuksiin
38	1	38	autonomously	47	1	47	ohjelmistopohjainen
39	1	39	made	48	1	48	lopuksi