TAMPEREEN TEKNILLINEN YLIOPISTO

MIKA MÄHÖNEN

TEKSTIN LUOKITTELU

Diplomityö

Tarkastaja: professori Hannu Jaakkola
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 6. helmikuuta 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
MÄHÖNEN, MIKA: Tekstin luokittelu
Diplomityö, 100 sivua
Kesäkuu 2013
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Hannu Jaakkola
Avainsanat: tiedonhaku, luokittelu, informaatioteoriat ja tilastomenetelmät

Tämän diplomityön tarkoituksena oli tutkia tekstin luokittelua ja avainsanojen poimintaa. Tähän tarkasteluun tärkein yksittäinen tekijä on datan rakenne, jonka avulla työssä perusteellaan luokittelun tarpeellisuutta. Informaation etsintään on saatavilla kaksi keskeistä menetelmää, jotka ovat informaation poiminta strukturoimattomasta datasta ja strukturoidun datan käyttöönotto eli metadata. Työssä nämä menetelmät esitellään huolellisesti samalla argumentoiden, minkä tyyppisiä heikkouksia ja vahvuuksia niihin liittyy. Tämän tutkimuksen perusteella saatu lopputulos oli, että molempia menetelmiä tarvitaan osana kokonaisvaltaista sisällönhallintaratkaisua.

Sisällöstä kirjattujen avainsanojen ja luokittelun voidaan ajatella olevan sisällöstä saatavilla olevia havaintoja. Näiden havaintojen tarkoitus on tiivistää tekstiä niin, että dokumentin löytäminen on yksinkertaisempaa. Luokittelu ja avainsanojen kerääminen on edellyttänyt perinteisesti ihmistyötä, koska teksti edellyttää tulkintaa. Tämä on myös syy, miksi ihmiset suorittavat edelleen avainsanojen poimintaa ja luokittelua. Tämän prosessin automatisointi voi parantaa monien tietoteknillisen järjestelmien tehokkuutta ja säästää aikaa prosessoitaessa suurta määrää tekstidokumentteja. Aihealuetta työssä tutkitaan esittelemällä toimenpiteet, joita tekstin luokitteluun ja avainsanojen poimintaan tarvitaan. Tämä tutkimus on jaettu NLP-menetelmiin (engl. *natural language processing*) ja luokittelualgoritmeihin. NLP-tekniikoiden tehtävänä on poistaa haasteita, jotka liittyvät merkkijonojen vertailuun tietokoneen muistissa. Näiden tekniikoiden osalta työssä esitellään kielen tunnistusta, tekstin jakamista avaimiin, sanojen palauttamista perusmuotoon, konseptien mallintamista ja ominaisuuksien valintaa. Luokittelualgoritmien osalta työssä tutkitaan naiivia Bayesian luokittelua ja päätöspuita. Näistä algoritmeista annetaan myös käytännön esimerkki, joka vahvistaa esitellyn teorian käytännössä. Tutkimuksen aikana luokittelujärjestelmissä havaittiin muutamia rajoituksia. Näistä rajoituksista ensimmäinen on, ettei luokittelujärjestelmä omaa ihmiselle tunnusomaisia abstraktiotasoja. Näin ollen tietokone ei pysty yhdistämään esimerkiksi sanoja auto ja ajoneuvo toisiinsa. Toinen löydetty rajoite oli, ettei sanojen sijaintia huomioida tekstissä. Löydetyistä rajoitteista huolimatta, monet algoritmit toimivat todellisuudessa varsin hyvin. Tämä on todennettu myös useissa tieteellisissä julkaisuissa.

Työssä luokittelua ja avainsanojen keräämistä tutkittiin myös käytännön ympäristössä eräässä Suomessa toimivassa pankki- ja vakuutusyhtiössä. Tässä projektissa hyödynnettiin *IBM:n Content Classification Modulea*, joka käyttöönotettiin asiakasympäristössä. Tämän projektin osalta työssä esitellään saatuja kokemuksia ja muutama parannusehdotus nykyiseen järjestelmään. Projektista saatujen kokemusten perusteella tuote todettiin käyttökelpoiseksi tekstin luokitteluun ja avainsanojen poimintaan.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
MÄHÖNEN, MIKA: Text Classification
Master of Science Thesis, 100 pages
June 2013
Major: Software Engineering
Examiner: Professor Hannu Jaakkola
Keywords: information retrieval, classification, information theories and statistical methods

The purpose of this master's thesis was to study text classification and keyword extraction methods. Data structure is the most important factor when one considers, how important information can be located from a vast amount of data. There are two ways to approach locating relevant information: the first one relies on unstructured data and the second one on structured information which is known as metadata. These methods are carefully introduced with their advantages and disadvantages to the argument of why classification and keywords are needed with data warehouses. Conclusion of this study was that both approaches are required as a part of a comprehensive content management solution.

Keywords and text classification can be seen as a limited amount of observations from the text content. In fact the purpose of keywords and text classification is to provide all the necessary information. This information can then be used to locate documents that satisfy our information needs. Classification and keyword extraction process has traditionally required human interpretation known as cognition which computers do not have. Cognition has been the main reason why humans are still required in this process. To have this process automated could enhance functionality of many computer systems and save time while processing large amount of data. This matter is studied by introducing operations that are required to classify a text document and extract its keywords. This subject is divided into natural language processing and text classification algorithms. The aim of natural language processing is to remove challenges that arise from comparison of character strings in the memory of a computer. The following natural language techniques were studied: language recognition, text tokenization, lemmatization, stemming, concept modeling and feature selection algorithms. This thesis introduces two classification algorithms which are naive Bayes and decision trees. An example is given of both of them to proof theories in practice. Conclusion of this study was that the studied text algorithms have few limitations. The first limitation is that computers do not have similar understanding of words occurred in text. For example humans are able to automatically connect the word *car* to *vehicle* while computers are not. The second limitation is that word position in the text is not taken into account. Despite limitations found from classification algorithms, they do work relatively well and it has been proven by many scientific studies and publications.

Keyword extraction and text classification were studied in practice. This part of study was carried out for a company that operates within the insurance and bank sector in Finland. During the project IBM's product *Content Classification Module* was commissioned in use. Conclusion of the project was that the studied product works very well in practice. Based on this project a few improvements were found and they are being introduced to the customer.

ALKUSANAT

Tämän työn kirjoittaminen on ollut mielenkiintoinen, mutta samalla pitkäkestoinen oppimisprosessi. Haluankin kiittää kärsivällisyydestä ja tuesta erityisesti valvojaani Hannu Jaakkolaa. Kiitosta ansaitsee myös Katri Piekkola, joka on antanut tukensa kirjoitusprosessille ja kannustanut sen loppuun viemiseen. Työnantajaani Elinar Oy Ltd:tä ja Ari Juntusta haluan kiittää saadusta aiheesta ja tuesta kirjoitusprosessin loppuun viemiseksi.

Työssä tutkittava aihe on laaja-alainen kokonaisuus, sillä luokittelu ei sivua ainoastaan tietotekniikkaa, vaan myös useita muita tieteenaloja. Tästä syystä työhön on valittu monitieteellinen lähestymistapa. Työssä perehdytän lukijan ensin tiedonhaullisiin konsepteihin rakentaakseni kokonaisvaltaisen kuvan aihealueesta. Samalla pyrin perustelemaan, miksi luokittelua oikein tarvitaan. Useat työssä käsitellyistä menetelmistä ovat myös osana päivittäistä elämäämme, vaikka emme ajattele sitä. Omasta näkökulmasta erityisesti tiedonhakuun ja luokitteluun liittyvien konseptien selvittäminen on ollut yhtäaikaisesti innostavaa ja haastavaa. Näiden asioiden tutkiminen on samalla kehittänyt myös matemaattista ajatteluani osana abstraktien ongelmien ratkaisua. Aiheen matemaattisuudesta johtuen pyrin käsittelemään tärkeimpiä ideoita esimerkkien avulla. Niiden tavoitteena on avata lukijalle käsiteltyjen matemaattisten menetelmien hyödyllisyyttä. Oma tavoitteeni on ollut myös esittää menetelmiin liittyviä haasteita ja mahdollisia ratkaisumalleja.

Toivon samalla diplomityön innostavani myös muita jatkamaan aihealueen tutkimista. Monia aihealueeseen liittyviä haasteita ei ole läheskään ratkaistu. Tekstin luokitteleminen edellyttää monia eri vaiheita, jotka ovat edelleen hyvin laaja-alaisen tutkimuksen kohteita. Osasta aihealueista on varsin niukalti saatavilla tutkimusmateriaalia suomen kielen erityispiirteiden kannalta. Kielellisistä haasteista pyrin esittämään muutamia havaitsemiani esimerkkejä.

Porissa 29.4.2013,

_____

Mika Mähönen

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

BLOB	Binary Large Object. Tietokannan tietotyyppi, johon voidaan tallentaa binääristä dataa.
DBMS	Database Management System. Tietokannan hallintajärjestelmä.
DP	Decision Plan. IBM Content Classification Modulen osa, joka sisältää suoritettavia toimenpiteitä ja ehtoja.
DTD	Document Type Definition. XML-dokumenttien rakenteen kuvaamiseksi kehitetty erityinen kuvausmenetelmä.
Filenet	IBM:n sisällönhallintajärjestelmä, joka tunnetaan myös P8 tuoteperheenä.
IDF	Vektorimallissa käytetty matemaattinen menetelmä, jolla voidaan arvioida sanan merkityksellisyyttä dokumentin erottamiseksi tietovarastosta.
IR	Information Retrieval. Koostuu joukosta matemaattisia menetelmiä informaation hakemiseksi tietovarastosta.
KB	Knowledge Base. IBM Content Classification Modulen komponentti, joka oppii sille annetun palautteen perusteella.
LN	Learning Node. IBM Content Classification Modulessa tekstille valittavissa oleva luokka.
MI	Mutual Information. Matemaattinen menetelmä ominaisuuksien valintaan.
MLE	Maximum Likelihood Estimates. Yleinen tapa arvioida naiivissa Bayesian päättelyssä posteriori- ja priori-todennäköisyydet.
Naiivi Bayesian päättely	Tekstin luokittelumenetelmä, joka perustuu posteriori- ja priori-todennäköisyyksien käyttöön.

NER	Named Entity Recognition. Järjestelmä, jolla pyritään tunnistamaan tekstissä esiintyviä nimiä.
NLP	Natural Language Processing. Joukko matemaattisia menetelmiä, joiden tehtävänä on matkia ihmisen kykyä tulkita ja ymmärtää ihmiskieltä.
POST	Part-of-Speech Tagger. Ohjelmisto, jonka avulla voidaan tunnistaa automaattisesti tekstissä esiintyvien sanojen sanaluokat.
Päätöspuut	Luokittelumenetelmä, joka rakentaa hierarkkisen puun koulutusmateriaalista.
RN	Rule Node. IBM Content Classification Modulen konsepti, jolla käytettävissä olevien luokkien määrää voidaan rajata.
S.M.A.R.T	System for the Mechanical Analysis and Retrieval of Text. Vektorimalliin perustuva hakujärjestelmä.
SVM	Support Vector Machines. Luokittelumenetelmä, jossa dokumentit esitetään n-ulotteisessa vektoriavaruudessa.
TF	Term Frequency. Vektorimallissa käytetty termin esiintymistiheys.
VSM	Vector Space Model. Vektorimalli eli hakumenetelmä, jossa dokumenttia koskevat sanat esitetään n-ulotteisessa vektoriavaruudessa.
XML	Extensible Markup Language. Menetelmä, jonka avulla informaation sijaintia voidaan kuvata.
XSD	XML Schema Definition. XML-dokumenttien rakenteen kuvaamiseksi kehitetty erityinen kuvausmenetelmä.

1 JOHDANTO

Diplomityössä tutkitaan millaisia tekniikoita tekstin tilastollisessa luokittelussa ja avainsanojen poiminnassa voidaan hyödyntää. Tavoitteena on selvittää lukijalle, miksi luokittelusta voi olla hyötyä ja millaisia mahdollisia rajoitteita siihen liittyy. Työhön valittujen menetelmien tarkastelussa pyritään käyttämään yleisesti alan tutkimusta ja kirjallisuutta. Luokittelusta ja avainsanojen poiminnasta saatavia hyötyjä on tarkoitus tutkia hyödyntämällä *IBM Content Classification Module* -ohjelmistoa. Ohjelmiston tutkiminen tapahtuu eräässä pankki- ja vakuutusalan yhtiössä. Esitetyllä teorialla tekijä ei halua väittää, että tarkasteltava ratkaisu käyttäisi yksinomaan esiteltyjä menetelmiä. Menetelmien tutkimisen tarkoituksena on luoda lukijalle käsitys, millaisista asioista tilastollisessa luokittelussa on kysymys. Motivaationa työn kirjoittamiselle tekijällä on ollut parantaa lukijoiden ymmärrystä luokittelun tärkeydestä osana tiedonhallinnan tarpeita.

Tietotekniikan yleistyttyä informaation hallinnassa, datan määrä on kasvanut valtavaa vauhtia. Tietovarastoista on tullut samalla yhä tärkeämpi apuväline liiketoimintaprosesseille, sillä ne sisältävät tärkeää informaatiota päätösten tekemiseksi. Usein kirjallisuudessa puhutaan myös liiketoimintatiedon hallinnasta (engl. *businness Inteligence*), jonka tavoitteena on tarjota oikeaa informaatiota, oikeille ihmisille ja oikeaan aikaan (Rud 2009, s. 3). Samalla yhä suurempi osa ihmisistä ja organisaatioista käyttää tietotekniikkaa informaation tuottamiseen ja käsittelyyn. Samalla on syntynyt uusia tarpeita myös informaation käsittelylle, välittämiselle ja tallentamiselle. Yhtäaikaisesti organisaatioilla on syntynyt tarve myös tehostaa tietovarastojen toimintaa, koska tallennettua dataa on hyvin paljon. Tehottoman järjestelmän vaarana on ajaa sen käyttäjät informaatioähkyyn. Ilman riittäviä apuvälineitä informaatioksi prosessoitavaa dataa on saatavilla enemmän kuin pystymme sitä käsittelemään. Tästä seurauksena henkilö voi kokea turhautumista, stressiä tai työmotivaation laskua. Tällöin henkilö kokee kontrollin puutteetta omiin työtehtäviinsä liittyen (Edmunds & Morris 2000, ss. 17-18). Parantamalla nykyisten menetelmien tehokkuutta voidaan mahdollisesti vähentää käyttäjien kokemaa kuormitusta. Tuloksena organisaation kyvykkyys käsitellä dataa voi tehostua. Informaation tallennus, käsittely ja hakeminen ovat tyypillisiä tapahtumia, jotka kuormittavat käyttäjän resursseja. Tutkittavalla ohjelmalla kuormitusta pyritään vähentämään tuomalla käyttäjälle esille valmiiksi dokumenttia koskeva oleellinen informaatio.

Työn tavoitteet ovat:

A. Avata luokitteluun ja tiedonhakuun liittyvät keskeiset käsitteet lukijalle
B. Selvittää tiedonhakua konseptitasolla ja käytännössä
C. Esitellä toimintamalleja ja periaatteita, joita tekstin luokittelemiseen tarvitaan

D. Etsiä millaisia rajoituksia tiedonhaku- ja luokittelutekniikoihin liittyy
E. Selvittää tilastollista luokittelua ja avainsanojen poimintaa käytännön ympäristössä

Diplomityössä luvussa 2 kuvataan tiedonhallintaa ja siihen liittyviä konsepteja. Luku perustelee, miksi tekstin luokittelua ja avainsanoja tarvitaan osana tiedonhakua tietovarastoista. Tarkoituksena on esittää haasteita, joita nykyisten menetelmien käyttöön liittyy. Luvussa 3 käsitellään avainsanojen poimintaa ja tilastollista tekstin luokittelua. Luvun tavoitteena on esittää toimenpiteet, joita tarvitaan tekstin luokittelemiseksi ja avainsanojen keräämiseen tietokoneella. Tarkoituksena on selvittää, millaisia haasteita tekstin luokittelussa tulee huomioida. Luokitteluun käytettyjä menetelmiä työssä tutkitaan, jotta varsinaisen tutkittavan ohjelmiston toimintaperiaatteita olisi mahdollista ymmärtää. Luvussa 4 esitetään esimerkki luokitteluratkaisusta eräässä Suomessa toimivassa vakuutus- ja pankkialan yhtiössä. Luvun tarkoituksena on esitellä tuotetta, jonka avulla luokittelu ja avainsanat voidaan tuottaa automaattisesti. Luvussa 5 esitetään yhteenveto diplomityöstä. Yhteenvedossa esitetään työn aikana havaitut jatkotutkimuskohteet ja saadut johtopäätökset.

2 INFORMAATION HALLINTA

Luvussa 2 käsitellään tiedonhakua ja informaation hallintaan liittyviä käytäntöjä ja tekniikoita. Tämä tapahtuu perehdyttämällä lukija keskeisimpien menetelmien toimintaan informaation rakenteen näkökulmasta. Samalla tarkastellaan kriittisesti, mitä haasteita tekniikoiden käyttöön liittyy. Tarkoituksena on selvittää lukijalle syitä, miksi ja mihin luokittelua ja avainsanoja tarvitaan osana tietovarastojen hallintaa.

2.1 Käsitteet ja tietotekniikka

Kohdan 2.1 motivaationa on selventää tiedon, informaation ja datan suhdetta toisiinsa. Työn kannalta tarkoituksena on yhtenäistää käytettyjä käsitteitä. Näkökohdaksi tiedon, informaation ja datan selvittämiseksi on valittu Niiniluodon ja Kurosen mukainen määritelmä.

Suomen kielessä sanoja informaatio, tieto ja data käytetään monissa merkityksissä. Sanalla tieto (engl. *knowledge*) viitataan usein informaatioon (engl. *information*) tai dataan (engl. *data*). Se on kuitenkin virheellistä tarkasteltaessa tiedon alkuperäistä määritelmää. Sanalla tieto on alun alkaen tarkoitettu tien tuntemista. Tieto on Platonin klassisen filosofisen määritelmän mukaan hyvin perusteltu tosi uskomus. Tietokoneella, tiedostolla tai tietokannalla ei ole uskomuksia. Näin ollen ei voida väittää, että ne sisältäisivät suoranaisesti tietoa. Tuolloin kysymys on tallennetusta datasta. Data on luonteeltaan sellaista, mihin ei välttämättä liity merkitystä. Käsitteenä informaatio on dataa, johon liittyy merkitys ja tulkinta. (Niiniluoto 1996, ss. 7-9; Kuronen 1998) Yllä esitettyjen yhdyssanojen osalta työssä käytetään kieleen vakiintuneita sanoja tietokone, tietovarasto ja tietokanta, vaikka sanoilla viitataan tietoon.

Niiniluodon ja Kurosen määritelmä ei kuitenkaan kirjoittajan mielestä vastaa kysymykseen, mitä data oikeastaan on. Tietoteknillisessä mielessä data koostuu binäärisistä sarjoista. Kyseiset sarjat koostuvat arvoista tosi (1) (engl. *true*) ja epätosi (0) (engl. *false*). Nykymuotoisten tietokoneiden toiminta perustuu näiden tilojen hyödyntämiseen. Tilojen tehtäviin kuuluu ohjata mikropiirejä ja säilöä dataa. Mikropiirien näkökulmasta nollat ja ykköset ovat rinnastettavissa aakkosiksi, joita käytetään ohjaukseen ja informaation tallentamiseen. Analogiana luonnollisella kielellä kirjoitetut sanat puolestaan koostuvat myös sovituista symboleista, joita länsimaissa kutsutaan yleisesti aakkosiksi. Tietokoneella kyseisten symbolien esittäminen edellyttää muunnosta binääriseen muotoon. Tällä tavoin ne voidaan tallentaa mikropiirin muistiin. Kyseinen operaatio edellyttää sopimusta kirjaimen esitystavasta bitteinä. Tästä sopimuksesta käytetään nimeä merkistöstandardi (engl. *character encoding*). Tunnettuja merkistöstandardeja ovat UTF-8 ja ANSI. Merkistöstandardin tehtävä on määrittää yhteinen tapa esittää kirjain-

muotoisia symboleita binääriluvuilla. Tietokoneen muistiin esimerkinomaisesti tallennettu termi *data* näyttää ANSI-merkistössä seuraavalta (kuva 1).

Kirjaimet	**d**	**a**	**t**	**a**
Binääri	01100100	01100001	01110100	01100001

Kuva 1. *Sana data esitettynä ANSI-merkistössä*

Annetun esimerkin perusteella voidaan huomata, että luonnollisella kielellä kirjoitetut sanat tietokoneen muistissa koostuvat joukosta binäärilukuja. Sanoja esittävät binääriluvut eivät suoranaisesti ole tietokoneelle komentoja, joten niillä ei ole tietokoneelle samanlaista merkitystä kuin meille. Oma ymmärryksemme koostuu monimutkaisemmasta ilmiöstä kuin joukosta symboleita. Sopimalla esitettäville kirjaimille symbolit ei voida puhua tekstin ymmärtämisestä. Tilanne on verrattavissa vieraaseen kieleen, jota emme ole koskaan oppineet. Tuolloin aivomme eivät kykene antamaan käytetyille ilmaisuille mitään merkitystä. Toisin sanoen mikropiireillä ei ole käsitystä tekstin merkityksistä (PcMag 2012; Nrsi 2001). Lauseita ja sanoja on yritetty usein mallintaa kirjoitettujen sääntöjen avulla. Tässä ei kuitenkaan toistaiseksi ole onnistuttu, koska ihmisten käyttämät sanat ja lauseet ovat usein merkitykseltään epätäydellisiä ja abstrakteja. Lisäksi lauseissa käytetty kielioppi koostuu joukosta epätarkkoja sääntöjä, joita ei ole toistaiseksi onnistuttu mallintamaan täydellisesti loogisiksi ehdoiksi. Tämän ongelman vaikeutta ja laajuutta kuvastaa kielentutkijana tunnetun *Edward Sapirin* sanat ”*jos kieli olisi koskaan niin täydellisen johdonmukainen, se olisi täydellinen tapa ilmaista eri käsitteitä. Valitettavasti tai ehkä onneksemme yksikään kieli ole johdonmukainen ja kaikki kieliopit vuotavat*” (Sapir 1921, s. 38). Ihmiskielen sisältämistä epätarkkuuksista ja tulkinnanvaraisuuksista johtuen kielenrakenteen mallintaminen loogisin lausein on ehkä mahdotonta. Nykytutkimus onkin keskittynyt suurelta osin kielen tilastolliseen mallintamiseen sen sijaan, että tekstiä yritettäisiin tulkita hyödyntäen kiinteästi muodostettuja sääntöjä.

Termillä sisältö työssä viitataan sellaiseen joukkoon bittejä, jotka edellyttävät tulkintaa tullakseen informaatioksi. Sisältöä ovat sähköpostit, kuvat ja dokumentit (Zhu et al. 2009, ss. 5-6). Sanalla dokumentti työssä tarkoitetaan tiedostoa, joka sisältää tekstiä. Määritelmä termille dokumentti on myös subjektiivinen. Käsiteltäessä sähköpostia on usein hankalaa sanoa muodostavatko liitteet ja viesti yhden dokumentin vai pitäisikö ne käsittää erillisinä dokumentteina. Toisaalta sähköpostiviestillä ja siihen kuuluvilla liitteillä on yleensä jokin sidos keskenään. Kyseinen side on usein vaikeasti määriteltävä, joten työssä dokumentilla tarkoitetaan yksittäistä tiedostoa (Manning et al. 2009, s. 21). Kirjoittajan tulkinnan mukaan sähköposti ja sen liitteet ovat eri dokumentteja. Suurin osa informaatioon liittyvistä käsitteistä on subjektiivisia. Tuolloin on hankalaa sopia sanoille vain yhtä merkitystä, jonka kaikki osapuolet hyväksyisivät keskenään. Samasta syystä myös eri ihmiset voivat ymmärtää lukemansa hieman eri tavoin.

2.2 Menneisyyden tietovarastot

Kohdassa tarkastellaan tietovarastojen toimintaa ennen nykyisten järjestelmien kehittymistä. Tarkoituksena on avata lukijalle, etteivät tietovarastot liity ainoastaan tietotekniikkaan. Erilaisia tietovarastoja on ollut olemassa jo paljon ennen tietotekniikkaa. Samalla tarkoituksena on tutkia aikaisempia käytäntöjä, joita voidaan hyödyntää informaation hallinnassa. Tämä tapahtuu perehdyttämällä lukija informaation eri aikakausiin ja tietovarastojen hallitsemisessa käytettyjen menetelmiin ja niiden kehitykseen.

Informaation aikakaudet

Ihmiskunnan historiassa opitun tiedon tallentamisessa ja välittämisessä eteenpäin voidaan jakaa kolmeen aikakauteen. Ne ovat esimoderni, moderni ja postmoderni aikakausi. Jokaisella aikakaudella on tyypillisesti oma informaation siirtotapansa, välineistönsä ja käytäntönsä. Esimodernilla aikakaudella informaatiota siirrettiin sanallisesti sukupolvelta toiselle. Tuolloin ihmiset luottivat oman kehonsa asettamin rajoihin opitun tiedon varastoimiseksi. Osa opitusta tiedosta saattoi tuhoutua tai muuttua olennaisesti ennen sen päätymistä seuraavalle sukupolvelle. Modernia aikakautta kuvaa perinteisesti kirjoitusvälineiden ja kirjoitustaidon kehittyminen. Uusien välineiden ja tekniikoiden ansiosta ihminen saattoi säilöä tietämystään sukupolvelta toiselle, jos tallennusväline säilyi fyysisesti. Aikakauden alkupuolella käytettiin välineinä savitauluja, papyruksia, jonka jälkeen yleistyivät kirjat ja erilaiset painotuotteet (Kapitzke 2001, ss. 1-3). Postmodernille aikakaudelle siirryttäessä merkittävä edistysaskel tapahtui, kun Vanner Bush vuonna 1945 esitti tietokoneiden soveltamista informaation säilömiseen artikkelissa *As We May Think*. Hän totesi sen aikaisten käytössä olevien menetelmien olevan liian tehottomia tuotetun informaation hyödyntämiseen järkevästi (Bush 1979, ss. 37-37). Tähän oli syynä saatavilla olevan datan määrän kasvu. Tietokoneiden kehittyminen mahdollisti informaation tallentamisen digitaaliseen muotoon nolliksi ja ykkösiksi (Kapitzke 2001, ss. 1-3). Tämän menetelmän ansiosta voimme tallentaa informaatiota fyysisesti yhä pienempään tilaan ja edullisemmin. Nyt yli 50 vuotta myöhemmin informaation tallentamisesta on tullut arkipäivää ja käytetyt laitteet ja välineet ovat yhä suuremman ihmisjoukon saatavilla. Samalla saatavilla olevan informaation määrä on kasvanut räjähdysmäisesti (Feather 1998; Edmunds & Morris 2000). Henkilökohtainen kykymme prosessoida dataa ei ole kuitenkaan kasvanut samassa suhteessa kuin keksittyjen välineiden. Tallennetun informaatiomäärän kasvaessa joudumme käsittelemään dataa yhä nopeammin ja enemmän. Tästä johtuen käsiteltävää dataa on saatavilla enemmän kuin ehdimme sitä prosessoida. Kyseistä tilannetta nimitetään informaatioähkyksi (engl. *information overload*) (Edmunds & Morris 2000, ss. 17-22). Tarkastelemalla tietokoneiden tallennuskapasiteetin kasvua voidaan havaita, että nykyään saatavilla olevat tallennusmediat ovat huomattavasti suurempia kuin 50 vuotta sitten käytetyt. Vuonna 1956 IBM julkaisi ensimmäisen kiintolevyn, jonka koko oli vain 5 megatavua ja se oli kooltaan noin kahden jääkaapin kokoinen. (Farrance 2006). Vuonna 2012 saatavilla oli jo 4 teratavun kokoisia kiintolevyjä, joiden koko on vain 3.5 tuumaa (Hgst

2012, ss. 1-2). Yksittäiselle kiintolevylle tallennettavissa oleva datamäärä on kasvanut lähes tuhatkertaiseksi 56 vuodessa. Samalla fyysinen koko on pienentynyt murto-osaan ensimmäisestä kiintolevystä. Internetin kehittymisen myötä myös välineet datan siirtämiseksi laitteiden välillä kehittyivät. Nykyään dataa siirretään yhä enemmän sähköisessä muodossa.

Välineet tietovarastojen hallintaan

Tiedonhallinnassa käytettyjen välineiden kehityksen voidaan ajatella lähteneen kirjoitustaidon kehittymisestä. Ensimmäiset viittaukset tarpeesta järjestää tietovaraston kirjoituksia löytyvät noin 3000 vuotta ennen ajanlaskumme alkua. Muinaiset sumerialaiset kirjasivat nuolenpääkirjoituksella tietämystään tauluihin, joiden tehtävänä oli auttaa valtakunnan hallinnassa. Arkeologiset todisteet viittaat siihen, että näitä tauluja pidettiin tietyssä järjestyksessä informaation löytämiseksi (Kapitzke 2001, ss, 3-5; Singhal 2001, ss. 1-2). Tämä voidaan nähdä alkusysäykseksi kohti nykymuotoista tietojenhallintaa. Tämän jälkeen tietovarastojen kannalta merkittäviä keksintöjä ovat olleet indeksointi (engl. *indexing*) ja lyhennelmät (engl. *abstracting*). Alun perin indeksoinnilla tarkoitettiin merkintää papyrusrullissa, jonka perusteella ne on voitu tunnistaa. Historiassa indeksointi alkoi pitkien teosten otsikoinnilla ja teosten lyhentämisellä. Otsikointi mahdollisti viittauksien luomisen eri teosten välillä ja paransi samalla tiedonhakua. Lyhentämisellä tarkoitetaan teoksen alkuun tai ulkopuolelle tai kappaleen alkuun tehtyä tiivistelmää sen pitämästä sisällöstä. Lyhennelmien käytön tiedetään yleistyneen Callimachuksen aikoihin Aleksandriassa. Tuolloin pitkiä teoksia on lyhennytty, jotta niiden hyödyntäminen olisi vaivattomampaa. (Francis. 1973, ss. 194-196). Lyhentämisen ansiosta lukija saattoi käydä läpi vain teoksen lyhennelmän ja tehdä omat johtopäätöksensä lukematta koko teosta. Edellä kuvatut kaksi käsitettä liittyvätkin historian alkutaipaleella vahvasti toisiinsa. Indeksointi on sittemmin kehittynyt nykyiseen muotoonsa ja se on saanut uusia käytäntöjä ja piirteitä. Indeksoinnilla kirjastotieteissä tarkoitetaan nykyään teosta kuvaavien asiasanojen määrittämistä. Asiasanastosta käytetään myös nimeä Tesaurus, jolla viitataan tiettyyn vakiintuneeseen tapaan kuvata tekstin sisältöä (Alaterä et al. 2005). Käsiteltäessä varhaista indeksointia on otettava huomioon, että varhaiset indeksit muistuttivat enemmän lyhennelmiä kuin nykymuotoisia indeksejä. Indeksoinnin kehittyminen nykymuotoonsa on sittemmin mahdollistanut kirjaston teosten lajittelun hyllyihin asiasanoittain. Se onkin eräs merkittävimmistä apuvälineistä informaation paikantamiseksi tietovarastosta. Ilman loogisesti suunniteltua järjestystä suurien tietovarastojen käsittely olisi hyvin hankalaa tai epäkäytännöllistä. Indeksointi on tapa muodostaa looginen järjestys. Käytettäessä nykymuotoista indeksiä teokset järjestellään usein nimen perusteella aakkosittain. Historiassa ensimmäiset järjestykset eivät kuitenkaan ole perustuneet aakkosten käyttöön. Indeksoinnin merkityksellisyydestä huolimatta, historia ei tunne sen keksijää. Kirjainjärjestyksen käyttöönoton arvellaan tapahtuneen kuitenkin niihin aikoihin, kun muinaiset kreikkalaiset ottivat käyttöön semiittisen kirjoitusjärjestelmän (engl. *semitic writing system*). Kreikkalaisten uskotaan perineen kirjainjärjestys käytettyjen aakkosten mukana. Tämä on tapahtunut ennen ajanlaskumme al-

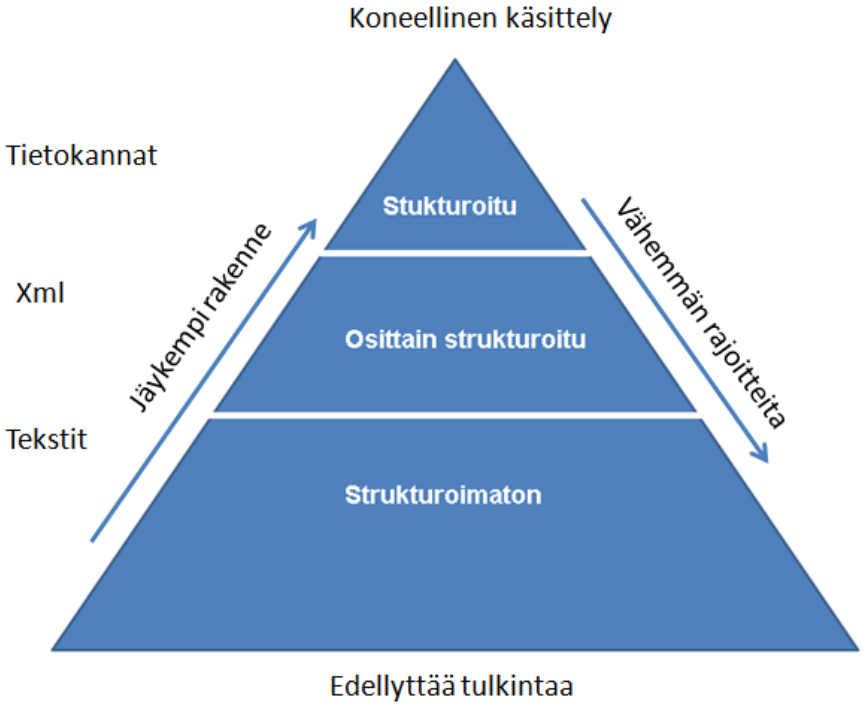
kua, mahdollisesti 700-800 luvulla tai aikaisemmin. Francis J. Wityn mukaan indeksointi ei ollut kovin yleistä ja se koki huomattavia parannuksia vasta 1500-luvulla (Francis 1973, ss. 195-196). Nykymuotoinen indeksointi on vakiinnuttanut paikkansa vasta varsin myöhään historiassa. Luokittelu ja sitä koskevat teoriat ovat syntyneet länsimaisiin tieteisiin pitkän ajan kuluessa. Ne eivät ole olleet historiassa yhtä selkeästi liitoksissa tietovarastojen käsittelyyn kuin indeksointi tai lyhentäminen. Luokittelun kehittyminen länsimaisissa tieteissä juontaa luonnontieteisiin ja filosofiaan. Tiettävästi länsimainen luokittelu (engl. *taxonomies*) kehitettiin Kreikassa muutamia satoja vuosia ennen ajanlaskumme alkua. Aristoteleen tiedetään pyrkineen määrittämään eläimille luokittelua (Manktelow 2012, ss. 1-5). Aristoteleen kirjoitukset sisältävät neljä periaatetta eläinten luokittelemiseksi, jotka ovat elintapa (engl. *mode of life*), toiminnot (engl. *activities*), piirteet (engl. *characters*) ja eläinten ruumiin osat (engl. *parts*). (Pellegrin 1986, ss. 5-7). Tämän lisäksi Aristoteleen tiedetään pohtineen luokittelua myös väitelauseiden termeille (Edghill 1994). Sen jälkeen luokittelua ovat tutkineet tai soveltaneet useat merkittävät historian henkilöt kuten Francis Bacon, Carl Linnaues, Denis Diderot ja Jean le Rond d'Alembert. Heistä Francis Bacon kehitti teorian tietämyksen puusta (engl. *tree of knowledge*). Kyseisen teorian pohjalta Denis Diderot ja Jean le Rond d'Alembert kehittivät 1750-luvulla Encyclopedian (Welty & Jenkins 1999, ss. 162-165). 1700-luvulla elänyttä Carl Linnausta voidaan pitää myös merkittävänä luokitteluteorian kehittäjänä. Hänen esittämänsä malli mahdollisti koko eliökunnan jakamisen luokkiin, sukuihin ja lajeihin. Linnausta pidetäänkin usein modernin luokittelun isänä (Manktelow 2012, ss. 1-5). Kirjastot puolestaan alkoivat järjestellä teoksiaan aiheittain (engl. *subject*) vasta 1800-luvulla, jolloin LOC (engl. *library of congress*) perustettiin. Tuolloin vasta luokittelu vakiinnutti asemaansa yleisesti hyväksyttynä tapana järjestellä aineistoa. Historiasta kuitenkin löytyy tätäkin varhaisempia viitteitä luokittelulle kirjastoissa. (Verner & Adrien 1968) mukaan varhaisin aiheittain järjestelty katalogi tiedetään olevan peräisin Itävallasta vuodelta 1483. Luokittelun nähdään kuitenkin yleistyneen tietovarastojen yhteydessä vasta LOC:in perustamisen jälkeen (Welty & Jenkins 1999, ss. 162-165). Tietovarastoissa käytettyjen menetelmien kehityksen voidaan nähdä alkaneen savitaulujen järjestelemisestä. Tätä seurasi teosten indeksointi ja lyhentäminen informaation hakemiseksi. Luokittelu on alkanut luonnontieteiden puolella huomattavasti aikaisemmin kuin kirjastoissa. Tiedonhaku nykyisistä tietovarastoista perustuu edelleen ikivanhojen ideoiden hyödyntämiseen.

2.3 Datan rakenne

Datan rakenne on keskeisessä asemassa, kun tarkastellaan tietovarastojen toimintaa, haasteita, hyödyntämistä ja informaation etsintää. Kohdan motivaationa on selvittää haasteita ja mahdollisia ratkaisuvaihtoehtoja, joita eri datan muotoihin liittyy. Tarkoituksena on avata keskustelu siitä, miten dataa voitaisiin hyödyntää vaivattomammin.

Datan rakenne on keskeinen tekijä puhuttaessa tiedonhausta (engl. *information retrieval, ir*) ja sen eri muodoista. Tästä huolimatta datan rakennetta käsittelevässä ter-

minologiassa on usein eroavaisuuksia riippuen käytetystä lähteestä. Osa lähteistä jakaa rakenteen karkeasti strukturoituun (engl. *structured data*, *structured content*) ja strukturoimattomaan (engl. *unstructured data*, *unstructured content*) dataan. Osassa lähdemateriaalista esitellään lisäksi välimuoto nimeltään osittain strukturoitu data (engl. *semi-structured data*, *semi-structured content*). (Zhu et al. 2009, ss. 1-6; Zhu et al. 2011, ss. 29-39; Sint et al. 2012, ss. 1-7; Blumberg & Atre 2003, ss. 1-2; Butler 2012). Työhön on valittu tarkasteltavaksi strukturoitu, osittain strukturoitu ja strukturoimaton data (kuva 2). Tarkoituksena on esittää kunkin datan rakenteen vahvuudet ja heikkoudet informaation hyödyntämisen näkökulmasta.



Kuva 2. *Datan rakenne*

Strukturoimattomalla datalla tarkoitetaan informaation tallennustavan rakennetta, joka edellyttää tulkintaa. Tyypillisiä strukturoimattoman datan lähteitä ovat tekstit, valokuvat, video- ja äänitiedostot. Työssä keskitytään luonnollisella kielellä tuotetun tekstin tarkasteluun. Ensimmäinen käytetty määritelmä liittyy tekstin ennalta määrittämättömään rakenteeseen. Toinen kirjallisuudessa esitetty määritelmä strukturoimattomalle datalle on, että sen tallentaminen tietokantaan edellyttää BLOB-tietotyypin (engl. *binary large object*) käyttöä. (Sint et al. 2012, ss. 1-7) Kirjoittajan mielestä kumpikaan määritelmä ei ole hyvä. Osalla teksteistä on ennalta määritelty ja selkeä rakenne. Asiakirjana laskun on täytettävä arvonlisäverolain pykälässä 209 b asetetut vaatimukset. Lain ensimmäisen kohdan mukaisesti laskulta tulee olla antamispäivä ja kohdan 10 mukaan *suoritettavan veron määrä euroissa, ei kuitenkaan 13 kohdassa tarkoitetun myynnin osalta.* (L 25.4.2003/325. Arvonlisäverolaki) Laki ei kuitenkaan aseta tarkkaa järjestystä asioiden esittämiselle, voimme edelleen päättää asioiden esitysjärjestyksestä itse. Laskun tulee täyttää ainoastaan lain vaatimukset. Tietenkin yleisesti hyvän tavan mu-

kaisesti lasku noudattaa jotakin tuttua pohjaa. Laki ei takaa kuitenkaan asioiden esitysjärjestystä itse paperilla tai sähköisessä muodossa. Annettu määritelmä ei poista yksiselitteisesti asiakirjaan liittyvää tulkinnan tarvetta. Tämän kaltaista dokumenttia tulisi kutsua enemminkin osittain strukturoiduksi dokumentiksi (engl. *semi-structured document*). Sitä ei tule sotkea kuitenkaan strukturoituun tai osittain strukturoituun dataan (Zhu et al. 2009, s. 4; Butler 2012; Blumberg & Atre 2003). Määritelmä BLOB-tietotyypistä on myös usein ontuva, koska tietyissä tilanteissa pitkiäkin tekstejä voidaan tallentaa tietokannan perustietotyyppeihin. Nämä tietotyypit eivät sinällään vakioi tekstin rakennetta tai muotoa ainoastaan sen, montako merkkiä voimme syöttää kyseiseen kenttään. Tietotyyppiin viittaava määritelmä kuvaa näin ollen paremmin ääni-, video- ja kuvatiedostoja kuin itse strukturoimatonta tekstiä.

Tietoteknillisessä mielessä kysymys on haasteesta, sillä strukturoimattoman informaation käsittelyyn liittyy yleensä tarve tulkintaan. Tuolloin informaation paikantamisen kannalta ongelman ydin on pystyä löytämään sellaiset merkit, jotka esittävät tiedonhaun kannalta relevanttia informaatiota. Asiaa voidaan kuvata vertauskuvalla, jossa lukutaidotonta pyydetään etsimään kaikki tekstissä esiintyvät nimet. Tehtävän suorittamiseksi tarvitaan ymmärrys tekstin sisällöstä tai lista paikannettavista symboleista nimien löytämiseksi. Saman ongelman voidaan ajatella myös koskevan mikropiirejä, koska niillä ei ole ymmärrystä tekstiin liittyvistä merkityksistä. Kyseessä on näin ollen merkittävä tietoteknillinen haaste. Ongelmaa ei ole kyetty tähän päivään mennessä ratkaisemaan. Tiedonhakuun suunnitellut järjestelmät ovat lähinnä helpottaneet informaation paikantamista ja käsittelyä. Monissa organisaatioissa tallennetun informaation hyödyntäminen on edelleen turhan hidasta ja vaivalloista. Kaikesta informaatiosta arvioidaan olevan 80 % tai enemmän strukturoimatonta dataa. Lisäksi määrän oletetaan kasvavan tulevaisuudessa entisestään (Butler 2012; Shilakes & Tylman). Datan määrän kasvu ja halu hyödyntää sitä entistä tehokkaammin ovat toimineet motivaatioina tuottaa entistä parempia ratkaisuja. Lähestymistapoja ongelmaan ovat datan rakenteen vakiointi ja eri tiedonhaun menetelmät. Tiedonhaun menetelmiin kuuluu myös tiedonlouhinta (engl. *data mining*). Tiedonlouhinnalla tarkoitetaan laajaa joukkoa eri tekniikoita, joiden avulla voidaan tunnistaa automaattisesti aikaisemmin tuntemattomia, sopivia, uusia ja käyttökelpoisia malleja tietokannoista (Gupta 2012). Tiedonlouhinta on löyhästi määriteltynä joukko matemaattisia menetelmiä informaation etsimiseksi automaattisesti datasta.

Osittain strukturoimattomalla datalla (engl. *semi-structured data*) on myös lähteestä riippuen hieman erilaisia tulkintoja. Yleensä sillä tarkoitetaan datan tallennusmuotoa, joka kykenee kuvaamaan omaa rakennettaan ilman erillistä kuvausta (engl. *description*) (Sint et al. 2012, ss. 1-7). Tyypillinen esimerkki osittain strukturoidusta datasta on XML-dokumentti (engl. *extensible markup language*). XML mahdollistaa ennalta määrittämättömien elementtien (engl. *element*) käytön datan kuvailemiseksi. Elementit mahdollistavat informaation paikantamisen dokumentista ilman sen absoluuttisen sijainnin tuntemista. Nykyään XML-standardi on eräs suosituimmista menetelmistä informaation tallentamiseksi ja siirtämiseksi järjestelmien välillä. Standardia ei alun alkaen suunniteltu kuitenkaan datan esittämiseksi käyttäjälle XML-muodossa, vaikka

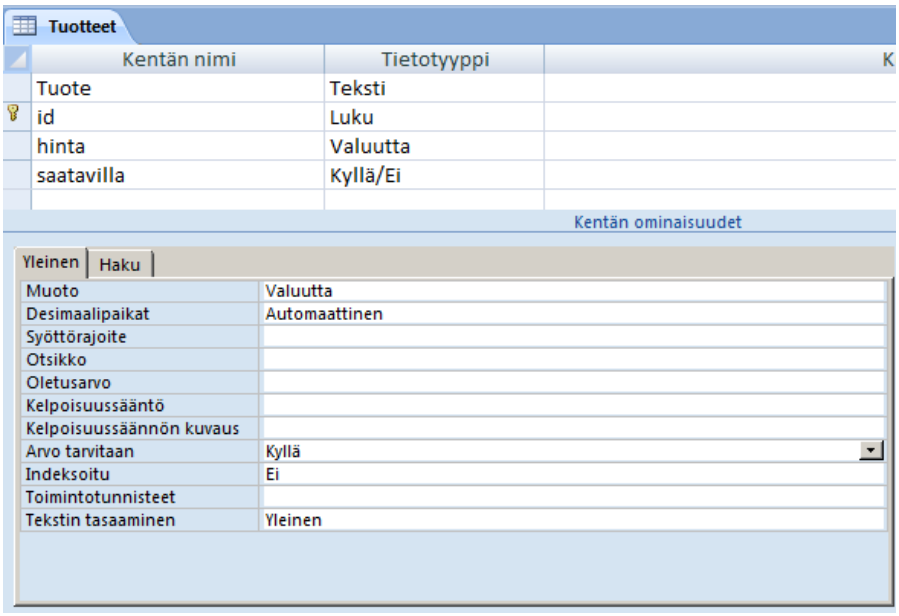
lyhyiden tiedostojen esittäminen on havainnollista (kuva 3). XML:n tehokkaan hyödyntämisen rajoitteena nähdään usein tulkkien (engl. *parser*) heikompi suorituskyky relaatiotietokantoihin verrattuna (Nicola & John 2003, ss. 1-2).

```
<?xml version="1.0"?>
<diplomityo>
    <aihe>Tekstin luokittelu</aihe>
    <tekija>Mika Mähönen</tekija>
    <valvoja>Hannu Jaakkola</valvoja>
</diplomityo>
```

Kuva 3. *Esimerkki diplomityötä kuvaavista avainsanoista*

XML:n käyttö perustuu dokumentin rakenteen määrittämiseen hyödyntäen loppu ja aloitus elementtejä. Kuvassa 3 tekijä on löydettävissä *<tekija> ja </tekija>* sisältä. Dokumentin rakenteen ollessa vakioitu myös sen ohjelmallinen lukeminen on yksinkertaisempaa. Elementtien avulla tekstin sisällön poiminta on mahdollista ilman monimutkaisia sääntöjä. Tilanne olisi aivan toinen, jos kuvaavat elementit poistettaisiin ja sisällön järjestystä muutettaisiin. Tuolloin nimien rajaaminen olisi monimutkaista, koska se edellyttäisi päättelyä. XML-standardi ratkaisee tämän ongelman kuvaamalla informaation sijaintia elementeillä. Itsessään XML-standardi ei edellytä kuvausta järjestyksestä tai käytetyistä elementeistä, mutta elementtien sijainti, määrä tai järjestys on mahdollista määrittää erillisellä dokumentilla. Määritteleminen voidaan tehdä kahdella vaihtoehtoisella standardilla, joita ovat DTD (engl. *document type definition*) ja XSD (engl. *xml schema definition*). Rakenne määritellään yleensä, kun ohjelmoija haluaa varmistua dokumentin sopivuudesta sitä tulkitsevaan ohjelmaan (W3schools 2012 a; W3schools 2012 b).

Strukturoidulla datalla (engl. *structured data*) työssä tarkoitetaan sellaista informaatiota, joka on tallennettu ennalta sovitun rakenteen ja muodon mukaisesti. Tietokannat ovat klassinen esimerkki strukturoidun datan lähteestä. Käyttäjän näkökulmasta tietokanta on kokoelma tallennettua dataa. Tätä dataa hallitaan käyttäen hallintajärjestelmää (engl. *database management system*, DBMS) (Rouse 2012; Abiteboul et al. 1994, ss. 3-4). Tietokanta perustuu datamallin (engl. *data schema*) käyttöön. Se määritetään ennen kuin tietokantaan voidaan tallentaa dataa (kuva 4).



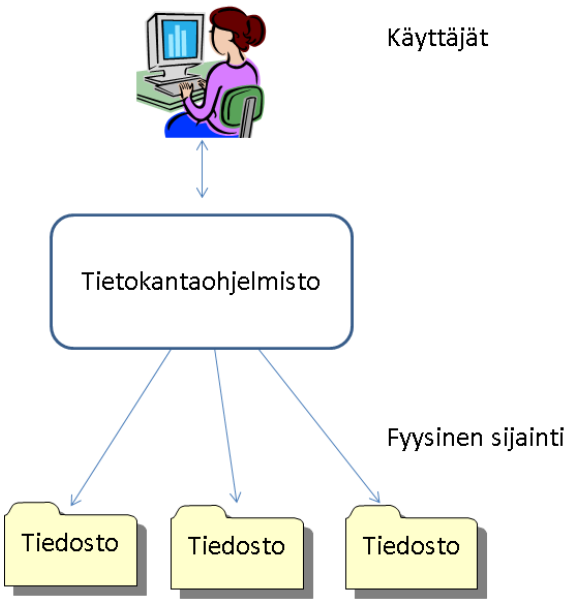
Kuva 4. *Datamalli taulusta tuotteet*

Datamalli toimii formaalina rajoituksena tietokantaan tallennettavan datan muodolle (Rybinski 1987, ss. 325-349). Formaalilla määrityksellä relaatiotietokantojen yhteydessä tarkoitetaan taulun (engl. *table*) sisältämien kenttien määrittämistä (kuva 4). Oliokantojen kohdalla kuvauksesta käytetään yleisesti nimeä objekti- tai luokkakuvaus. Muodostettu määritys toimii rajoituksena datan muodolle ja helpottaa myös informaation löytämistä. Kuvassa 5 kenttään hinta, ei voida antaa tekstimuotoista syötettä. Samalla voidaan varmistua, että tuotteen hinta on määritetty ja se löytyy samasta paikasta.

Tuote	id	hinta	saatavilla
Tuote 1	1	200,00	☑
Tuote 2	2	150,00	☐
Tuote 3	3	23,00	☑
			☐

Kuva 5. *Taulun data näkyvillä esimerkkinäkymässä*

Yleisesti tietokantoihin liittyy kaksi keskeistä periaatetta. Ensimmäinen periaate on piilottaa datan fyysinen sijainti käyttäjältä (engl. *data independence principle*). Periaatteen mukaisesti tietokantaa käyttävän henkilön ei tarvitse olla tietoinen datan fyysisestä sijainnista levyllä tai tiedostossa (kuva 6). Toinen periaate on, että dataan tarjotaan näkymiä (engl. *views*). Näin ollen tietokannat ovat kokoelmia strukturoitua dataa. Datan rakenne on strukturoitua, koska sen tallentaminen tapahtuu selkeään ja ennalta määriteltyyn rakenteeseen. (Rouse 2012; Abiteboul et al. 1994, ss. 3-4). Se ei takaa kuitenkaan, etteikö tulkintaa tarvittaisi.



Kuva 6. *Tietokantaohjelmiston tehtävät mukaillen (Abiteboul et al. 1994, s. 4.)*

Luonnollisella kielellä tuotettu informaatio ei omaa sellaista rakennetta tai yksiselitteisyyttä. Tätä rakennetta ei voida kuvata muodossa, jonka tietokone ymmärtäisi. Tietokoneet käsittelevät tekstiä joukkoina bittejä, jotka esittävät merkkejä. Informaation etsimisen kannalta on yksinkertaisempaa, jos etsintä voidaan rajata tiettyyn rakenteeseen. Rakenteen vakiinnuttamisen tavoite on kuvata informaation sijaintia ja vakioida sen muotoa. Annettuja rajoitteita hyödyntämällä aineiston rajaaminen onnistuu yksinkertaisemmin, koska kaikkia esitystapoja informaatiolle ei tarvitse ottaa huomioon. Toisaalta strukturoimalla ilmaisua menetetään osa sanallisen ilmaisun rikkaudesta. Tästä syystä kaiken datan strukturoiminen ei ole usein mahdollista. Se on toiminut motivaationa myös tarkastella, mitä vaihtoehtoja informaation paikantamiseksi datasta on saatavilla. Menetelmien esittelyn tarkoituksena on luoda kokonaisvaltainen kuva, mistä tiedonhaussa on kysymys. Esitellyt menetelmät luovat myös perustan tiedonlouhinnassa käytettyjen ajatusten esittämiselle.

2.4 Tiedonhaku

Kohdan tavoitteena on esitellä lukijalle, mitä tiedonhaku strukturoimattomasta datasta tarkoittaa tietojärjestelmien näkökulmasta.

Tiedonhaku on laaja-alainen aihe ja sillä voidaan tarkoittaa lähes minkä tahansa informaation etsimistä. Yleisesti se on informaatiotarpeen tyydyttävän materiaalin etsimistä tietovarastosta. Työssä etsittävä kohde on dokumentti, joka sisältää strukturoimatonta dataa eli tekstiä. Monet tietovarastot koostuvat strukturoimattomasta datasta. Tietovarastolla ei tarkoiteta tässä yhteydessä ainoastaan tietokantaa, vaan paikkaa dokumenttien tallentamiseksi. Levylle määritetty hakemistorakenne voi toimia näin ollen alkeellisena tietovarastona. Se tarjoaa rajalliset välineet tiedonhakuun. (Manning et al. 2009, ss. 1-20) Yleisellä tasolla tarkasteltaessa, tiedonhakujärjestelmissä hakeminen voi

kohdistua myös muuhun sisältöön, kuten valokuviin, ääni- ja videotallenteisiin. Tiedonhakujärjestelmällä tarkoitetaan työssä sellaista ohjelmaa tai joukkoa sovelluksia, joilla informaatiota voidaan hakea hyödyntäen kyselyä (engl. *query*). Web-hakukoneet ovat tiedonhakujärjestelmiä. Tietokantajärjestelmä on puolestaan ohjelmisto, jonka pääasiallinen tehtävä on varastoida dataa. Sen tehtävänä on tarjota erilaisia hakuominaisuuksia aineiston etsintään. Edellä mainittujen käsitteiden erottaminen on usein hankalaa, koska järjestelmät jakavat paljon yhteisiä piirteitä. Merkittävin ero liittyy siihen, että tiedonhakujärjestelmät keskittyvät informaation hyödyntämiseen ja tietokantajärjestelmät datan säilyttämiseen.

Tiedonhakujärjestelmät voidaan luokitella niiden koon ja käyttötarpeen mukaisesti. Niitä on saatavilla niin henkilökohtaiseen, organisaation kuin Web-käyttöön (Manning et al. 2009, ss. 1-20). Edellä mainituista järjestelmistä jokaisella on omat käyttäjäryhmät, joilla on myös omat tarpeet tiedonhaulle. Tiedonhakujärjestelmien välillä merkittävimmät erot löytyvät tietovarastojen hallinnasta. Tämä johtuu eri tietovarastojen rakenteesta ja ominaisuuksista. Henkilökohtaisissa ja organisaatiolle suunnatuissa järjestelmissä on yleensä mahdollista vaikuttaa datan varastoimistapaan. Web-hakukoneen ylläpitäjän näkökulmasta tietovaraston sisältöön tai rakenteeseen ei voida vaikuttaa. Tämä johtuu Internetin rakenteesta, joka ei ole yksittäisen henkilön hallinnoima järjestelmä. Se muodostuu useista toisiinsa kytketyistä tietokoneista, joita hallitsee useat eri tahot. Tästä syystä Internetissä on vaikeaa soveltaa datan strukturointiin perustuvia menetelmiä. (Manning et al. 2009, ss. 2-3) Seuraavaksi tutkitaan haku- ja rajausmenetelmiä, joita voidaan hyödyntää informaation hakemiseen dokumenteista. Menetelmien käsittelyn tarkoituksena on luoda ymmärrys tiedonhaun perusmenetelmistä. Käsitteiden avaaminen antaa samalla mahdollisuuden vertailla menetelmien hyötyjä ja huonoja puolia. Tarkastelun tavoitteena on perustalla, miksi strukturoitua dataa tarvitaan yhä osana kokonaisratkaisua.

2.4.1 Yleiset haku- ja rajausmenetelmät

Merkkijonohaku (engl. *full text search*) on yksi yksinkertaisemmista tekstinhakutekniikoista. Sen avulla voidaan etsiä sopivaa merkkijonoa tiedostosta tai tietorakenteesta. Kyseinen haku kohdistetaan yleensä tiedostoihin, mutta myös tietokantaa voidaan käyttää. Unix-järjestelmissä tiedostoista informaatiota voidaan hakea tällä tavoin grep-komennolla. Tuolloin puhutaan merkkijonon etsimisestä peräkkäisistä tiedostoista (engl. *linear scanning*). Tiedostojen määrän kasvaessa haasteeksi muodostuu kuitenkin hakumenetelmän tehottomuus. Tämä johtuu siitä, että jokainen tiedosto tulee käsitellä erikseen merkkijonon hakemiseksi. Tästä syystä menetelmää ei sovelleta modernien hakujärjestelmien toteuttamisessa. Tehokkaiden rajausten tekeminen on merkkijonohaun eräs suurimmista ongelmista. Tämän menetelmän haasteena on myös sopivien rajausten muodostaminen. Lyhyellä merkkijonolla haettaessa haun rajaavuus on usein liian pieni ja se tarkoittaa suurta määrää hakutuloksia. Pitkällä merkkijonolla haettaessa ongelmaksi muodostuu puolestaan liian suuri rajaus. Tuolloin saatuja hakutuloksia on liian vähän. Tämä johtuu menetelmän vaatimuksesta täydelliseen täsmäävyyteen (engl. *exact match*)

(Manning et al. 2009, s. 3). Erityisesti suomen kielessä haasteeksi muodostuvat sanojen päätteet, taivutukset ja yhdyssanat. Haku *”tilastollinen tekstin luokittelu”* ei täsmää dokumenttiin, jossa esiintyvät sanat *"tilastolliset tekstin luokittelumenetelmät"*. Vapaatekstihaku (engl. *free text query)* on menetelmä, jossa käyttäjä antaa yleensä listan asiasanoista. Tuolloin hakujärjestelmän tehtävä on tarkastella annettujen avainsanojen esiintymistä käänteistiedostossa (engl. *inverted file* tai *index*). Käänteistiedostoa tarvitaan, koska dokumentteja ei ole mahdollista käydä läpi peräkkäin. Sen toiminta perustuu ideaan esikäsitellä haettavat termit valmiiksi tietorakenteeseen. Tämä poistaa tarpeen etsiä sana tai sanoja peräkkäisistä tiedostoista (Manning et al. 2009, ss. 3-10). Itsessään käänteistiedoston avulla ei voida rajata hakuun vastaavia dokumentteja. Tästä syystä on kehitetty joukko erilaisia hakumalleja. Muutamia niistä ovat Boolen malli (engl. *boolean model, boolean information retrieval*) ja vektorimalli (engl. *vector space model*). Näitä hakumalleja työssä on tarkoitus käsitellä omina kohtinaan.

2.4.2 Käänteistiedosto

Käänteistiedosto on perustekniikka, jota hyödynnetään useissa eri tiedonhakujärjestelmissä. Sen idea perustuu ajatukseen esiprosessoida tiedostoissa esiintyvät sanat erillisiksi avaimiksi (engl. *token*) käänteistiedostoon. Tuolloin käänteistiedoston avulla voidaan selvittää, jos avain esiintyy dokumentissa. Käänteistiedoston muodostamiseen kuuluu neljä eri vaihetta, jotka ovat:

1. Indeksoitavien dokumenttien valinta
2. Tekstin pilkkominen avaimiksi (engl. *tokenization*)
3. Sanojen kielellinen käsittely (engl. *linguistic preprocessing*)
4. Käänteistiedoston muodostus

Tiedoston muodostaminen alkaa pilkkomalla tekstissä esiintyvät lauseet (engl. *tokenization*) avaimiksi (engl. *tokens*). Avain on eräänlainen ehdotelma löydetystä sanasta. Se ei kuitenkaan ole sama asia kuin sana. Tämä johtuu oletuksesta, jolla avaimet muodostetaan. Sanojen erottamiseksi tekstistä yleinen oletus on käyttää välilyöntiä, pistettä tai pilkkua (Manning et al. 2009, ss. 22-33). Tämä oletus ei toimi kaikissa tilanteissa oikein. Kaupungin nimi "*Hong Kong*" tuottaa kaksi avainta. Merkkijono "*äidinkieli*" tuottaa yhden avaimen, kun taas merkkijono "*äidin kieli*" tuottaa kaksi avainta. Ilman tekstiin liittyvää merkitystä ei ole mahdollista päätellä, onko kyseessä yhdyssanavirhe. Tästä syystä käänteistiedoston muodostamiseen tarvitaan myös kielellistä prosessointia. Muita keskeisiä haasteita merkkijonojen vertaamiseksi keskenään ovat sanojen taivutukset. Yleinen tapa onkin poistaa avaimiin liitetyt sanojen taivutusmuodot, jotta merkkijonoja voitaisiin verrata keskenään. Muutamia kirjallisuudessa esiteltyjä menetelmiä ovat *stemmimng* ja *lemmatization* (Manning et al. 2009, ss. 22-33). Poistamalla sanojen taivutukset vähennetään myös muodostuvan käänteistiedoston kokoa. Tuolloin jokaista sanamuotoa ei tarvitse käsitellä omana indeksinään. Kyseisiä tekniikoita on tarkoitus käsitellä tarkemmin osana luokittelua.

Annettujen työvaiheiden perusteella dokumenteista muodostetaan korpus (engl. *corpus*). Se pitää myös sisällään listan dokumenteista kerätyistä avaimista. Yhdessä

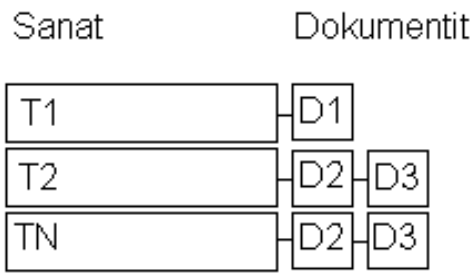
dokumentit ja niihin liitetyt avaimet muodostavat dokumenttitermimatriisin (engl. *document term matrix, term-document incidence matrix*). Kuvitteellinen esimerkki dokumenttitermimatriisista annetaan taulukossa 1.

Taulukko 1. *Kuvitteellinen dokumenttitermimatriisi*

	Termi 1	**Termi 2**	**Termi N**
D_1	1	0	0
D_2	0	1	0
D_3	0	1	1

Annetussa dokumenttitermimatriisissa korpukseen kuuluvat termit (1,2,..N). Ne esittävät, mitä avaimia kussakin dokumenteissa D_1, D_2 ja D_3 on. Dokumenttitermimatriisiin kirjatut arvot riippuvat usein käytetystä hakumallista. Boolen mallissa kuinkin termin kohdalle merkitään (1), jos se löytyy dokumentista. Tuolloin dokumenttitermimatriisi ei pidä kirjaa sanojen esiintymistiheyksistä dokumentissa (Manning et al. 2009, s. 3). Vektorimallissa puolestaan esiintymistiheydet on merkitty taulukkoon. Tuolloin matriisista käytetään myös nimitystä *bag-of-words model* (Manning et al. 2009, ss. 22-33). Yhteistä molemmille malleille on, etteivät ne ota kantaa avaimen sijaintiin dokumentissa.

Todellisuudessa dokumenttitermimatriisi ei ole sellainen tietorakenne, jossa avaimet ovat tietokoneen muistissa. Menetelmänä se on lähinnä havainnollinen tapa esittää dokumentteihin kuuluvat avaimet. Muistin kannalta matriisin ongelmana on sen vaatima tila sanojen, kun dokumenttien tai sanojen määrä kasvaa (Manning et al. 2009, s. 7). Asia on havaittavissa kahden esimerkin avulla. Oletetaan, että jokaiselle solulle esitetyssä taulukossa 1 tarvitaan yhden bitin verran muistia. Tässä tapauksessa taulukon vaatima tila saadaan kertomalla matriisin sarakkeiden ja rivien määrä keskenään. Annettu taulukko vaatii 3x3 = 9 bittiä muistia, joka on vain ~1,124 tavua. Tilan käytön tarve kasvaa voimakkaasti, jos sanojen tai dokumenttien määrä kasvaa. Oletetaan sanojen määrän olevan kiinteästi satatuhatta ja tietovarastoon kuuluvan miljoona dokumenttia. Tuolloin muistivaatimus on jo 100 000 x 1 000 000 = 100 gigabittiä, joka on ~11,64 gigatavua muistia. Jos matriisissa esitettäisiin sanojen esiintymistiheydet, vaadittu tila tulisi olemaan tätäkin suurempi. Tästä syystä matriisi ei ole paras vaihtoehto suurien tietovarastojen esittämiseen muistissa. Sen vaatimaa tilaa voidaan vähentää tietämyksellä, ettei kaikki avaimista esiinny kaikissa dokumenteissa (engl. *sparse data*). Tuolloin matriisista esitetään vain ne avaimet, jotka esiintyvät dokumentissa. Periaatteellinen ratkaisu käänteistiedoston sisällöstä on esitetty kuvassa 7. Kuvassa esitetään taulukossa 1 annetut termit.



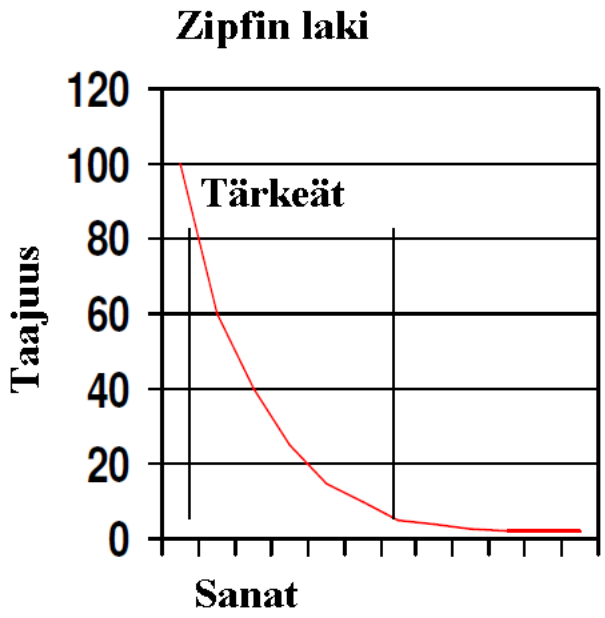
Kuva 7. *Käänteistiedosto mukaillen (Manning et al. 2009, s. 7)*

Kyseistä tietorakennetta kutsutaan termillä *postings list*. Tässä rakenteessa jokaisen termin kohdalle kirjataan ne dokumentit, joihin termi kuuluu (Manning et al. 2009, s. 3). Kyseinen tietorakenne ei vielä ratkaise kaikkia käänteistiedoston kokoon liittyviä ongelmia. Yksi ongelmista on, että tietyt sanat esiintyvät kirjoitetussa kielessä useammin kuin toiset (Manning et al. 2009, ss. 90-91, 27). Englannin kielessä tällaisia merkkijonoja ovat esimerkiksi artikkelit *the*, *an* ja *a*. Suomen kielessä puolestaan yleisimpien termien listaan kuuluu *ja*, *on*, *koska* ja *oli* (Kotimaisten kielten keskus 2012a). Näille sanoille on kuitenkin yhteistä, että tiedonhaun kannalta niiden merkitys on usein vähäinen. Nämä sanat ovat niin yleisiä, että niiden voidaan olettaa löytyvän lähes jokaisesta dokumentista. Tästä syystä ne eivät myöskään tuo lisäarvoa dokumentin löytämiseen. Yleisesti esiintyvien sanojen ongelmallisuus piilee niiden järjestelmälle aiheuttamasta kuormasta, jos käänteistiedostoa ei optimoida.

Erään tärkeän periaatteen optimointiin antaa Zipfin laki (engl. *zipf's law*), joka esittää sanojen jakautumista. Se tunnetaan tilastotieteissä myös potenssijakaumana (engl. *power law*). Mallina se on käyttökelpoinen, kun halutaan kuvata sanan esiintymistä kielessä. Zipfin laki on esitetty kaavassa 1.

$$f \propto \frac{1}{r} \tag{1}$$

Lain mukaan sanan sijoitus (r) on kääntäen verrannollinen sanan esiintymistaajuuteen (f), jollakin vakiolla. Lain mukaan, jos termi (t_1) esiintyy n kertaa termi (t_2) esiintyy puolet (t_1) määrästä. Termien esiintymistiheys laskee tuolloin melko nopeasti sijoituksen kanssa (Manning et al. 2009, s. 89). Samalla löytyy aina termejä, jotka esiintyvät usein, melko usein tai harvoin. Tämä tarkoittaa, että kielessä harvoin esiintyviä termejä on paljon (Luojola 2006, s. 82). Zipfin laista saatavaa tietämystä voidaan käyttää suunniteltaessa kielen mallintamisessa käytettyjä algoritmeja (kuva 8).



Kuva 8. *Zipfin laki mukaillen (Kummamuru 2012. s. 10)*

2.4.3 Boolen hakumalli

Boolen hakumalli on yksi ensimmäisistä ja yksinkertaisimmista menetelmistä tiedonhakuun. Malli perustuu boolen algebran käyttöön. Siinä indeksistä löytyviä sanoja voidaan yhdistää operaattoreilla *ja* (engl. *and*), *tai* (engl. *or*) ja *ei* (engl. *not*). Tätä mallia voidaan hyödyntää hakulauseissa pilkkomalla haettava lause ensin avaimiksi. Tämän jälkeen haussa annetut avaimet voidaan yhdistää toisiinsa ja-ehdolla. Menetelmästä käytetään joissakin yhteyksissä nimitystä täysitäsmäys (engl. *exact match*). Se johtuu menetelmän rajauksen luonteesta, joka edellyttää kaikkien annettujen ehtojen täyttymistä (Manning et al. 2009, ss. 14-16; Salton et al. 1983, ss. 1022-1023). Haku *tekstin tilastollinen luokittelu* voidaan purkaa hakulauseeksi *tekstin AND tilastollinen AND luokittelu*, joka tarkoittaa vaatimusta jokaisen termin esiintymisestä hakuun sopivassa dokumentissa. Boolen mallia on kritisoitu siitä, ettei se tarjoa välimuotoa ehdon toteutumiselle. Mallissa ehdon toteuttamiseen riittää, kun annetut avainsanat esiintyvät, missä tahansa dokumentin sisällä. Joukot eivät pidä sisällään informaatiota, missä kohtaa dokumenttia sana esiintyy. Lisäksi sopivien ehtolauseiden löytäminen hankalaa, koska dokumenteissa käytetty sanasto ei ole välttämättä vakiintunutta. Boolen malli ei ota kantaa myöskään siihen, kuinka hyvin dokumentti vastaa annettua hakulausetta. Tiedonhaun kannalta olisi järkevämpää, jos järjestelmä pystyisi mittaamaan dokumentin sopivuutta. Tämä on erityisen tärkeää isoissa järjestelmissä, joissa hakuun sopivia dokumentteja on yleensä paljon. Sopivuuden mittaaminen perustuu yleisellä tasolla avainten esiintymistiheyteen. Tuolloin mitä useammin termi esiintyy dokumentissa, sitä paremmin se kuvaa dokumenttia. Esiteltyjen haasteiden ratkaisemiseksi boolen hakumallista on myös saatavilla laajennettu versio (engl. *extended boolean model*). Se täydentää alkuperäistä mallia vektorimallien piirteillä (engl. *vector processing models*). Vektorimallin piirteiden lisäksi

mukana tulevat ominaisuudet haettavien termien etäisyyden määrittämiseen toisistaan. (Manning et al. 2009, ss. 2-15; Salton et al. 1983, ss. 1022-1023).

2.4.4 Boolen hakumalli käytännössä

Boolen malli on eräs yksinkertaisemmista tavoista informaation etsintää. Työssä sen esitteleminen on katsottu tarpeelliseksi perusajatusten selvittämiseksi ennen monimutkaisempien hakumallien esittelemistä.

Taulukkoon 2 on koottu esimerkki korpuksen sisältämistä dokumenteista ja sanoista. Olkoon hakulause (Q) esimerkiksi: *metadata AND luokka*.

Taulukko 2. *Dokumenttitermimatriisi dokumenteissa esiintyvistä termeistä*

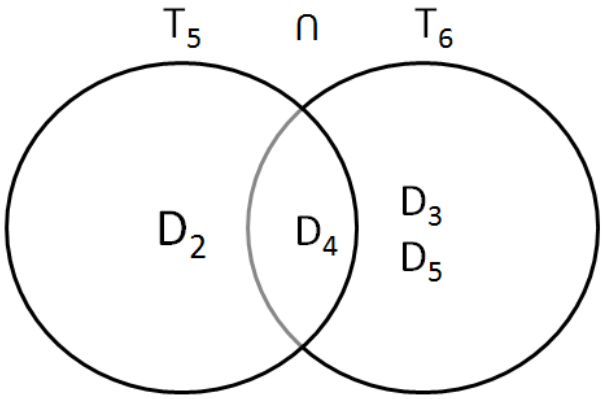
	hei (T_1)	**huomio (T_2)**	**lasku (T_3)**	**viesti (T_4)**	**luokka (T_5)**	**metadata (T_6)**	**muistutus (T_7)**
D_1	1	0	1	0	0	0	1
D_2	1	1	0	0	1	0	0
D_3	0	0	0	1	0	1	0
D_4	0	0	0	1	1	1	0
D_5	0	0	0	0	0	1	1

Dokumenttitermimatriisista muodostetut joukot annetaan taulukossa 3.

Taulukko 3. *Matriisista saadut joukot*

Termit	**Dokumentit, joissa termi esiintyy**
T_1	$\{D_1, D_2\}$
T_2	$\{D_2\}$
T_3	$\{D_1\}$
T_4	$\{D_3, D_4\}$
T_5	$\{D_2, D_4\}$
T_6	$\{D_3, D_4, D_5\}$
T_7	$\{D_1, D_5\}$

Koska kyseessä on ja-tyyppinen ehto, tulosjoukko voidaan kirjoittaa termien T5 ja T6 leikkauksena. Tuolloin dokumenttien tulosjoukko on $T_5 \cap T_6$. Leikkauksella tarkoitetaan ehtoa, että dokumentti kuuluu molempiin joukkoihin T_5 ja T_6. Annetussa mallitapauksessa ainoastaan D_4, kuuluu molempiin joukkoihin T_5 ja T_6. Tilannetta selkeyttää kuvan 9 Venn-diagrammi.



Kuva 9. *Tulosjoukko Venn-diagrammina*

Annetun kuvan perusteella voidaan huomata, että menetelmä on varsin yksinkertainen. Tähän menetelmään liittyvät ideat ovat myös erittäin keskeisessä asemassa. Menetelmässä hyödynnetty Boolen algebra luo pohjan tietokantojen toiminalle. Tästä syystä sitä käytetään myös osana strukturoituja menetelmiä.

2.4.5 Vektorimalli

Vektorimalli (engl. *vector space model, VSM*) on Gerard Saltonin kehittämä menetelmä hakulauseen vertaamiseksi dokumenttiin. Menetelmässä kysymys on osittaisesta täsmäyksestä, koska malli ei edellytä kaikkien haettavien termien löytymistä hakuun sopivista dokumenteista. Se mahdollistaa myös kahden dokumentin vertaamisen toisiinsa. Vektorimalli tuli tutuksi Smart-järjestelmän (engl. *system for the mechanical analysis and retrieval of text*) kehityksen myötä (Singhal 2001, ss. 1-2). Sen käyttö perustuu nimensä mukaan ajatukseen tekstin sanojen eli termien vertaamisesta vektoriavaruudessa. Mallissa hyödynnetään ilmiöitä, jota kutsutaan klusteroitumiseksi (engl. *clustering*). Se tarkoittaa tiettyjen termien esiintymistä tietyn tyyppisissä dokumenteissa. Varsinainen vertaaminen tapahtuu muodostamalla hakulauseelle ja dokumenteille omat vektorit, joita verrataan keskenään. Hakulauseen täsmäävyys dokumenttiin saadaan tuolloin tarkastelemalla hakuvektorin ja dokumenttien välistä kulmaa. Kulma esitetään asteikolla nollasta yhteen, missä nolla tarkoittaa hakuvektorin täydellistä erilaisuutta ja yksi täydellistä vastaavuutta dokumenttiin. Tätä arvoa käytetään myös asetettaessa dokumentteja paremmuusjärjestykseen. Tästä syystä kulman arvo on tärkeä myös hakukoneille. Tämä johtuu siitä, että saatuja hakutuloksia on yleensä paljon ja ne tulee järjestää. Vektorimallissa hyödynnetään kahta keskeistä ajatusta. Ensimmäisen ajatuksen mukaan, mitä useammin sana esiintyy dokumentissa, sitä paremmin se kuvaa dokumenttia. Toinen ajatus liittyy sanan erottavuuden arviointiin. Tuolloin mittaan, kuinka hyvin hakutermin avulla dokumentti voidaan erottaa muista tietovaraston dokumenteista (Manning et al. 2009, ss. 64-128). Suomen kielessä usein esiintyviä sanoja ovat *ja*, *on*, *että*, *tämä* ja *mutta* (Kotimaisten kielten keskus 2012a). Tämän tyyppiset termit eivät tuo hakujen kannalta suurta lisäarvoa, joten niiden merkitystä hakulauseessa voidaan pienentää. Se perustuu

ajatukseen, että kyseiset termit esiintyvät lähes jokaisessa dokumentissa. Tämä tarkoittaa, ettei niiden perusteella voida luoda ryhmiä. Yleisin tapa arvioida dokumenttien ja hakulauseen välistä sopivuutta on käyttää tf-idf-painotuksia (engl. *term frequency - inverse document frequency*). Näiden painotuksien laskemiseksi tulee muodostaa jokaiselle dokumentille ja hakulauseelle omat vektorit. Tämä voidaan kirjoittaa kaavan 2 muodossa.

$$d = [t_{i1} \quad t_{i2} \quad \dots \quad t_{in}] \tag{2}$$

Kaavassa 2 esitetyt alkiot ($t_{1\text{-}n}$) ovat avainten eli termien esiintymisiä tutkittavassa dokumentissa (d) tai hakulauseessa (q). Hakulauseen ja dokumentin välinen arviointi tehdään kaavalla 3. (Manning et al. 2009, s. 119). On hyvä huomata, että varsinainen sopivuuden arviointi tehdään käyttäen kulmaa. Kaava 3 on kehitelmä, miten arviointi voidaan tehdä teoriassa.

$$score(q,d) = \sum_{t \epsilon q} tf - idf_{t,d} \tag{3}$$

Kaavassa 3 $tf - idf_{t,d}$ sisältää kaksi alifunktiota. Niiden arvo voidaan laskea kaavalla 4.

$$tf - idf_{(t,d)} = tf_{(t,d)} \times idf_{(t)} \tag{4}$$

Kaavassa 4 esitetyt alifunktiot ovat tf (engl. *term frequency*) ja idf (engl. *inverse document frequency)*. Näistä (tf) mittaa, kuinka usein termi esiintyy tutkittavassa kohteessa. Funktion tarkoituksena on antaa suurempi painokerroin niille dokumenteille, jossa termi (t) esiintyy useammin. Funktio (idf) mittaa termin (t) esiintymistä globaalisti kaikissa tietovaraston kuuluvissa dokumenteissa. Sen avulla painotetaan sellaisia sanoja, jotka erottavat dokumentin parhaiten tietovarastosta. Molempien alifunktioiden valintaan kirjallisuudessa esitetään useita vaihtoehtoja, joista muutamia vaihtoehtoja on taulukossa 4.

Taulukko 4. *Tf-idf- funktiot (Manning et al. 2009, s.128)*

tf (Term Frequency)		**idf (Document Frequency)**	
n (Normaali)	$tf_{t,d}$	n (Ei käytössä)	1
l (Logaritminen)	$1 + \log_{10}(tf_{t,d})$ *	t (Käänteinen)	$\log_{10}(\frac{n}{df_t})$
a (Suurennettu)	$0.5 + \frac{0.5 * tf_{t,d}}{\max_t(\mathrm{tf_{t,d}})}$	p (Todennäköisyys)	$\max\{0,\log\frac{N-Df_t}{Df_t}\}$
b (Boolen)	$\begin{cases} jos\ tf_{t,d} > 0 = 1 \\ muutoin = 0 \end{cases}$	* Logaritmisen tf:n osalta yleinen tapa on muokata funktiota niin, että painokerroin lasketaan annetulla kaavalla kun $tf_{t,d} > 0$.	
l (Logaritminen keskiarvo)	$\frac{1 + \log_{10}(tf_{t,d})}{1 + \log_{10}(ave_{t \in d} \times (tf_{t,d}))}$		

Taulukkoon 4 liittyviä merkintöjä:
$\mathrm{tf}_{t,d}$ = Termin (t) määrä dokumentissa (d)
df_t = Monessako dokumentissa (d) termi (t) esiintyy
n = Kaikkien dokumenttien määrä korpuksessa

Kaavaa 3 ei suoranaisesti voida käyttää hakulauseen ja dokumentin arviointiin. Asiaa voidaan perustella dokumentin ja hakuvektorien etäisyydellä vektoriavaruudessa. Jos vertaaminen tehtäisiin euklidisen etäisyyden perusteella, hakulauseessa ja dokumentissa esiintyvien termien erotuksella olisi suuri merkitys hakutulokseen. Euklidisella tarkoitetaan kahden pisteen välistä etäisyyttä n-ulotteisessa vektoriavaruudessa. Dokumentti, joka sisältää hakuvektorin sanat kahdesti sijaitsee käytetyn tf-idf-funktion etäisyyden päässä hakulauseesta. Tämä etäisyys saattaa olla pitkä, joten se ei sovellu suoraan samankaltaisuuden arviointiin. Näin tapahtuu erityisesti dokumentin ollessa pitkä verrattuna hakuvektoriin. Tästä syystä arviointi tehdään hyödyntäen dokumentin ja hakulauseen välistä kulmaa ongelman ratkaisemiseksi. Tämä on esitetty kaavassa 5.

$$\cos\theta = \frac{d \cdot q}{|d||q|} \tag{5}$$

Kaavassa kulma lasketaan (d) ja (q) pistetulon avulla ja lopuksi saatu tulos jaetaan (d) ja (q) yksikkövektorien tulolla. Yksikkövektorit hakulauseelle (q) ja dokumentille (d) voidaan määrittää kaavan 6 avulla.

$$|d| = \sqrt{t_1^2 + t_2^2 + \ldots + t_n^2} \tag{6}$$

(Manning et al. 2009, s. 64-128)

Matemaattista teoriaa on tarkoitus perustella tarkemmin osana annettavaa käytännön mallia.

2.4.6 Vektorimalli käytännössä

Vektorimalli on eräs tärkeimmistä menetelmistä hakujen suorittamiseen strukturoimattomasta datasta. Tästä syystä työssä esitetään yksinkertainen esimerkki vektorimallista. Annetun esimerkin tavoitteena on selvittää lukijalle perusideaa, joka liittyy dokumenttien esittämiseen *bag-of-words* mallilla. Tähän esimerkkiin on valittu taulukosta 4 logaritminen tf funktio ja käänteinen idf funktio (taulukko 4). Tavoitteena on esittää, miten tf-idf vektorimalli käyttäytyy seuraavissa tilanteissa:

1. Termi esiintyy dokumentissa useita kertoja
2. Termi esiintyy kaikissa tietovaraston dokumenteissa
3. Vain osa haetuista termeistä esiintyy tutkittavassa dokumentissa

Annettuun korpukseen kuuluu 5 dokumenttia, jotka sisältävät 7 termiä. Nämä ovat annettu taulukossa 5. Olkoon hakulauseessa (q) avaimet: *metadata*, *ja*, ja *luokka*.

Taulukko 5. *Korpuksen dokumentit ja termit*

	ja (t_1)	**huomio (t_2)**	**lasku (t_3)**	**viesti (t_4)**	**luokka (t_5)**	**metadata (t_6)**	**muistutus (t_7)**
d_1	15	0	5	0	0	0	3
d_2	2	1	0	0	6	0	0
d_3	4	0	2	3	0	1	0
d_4	5	0	0	1	5	100	0
d_5	6	0	0	0	0	4	2

Taulukossa 5 esitetyille arvoille voidaan laskea tf-painotukset valitun kaavan mukaisesti. Työssä siihen on käytetty kaavaa $1 + \log_{10}(tf_{t,d})$, kun $tf_{t,d} > 0$ (taulukko 4). Kaikki lasketut tf-painokertoimet annetaan taulukossa 6.

Taulukko 6. *Termien tf-painotukset*

	ja (t_1)	**huomio (t_2)**	**lasku (t_3)**	**viesti (t_4)**	**luokka (t_5)**	**metadata (t_6)**	**muistutus (t_7)**
d_1	2,18	0	1,7	0	0	0	1,48
d_2	1,30	1,0	0	0	1,78	0	0
d_3	1,60	0	1,3	1,48	0	1,0	0
d_4	1,70	0	0	1,0	1,7	3,0	0
d_5	1,78	0	0	0	0	1,60	1,3

Taulukossa 6 sanan *ja* painotus on saatu dokumentille (d_1), esimerkinomaisesti laskemalla $1 + \log_{10}(15) \approx 2.18$. Sanalle *huomio* ($t_2$) painotusta ei puolestaan lasketa, koska (t_2) ei esiinny dokumentissa (d_1). Tämä tarkoittaa, ettei annettu ehto $tf_{t,d} > 0$ täyty.

Taulukon 6 arvoista voidaan huomata, että sanan esiintyessä useamman kerran dokumentissa ($d_{1\text{-}5}$) tf-arvo kasvaa logaritmisella asteikolla. Se tarkoittaa sitä, että sata kertaa useammin esiintyvä termi saa painokertoimen kolme (3). Kun termi esiintyy kerran, painokerroin on yksi (1). Toisin sanoen menetelmä painottaa maltillisesti dokumentteja, joissa sama termi esiintyy useammin.

Idf-painotukset määritellään $\log_{10}(\frac{n}{df_t})$ kaavan mukaisesti jokaiselle korpuksen termille (taulukko 4). Kaavassa esitetty (n) on korpukseen kuuluvien dokumenttien määrä eli viisi (5). Arvo (df_t) tarkoittaa sitä, monestako dokumentista kyseinen termi on löydettävissä. Lasketut termien idf-painotukset on esitetty taulukossa 7.

Taulukko 7. *Termien Idf -painotukset*

ja (t_1)	**huomio** (t_2)	**lasku** (t_3)	**viesti** (t_4)	**luokka** (t_5)	**metadata** (t_6)	**muistutus** (t_7)
0,0	0.70	0,40	0,40	0,40	0,22	0,40

Taulukosta 7 voidaan huomata termin painoarvon laskevan sen mukaan, mitä useammassa dokumentissa se esiintyy. Sellaiset termit, jotka esiintyvät kaikissa dokumenteissa saavat painoarvon nolla. Annetussa esimerkissä näin kävi termille (t_1). Perustelu asialle on $\log_{10}(\frac{5}{5}) = 0$. Toisaalta, mitä harvemmin termi esiintyy korpuksessa, sitä paremmin se yksilöi dokumentin tietovarastosta. Paras painoarvo saadaan sanalle (t_2), joka esiintyi vain yhdessä dokumentissa $\log_{10}(\frac{5}{1}) \approx 0.7$. Tämän jälkeen voidaan laskea varsinaiset tf-idf painotukset sanoille. Tämä tapahtuu kaavan 4 mukaisesti kertomalla taulukon 6 ja 7 arvot keskenään. Saadut tulokset on annettu taulukossa 8.

Taulukko 8. *Dokumenttivektorit*

	ja (t_1)	**huomio** (t_2)	**lasku** (t_3)	**viesti** (t_4)	**luokka** (t_5)	**metadata** (t_6)	**muistutus** (t_7)
d_1	0,00	0,00	0,68	0,00	0,00	0,00	0,59
d_2	0,00	0,70	0,00	0,00	0,71	0,00	0,00
d_3	0,00	0,00	0,52	0,59	0,00	0,22	0,00
d_4	0,00	0,00	0,00	0,40	0,68	0,67	0,00
d_5	0,00	0,00	0,00	0,00	0,00	0,36	0,52

Saatuja arvoja voidaan verrata haku- tai dokumenttivektoriin kaavan 5 avulla. Ennen vertaamista on valittava hakuvektori (q). Annetun esimerkin tapauksessa hakuvektoriksi valittiin kolme erillistä termiä, jotka olivat *metadata*, *ja*, ja *luokka.* Itse hakuvektoria ei ole tapana esittää osana korpusta, koska kyseessä ei ole dokumentti. Yleinen tapa on olettaa hakulauseen suhteen, että kaikki siinä esiintyvät avainsanat saavat tf-funktion arvon yksi (1). Idf-painotusten osalta hakuvektorille voidaan käyttää dokumenteille laskettuja painotuksia.Tämän tietämyksen perusteella voidaan muodostaa seuraavaksi tf-idf-painotettu hakuvektori. Saatu hakuvektori (q) on esitetty taulukossa 9.

Taulukko 9. *Hakuvektori*

	ja (t_1)	**huomio** (t_2)	**lasku** (t_3)	**viesti** (t_4)	**luokka** (t_5)	**metadata** (t_6)	**muistutus** (t_7)
q	0	0	0	0	0,44	0,22	0

Hakuvektorista voidaan huomata, että kaikissa dokumenteissa esiintyvä termi *ja* poistuu hakulauseesta idf-painotusten ansiosta. Taulukon arvo nolla tarkoittaa, ettei kyseistä termiä huomioida lainkaan osana hakutulosta.

Varsinaiset tulokset saadaan vertaamalla hakuvektorin (q) ja dokumenttien ($d_{1\text{-}5}$) välistä kulmaa (kaava 5). Työssä kaavaa on selkeytetty laskemalla ensin vektorien komponenttien normalisoidut arvot. Tämän jälkeen lasketaan vasta varsinainen pistetulo. Normalisointi tehdään jakamalla jokainen vektorien alkioista euklidisella normilla (kaava 6). Hakuvektoriin (q) kuuluvalle termille (t_5) normalisoitu arvo on $\frac{0{,}44}{\sqrt{(0{,}44)^2+(0{,}22)^2}} \approx 0{,}89$. Dokumentti- ja hakuvektorin normalisoidut komponentit esitetään taulukossa 10. Taulukossa esitetyt arvot ovat laskettu Microsoft Excelissä, jolloin ne ovat tarkempia kuin kahden desimaalin perusteella saatavat arvot. Tästä syystä esimerkistä saatu arvo eroaa 0,02 verran taulukossa annetusta.

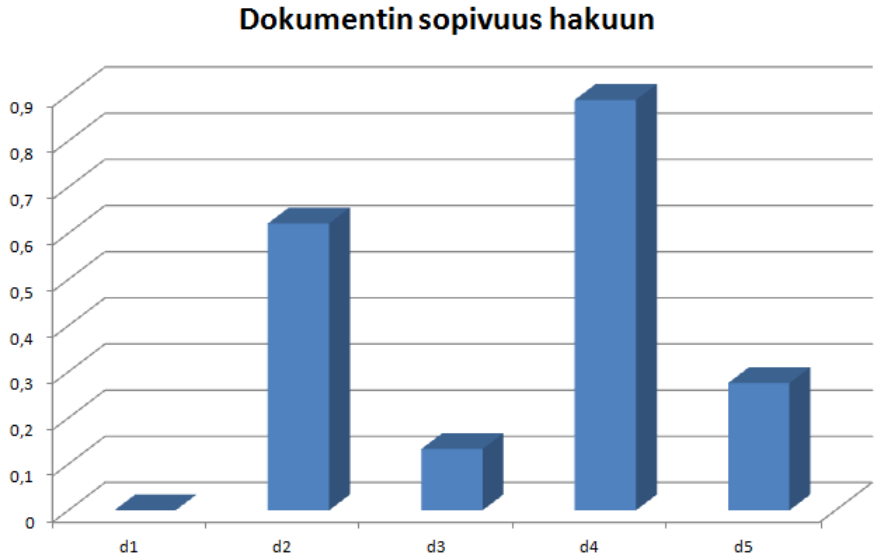
Taulukko 10. *Vektorien normalisoidut komponentit*

	ja (t_1)	**huomio (t_2)**	**lasku (t_3)**	**viesti (t_4)**	**luokka (t_5)**	**metadata (t_6)**	**muistutus (t_7)**
|q|	0,0	0,0	0,0	0,0	0,87	0,49	0,0
$\|d_1\|$	0,0	0,0	0,75	0,0	0,0	0,0	0,66
$\|d_2\|$	0,0	0,70	0,0	0,0	0,71	0,0	0,0
$\|d_3\|$	0,0	0,0	0,64	0,72	0,0	0,27	0,0
$\|d_4\|$	0,0	0,0	0,0	0,39	0,66	0,65	0,0
$\|d_5\|$	0,0	0,0	0,0	0,0	0,0	0,57	0,87

Lopuksi jokaiselle dokumenteista ($d_{1\text{-}5}$) ja (q) määritellään kulma. Hakuvektorin esimerkinomaisesti (q) ja (d_4) välinen kulma on määritetty seuraavasti:

$$(0{,}0 * 0{,}0) + (0{,}0 * 0{,}0) + (0{,}0 * 0{,}0) + (0{,}0 * 0{,}39) + (0{,}66 * 0{,}87) + (0{,}65 * 0{,}49) + (0{,}0 * 0{,}0) \approx 0{,}9$$

Lopulliset tulokset on esitetty kuvassa 10.



Kuva 10. *q:n yhdenmukaisuus dokumenttien ($d_{1\text{-}5}$) kanssa*

Kuvion perusteella voidaan havaita tulosten olevan odotetun mukaiset. Dokumentti (d_4), joka sisälsi termit *metadata*, *ja* ja *luokka* sopii hakuun parhaiten. Kaikissa dokumenteissa esiintynyttä termiä *ja*, ei huomioitu lainkaan tuloksissa. Hakutuloksessa otettiin huomioon myös ne dokumentit, jotka sisälsivät toisen haettavista termeistä. Tuloksesta voidaan huomata, että (d_2) sopii kyselyyn paremmin kuin (d_5). Tämä johtuu siitä, että termi *metadata* on korpuksessa yleisempi kuin *luokka*. Tarkasteltaessa dokumenttien (d_3) ja (d_5) sopivuutta hakuun huomataan, ettei suurempi esiintymistiheys paranna sopivuutta hakutulokseen lineaarisesti. Termi *metadata* esiintyy dokumentissa (d_5) neljä kertaa useammin kuin dokumentissa (d_3). Sama havainto voidaan tehdä dokumentista (d_4), joka sisältää termin metadata sata kertaa.

2.4.7 Hakumenetelmien mittaaminen

Koska tietovarastoissa tiedonhaussa on kysymys monimutkaisesta ilmiöstä, tietovarastojen toimintaa tulisi pystyä mittaamaan. Tietovarastojen tehokkuuden mittaamiseksi esitetään usein kaksi keskeistä menetelmää, jotka ovat saanti (engl. *precision*) ja tarkkuus (engl. *recall*). Saannilla (engl. *precision*) tarkoitetaan dokumenttien osuutta, jotka ovat relevantteja tiedonhaun näkökulmasta (kaava 7).

$$saanti = \frac{|\{relevantit\ dokumentit\} \cap \{löydetyt\ dokumenti\}|}{|\{löydetyt\ dokumentit\}|} \quad (7)$$

Tarkkuudella (engl. *recall*) tarkoitetaan osuutta dokumenteista, jotka ovat relevantteja ja löydettiin haun tuloksena.

$$tarkkuus = \frac{|\{relevantit\ dokumentit\} \cap \{löydetyt\ dokumentit\}|}{|\{relevantit\ dokumentit\}|} \quad (8)$$

Kaavoissa 7 ja 8 esitetyllä merkinnällä |{joukko}| tarkoitetaan joukon sisältämien alkioiden määrää. Relevantilla tarkoitetaan dokumenttien joukkoa, jotka sopivat tekijän mielestä haun tuloksiksi. Löydetyillä dokumenteilla työssä viitataan dokumentteihin, jotka palautettiin haun tuloksena (Huijsmans & Dionysius 2001, s. 2). Näiden menetelmien osalta työssä on tarkoituksena jättää avoimeksi kysymys organisaation tietovarastojen kyvykkyydestä tiedonhakuun.

2.4.8 Yhteenveto rajausmenetelmistä

Tiedonhakumenetelmät perustuvat usein sanojen hyödyntämiseen sellaisenaan. Ne eivät ymmärrä suoranaisesti tekstin sisältöä ja tästä syystä niihin liittyy myös haasteita. Kirjoittajan käsityksen mukaan nämä haasteet johtuvat luonnollisen kielen tarpeesta tulkintaan. Kielessä on kysymys paljon monimutkaisemmasta ilmiöstä kuin yksittäisten sanojen tarkastelusta. Tästä syystä tehokkaiden rajausten muodostaminen on yksi strukturoimattoman tiedonhaun keskeisimmistä haasteista. Tarkastelemalla menetelmiä yleisel-

lä tasolla voidaan huomata, etteivät ne suoranaisesti kykene havaitsemaan sanojen synonyymeja. Tämä herättää kysymyksen, miten sanojen synonyymit tulisi hoitaa. Tulisiko merkkijonon *mukava* täsmätä merkkijonoon *kiva*? Menetelmiä tutkiessa herää myös kysymys, onko merkkijonon sijainnilla myös merkitystä. Työssä esitellyt mallit eivät ottaneet kantaa tähän kysymykseen, niiden kannalta merkkijonon sijainnilla dokumentissa ei ollut merkitystä. Menetelmät perustelevat, miksi strukturoituja tiedonhakumenetelmiä tarvitaan. Yleisesti tiedonhaku strukturoimattomasta datasta sisältää paljon epävarmuustekijöitä. Näiden epävarmuustekijöiden poissulkemiseksi organisaatioissa tarvitaan myös vaihtoehtoista lähestymistapaa tietovarastojen hallintaan. Tämä on toiminut motivaationa esitellä molemmat näistä menetelmistä.

Kirjoittajan käsityksen mukaan näitä menetelmiä tarvitaan vastaamaan kysymyksiin, joihin strukturoiduilla menetelmillä ei voida. Menetelmät mahdollistavat myös sellaisen informaation hakemisen dokumenttien sisältä, jota ei ole kuvattu dokumentin ulkopuolella. Tästä syystä menetelmät ovat myös keskeisessä roolissa hakukoneiden toiminnassa.Niiden esittely on myös työn yhteydessä katsottu tarpeelliseksi. Lisäksi malleissa toimintaperiaatteita voidaan hyödyntää tutkittaessa tilastollista luokittelua.

2.5 Sisältöä kuvailevat menetelmät

Kohdassa 2.5 tutkitaan, miten datan strukturoinnilla voidaan yksinkertaistaa tiedonhakua tietovarastosta. Tarkoituksena on selvittää metadataa, joka kuvaa resurssia. Kohdassa selvitetään, mitä metadatalla tarkoitetaan ja miten se toimii konseptitasolla osana tietovarastojen hallintaa. Työssä pyritään selventämään myös luokittelun ja metadatan välistä suhdetta toisiinsa.

2.5.1 Metadata

Metadatalle Internetissä on saatavilla useita määritelmiä, joista kaikki pyrkivät kuvaamaan metadataa omasta näkökulmastaan. Salmisen mukaan metadatalla tarkoitetaan *tietoa tiedosta* (Salminen 2005, ss. 1-2). Määritelmä (Arkistolaitos 2012) mukaan: *metadata on yleisnimitys tietoverkon tai muiden tietovarantojen sisältämien dokumenttien kuvailutiedoille. Metadata on tietoa tiedosta, tietosisältöä kuvaavaa ja sen merkitystä selittävää informaatiota. Metadatan käytöntarkoituksena on helpottaa aineiston hakua, paikallistamista, tunnistamista ja säilyttämistä sähköisessä muodossa. Metadatalla on tärkeä merkitys myös dokumenttien käsittelyvaiheessa. Kuvailutiedot voidaan tallentaa asiakirjaan jo dokumenttia kirjoitettaessa.* Amerikkalainen standardointijärjestö Niso mainitsee metadataa olevan kuvaavassa, strukturoivassa tai hallinnallisessa tarkoituksessa. Kuvaavaa metadataa (engl. *descriptive metadata*) käytetään yleisesti resurssin tunnistamiseen. Sen tehtävänä on helpottaa informaation paikantamista datamassasta. Kuvaavaa metadataa on dokumentin otsikko, luokittelu ja muut sen sisältöä kuvaavat asiasanat. Strukturoivaa metadataa (engl. *structural metadata*) käytetään esittämään fyysistä tai loogista objektin rakennetta. Sitä ovat myös diplomityössä käytetyt väliotsikot, koska niiden tehtävä on ohjata lukijaa informaation paikantamiseksi. Tietovarasto-

jen yhteydessä metadata säilötään kuitenkin erikseen varsinaisesta dokumentista. Hallinnallinen metadata (engl. *administrative metadata*) on informaatiota, jonka tehtävä on helpottaa resurssin hallintaa. Siihen kuuluvat luontiaika, tiedostomuoto ja käyttöoikeudet. (Niso 2001, ss. 1-3) Se on keskeisessä roolissa ylläpidettäessä myös arkistoa.

Metadatan tavoite on helpottaa näin ollen tietovaraston ylläpitoa ja yksinkertaistaa informaation löytämistä datamassasta. Metadata jakaa paljon yhteisiä piirteitä kirjastoissa käytetyn indeksoinnin kanssa, sillä molemmat menetelmistä pyrkivät kuvailemaan sisältöä ennalta kuvatulla ja systemaattisella tavalla. Tämä tarkoittaa eräänlaista sopimusta kuvailtavasta informaatiosta. Metadatan yhteydessä tästä sopimuksesta käytetään nimitystä metadatamalli (engl. *metadata scheme*). Sopimuksen tehtäviin kuuluu määrittää tarvittavat metaelementit ja niiden semantiikka. Metaelementeillä tarkoitetaan attribuuttia tai attribuutteja, jolla sisältöä kuvataan. Semantiikalla puolestaan tarkoitetaan kuvausta elementtiin tallennettavan informaation muodosta ja sen käyttötarkoituksesta. (Niso. 2001, s. 3) Käytetyssä metadatamallissa tulisi tunnistaa tärkeimmät attribuutit, joiden perusteella sisältöä halutaan etsiä. Metadatamalliin valittujen attribuuttien perusteella aineistosta tulisi pystyä muodostamaan katalogimainen rakenne. Tämä tarkoittaa aineiston jakamista pienempiin loogisiin osakokonaisuuksiin. Käytettyjen attribuuttien arvojen tulee olla systemaattisia, jotta niitä voidaan käyttää informaation paikantamiseksi. Se tarkoittaa asiasanaston (engl. *vocabulary, thesaurus*) määrittämistä. Metadatamallia määriteltäessä aineistosta kirjattavien attribuuttien tarpeellisuudesta tulee varmistua. Lisäksi niitä tulee päivittää tarvittaessa, jotta ne pysyisivät hyödyllisinä tiedonhaun kannalta. (Gillet et al. 2008; Niso 2001, ss. 1-3) Ylimääräisten attribuuttien keräämistä tulisi yleisesti välttää, koska niiden tuottamisella on hintansa kognitiivisen kuormituksen suhteen. Nison (2001, ss. 1-3) mukaan toimiva metadatamalli sisältää seuraavat piirteet:

1. Kerätyn metadatan tulee olla sopivaa aineiston, käyttäjien ja käyttötarkoituksen näkökulmista.
2. Tallennetun metadatan tulee olla ajan tasalla ja tarpeellista sisällön saatavuuden tai käytön kannalta.
3. Käytetyn metadatamallin tulee edistää tietojärjestelmien välistä yhteensopivuutta.
4. Malli määrittää kontrolloidun sanaston, jonka avulla voidaan vastata kysymyksiin mitä, missä, milloin ja kuka.
5. Malli määrittää selkeän esitystavan ja käyttötarkoituksen datalle.
6. Resurssia kuvaava informaatio tallennetaan erikseen varsinaisesta sisällöstä. Samalla malli tukee myös arkistoitavuutta, säilytettävyyttä ja jakoa loogisiin osakokonaisuuksiin.
7. Malli tukee pääsynvalvonnan näkökohtia. Kenellä on oikeus nähdä tai muokata sisältöä?
8. Tietovarastoon informaatiota tallentavat henkilöt koulutetaan käyttämään mallia ja ymmärtämään sen tarpeellisuus.
9. Käyttäjäyhteisölle annetaan mahdollisuus täydentää mallia sopivammaksi käyttötarpeisiin.
10. Kontrolloitua sanastoa ja sen käyttöohjeita päivitetään vastaamaan jatkuvasti uusia tarpeita.

11. Olemassa olevaa termistöä voidaan levittää laajempaan käyttöön, vaikka se olisi alun perin suunniteltu tietyn ryhmän käytettäväksi.

Standardoinnin yleisestä menestyksestä huolimatta, yksittäistä kaikille sopivaa metadatastandardia ei ole saatavilla. Saatavilla on useita keskenään kilpailevia standardeja (Gilliland 2008, s. 5). Niitä on saatavilla niin elementtien (engl. *element sets*) määrittämiseksi kuin tallennusmuodon valintaan (engl. *metadata format*). Yhteistä näille standardeille on yhteensopimattomuus toistensa kanssa. Lisäksi useat standardit ovat suunniteltu tiettyyn rajattuun käyttötarkoitukseen. Eräitä standardeja ovat Dublin Core, Z39.87, JHS-143 ja METS (engl. *metadata encoding and transmission standard*). JHS-143 on suunniteltu esimerkiksi julkishallinnon asiakirjojen tarpeisiin, mutta se on kansallinen standardi. Dublin Core puolestaan on kehitetty alun perin digitaalisten julkaisujen kuvaamiseen. Osa organisaatioista soveltaakin standardeja omista lähtökohdistaan tai on määrittänyt oman sisäisen standardin tallennettavan informaation kuvailemiseksi. METS on suunniteltu digitaalisen aineiston kuvailuun hyödyntäen XML:ää (Niso 2001, ss. 1-3). Organisaation näkökulmasta standardin tulisikin täyttää informaation säilyttämisen ja hyödyntämiseen liittyvät näkökulmat. Näihin tarpeisiin voivat vaikuttaa sellaiset seikat, kuten ollaanko informaatiota siirtämässä vai arkistoimassa pysyvästi. Samat tekijät voivat vaikuttaa myös sopivan tallennustavan valintaan. Yleisimmin kuitenkin dokumenttia koskevaa metadataa tallennetaan tietokantoihin, XML- ja SGML-tiedostoihin (Niso 2001, ss. 1-3). Konseptina metadatan tehtävä on kuvata resurssien sisältöä rajallisella määrällä ominaisuuksia. Mallin toiminta perustuu sopimukseen, mitä informaatiosta kuvataan ja miten. Käytetyn mallin tulisi heijastaa liiketoiminnallisia tarpeita, joihin vaikuttaa esimerkiksi lainsäädäntö. Metadatan etuna verrattuna struktuuroimattomiin menetelmiin on sen yksiselitteisyys. Se tekee menetelmästä myös suositun osana sisällönhallintaa. Menetelmänä se ei kuitenkaan sovellu kontrolloimattomiin ympäristöihin, kuten Internetiin. Lisäksi kaikkea dokumentteihin liittyvää sisältöä ei usein voida kuvata formaalilla tavalla.

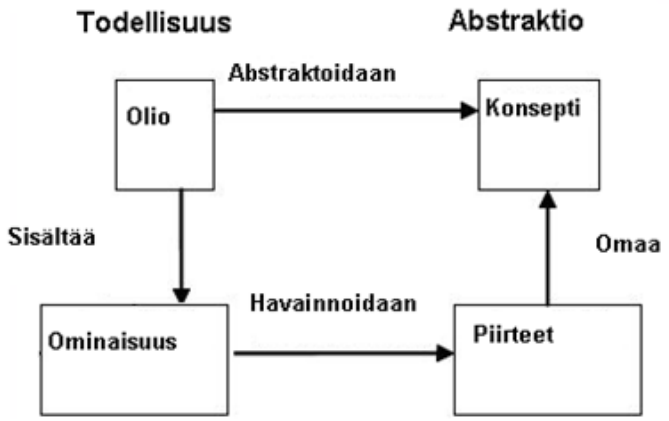
2.5.2 Luokittelu ilmiönä

Luokittelua ilmiönä on tutkittu eri tieteellisissä teksteissä jo yli 2000 vuotta. Tietotekniikan myötä luokitteluun liittyvä tutkimus on noussut jälleen ajankohtaiseksi aiheesi. Luokittelua tutkitaan kirjallisuudessa yhtenä tiedonlouhinnan (engl. *data mining*) osa-alueista. Luokittelua ympäröivä tutkimus ei koske pelkästään tietotekniikkaa, vaan se sivuaa metafysiikkaa, psykologiaa, kognitiivisia- ja kielitieteitä. Asiaa lähestytään monitieteellisestä näkökulmasta, jotta sitä voitaisiin ymmärtää ilmiönä. Tavoitteena on muodostaa käsitys, mitä luokittelu ja luokat ovat, miksi luokittelua tarvitaan ja miten luokittelemme eri asioita.

Diplomityön näkökulmasta luokittelulla (engl. *classification*) tarkoitetaan dataresurssin yhdistämistä luokkaan eli kategoriaan (engl. *class category*). Luokittelu tehdään arvioimalla kohteen ominaisuuksia (engl. *attributes, properties*). Kohde on luokan jäsen, jos se täyttää luokalle määritellyt piirteet. Näistä piirteistä usein voidaan laatia

myös luokkakuvaus (engl. *class description*). Luokalla tarkoitetaan määritettyä käsitettä eli konseptia (engl. *concept*). Luokan kuvataan joskus olevan lauseen substantiivi tai ryhmä yhteenkuuluvia ominaisuuksia. Näin ei tarvitse kuitenkaan olla, vaan myös muunlaisia luokitteluperusteita voidaan valita. Keskeinen ajatus on kuitenkin, että luokittelu perustuu tekemiimme havaintoihin ja luokka on joukko yhteen kuuluvia havaintoja. Tekstin luokittelemiseksi, tietovarastossa tulee tunnistaa käytetyt konseptit. Tästä työvaiheesta käytetään myös nimitystä terminologinen analyysi (engl. *terminological analysis*). Sen aikana rakennetaan yhtenäinen malli siitä, mitä luokkia on käytettävissä. Prosessin tuloksena syntyy luokkahierarkia (engl. *class hierarchy*), joka toimii perusteena datan järjestämiseksi (engl. *organize*) tietovarastossa (engl. *storage*). Luokkahierarkian tehtävänä on parantaa informaation saatavuutta ryhmittelemällä data pienempiin ja hallittavampiin osakokonaisuuksiin. Tällä tavoin saadaan alennettua informaation paikantamisesta syntyvää kuormitusta. (Zhu et al. 2009, s. 5; ISO 704 2009, ss. 4-7; Blumberg & Atre 2003, ss. 1-2) Kirjastossa esimerkiksi teokset ovat järjesteltyinä aihealueittaan niin, että lainaajan olisi helpompi löytää sopiva teos. Samaa ajatusta käytetään myös tietotekniikassa suurten tietovarastojen järjestelemiseksi. Tekstin luokittelu voidaan nähdä metadatana tekstille. Luokittelu tallennetaan tyypillisesti yksittäiseen attribuuttiin, josta se on käytettävissä ryhmittelyyn. Työssä käytettyä käsitettä luokka ei tulisi sekoittaa olio-ohjelmoinnissa (engl. *object oriented programming*) yleisesti käytettyyn käsitteeseen luokka. Olio-ohjelmoinnissa käytetyn luokan jälkeläinen (engl. *instance, object*) perii aina luokan määrittelemän muodon ja ominaisuuden. Tekstin luokkaa ja sen sisältämiä ominaisuuksia on usein vaikea määritellä tarkasti. Suhde ominaisuuksien periytymiseen ei ole yhtä selkeä, mitä ohjelmoinnissa. Tekstin luokka on subjektiivinen käsite ja riippuu tietomallista ja tulkinnasta (Zhu et al. 2009, s. 5; Oracle; ISO 704:2009).

Ontologia (engl. *ontology*) on filosofian haara, joka tutkii olevaisuuden perimmäistä olemusta. Se pyrkii selittämään mitä tyyppiä olevia asioita on olemassa ja minkälaisia suhteita niiden välillä on. Ontologia sanaa käytetään usein myös synonyymina metafysiikalle (engl. *metaphysics*). (Smith 2012). Historiassa luokittelua pohtinut Aristoteles esittää käsitteitä olevan luonteeltaan kahdenlaisia. On olemassa todellisia olioita (engl. *individual*). Lisäksi on olemassa yleisiä käsitteitä, joita kutsutaan konsepteiksi. Konseptit kuvaavat todellisia olioita joukkona abstrakteja piirteitä (kuva 11) (Edghill 1994; Cohen & Lefebvre 2005, ss. 848-849; Smith 2012). Tämä malli vastaa läheisesti myös ISO 704 käsitystä luokista ja konsepteista. Mallissa todellisista käsitteistä luodaan abstrakteja konsepteja. Oliot voidaan sovittaa konsepteihin havaittujen ominaisuuksien avulla. Konsepti sisältää yleisesti joukon piirteitä, joiden avulla se voidaan tunnistaa muista konsepteista. Olion ei tarvitse täyttää kuitenkaan kaikkia konseptin piirteistä. (ISO 704:2009)

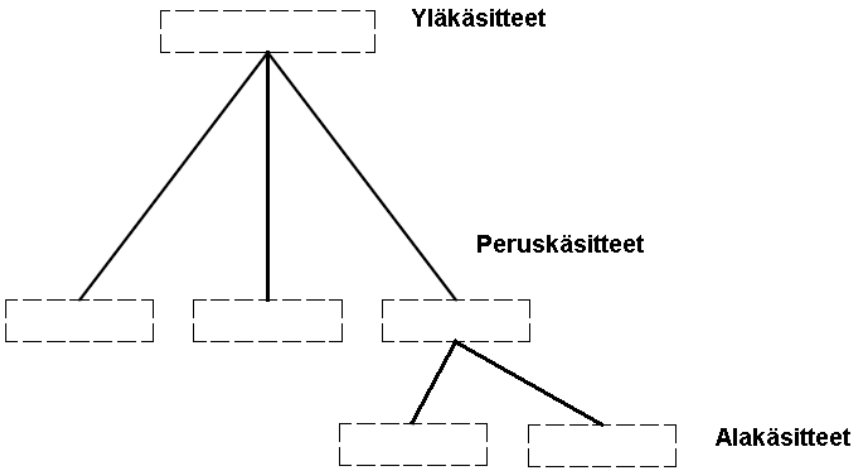


Kuva 11. *ISO 704:2009 luokat (ISO 704:2009)*

Catetegories in Cognitive Science teoksessa esitetään kognition olevan luokittelua. Väitettä perustellaan sillä, että organismit eli eliöt ovat sensomotorisia järjestelmiä (engl. *sensorimotor systems*). Se tarkoittaa, että käsitämme ympäröivää maailmaa aistiemme kautta ja kykenemme reagoimaan sen tuottamiin ärsykkeisiin motorisin liikkein. Luokkia voidaan kuvata ”juttuina”, jotka saavat meidät toimimaan toisin, kuten: mitä ruokaa syömme ja mitä emme, kenen kanssa viihdymme ja ketä välttelemme. Ne ovat myös asioita, joita kuvailemme totuuksina, alkulukuina, kohteen ominaisuuksina ja eroavaisuuksina. Toisin sanoen luokittelu on järjestelmä, jonka kautta hahmotamme ympäröivää todellisuutta. Tarkasteltaessa luonnossa esiintyviä ilmiöitä ne ovat yleisesti kaikki luonteeltaan dynaamisia eli muuttuvia. Tuulensuunta esimerkiksi vaihtelee riippuen fysikaalisista ominaisuuksista, jotka voidaan mitata. Yleisesti tällaisen järjestelmän tila on toistettava, jos kaikkien määreiden arvot ovat tismalleen samat. Samanlainen toistettavuus ei päde luokitteluun, koska se on erilainen järjestelmä verrattuna luonnon lainalaisuuksiin. Luokiteltaessa samoja olioita useampaan kertaan, tulos ei ole välttämättä aina sama. Se johtuu järjestelmän kyvystä sopeutua, joka periytyy kyvystä oppia uusia asioita. Näin oma käsityksemme luokasta on adaptiivinen ja riippuvainen järjestelmälle annetusta palautteesta (Cohen & Lefebvre 2005, ss. 22-23).

Principles of Categorization esitetään luokitteluun kuuluvan kaksi keskeistä periaatetta. Ensimmäinen periaate liittyy kategorisointijärjestelmän toimintaan. Kategorisointi mahdollistaa maksimaalisen määrän informaatiota pienimmällä kognitiivisella vaivalla. Toinen periaate liittyy siihen, kuinka ympäröivä maailma aistitaan. Aistittu maailma koostuu joukosta yhteen kuuluvia ominaisuuksia. Toisin sanottuna se ei ole joukko mielivaltaisia ja irrallisia havaintoja. Tämä tunnetaan paremmin nimellä prototyyppiteoria (engl. *prototype theory*). Ensimmäisen periaatteen mukaan, kategorisoinnin tulee tarjota mahdollisimman paljon ennalta arvattavaa informaatiota kuormituksen minimoimiseksi. Omien etujemme mukaista ei myöskään ole havaita kaikkia pieniä kohteen eroavaisuuksia. Se johtaisi äärettömän hienojakoiseen luokittelumalliin, joka olisi tehoton kuormittavuutensa takia. Luokkajaon tulee olla riittävän tarkka ollakseen tehokas, mutta toisaalta se ei voi olla myöskään liian hienojakoinen resurssien rajallisuuden

takia. Toisen periaatteen mukaan, ympäröivä todellisuus ei ole joukko strukturoimattomia toisistaan riippumattomia ominaisuuksia. Se tarkoittaa, että havaittujen ominaisuuksien välillä on jokin korrelaatio. Puhuttaessa esimerkiksi siivistä, voimme olettaa niissä olevan höyhenet, eikä turkkia. Luokittelu vähentää näin ollen kognitiivista kuormitusta tarjoten oletuksia sen kohteesta. Molempia periaatteita noudattaen luokittelujärjestelmät ovat nähtävissä niin, että niihin kuuluvat vaaka- ja pystysuorat ulottuvuudet (kuva 12). (Rosch & Lloyd 1978) Tämä vastaa myös ISO 704 standardin käsitystä luokittelujen tasoista (ISO 704:2009).



Kuva 12. *Käsitteistöt voidaan esittää puurakenteina (ISO 704:2009)*

Pystysuora ulottuvuus ilmaisee termin abstraktivisuutta. Alakäsitettä kuvaa koira, peruskäsitettä nisäkäs ja yläkäsitettä elävä olento (kuva 12). Vaakasuora ulottuvuus ilmaisee termejä samalla abstraktiotasolla. Vaakasuoralla ulottuvuudella on saman tason käsitteitä esimerkiksi kissa ja koira. Yläkäsitteet omaavat vähemmän täytettäviä ominaisuuksia kuin peruskäsitteet. Ne tarjoavat myös vähemmän informaatiota luokitellusta kohteesta. Alakäsitteet tarjoavat eniten informaatiota kohteestaan, mutta niiden käyttäminen lisää kuormitusta. Peruskäsitteet näyttävät olevan luokkia, jotka tarjoavat parhaan hyödyn ympäristön kuvaamiseksi. Peruskäsitteet ovat erityisen hyödyllisiä tehtäessä luokittelua, koska ne:

1. Sisältävät maksimaalisen määrän yhtenäisiä piirteitä alakäsitteiden kanssa, mikä vähentää vaivaa luokiteltaessa
2. Sisältävät minimaallisen määrän yhteisiä piirteitä rinnakkaisten käsitteiden kanssa, joka helpottaa käsitteen erottamista toisistaan

Luokittelujärjestelmämme ei ole yhteneväinen, sillä aistiemme rajat asettavat rajoituksia luokittelulle ja ominaisuuksien tunnistamiselle. Kokonaisen aistin puuttumisella tai sen voimakkuudella voi olla huomattavaa merkitystä luokittelujärjestelmän kannalta. Voidaan kysyä, mitä synnynnäisesti sokealle tarkoittaa musta tai vihreä? Lisäksi tekijät kuten aika, kulttuuri ja kielitausta vaikuttavat luokitteluun (Rosch & Lloyd 1978). Kirjoittajan näkemys on, että tekstin luokittelussa on kysymys edellä esitellystä ilmiöstä. Tekstistä havaittavat sanat ovat kohteesta tehtyjä havaintoja eli sen piirteitä. Nämä havainnot

pyritään tiivistämään konseptiksi, jolloin kohteen kognitiivinen kuormitus myös alenee. Aivojemme kannalta se tarkoittaa, ettei koko tekstiä tarvitse tulkita. Voimme tehdä näin ollen joukon oletuksia annetun luokittelun avulla. Tämä auttaa myös meitä lajittelemaan tietovarastoja tuttujen konseptien mukaisesti. Luokittelu on tästä syystä eräs tärkeimmistä keinoista tehostaa tietovarastojen toimintaa. Sen tehtävänä on tarjota oletuksia joiden perusteella tietovarastoa voidaan järjestää.

2.5.3 Yhteenveto sisältöä kuvailevista menetelmistä

Kirjoittajan näkemyksen mukaan metadatan tehtävänä on kuvata resurssin sisältöä ennalta määrätyllä tavalla. Ennalta määrätyllä tavalla tarkoitetaan metadatamallin asettamia rajoitteita. Mallin avulla tallennetut resurssit voidaan jakaa yksinkertaisemmin ryhmiin eli joukkoihin. Tähän operaatioon voidaan käyttää tietokantaa, joka hyödyntää Boolen algebraa joukkojen muodostamiseksi. Metadata on jossakin määrin sisältöä luokittelevaa informaatiota, mutta se ei vastaa käsitettä luokka tai luokittelu. Asiaa voidaan perustella sillä, että tallennuspäivää voidaan käyttää esimerkiksi jakamaan aineistoa helpommin hallittaviin kokonaisuuksiin. Tallennettu päivämäärä toimii kuitenkin yksittäisenä havaintona sisällöstä. Näin ollen metadataa voidaan verrata joukkoon sisällöstä havaittavia ominaisuuksia. Metadatamalli rajoittaa näiden havaittavien ominaisuuksien määrää ja muotoa. Kognitiivisessa mielessä metadatan tehtävänä on tarjota maksimaalisen määrä informaatiota pienimmällä mahdollisella kuormituksella. Se tarkoittaa, että käytettävissä on rajallinen määrä havaintoja aineistosta.

Luokka toimii yhdistävänä käsitteenä joukolle todellisia olioita. Luokan jäsenet omaavat toisiaan muistuttavia ominaisuuksia, jotka perustuvat kohteista tehtyihin havaintoihin tai oletuksiin. Luokan jäseniksi valitut konseptit ovat usein myös adaptiivisia ja epätarkkoja. Se tarkoittaa, että niiden luokittelu voi muuttua järjestelmälle annetun palautteen myötä. Epätarkkuudella tarkoitetaan, että todellisen olion ei välttämättä tarvitse täyttää kaikkia luokan tunnusmerkkejä. Todelliset luokiteltavat kohteet voivat kuulua yhteen tai useampaan konseptiin. Luokittelujärjestelmä sisältää näin ollen useita eri abstraktiotasoja. Tästä esimerkkinä toimii *tuoli*, joka kuuluu myös korkeamman tason abstraktioon objekti. Tekstin piirteet ovat myös samantyyppisiä, sillä usein ilman tulkintaa luokan päättely on hankalaa. Tulkinta muodostuu dynaamisesti saamiemme kokemusten eli palautteen perusteella. Tästä syystä kohteen kuuluminen luokkaan on todellisuudessa aina subjektiivista ja riippuu oppimastamme. Lisäksi se riippuu käytettävissä olevista havainnoista.

Ominaisuuden ja luokan keskeinen ero on siinä, että luokka tarjoaa oletuksia kohteesta. Nämä oletukset perustuvat aikaisemmin oppimiimme malleihin luokan sisällöstä. Metadata puolestaan on joukko erilaisia havaintoja sisällöstä. Näihin havaintoihin voi kuulua sisällölle kirjattu luokka. Hyödyntäen näitä havaintoja voimme jakaa aineiston pienemmiksi osakokonaisuuksiksi. Koska metadata on rajoittunut tapa ilmaista asioita, osa informaatiosta jää osaksi tekstiä. Tästä syystä strukturoimattomasta datasta tuttuja tiedonhakumenetelmiä tarvitaan osana kokonaisratkaisua.

3 TEKSTIN TILASTOLLINEN LUOKITTELU

Luvussa 3 selvitetään tekstin tilastollista luokittelua ja siinä tarvittavia työvaiheita. Tarkastelussa esitellään tekstin prosessointia luokittelua varten sekä muutama keskeinen luokittelualgoritmi. Luokittelumenetelmistä tarkoitus on selvittää päätöspuita (engl. *decision trees*) ja Bayesian päättelyä (engl. *bayesian classifiers*). Tekstin esikäsittelymenetelmille työssä on valittu Zhu. et al. esittämät työvaiheet (Zhu et al. 2009, s. 123):

- Kielen ja merkistöstandardin tunnistaminen (engl. *language identification*)
- Tekstin jakaminen avaimiin (engl. *tokenization*)
- Puhdistus sellaisesta informaatiosta, joka ei ole relevanttia luokittelun näkökulmasta (eng. *filtering*)
- Kirjoitus- ja muiden virheiden korjaus (engl. *spelling correction*)
- Morfologinen analyysi sanojen palauttamiseksi perusmuotoon (engl. *morphological analysis*)
- Muu kaupallinen kielen rakenteellinen analysointi (engl. *proprietary linguistic analysis*)
- Avainsanojen ja konseptien purkaminen (engl. *keywords and concept extraction*)

Luvun sisältö työssä on jaettu neljään eri osaan, jotka ovat käsitteet, NLP-menetelmät, tilastollinen luokittelu ja avainsanojen poiminta.

3.1 Käsitteet

Kohdan tarkoituksena on johdattaa lukija kappaleessa käytettyihin termeihin yleisellä tasolla.

Koneellista luokittelua on tutkittu yleisesti tietokantoja, tiedonlouhintaa ja tiedonhakua koskevassa kirjallisuudessa. Tilastollinen luokitteluongelma voidaan määrittää seuraavasti: järjestelmällä on annettu opetusmateriaalia $D = [x_1 \quad ... \quad x_n]$ (engl. *training set*), joista kaikille kohteille (x_n) on annettu luokka (c). Kohteille valittu luokka (c) kuuluu tunnettujen luokkien joukkoon $C = [1 \quad ... \quad c]$. Nämä luokat ovat ennalta järjestelmään määriteltyjä ja ne toimivat rajoitteena käytettävissä oleville vaihtoehdoille. Koulutusmateriaalin perusteella luokittelujärjestelmä muodostaa luokittelumallin (engl. *classification model*). Työssä koulutusmateriaalilla tarkoitetaan järjestelmälle annettuja esimerkkejä luokkaan kuuluvista dokumenteista. Näistä rakennetun mallin perusteella järjestelmä antaa luokittelun testattavalle kohteelle. Kysymys on näin ollen valvotusta oppimisesta (engl. *supervised learning*), koska luokittelumalli luodaan ihmisen annettujen esimerkkien perusteella.

Luokittelumenetelmistä esitetään kaksi eri versiota. Ensimmäisessä versiossa luokiteltava kohde saa aina yhden luokan (engl. *hard version*). Päätöspuut kuuluvat tämän tyyppisiin luokittelijoihin, koska ne eivät esitä todennäköisyyksiä kohteen kuulumiseksi muihin luokkiin. Toinen versio on määrittää todennäköisyys kohteen kuulumiseksi kuhunkin luokkaan (engl. *soft version*). Bayesian päättelyssä toimitaan edellä kuvatulla tavalla. Lisäksi on olemassa myös luokitteluongelma, jossa kohteelle voidaan antaa useampi kuin yksi luokka (engl. *multi-label classification*) (Aggarwal & Zhai 2012, ss. 163-164). Moniluokitusta tai siihen liittyviä algoritmeja, ei kuitenkaan ole tarkoitus käsitellä työn puitteissa.

Tekstin tilastollisessa luokittelussa on kysymys tekstissä esiintyvien avainten vertaamisesta malliin (engl. *statistical pattern matching*). Tilastollisen mallin rakentamiseksi tarvitaan usein eri NLP-menetelmiä (engl. *natural language processing*). (Zhu et al. 2009, s. 25) NLP-menetelmät ovat teknologioita, joita käytetään kielen kuvaamiseksi. Terminä NLP on laaja ja sillä voidaan tarkoittaa lähes kaikkia sellaisia menetelmiä, joiden tavoitteena on matkia ihmisen kykyä tulkita ja ymmärtää tekstiä (Liddy 2001). Tekstin tilastollisen luokittelun kannalta NLP-menetelmien tehtävä on poistaa esteitä, jotka liittyvät merkkijonojen vertailuun keskenään. Itse luokittelu suoritetaan kuitenkin käyttäen aina erillistä luokittelualgoritmia, joka hyödyntää esikäsiteltyjä merkkijonoja.

Avainsanalla työn yhteydessä tarkoitetaan tekstisisällöstä poimittavaa merkkijonoa. Se tarkoittaa mitä tahansa tekstistä paikannettavaa informaatiota, kuten henkilötunnusta tai päivämäärää. Avainsanojen poiminnassa käytetyt menetelmät perustuvat joko sijainnin tai etsittävän kohteen muodon hyödyntämiseen.

3.2 NLP-menetelmät

Kohdassa esitellään työvaiheita, jotka mahdollistavat tekstissä olevien ominaisuuksien vertaamisen keskenään. Osana esitellyistä menetelmistä tarvitaan luokittelualgoritmeja esimerkiksi Bayesian päättelyä. Varsinaiset luokittelualgoritmit työssä esitellään kuitenkin tarkoituksellisesti omana kohtanaan ja niihin lähinnä viitataan osana käsiteltäviä menetelmiä.

3.2.1 Esikäsittely ja kielentunnistus

Ennen kielellistä prosessointia tiedostosta tulee purkaa sen sisältö eli teksti. Tiedoston sisällön purkaminen edellyttää muutamia työvaiheita, joihin kuuluu tiedostomuodon ja merkistökoodauksen tunnistus. Näiden työvaiheiden tarkoitus on esittää teksti sellaisessa muodossa, jossa sitä voidaan prosessoida. Työ käsittelee aihetta lyhyesti kielentunnistuksen yhteydessä, koska esikäsittely voi tarjota myös tärkeää informaatiota kielentunnistuksen kannalta. Tiedoston sisältämää merkistökoodausta voidaan hyödyntää tunnistettaessa kirjoituskieltä (Zhu et al. 2009, ss. 25, 123-133). Merkistökoodauksen tunnistaminen ei ole itsessään riittävä toimenpide, koska osa kielistä jakaa saman merkistökoodauksen. Suomen kielessä käytetään yleisesti samaa merkistökoodausta kuin ruotsin

kielessä. Tästä syystä tarvitaan myös kehittyneempiä tekniikoita varsinaisen sisällön kielen tunnistamiseksi. Ennen näiden tekniikoiden soveltamista tulee ottaa huomioon, vielä käytetty tiedostomuoto ja siinä esiintyvät muotoilut. Tämä johtuu siitä, että useat markkinoilla olevista tiedostomuodoista sisältävät tekstin seassa muotoiluja. Muotoiluissa voidaan käyttää usein vierasperäisiä sanoja, jotka voivat hankaloittaa kielentunnistusta ja myös luokittelua. Tästä syystä tiedostomuotoon liittyvistä muotoiluista halutaan päästä yleisesti eroon. (Manning et al. 2009, s. 20) HTML- ja XML-dokumentit esimerkiksi sisältävät elementtien tai attribuuttien nimissä suomen kielelle vieraita sanoja. Kielitoimiston sivut esimerkiksi näyttävät ilman muotoiluelementtien poistoa kuvan 13 mukaisilta.

```
                                        <div class="centerblock float">
                                                                <div class="content"
id="C3O1v27541"><h1>Kielitoimisto</h1>Kielitoimisto eli
virallisesti Kotimaisten kielten keskuksen kielenhuolto-osasto vastaa <a title="Suomen kielen huolto"
id="pwd_internal_link_21:fi:www_ea13c760c1575693aeb51da711482413" target="" href="index.phtml?s=21">suomen kielen
huollosta</a>.
```

Kuva 13. *Html-raakadataa (Kotimaisten kielten keskus 2012b)*

Varsinaisesti muotoilujen poistaminen on yksinkertainen toimenpide, eikä sitä työn yhteydessä ole tarkoitus käsitellä tämän tarkemmin. Se on kuitenkin tarvittava työvaihe, jotta tekstiä voidaan ylipäätään käsitellä.

Kielentunnistukseen raakatekstistä on saatavilla useita menetelmiä, joihin kuuluvat yleisimpien sanojen laskeminen (engl. *frequent words counting*), uniikit yhdistelmät (engl. *unique tokens*) ja n-grammit (engl. *n-grams tai k-grams*).

Yleisimpien sanojen laskeminen (engl. *frequent words counting*) perustuu sanakirjan käyttöön. Menetelmässä käytettyyn sanakirjaan valitaan kielessä yleisimmin esiintyviä sanoja, joita etsitään arvioitavasta tekstistä. Kyseiseen lähestymistapaan liittyy kuitenkin hankaluus ylläpitää sanastoa. Sukulaiskielten erot myös tietyissä tapauksissa voivat olla riittävän vähäisiä, mikä estää menetelmän hyödyntämisen. Menetelmää kohtaan on esitettykin kritiikkiä tästä syystä. (Ölvecký 2005).

Uniikit yhdistelmät (engl. *unique tokens*) -menetelmän käyttö perustuu ideaan, että useat kielet käyttävät niille ominaisia kirjasimia tai kirjainyhdistelmiä (Ölvecký 2005). Suomen kielen osalta näitä ovat kirjaimet kuten ä ja ö ja yhdistelmät kuten *lla* ja *stä*. Asia voidaan havaita vertaamalla suomen kieltä englantiin: kuinka monta ”stä” -päätteistä sanaa englannin kielestä voidaan löytää? Menetelmän haaste on, että tekstit voivat omata myös vierasperäisiä sanoja. Lisäksi sukulaiskielet jakavat joukon samoja äänteitä keskenään (Ölvecký 2005).

N-grammien toiminta perustuu tekstissä esiintyvien sanojen pilkkomiseen n-pituisiksi merkkijonoiksi. Näitä merkkijonoja voidaan verrata menetelmän avulla kielen tilastolliseen profiiliin. Menetelmä hyödyntää ajatusta, jossa pilkotut merkkijonot edustavat sanojen äänteitä. Sen hyvänä puolena on myös, ettei se edellytä sanakirjan ylläpitoa. Sanasta *tekstiä* voidaan muodostaa 2-6 pituiset n-grammit, jotka ovat *{te,tek,teks,tekst,teksti}*. Operaatio toistetaan kaikille tekstissä esiintyville merkkijonoil-

le. Tämän perusteella muodostetaan tilastollinen profiili. (Ölvecký 2005; Vatanen et al. 2010) Mallin hyödyntäminen perustuu faktaan, että tietyt sanat ja äänteet esiintyvät kielessä useammin kuin toiset (Luojola 2006, s. 82). Suomen kielessä yleisimmät sanat ovat esimerkiksi *ja* ja *on* (Kotimaisten kielten keskus 2012a). Vastaavasti englannin kielessä kaksi yleisintä sanaa ovat *the* ja *and*. Tutkimalla annetuista sanoista saatavia bigrammeja *th ja nd*, on helppoa havaita niiden olevan harvinaisia suomen kielessä. Nämä bi-grammit löytyvät seuraavista englannin kielen sanoista *there*, *with* ja *they*. Kyseiset sanat kuuluvat myös 1000 käytetyimmän sanan joukkoon (the Lexiteria 2010). Tämä tarkoittaa myös, että kyseisten äänteiden osuus englannin kielellä kirjoitetussa on tekstissä varsin suuri. Tutkimusten mukaan n-grammeihin perustuvan tekniikan teho näyttäisi paranevan tekstin pituuden kasvaessa. Lyhyiden tekstien osalta menetelmä ei takaa aivan yhtä hyvää tarkkuutta. Tekstin kielen arviointi voidaan tehdä Bayesian päättelyllä tai vastaavalla luokittelumenetelmällä (Vatanen et al. 2010). Kysymys on luokitteluongelmasta, jossa tekstille valitaan tunnistettu kieli tunnistettavien kielien joukosta.

3.2.2 Tekstin jakaminen avaimiin

Tilastollisen mallin muodostamiseksi teksti tulee jakaa avaimiksi. Tästä työvaiheesta käytetään työssä nimitystä tekstin jakaminen avaimiin (engl. *tokenization*). Se tarkoittaa tekstissä esitettyjen sanarajojen paikantamista. Tämä tapahtuu hyödyntäen oletusta, että länsimaisissa kirjoituksissa sanoja erottaa toisistaan välimerkki, pilkku tai piste. Kyseinen oletus aiheuttaa myös haasteita tietyissä tapauksissa. Eräs näistä haasteista on moniosaiset konseptit, joita esiintyy tekstissä. Konseptilla tarkoitetaan yksiselitteistä käsitettä, joka voi koostua yhdestä tai useammasta avaimesta. Tästä hyvänä esimerkkinä toimii kaupungin nimi *Hong Kong*, joka sisältää avaimet *Hong* ja *Kong*. Moniosaiset konseptit ovat ongelmallisia, koska luokittelu- ja hakujärjestelmät eivät yleisesti huomioi sanojen etäisyyttä toisistaan tai sijaintia. Tuolloin avain *Hong* voidaan yhdistää myös konseptiin *Michael Hong*, joka on vain työtä varten keksitty nimi. Tästä syystä tietotekniillisten järjestelmien tulee pystyä tunnistamaan myös konseptit, jotka sisältävät useamman kuin yhden avaimen. Ongelma ratkaistaan tekstin käsittelyn myöhemmissä vaiheissa, koska sen ratkaiseminen ei ole mahdollista kiinteillä käsittelysäännöillä. (Manning et al. 2009, ss. 20-32) Työssä sanojen rajoja käsitellään tarkemmin osana konseptien mallintamista, mutta ongelman huomaaminen on tärkeää. On löydettävissä myös muita konsepteja, jotka tuottavat virheellisiä avaimia kyseisellä oletuksella: U.S.A tai 4,5 euroa.

Pilkkomisen tuloksena saadut avaimet sisältävät myös haasteita, joihin kuuluvat isot alkukirjaimet, taivutukset, synonyymit, homonyymit, lyhenteet ja kirjoitusvirheet. Nämä haasteet johtuvat siitä, ettei tietokone kykene suoraan samanlaiseen päättelyyn kuin ihminen. Tietokoneen näkökulmasta merkkijono joko vastaa toista merkkijonoa tai sitten ei. Avainten kirjoitusasuun liittyvien haasteiden ratkaisemiseksi yleisempiä tekniikoita ovat päätteiden poistaminen (engl. *stemming*), sanavartaloiden palautus (engl. *lemmatization*) ja yhtenäisen kirjoitusasun valinta (engl. *normalization*). Edellä mainit-

tujen menetelmien tavoitteena on muokata avaimet sellaiseen muotoon, jossa niitä voidaan vertailla keskenään.

Stemming-menetelmän idea perustuu sanan vartaloon liitetyn taivutuksen poistamiseen eli sanan esitykseen yksikössä. Menetelmänä se on alkeellinen, koska sen tehtävänä on vain poistaa avaimiin liitetyt loppupäätteet. (Manning et al. 2009, ss. 22-33) Avaimesta *kissat* voidaan johtaa avain *kissa* poistamalla t-kirjain. Erityisesti suomenkielisten sanojen muuttaminen vertailtavaan muotoon on monimutkainen operaatio. Tästä syystä t-kirjaimen poistaminen ei ole riittävä toimenpide kaikissa tapauksissa vrt. kaupunki ja kaupungit. Menetelmä sopiikin paremmin englannin kieleen, jossa sanojen taivutukset eivät yleensä muuta sanavartaloa.

Sanavartalon palautus (engl. *lemmatization*) on morfologinen menetelmä, jossa poistetaan sanojen runkoihin (engl. *lemma*) liitetyt osat eli affiksit. Morfologialla viitataan sanojen muoto-oppiin (Savolainen 2001). Sen avulla otetaan huomioon sanoihin ja niiden muotoon liittyviä erityisiä piirteitä. Morfologisen käsittelyn tarpeellisuudesta hyvänä esimerkkinä toimii englannin kielestä tuttu verbi nähdä (engl. *see*). Se taipuu eri aikamuodoissa *see*, *saw*, *seen.* Kyseisen verbin eri aikamuotojen vertailemiseksi keskenään, eri kirjoitusmuodoille tulee sopia yhteinen esitys. Se voitaisiin ajatella saavutettavan korvaamalla yksinkertaisesti muodot *saw* ja *seen* muodolla *see*. Näin ei kuitenkaan voida menetellä, koska avain *saw* voi viitata myös substantiiviin saha (engl. *a saw*). Tämä tarkoittaa, että sanan konteksti tulisi huomioida. (Manning et al. 2009, ss. 22-33) . Tämä ongelma voidaan yleensä ratkaista sanojen luokittelun avulla.

Normalisoinnin tehtävänä on yhdistää sanojen vaihtoehtoiset kirjoitusasut eli synonyymit toisiinsa. Suomen kielessä pizza voidaan kirjoittaa myös pitsa. Nämä kaksi sanaa voidaan yhdistää toisiinsa ekvivalenssiluokan avulla (engl. *equivalence classes*). Tuolloin alkuperäinen sana pitsa korvataan sanalla pizza. (Manning et al. 2009, s. 32) Normalisointi operaationa on yksinkertaisemmillaan merkkijonon korvaamista toisella merkkijonolla. Sama operaatio voidaan toistaa myös tekstissä esiintyville lyhenteille *Us* ja *USA*. Merkkijonojen korvaamista tulisi välttää myös ilman avaimen kontekstia. Tästä hyvänä esimerkkinä toimii sana *kuusi*, joka voi olla numero tai puu. Eräs keino edellä kuvatun ongelman ratkaisemiseksi on selvittää asiayhteys, jossa sana mainitaan. Tämä edellyttää viereisten sanojen tutkimista.

3.2.3 Konseptien mallintaminen

Aikaisemmin huomattiin, että kirjoitettu teksti voi sisältää avainten rajoja ylittäviä konsepteja. Tämän voitiin huomata aiheuttavan myös ongelman, jossa konseptiin kuuluva avain voitiin yhdistää toiseen konseptiin. Tästä syystä tekstissä esiintyvien konseptien paikantaminen on tarpeellinen toimenpide. Sen avulla voidaan myös parantaa luokittelujärjestelmän tarkkuutta. Tuolloin luokittelualgoritmi osaa käsitellä moniosaisia konsepteja yhtenä avaimena. Ongelmasta esimerkkinä toimii aikaisemmin annettu konsepti *Hong Kong*, johon kuuluvaa avainta *Hong* ei tulisi sotkea konseptin *Michael Hong* kanssa. Hakukoneissa kyseinen haaste voidaan yleensä ratkaista hyödyntämällä fraasihakua. Tuolloin käyttäjät merkitsevät vierekkäin esiintyvät termit lainausmerkkeihin.

Tästä hakukone tietää, että lainausmerkkien sisällä olevien avainten tulee sijaita vierekkäin. Käänteistiedostossa ongelmaksi muodostuu kuitenkin se, ettei sijainneista pidetä kirjaa. Yleisesti ongelma voidaan ratkaista tallentamalla avaimen sijainti osaksi indeksiä. Tuolloin puhutaan sijainnillisesta indeksistä (engl. *positional indexes*). Sijainnin ottamisella mukaan indeksiin on kuitenkin oma hintansa levykapasiteetin ja suorituskyvyn suhteen. (Manning et al. 2009, s. 39)

Luokittelujärjestelmässä fraasihaun kaltaista menetelmää ei voida soveltaa. Tämä herättää kysymyksen, olisiko moniosaiset konseptit mahdollista tunnistaa automaattisesti. Konseptien etsimisestä käytetään nimitystä informaation purkaminen (engl. *information extraction, ie*). Käsitteenä Ie on laaja ja sillä voidaan usein viitata lähes mihin tahansa luonnollisella kielellä esitetyn informaation tunnistamiseen sisällöstä. Työn kannalta tarkasteluun on valittu konseptien mallintaminen, joista erityisesti on valittu käsiteltäväksi nimiin liittyvät ongelmat. Tämä johtuu siitä, että nimissä kuvailtuun ongelmaan on helppo törmätä. Tutkittuja menetelmiä, joita käytetään nimien tunnistamiseen, käytetään myös POS-taggereissä (engl. *part-of-speech tagger*). POS -taggerit pyrkivät luokittelemaan tekstin sanoja substantiiveihin, verbeihin ja niin edespäin. (Bunescu, R. & Mooney 2005, ss. 3-5). Tarkoituksena ei ole tarkastella suoranaisesti POS -taggereita, vaikka ne jakavat paljon yhteisiä piirteitä nimien tunnistuksen kanssa. Erityisesti nimien etsimiseen suunnatuista järjestelmistä käytetään nimitystä NER (engl. *named entity recognition).* NER ja POS-taggeri eroavat lähinnä valittavissa olevien tagien eli luokkien määrän suhteen. NER-taggerit luokittelevat sanat joko luokkaan muut tai nimi. Seuraavaksi työssä tarkastellaan, miten ihmisten, yritysten, paikkojen, katujen, tai kemikaalien nimiä voidaan paikantaa.

Sanojen luokittelemiseksi on esitetty eri menetelmiä, joihin kuuluvat käsikirjoitetut säännöt (engl. *handcrafted rules*), maksimi entropiamallit (engl. *maximum entropy models, MEM*) ja Markovin piilomalli (engl. *hidden markov models*) (Bunescu & Mooney 2005, ss. 3-5; Borthwick1999, ss. 7-9) Työhön tarkasteltavaksi on valittu käsin laaditut säännöt ja MEM-menetelmä. Näistä MEM-menetelmä on käytössä myös kielenkääntäjissä (engl. *machine translation, MT*). MEM-mallin esittelyn tarkoituksena on antaa lukijalle käsitys yleisellä tasolla sanojen luokittelusta ja niissä hyödynnetyistä periaatteista.

Käsin laaditut säännöt on suorituskykyinen ja yksinkertainen tapa tunnistaa tekstissä esiintyviä käsitteitä. Sääntöjen laatiminen edellyttää usein tietämystä ratkaistavasta ongelma-alueesta (Borthwick 1999, ss. 7-9). Erityisesti paikkojen, henkilöiden ja yritysten nimet ovat hankalia tunnistaa vaihtoehtojen suuresta määrästä ja sääntöjen monimuotoisuudesta johtuen. Ajatellaanpa konsepteja *Suomen valtio* tai *Euroopan keskuspankki*. Tarkastelemalla kielioppisääntöjä tiedämme, että yleensä nimi alkaa isolla alkukirjaimella. Sääntö ei ole täysin pitävä: *liikkuva poliisi* ja *korkein oikeus* (Nurmi 2004). On selvää, että kaikkien vaihtoehtojen ja poikkeuksien määrittäminen on hankalaa (Borthwick 1999, s. 9). Samalla on hyvä muistaa, että ihmiset kirjoittavat asioita säännöistä poiketen. Se tarkoittaa, ettei käsiteltävä teksti noudata välttämättä kielioppia

täydellisesti. Toisaalta poikkeuksien määrä voi olla niin vähäinen, ettei tehdyillä virheillä ole merkitystä.

MEM (engl. *maximum entropy model*, *MEM*) on opetusmateriaalista rakennettu stokastinen malli (engl. *stochastic model*). Stokastisen mallin avulla pyritään ennustamaan satunnaista prosessia. Sen toiminta perustuu opetusmateriaalista kerättävien näytteiden hyödyntämiseen. (Berger 1996) Annettujen mallien perusteella voidaan määrittää jokaiselle tekstissä esiintyvälle sanalle oma tagi eli luokka. Mallissa valitaan sanalle (h) tagi (t), jonka entropia on suurin. Merkintöinä Bergerin mallista poiketen työssä käytetään h-kirjainta ominaisuudelle eli sanalle ja t-kirjainta tagille Ratnaparkhin mukaisesti (Ratnaparkhi 1996).

$$\acute{p}(h,t) \equiv \frac{1}{N} \tag{9}$$

$$f(h,t) = \begin{cases} 1, & t = nimi, \text{jos sana alkaa isolla kirjaimella} \\ 0, & t = muu \end{cases} \tag{10}$$

$$p(f) \equiv \sum_{h,t} \acute{p}(h,t) f(h,t) \tag{11}$$

Sanojen luokittelu tapahtuu tarkastelemalla opetusmateriaalissa esiintyvien sanojen tageja. Kaavassa 9 määritetään, montako kertaa opetusmateriaalissa sanalle (h) on annettu tagi (t). Mukaan otettavat parit valitaan hyödyntäen binääristä valintafunktiota (engl. *feature function*), joka on annettu kaavassa 10. Arvioitavien mallien valinnassa voidaan hyödyntää sanojen muotoa. (Klein & Manning 2003, s. 2) Kirjallisuudessa esitetään myös muun tyyppisiä valintafunktioita, joten isokirjain on vain yksi vaihtoehdoista. Kirjoittajan näkemyksen mukaan nimessä esiintyvä avain voi alkaa myös pienellä, josta esimerkkinä *korkein oikeus*. Arvo tagille (t) saadaan kaavasta 11. Saadulle arvolle on tietyissä tapauksissa tarve asettaa jokin erityinen rajoite. (Berger 1996) Tämä tapahtuu kaavan 12 avulla.

$$\acute{p}(f) = \sum_{h,t} \acute{p}(h) p(t|h) f(h,t) \tag{12}$$

Kaavassa $\acute{p}(h)$ on koulutusmateriaalista määritelty (h):n odotusarvo ja $p(t|h)$ on mallille asetettu erityinen rajoite (engl. *constraint equation*). Merkinnällä $p(t|h)$ tarkoitetaan ehdollista todennäköisyyttä sille, että kohtaamme tagin (t) sanan (h) kanssa. Kyseessä on aineistosta johdettu arvo, jossa voidaan huomioida myös lähellä olevat sanat. Tuolloin malli voidaan laajentaa myös yksittäisen tutkittavan avaimen ulkopuolelle. Tässä tapauksessa arvioitavien ominaisuuksien valintaan voidaan käyttää etsintäsädettä (engl. *beam Search*). (Toutanova & Manning 2000, s. 3; Ratnaparkhi 1996, s. 136). Tietyissä tilanteissa avaimen tagi on riippuvainen muista ympäröivistä avaimista. Tämä voi tarjota hyödyllistä informaatiota käytetylle luokittelumallille siitä, annettaanko luokiteltavalle avaimelle tagiksi nimi vai muu. Se mahdollista arvioida kuuluuko merkkijono *valtio* edeltävään merkkijonoon *Suomen*. Avaimelle valtio ei aina haluta antaa tagia nimi, koska *valtio tarvinnut lainaa*. Tämä päätös MEM-menetelmässä tehdään käytännössä luo-

kittelualgoritmilla, kuten Bayesian päättelyllä (Berger 1996). Kysymys on näin ollen avainten luokittelusta. Sanojen tagituksen jälkeen ne voidaan esittää omana listanaan, joko Io- tai Iob -koodattuina. Kuvitteellinen lause: *Mika Mähönen kirjoitti tämän diplomityön, jossa Hannu Jaakkola oli valvojana*. Näyttäisi Io-koodattuna seuraavalta (kuva 14) .

Sana	Tagi
Mika	Nimi
Mähönen	Nimi
kirjoitti	Muu
tämän	Muu
diplomityön	Muu
jossa	Muu
Hannu	Nimi
Jaakkola	Nimi
oli	Muu
valvojana	Muu

Kuva 14. *Sanojen tagit IO -koodattuna (engl. IO encoding)*

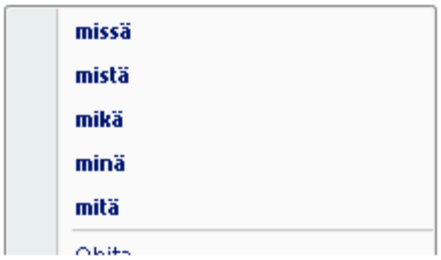
Suoranaisesti Io-koodauksella ei pyritä tunnistamaan nimen alkamista tai päättymistä, vaan se niputtaa toisiaan lähellä olevat nimet yhdeksi poimittavaksi kokonaisuudeksi. Vaihtoehtoisesti järjestelmä voi hyödyntää myös Iob-koodausta, joka merkitsee nimille mahdollisen aloituksen ja lopetuksen. (Manning 2013)

3.2.4 Kirjoitusvirheiden korjaus

Ihmiskielellä kirjoitettu aineisto sisältää usein myös kirjoitusvirheitä. Tietokoneesta poiketen ihminen kykenee tulkitsemaan tekstiä usein täysin oikein huolimatta virheistä. Tietokoneen suorittamassa vertailussa yhdenkin merkin muuttaminen tarkoittaa eri avainta. Tilastollinen arviointi puolestaan perustuu merkkijonojen vertailuun (engl. *pattern matching*), joten kyseessä on haaste. Ihmisen tulkinta on kyvykästä, koska seuraava teksti on tulkittavissa: *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.* (Davis 2003) Lukijan on hyvä tietää, ettei kyseistä tutkimusta tehty Cambridgessä, vaan tekstillä pyritään demonstroimaan aivojemme kyvykkyyttä. Tutkimusten mukaan tulkinta onnistuu, koska ihmisen aivot eivät tulkitse jokaista merkkiä erikseen. Sanojen tunnistamiseksi on tärkeää, että yli neljä merkkiä pitkien sanojen ensimmäinen ja viimeinen kirjain tulee olla oikeilla paikoillaan. On kuitenkin huomattu, että tämän tyyppisen tekstin lukemisella on suurempi kognitiivinen kuormittavuus (Rawlinson 2007, ss. 26-27; Davis 2003; Rayner et al. 2006, ss. 192-193). Tietokone ei kykene edellä kuvatun tyyppiseen virheiden korjaukseen. Tästä syystä riittävä määrä kirjoitusvirheitä voi johtaa myös väärään luokittelupäätökseen. Kirjoitusvirheitä näkee hyödynnettävän myös roskaposteissa, jolloin tavoitteena on estää luokittelujärjes-

telmää tunnistamasta viestiä. Kirjoitusvirheiden korjaamiseksi kirjallisuudessa esitellään kaksi menetelmää, jotka ovat merkkijonoon rajoittunut päättely (engl. *isolated-term correction*) ja asiayhteyden huomioiva virheiden korjaus (engl. *context sensitive spelling correction*).

Merkkijonoon rajoittunut päättely (engl. *isolated-term correction*) pyrkii nimensä mukaisesti korjaamaan virheitä yksittäisten sanojen osalta. Se ei huomioi kuitenkaan asiayhteyttä, jossa kyseinen sana esitetään. Virheiden tunnistamisessa yleinen menetelmä on hyödyntää sanakirjaa. Tuolloin järjestelmä vertaa kaikkia tekstissä kohdattuja avaimia sanakirjassa oleviin. (Manning et al. 2009, ss. 62-65) Ajatellaanpa kirjoitusvirhettä *misä ja million*. Ne eivät esiinny suomen kielen sanakirjassa, joten ne voidaan tunnistaa tekstin joukosta melko yksinkertaisesti. Tämän tyyppisten virheiden korjaamiseksi ehdotetaan menetelmää, joka tunnetaan nimillä *Edit Distance* tai *Levenshtein Distance.* Se perustuu ideaan etsiä tuntemattoman merkkijonon (S1) lähellä olevia tunnettuja sanoja. Menetelmä laskee pienintä mahdollista poisto-, lisäys- tai muokkausoperaatioiden määrää, joka vaaditaan merkkijonon (S1) muuttamiseksi sanakirjan sanaksi (S2). Samasta algoritmista on myös saatavilla kehittyneempi versio, joka sallii myös kirjainten vaihtamisen keskenään. (Manning et al. 2009, s. 58) Yksi menetelmään liittyvistä haasteista on verrattavien sanojen valinta sanakirjasta. Kaikille sanakirjan sanoille etäisyyttä ei ole usein järkevää laskea, koska sanakirja sisältää paljon sanoja. Arvioitavat termit voidaan valita hyödyntäen n-grammeja, jolloin arvioidaan sanan yhteisten n-grammien määrää verrattavaan merkkijonoon. Se voidaan tehdä määrittämällä ns. *Jaccard Coefficient*. Tietyn osuuden yhteisiä n-grammeja sisältävät sanakirjan sanat valitaan menetelmän avulla mukaan tarkasteluun. (Manning et al. 2009, ss. 60-61) Tutkimalla annettuja merkkijonoja *misä* ja *million* voidaan tehdä seuraavat johtopäätökset. Ensimmäiseen jonoon lisäämällä yksi kirjain saadaan *missä*. Jälkimmäisestä merkkijonosta saadaan *milloin* vaihtamalla i:n ja o:n paikat keskenään. Monia sanakirjan tuntemia sanoja voi myös sijaita samalla etäisyydellä, joten korjaavan termin valinta ei ole yksinkertaista. Sanat *mikä*, *mistä*, *missä, mitä isä*, ja *lisä* sijaitsevat samalla etäisyydellä merkkijonosta *misä*. Tuolloin yksittäistä sanaa ei voida valita, koska kaikki korjauksista ovat yhtä todennäköisiä. Jos korjauksesta ei ole varmuutta, jolloin on turvallisempaa ehdottaa mahdollisia ratkaisuvaihtoehtoja. Näin toimii esimerkiksi Microsoft Word 2007 (kuva 15)**.**

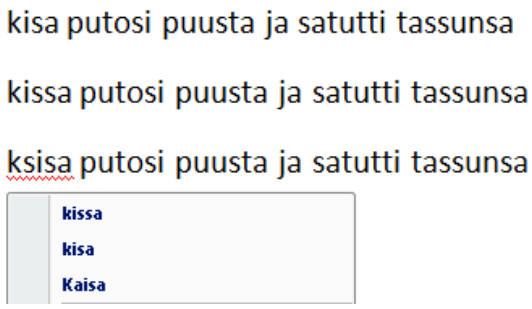


Kuva 15. *Suomen kielen oikoluku Microsoft Wordissa*

Yleensä korjauksen valinta ei ole yksinkertaista, koska vaihtoehtoja on saatavilla useita. Korjausvaihtoehtojen listaamiseen paremmuusjärjestykseen esitetään muutamia tapoja.

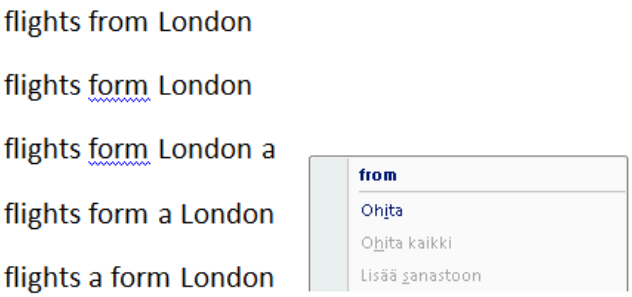
Yksi keino on hyödyntää näppäinasettelua ja kirjainten etäisyyttä toisistaan näppäimistöllä. Tuolloin hyödynnetään ajatusta siitä, että mahdollinen kirjoitusvirhe on voinut syntyä huolimattomasta näppäimen painalluksesta. (Manning et al. 2009, s. 62) Merkkijonossa *misä* esiintyvä kirjain *s* sijaitsee näppäimistöllä varsin kaukana kirjaimista *t* ja *k*. Sen perusteella voidaan olettaa, että käyttäjä ei todennäköisesti tarkoittanut vaihtoehtoja *mikä* tai *mitä*. Tämä tarkoittaa painokertoimien asentamista eri virheille. Työn puitteissa asiaa ei kuitenkaan ole tarkoitus käsitellä tämän tarkemmin.

Merkkijonoon rajoittuneen päättelyn suurin haaste on kuitenkin, ettei sen avulla voida havaita kontekstisidonnaisia virheitä. Lause *kisa tipahti puusta ja loukkasi tassunsa*, vaikutti Wordin mielestä järkevältä (kuva 16). Ihminen puolestaan kykenee havainnoimaan puuttuvan s-kirjaimen toisin kuin tietokone. Tätä virhettä ei havaita, koska sana *kisa* löytyy sanakirjasta. On selvää, että käyttäjä halusi tarkoittaa sanaa *kissa* sanan *kisa* asemasta. Kirjoittajan tekemien havaintojen mukaan työn kirjoituksessa käytetty Microsoft Wordin versio 2007 ei suorita suomeksi asiayhteyden huomioivaa oikolukua (kuva 16)



Kuva 16. *Microsoft Word 2007 virheiden tunnistus suomesta*

Kyseinen ominaisuus näyttää kuitenkin olevan saatavilla englannin kielelle (kuva 17). Kuvasta 17 on nähtävillä, kuinka Microsoft Word 2007 tunnistaa kontekstisidonnaisen virheen.



Kuva 17. *Microsoft Word 2007 kontekstisidonnainen oikoluku*

Asiayhteyden huomioiva kirjoitusvirheiden korjaus havaitsee tämän tyyppiset virheet. Kyseinen menetelmä ei ole rajoittunut ainoastaan yksittäisen sanan kontekstiin. (Manning et al. 2009, s. 62) Se laajentaa tutkimista myös muihin lauseen jäseniin, jolloin voidaan selvittää esitetäänkö sana oikeassa kontekstissa. Sanoista kaikki eivät ole sellaisia, että niitä voidaan sotkea helposti sanakirjan muihin sanoihin. Tästä syystä yleinen tapa on laatia ns. valintalista, jonka perusteella tutkittavat kohteet valitaan teks-

tistä. Listasta käytetään myös nimitystä keskenään sotkettavien sanojen joukko (engl. *confusion set*). Tähän joukkoon pyritään lisäämään sellaiset merkkijonot, jotka ovat lähellä jotain tunnettua sanaa. Joukkoon halutaan myös lisätä sellaiset sanat, joiden lausumisasu muistuttaa läheisesti jotakin toista sanaa (Golding 1995, s. 40). Suomen kielessä esimerkkinä asiasta toimivat sanat *kisa* ja *kissa*, jotka voivat sekoittua toisiinsa kirjoitusvirheen seurauksena. Molemmilla sanoista on oma konteksti, jossa ne esitetään tyypillisesti. Tätä virhettä ei voida havaita kuitenkaan sanakirjan avulla, koska molemmat ovat sanoja. Suomen kielessä oman haasteensa tuovat lisäksi yhdyssanat vrt. *äidin kieli* ja *äidinkieli*. Molemmat vaihtoehdoista voivat olla oikein, joskin tämä riippuu sanan kontekstista.

Perusmenetelmä (engl. *baseline methods*) hyödyntää sanojen esiintymistiheyttä. Tuolloin kohdattaessa sekoitettava sana tehdään seuraavaa: Tutkitaan kuinka monta kertaa nykyinen ja mahdollisesti sekoitettava vaihtoehto esiintyy tekstissä. Tämän perusteella valitaan useammin esiintynyt vaihtoehto korjausehdotukseksi. Käytännössä tällä tavoin ei voida huomioida varsinaista asiayhteyttä. Tuolloin korjausta ehdotetaan ainoastaan, jos termi esiintyy harvemmin sanastossa kuin toinen (Golding 1995, s. 41). Ajatellaanpa virhettä *flights form London* (kuva 17). Tarkastellessa englannin kielen korpusta on varsin selvää, että sana *from* esiintyy tiheämmin kuin *form*. Se ei kuitenkaan tarkoita, että *form* tulisi korjata aina sanaksi *from.*

Lähettyvillä olevien termien (engl. *context words*) hyödyntäminen perustuu ideaan, että jokaista tutkittavaa sanaa ympäröi tyypillinen joukko muita sanoja. Sanan *form* lähettyvillä on todennäköisesti artikkeli *a*. Tätä tietämystä voidaan hyödyntää myös arvioitaessa esiintyykö sana oikeassa kontekstissa. Menetelmässä valitaan ikkuna (engl. *window*), jonka perusteella naapurisanat valitaan tutkinnan kohteeksi. Valittuja sanoja verrataan tämän jälkeen koulutusmateriaalista saatuihin esimerkkeihin. Tuolloin kysymys on luokittelusta, jossa sana joko voi esiintyä toisen sanan parina tai sitten ei. Tämä päätös voidaan tehdä esimerkiksi Bayesian luokittelulla (Golding 1995, s. 41-42). Käytettyä esimerkkiä ajatellen on varsin epätodennäköistä, että sanan *kisa* lähettyvillä löydettäisiin viittaus sanaan *tassu*. Työn yhteydessä havaittiin, että Microsoft Word 2007 havaitsi artikkelin *a*, joka sijaitsi sanan *form* vasemmalla tai oikealla puolella (kuva 17). Tästä syystä se ei tunnistanut *flights a form London* virheellisyyttä.

Sanojen järjestyksen hyödyntäminen (engl. *collactions*) perustuu ideaan tunnistaa eri sanaluokkiin kuuluvat avaimet. Menetelmässä hyödynnetään POS-tageja (engl. *part-of-speech tagging*). Sen ansiosta on mahdollistaa tutkia, onko sana mahdollisesti riippuvainen toiseen sanaluokkaan kuuluvasta sanasta. Englannin kielessä sanaa autiomaa (engl. *the desert*) edeltää aina määräinen artikkeli *the*. Tämä tarkoittaa, että sanalla *desert* on riippuvuus artikkeliin *the*. Vastaavasti sanan jälkiruoka (engl. *dessert)* kanssa määräistä artikkelia ei oletuksena käytetä. (Golding 1995, s. 44-46) Nämä kaksi sanaa on varsin helppoa sotkea toisiinsa. Kirjoittajan käsityksen mukaan suomen kielessä vastaavaa päättelyä ei suoraan voida hyödyntää. Suomen kielestä puuttuvat artikkelit, jotka toimivat hyvänä tunnusmerkkinä substantiiveille. Sanojen päätteet voisivat tarjota

tämän tyyppistä informaatiota. Asiaa ei ollut työn puitteissa mahdollista tutkia tätä tarkemmin.

Kontekstiin sidottujen kirjoitusvirheiden korjaamisessa voidaan tyypillisesti hyödyntää tilastollista mallia, jonka avulla tutkitaan esiintyykö sana viereisten sanojen parina. Menetelmässä voidaan hyödyntää myös sanajärjestyksiä ja muita vastaavantyyppisiä menetelmiä.

3.2.5 Ominaisuuksien valinta

Ominaisuuksien valinta on eräs tärkeimmistä työtehtävistä luokitteluprosessissa, koska luokiteltavat dokumentit pitävät sisällään häiritseviä tekijöitä. Ilmiöstä käytetään myös nimitystä kohina (engl. *noise*). Ominaisuuksien valintaprosessin (engl. *feature selection*) tarkoituksena on parantaa luokittelun tarkkuutta ja algoritmien suorituskykyä. Todellisuudessa luokittelussa tarkoituksena ei ole arvioida kaikkia tekstissä esiintyviä sanoja. Tämä tarkoittaa sellaisten ominaisuuksien eli avainten karsimista, joilla ei ole tilastollista merkitsevyyttä. Niiden poistamiseen voidaan hyödyntää sulkusanalistoja (engl. *stop word lists*) ja tilastollisia menetelmiä.

Sulkusanalistat ovat avainten karsimiseen yksinkertaisin keino. Menetelmällä tarkoitetaan sanalistaa, josta löytyvät useimmin kielessä esiintyvät sanat. Suomen kielissä niitä ovat *on*, *ja*, *se*, *siksi*, *tämä* ja *ja*. (Kotimaisten kielten keskus 2012a) Näiden sanojen merkitys on myös todennäköisesti vähäinen luokittelupäätöksen kannalta, koska ne esiintyvät lähes jokaisessa dokumentissa (Manning et al. 2009, ss. 22-25; Zhu s. 81). Se tarkoittaa yleensä, ettei tämän tyyppisten avainten perusteella muodostaa ryhmiä. Sulkusanalistojen yksinkertaisuuden takia ne ovat myös suosittuja. Lista ei kuitenkaan ole aina riittävän tehokas menetelmä ominaisuuksien valintaan. Menetelmä on lähinnä yleiskäyttöinen tapa poistaa ominaisuuksia huomioimatta aineistoon liittyviä erityispiirteitä. Listojen laatiminen voidaan nähdä myös ongelmaksi, koska se mikä pätee yhdessä paikassa ei päde kaikkialla. Organisaation oma nimi voidaan löytää usein lähes jokaisesta dokumentista, jolloin se ei tuo tilastollista erottavuutta dokumenttien välillä. Tämä ei tarkoita sitä, ettei sillä voisi olla merkitystä tietyissä käyttökohteissa.

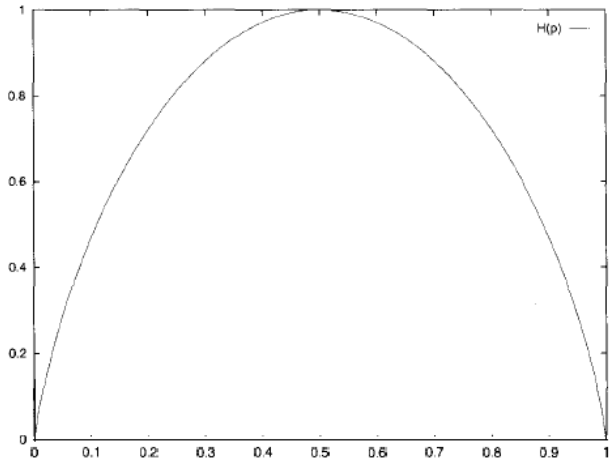
Avainten valitseminen tilastollisesti ratkaisee tämän haasteen, koska sen avulla voidaan ottaa huomioon aineistoon liittyviä erityispiirteitä. Eräitä kirjallisuudessa tyypillisesti esiteltyjä menetelmiä ovat informaationsaanti (engl. *information gain*), giniindeksi, termin tiheys (engl. *document frequency tresholding*), khii toiseen jakauma (engl. *chi-squared distribution*) ja yhteinen informaatio (engl. *mutual information*, MI). Osa näistä menetelmistä esiintyy myös muualla tilastotieteissä. Samalla osaa kaavoista on myös optimoitu luokittelun tarpeisiin. Tuolloin kaava voi poiketa sen tutusta esityksestä muilla tieteenaloilla. Näitä kaavoja käytetään myös osana muutamia luokittelumenetelmiä. Menetelmissä ideana on löytää ne sanat eli ominaisuudet, joiden perusteella aineisto voidaan jakaa luokkiin. Tästä syystä työssä pyritään selvittämään ensin, mitä informaatio oikeastaan on. Asiaa on helpoin lähestyä Claude Shannonin informaatioteorian (engl. *information theory*) kautta. Se tunnetaan tietojenkäsittelytieteissä myös nimillä entropia tai informaatio. Teorian mukaan informaatiota on olemassa, jos jokin on

epävarmaa. Informaatiota ei ole olemassa, jos jokin on täysin varmaa. Teorian merkityksen ymmärtäminen on helppoa tarkastellessa kuvitteellista kolikkoa, jossa molemmat puolet ovat klaavoja. Heittämällä tällaista kolikkoa ei voida saada mitään informaatiota. Kannattaa pohtia asiaa sen kannalta, mikä on todennäköisyys saada kruuna tai klaava kyseisellä kolikolla. Vastaus tähän kysymykseen on, ettei kruunaa voi saada eli todennäköisyys on nolla (0). Klaavalle puolestaan todennäköisyys on yksi (1), koska tulos on sama riippumatta heittojen määrästä. Nyt voimme esittää saman kysymyksen uudelleen: mitä informaatiota heitto tuo, jos tiedämme tuloksen etukäteen? Vastaus on tietenkin: ei mitään. Normaali kolikko eli painottamaton kolikko voi puolestaan päätyä tilaan klaava tai kruuna. Se tarkoittaa, että molemmat tiloista ovat teoreettisesti yhtä todennäköisiä. Tuolloin todennäköisyys saada kruuna on $\frac{1}{2}$ ja todennäköisyys saada klaava on $\frac{1}{2}$. Teoria osoittaa, että normaalin kolikon heittäminen tuottaa informaatiota. (Gupta 2012, ss. 116-117) Asiaa voidaan perustella informaatiolla. Tiloihin liittyvä informaatio voidaan kirjoittaa tuolloin kaavan 13 mukaan.

$$I = \sum_i (-p_i \log p_i) \qquad (13)$$

Kaavassa (p_i) on tapahtuman (i) todennäköisyys, jolloin informaatio koostuu tilojen todennäköisyyksien tai odotusarvojen summasta. Kaavasta huomautuksena, että siinä käytetään yleisesti 2-kantaista logaritmifunktiota. Sen käyttö ei ole kuitenkaan edellytys, joten siihen ei haluta ottaa kantaa.

Jos tarkastellaan esimerkkinä käytettyjä kolikkoja, jotka olivat täysin painotettu ja painottamaton kolikko. Ainoastaan klaavoja heittäessä tuottavaa kolikkoa kutsutaan täysin painotetuksi kolikoksi ja se omaa vain yhden tilan. Sen heitosta saatava informaatio on näin ollen $-1\log_2(1) = 0$. Normaalin kolikon eli painottamatonheitosta saatava informaatio on $-\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1$. Asiaa on hyvä tarkastella kuvaajasta (kuva 18) tilanteen ymmärtämiseksi.



Kuva 18. *Kolikon heiton entropia (Manning & Schütze 1999, s. 63)*

Esitetyssä kuvassa 18 horisontaalisella akselilla on esitetty todennäköisyys saada tulokseksi klaava. Pystyakselilla on esitetty kolikon heiton entropia **(**Manning & Schütze 1999, ss. 60-63). Kuviosta on helppoa huomata, että painottamaton kolikon heiton todennäköisyys $\frac{1}{2}$ maksimoi entropian. Kyseessä on toisin sanottuna maksimaalinen epävarmuus saada tulokseksi klaava. Maksimaalinen epävarmuus saavutetaan, jos mitattu jakauma on tasainen (engl. *uniform distribution*). Tuolloin kruunien ja klaavojen osuudet mitatusta jakaumasta ovat teoreettisesti yhtä suuret. Todellisuudessa kolikko voi olla osittain painotettu, jolloin saatu jakauma ei kuitenkaan ole tasainen. Se tarkoittaa, että heittojen jakaumasta määritetty odotusarvo on yleensä entropian maksimin vasemmalla tai oikealla puolella. Kolikko voi olla myös osittain painottamaton, jolloin esimerkiksi mittauksesta voimme saada seuraavat tulokset. Kolikkoa heitettiin 1000 kertaa, joista 700 kpl oli klaavoja ja 300 kpl kruunuja. Tässä tilanteessa kysymys voi olla painotetusta kolikosta, jonka heitot eivät tuota tasaista jakaumaa. Tuolloin mittaustulosten perusteella voimme tehdä johtopäätöksen, ettei todennäköisyysjakauma kuvaa vallitsevaa todellisuutta. Voidaan sanoa, että odotusarvot ovat kruunalle $\frac{3}{10}$ ja klaavalle $\frac{7}{10}$. Todennäköisyysjakauma on yleensä hyvä arvaus tuloksien jakautumisesta. Sen käyttö on järkevää, jos parempaakaan tietämystä ei ole saatavilla. Asiaa voidaan perustella sillä, että emme halua olettaa mitään jakaumasta ilman parempaa tietämystä. Sitä kuvaa hyvin Thomas Jeffersonin sanat: *Ignorance is preferable to error; and he is less remote from the truth who believes nothing, than he who believes what is wrong*. Käännettynä: on parempi olla tietämätön virheestä. Se joka ei usko mihinkään on lähempänä totuutta kuin hän, jonka uskomus on väärä. (Walker 2002)

Lyhyesti tasainen jakauma tarkoittaa maksimientropiaa eli suurinta mahdollista epävarmuutta saatavasta lopputuloksesta. Luokittelun kannalta tasaisesti jakautuneet avaimet ovat huonoja, koska ne tarkoittavat suurinta mahdollista epävarmuutta valinnan kannalta. Haluamme valita toisin sanoen avaimet, jotka vähentävät epävarmuutta eniten päätöksen tekemiseksi. Onneksemme kaikki tekstin avaimet eivät noudata tasaista jakaumaa ja tietyn tyyppiset dokumentit sisältävät yleensä tietyn tyyppisiä avaimia. Asiasta antaa myös viitteitä aikaisemmin käsitelty Zipfin laki. Lisäksi tähän ideaan perustuvat myös tiedonhakumenetelmät. Todellisuudessa aineisto sisältää avaimia, jotka sisältyvät lähes jokaiseen dokumenttiin. Näiden lisäksi meillä on avaimia, jotka löytyvät vain tietyn tyyppisistä dokumenteista. (Manning et al. 2009, s. 89; Kummamuru 2012, s. 10)

Informaation saanti on vain yksi tavoista suorittaa ominaisuuksien valintaa. Sen hyödyntäminen perustuu ideaan etsiä ne attribuutit, jotka minimoivat epävarmuuden. Tämä tarkoittaa (I) informaation eli entropian määrittämistä jokaiselle attribuuteista (kaava 13).

$$Gain(S, A) = I - \sum_{i \in values(A)} \left(\frac{t_i}{s}\right) I_i \qquad (14)$$

Kaavassa 14 (I) on informaatio ennen aineiston ositusta eli entropia. $\sum_{i \in values(A)} \left(\frac{t_i}{s}\right) I_i$ on informaation määrä aineiston osituksen jälkeen, jossa (I_i) solmun (i) sisältämä informaatio. Tässä (t_i) on niiden objektien määrä, jotka kuuluvat solmun (i) ja (s) kaikkien solmun jäsenten määrä. Merkinnällä (A) tarkoitetaan arvioitavien vaihtoehtojen joukkoa ja (S) koulutusmateriaaliin kuuluvia kohteita (Gupta 2012, s. 118). Binäärisessä luokittelussa esimerkiksi symbolilla (A) tarkoitetaan vaihtoehtoja *On* tai *Ei*. Kaavasta Zhai esittää samasta myös kehittyneemmän version. Se soveltuu arviointiin, kun halutaan huomioida avaimen esiintymistiheys (Zhai 2012, s. 169).

Gini-indeksillä voidaan mitata avaimen (t) merkityksellisyyttä. Alun perin gini-indeksi kehitettiin mittaamaan eroavaisuuksia frekvenssijakaumassa, mutta sittemmin gini-indeksiä on alettu soveltaa myös muualla. Olkoon $p_i(t)...p_k(t)$ ne luokkien nimet, joihin avain (t) kuuluu. Tuolloin gini-indeksi voidaan kirjoittaa kaavan 15 avulla:

$$\sum_{i=1}^{k} p_i(t) = 1 \tag{15}$$

Esitetyssä kaavassa 15 (p_i) on tiheys ehdolla, että dokumentti kuuluu luokkaan (i) ja sisältää avain (t). Tässä (k) on eri luokkien lukumäärä, joista avain eli sana (t) on löydettävissä. Avaimelle indeksin arvo saadaan tuolloin kaavan 16 mukaisesti.

$$G(t) = \sum_{i=1}^{k} p_i(t)^2 \tag{16}$$

G(t) arvo on aina ($\frac{1}{k}$, 1) välillä. Suuremmat G(t) arvot tarkoittavat paremmin materiaalin erottavaa avainta. Jos kaikki luokkaan kuuluvat dokumentit sisältävät aina avaimen (t), tuolloin G(t) on 1. Jos avain on jakautunut tasaisesti luokkien välillä, arvoksi saadaan $\left(\frac{1}{k}\right)$. Huonona puolena perusversiossa on, ettei se ota huomioon avaimen esiintymistä dokumenttitasolla. Samalla se puolestaan voi vääristää valittuja avaimia tilanteissa, joissa vain luokan osa dokumentista sisältää avaimen. Tästä syystä on kehitetty myös normalisoitu gini-indeksi, jolla voidaan ottaa huomioon edellä esitetty puute (kaava 18). Sen laskemiseksi tarvitaan myös kaavaa 17.

$$p_i'(t) = \frac{p_i(t)/P_i}{\sum_j^k p_j(t)/P_j} \tag{17}$$

$$G(t) = \sum_{i=1}^{k} p_i'(t)^2 \tag{18}$$

Kaavassa 17 $P_1 \ldots P_k$ tarkoitetaan sitä, miten dokumentit ovat jakautuneet globaalisti eri luokkien välille (Aggarwal & Zhai 2012, ss. 168-169).

Df-menetelmä mittaa, kuinka monessa dokumentissa kyseinen avain (t) esiintyy. Ideana menetelmässä on poistaa ne avaimet, jotka esiintyvät harvoin korpuksessa. Menetelmässä valitaan kynnys (engl. *threshold*), jonka avaimen tulee ylittää ollakseen mukana arvioinnissa. Menetelmän perusoletus on tarkastella sanan (t) jakautumista eri do-

kumenttien välillä. Avaimen löytyessä harvoin, se voidaan poistaa arvioitavien avaimien listalta. Yksittäisissä dokumenteissa esiintyviä avaimia ei yleisesti ole järkevää arvioida niiden vähäisen informaation takia. Menetelmän ongelmana on, ettei tämän tyyppinen valinta heijasta tehokkaasti alla olevaa luokkarakennetta. Tästä syystä se soveltuu paremmin tehokkuuden parantamiseen kuin varsinaisten ominaisuuksien valintaan. (Yang & Pedersen 1997). Lisäksi kaikissa dokumenteissa esiintyvät avaimet halutaan poistaa, koska ne eivät lisää informaatiota luokan ennustamiseksi.

Khii toiseen jakaumalla voidaan mitata myös avaimen ja luokan välistä riippumattomuutta toisistaan hyödyntäen kaksisuuntaista kontingenssitaulua (Yang & Pedersen 1997). Huomattavaa on, että kyseinen kaava poikkeaa yleisesti tilastotieteissä tunnetusta khii toiseen jakaumasta. Kyseessä on tietokoneelle optimoitu versio (Manning et al. 2009, s. 276).

$$X_i^2(i,t) = \frac{N*(AD-CB)^2}{(A+C)*(B+D)*(A+B)*(C+D)} \quad (19)$$

Kaavassa 19 (A) on montako kertaa avain esiintyy luokassa. Merkinnällä (B) tarkoitetaan, montako kertaa avainta (t) esiintyy ilman luokkaa (i). Kaavassa esiintyvällä (C) tarkoitetaan, montako kertaa luokka (i) esiintyy ilman avainta (t). Puolestaan (D) tarkoitetaan tapauksia, jossa dokumentti ei kuulu luokkaan (*i*) ja ei sisällä avainta (*t*). Kaavassa dokumenttien kokonaismäärä on (*N*). Esitetystä kaavasta saatu arvo on nolla, mikäli avain (t) ja luokka (i) ovat riippumattomia toisistaan. Suurin arvo puolestaan kuvaa parasta riippuvuutta luokan ja avaimen välillä (kaava 20). Arvo määritellään koulutusmateriaalissa jokaisen avaimen ja kategorian välille. Kaavassa 21 on esitetty keskimääräinen termin korrelaatio luokan ja avaimen välillä (Yang & Pedersen 1997).

$$X_{max}^2(t) = \max_{i=0}^{m}\{X_i^2(t,c_i)\} \quad (20)$$

$$X_{avg}^2(t) = \sum_{i=0}^{m} P_r(c_i)\, X_i^2(t,c_i) \quad (21)$$

Yhteinen informaatio (engl. *mutual information*) käytetään mittamaan, kuinka paljon kaksi satunnaista muuttujaa kertoo toisistaan. Menetelmää käytetään yleisesti tilastollisessa kielen mallinnuksessa (engl. *statistical language modelling)* mittaamaan sanojen välistä yhteyttä . Avaimen (t) ja luokan (i) välille yhteinen informaatio voidaan määrittää kaavan 22 avulla. (Yang & Pedersen 1997)

$$I(t,i) = \log\frac{A*N}{(A+C)*(A+B)} \quad (22)$$

Missä (A) on montako kertaa avain (t) esiintyy luokan (i) kanssa. Merkinnällä (B) tarkoitetaan, montako kertaa sana (t) esiintyy ilman luokkaa (i). Symbolilla (C) tarkoitetaan, montako kertaa luokka (i) esiintyy ilman avainta (t). Merkinnällä (N) tarkoitetaan, kaikkien dokumenttien määrä. Arvon ollessa lähellä nollaa avain (t) ja luokka (i) ovat

riippumattomia toisistaan. Tästä kaavasta Yang & Pedersen esittävät myös kaksi vaihtoehtoista laskentatapaa. Niitä ei kuitenkaan ole työn kannalta katsottu tarpeelliseksi esitellä syvällisemmin.

3.2.6 Yhteenveto tekstin prosessoinnista

NLP-menetelmien tehtävänä on poistaa kielen tilastolliseen analysointiin liittyviä haasteita. Näihin haasteisiin kuuluvat usealla tavalla kirjoitettujen merkkijonojen yhdistäminen toisiinsa. Merkkijonojen käsittelyn voidaan katsoa alkavaksi kirjoitetun kielen tunnistamisesta, sillä jokainen kieli pitää sisällään joukon omia käsittelysääntöjä. Luokiteltavaa tekstiä on pakko prosessoida, koska tietokone ei kykene suoraan yhdistämään sanojen eri taivutusmuotoja toisiinsa. Haasteen ratkaisemiseksi yleinen tapa on palauttaa kaikki tekstissä esiintyvät avaimet perusmuotoon, jolloin niitä voidaan vertailla keskenään. Kirjoittajan käsityksen mukaan tämä prosessi hävittää informaatiota. Näin tapahtuu erityisesti suomen kielessä, koska kyseessä on morfologisesti rikas kieli. Englannissa puolestaan informaatiota on vähemmän sanojen loppupäätteissä kuin suomen kielessä. Yksi merkittävä ongelma suomen kielessä ovat myös yhdyssanat, joiden avulla voidaan laajentaa käytettyä sanaa. Voimme liittää esimerkiksi substantiiveihin tarkentavia käsitteitä. Autonkuljettajaa voidaan laajentaa esimerkiksi linja-autonkuljettajaksi. Näiden kahden merkkijonon esitys tietokoneelle on kuitenkin eri, koska niitä ei ole mahdollista yhdistää toisiinsa. Tällaisessa tilanteessa on epäselvää tulisiko autonkuljettaja pystyä yhdistämään linja-autonkuljettajaan luokittelujärjestelmässä. Tämän haasteen osalta hakukoneissa on ehdotettu käytettäväksi yhdyssanojen pilkkomista. Tämä lähestymistapa voi johtaa sanaan kuuluvien avainten sotkeutumiseen virheellisesti muihin konsepteihin. Asia liittyy läheisesti Rosch & Lloyd esittämään teoriaan prototyypeistä. Tämä teoria voi kirjoittajan mukaan tuoda myös arvokasta informaatiota luokittelujärjestelmälle.

Merkkijonovertailun kannalta myös kirjoitusvirheiden huomattiin aiheuttavan ongelmia. Tämä johtui siitä, ettei tietokone kykene yhdistämään automaattisesti kahta eritavoin kirjoitettua merkkijonoa toisiinsa. Luokiteltava teksti ei ole myöskään läheskään aina täydellistä, joten kirjoitusvirheiden korjaaminen on tarpeellinen toimenpide ennen luokittelua. Tätä ideaa hyväksikäytetään myös roskaposteissa, joissa lääkeaineiden nimiä nähdään kirjoitettavan usein väärin. Tämä ei estä kuitenkaan ihmistä lukemasta saatua viestiä, sillä aivomme eivät tulkitse jokaista merkkiä erikseen. Tutkimusten mukaan yli neljä merkkiä pitkien sanojen keskimmäisten kirjainten paikat voidaan vaihtaa ilman, että se estäisi sanan lukemista.

Muita työssä havaittuja ongelmia olivat sanojen rajoihin liittyvät ongelmat. Eräs merkittävimmistä ongelmista syntyy moniosaisista konsepteista. Tämän seurauksena kaksi konseptia voidaan yhdistää virheellisesti toisiinsa. Tästä työssä annettiin esimerkki *Hong Kong*, jonka avain *Hong* voitiin yhdistää konseptiin *Michael Hong*. Tämän tyyppisten ongelmien ratkaisu edellyttääkin moniosaisten konseptien käsittelyä yhtenä avaimena luokittelujärjestelmässä, sillä käänteistiedosto ei pidä kirjaa avainten sijainnista tekstissä.

Tilastollinen analysointi edellyttää usein myös avainten valintaa. Tämä tarkoittaa sellaisten avainten etsimistä, jotka ovat tilastollisesti merkittäviä luokittelupäätöksen tekemiseksi. Sellaiset avaimet, jotka esiintyvät kaikissa tai vain harvoissa dokumenteissa eivät tuo luokittelun kannalta lisää informaatiota. Tämän tyyppiset avaimet poistetaan tai jätetään huomioimatta. Tämän operaation tavoitteena on tehostaa käytetyn luokittelualgoritmin toimintaa.

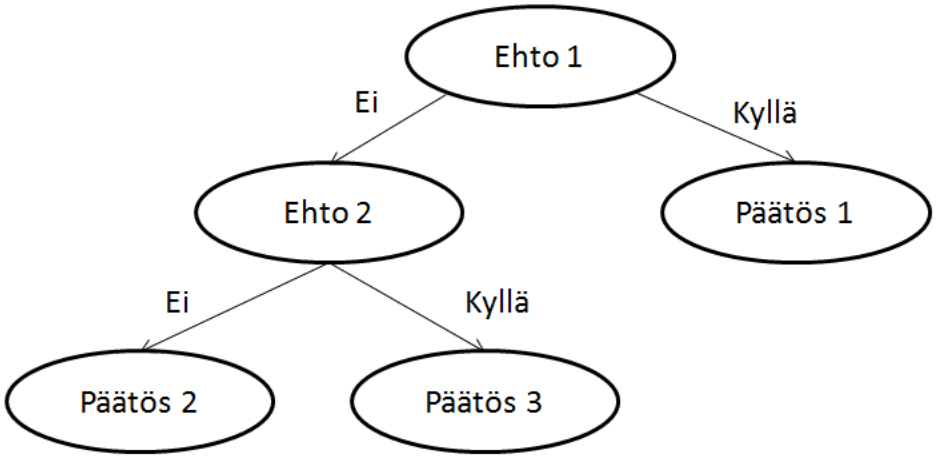
Lähes kaikissa esitellyissä menetelmissä tarvittiin luokittelua kielellisten ongelmien ratkaisemiseksi. Luonnollinen kieli ei usein omaa sellaista yksiselitteisyyttä, että siihen liittyviä ongelmia voitaisiin ratkaista loogisin käsittelysäännöin. Tästä syystä myös NLP-menetelmissä tarvitaan luokitteluun pohjautuvaa päätöksentekoa. Sen avulla voidaan ratkaista ongelmia, kuten esitetäänkö sana oikeassa kontekstissa tai kuuluvatko avaimet yhteen konseptiin. NLP-menetelmät ovat näin lähinnä työkaluja tilastollisen mallin muodostamiseksi.

3.3 Luokittelumenetelmät

Kohdassa perehdytään muutamiin yleisempiin luokittelumenetelmiin ja niiden toimintaperiaatteisiin. Työhön rajaukseksi on valittu päätöspuut (engl. *decision tree*) ja Bayesian päättely (engl. *naive bayes*), jotka ovat kirjallisuudessa yleisimmin esitettyjä menetelmiä. Tarkoitus on antaa käsitellyistä menetelmistä yksinkertainen esimerkki, jonka avulla menetelmien toimintaa voidaan ymmärtää käytännössä.

3.3.1 Päätöspuut

Päätöspuut (engl. *decision trees*) jakavat koulutusmateriaalin perusteella datan hierarkkiseksi rakenteeksi, jossa on loogisia ehtoja ja päätöksiä. Tämä rakenne voidaan esittää puuna (kuva 19). Puun muodostamiseksi koulutusmateriaali jaetaan pienempiin alijoukkoihin (engl. *subset*) eli alipuihin. Se tapahtuu hyödyntämällä ehtoja, jotka esitetään solmuina (engl. *node*) muodostettavassa puussa. Johtopäätökset puolestaan esitetään aina lehtinä (engl. *leaf*).



Kuva 19. *Kuvitteellinen päätöspuu*

Käytettävät ehdot voidaan johtaa aina tutkimalla opetusmateriaalia ja siinä esiintyviä ominaisuuksia. Tekstin yhteydessä ominaisuudet ovat tyypillisesti avaimia tai niiden klustereita. Ehtojen muodostamisessa yleinen periaate on valita se ominaisuus ensin, joka vähentää epävarmuutta eniten tutkittavassa alipuussa. Valitun ehdon perusteella data jaetaan uusiin alipuihin kunnes:

- Muodostettu alipuu ei sisällä muita luokkia
- Arvioitavia ominaisuuksia ei ole enempää saatavilla
- Asetettu maksimi syvyys saavutetaan

Tästä prosessista käytetään myös nimitystä induktio. Se tarkoittaa prosessin jatkamista kunnes saavutetaan johtopäätös eli luokittelu. Tekstin yhteydessä alipuiden muodostuksessa käytettyjen ehtojen ei tarvitse olla *kyllä*- ja *ei*-tyyppisiä. Tuolloin ne voivat jakaa myös aineiston useampaan kuin kahteen alipuuhun. Tässä tapauksessa sanojen esiintymistiheyksiä voidaan hyödyntää tutkittavana ehtona. Tämän tyyppiset ehdot muodostavat monimutkaisempia rakenteita eli ehtoja. Päätöspuussa luokittelu saadaan kulkemalla puuta ylhäältä alaspäin ja tutkimalla annettuja ehtoja. (Manning & Schütze 1999, ss. 576-578; Gupta, G. 2012, ss. 116-130)

Päätöspuun rakentamiseksi saatavilla on useita menetelmiä, joihin kuuluvat informaation saanti (kaava 14) ja gini-indeksi (kaava 15). Näistä menetelmistä käytetään myös nimitystä ositusmenetelmät, koska niiden perusteella aineisto voidaan jakaa pienempiin alipuihin. (Gupta 2012, s. 130; Manning & Schütze 1999, s. 583) Päätöspuihin liittyy usein myös pituudellinen haaste, koska induktiota jatketaan kunnes joku yllä annetuista ehdoista toteutuu. Tämä ilmiö tunnetaan myös nimellä ylisovitus (engl. *overfitting*). Se voi johtaa huonoon suorituskykyyn ja jopa virheellisiin luokittelupäätöksiin. Syynä ilmiöön on, ettei ominaisuuksien perusteella aineistosta aina voida muodostaa selkeitä ryhmiä. Usein ryhmät pitävät sisällään muutamia poikkeuksia, joiden mallintaminen on turhaa. Tämä johtuu siitä, ettei opetusmateriaali ole useinkaan täydellistä tai se sisältää epävarmuustekijöitä. Tästä syystä tarvitaan käytetyn päätöspuun yksinkertaistamista (engl. *pruning*). Se tarkoittaa joidenkin haarojen poistamista varsinaisesta lopputuloksesta. (Manning & Schütze 1999, s. 582) Puun yksinkertaistaminen perustuu Occam Razorsin periaatteeseen, jonka perusteella yksinkertainen malli toimii todennäköisesti usein paremmin kuin monimutkainen malli (Gupta 2012, s. 129). Muutamia menetelmiä tähän ovat Minimal Cost Complexity Pruning, Pessimistic Error Pruning , Error Based Pruning ja Reduced Error Pruning. Kyseisten menetelmien toiminta perustuu ideaan karsia puuta minimoiden samalla syntyvää virhettä. Puun yksinkertaistaminen tarkoittaa sitä, että osa koulutusaineistosta luokitellaan virheellisesti. Tästä syystä syntyvä virhe halutaan pitää mahdollisimman pienenä. (Maimon & Rokach 2010, ss. 175-178) Tämän tarkemmin kyseisiä menetelmiä työssä ei ole tarkoitus käsitellä. Niistä löytyy kuitenkin kattavasti lisää informaatiota käytetystä lähteestä.

Päätöspuut ovat suosituimpia luokittelun ennustamiseen käytettyjä menetelmiä. Yksi suosiota selittävistä tekijöistä on menetelmän havainnollisuus, sillä muodostunut puu voidaan kirjoittaa loogisina ehtoina. Lisäksi samaa menetelmään on mahdollista hyödyntää myös muilla tiedonlouhinnan osa-alueilla. Kyseessä ei ole pelkästään tekstin

luokitteluun soveltuva menetelmä. Kyseistä algoritmia on yksinkertaista laajentaa lähes kaikkeen koulutusmateriaalin pohjalta tehtävään ennustamiseen. Yksi tunnetuimmista päätöspuualgoritmeista on ID3, jonka pohjalta on kehitetty C4.5 eli CART (Gupta 2012, s. 123; Maimon & Rokach 2010, s. 168).

3.3.2 Päätöspuut käytännössä

Työssä on seuraavaksi tarkoituksena esittää, miten päätöspuu voidaan rakentaa. Annettava esimerkki toteutetaan binäärisille arvoille. On hyvä muistaa, että tekstin luokittelemiseksi todellisuudessa tutkitaan huomattavasti suurempaa määrää avaimia kuin annettavassa esimerkissä. Lisäksi luokittelussa voidaan hyödyntää varsinaisia esiintymistiheyksiä. Työssä binäärinen luokittelu on valittu, koska se on yksinkertaisin tapa esittää päätöspuun toimintaa käytännössä. Annettavassa esimerkissä pyritään havainnollistamaan seuraavaa:

1. Milloin ominaisuus luokittelee dataa hyvin
2. Miten induktio päätöspuun määrittämiseksi tehdään

Tutkittava ongelma on sähköpostiviestien luokittelu. Käytössä on kaksi luokkaa, jotka ovat *Spam* tai *Muu*. Luokittelijalle on annettu 16 kpl esiluokiteltuja sähköpostiviestejä $S=\{S_1...S_{16}\}$, jotka muodostavat koulutusmateriaalin. Kaikkiaan viestit sisältävät yhteensä kymmenen eri sanaa eli ominaisuutta. Käytettyjen sanojen perusteella pitäisi pystyä tekemään päättely, onko kyseessä *Spam* vai *Muu* viesti. Annetussa esimerkissä hyödynnetään ositusmenetelmänä informaation saantia (kaava 14). Kaikki arvioitavaksi valitut ominaisuudet ovat *On* ja *Ei* tyyppisiä eli ne kuuluvat joukkoon *A={On,Ei}.* Kuvitteellinen esimerkki koulutusmateriaalista annetaan taulukossa 11.

Taulukko 11. *Sähköposteissa käytetyt sanat (ensimmäinen alijoukko)*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_1	Ei	Ei	On	On	Ei	On	Ei	On	On	**Muu**
S_2	Ei	Ei	On	On	Ei	Ei	Ei	On	Ei	**Muu**
S_3	Ei	Ei	On	Ei	Ei	On	On	Ei	On	**Muu**
S_4	On	Ei	On	Ei	Ei	Ei	Ei	Ei	Ei	**Muu**
S_5	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_6	Ei	Ei	On	On	On	On	On	Ei	Ei	**Muu**
S_7	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_8	Ei	Ei	On	Ei	On	Ei	Ei	Ei	Ei	**Muu**
S_9	On	Ei	On	Ei	Ei	Ei	Ei	Ei	On	**Spam**
S_{10}	On	Ei	Ei	Ei	Ei	Ei	On	Ei	Ei	**Spam**
S_{11}	On	On	On	Ei	On	On	Ei	Ei	On	**Spam**
S_{12}	Ei	On	On	Ei	Ei	Ei	Ei	Ei	Ei	**Spam**
S_{13}	On	On	On	Ei	Ei	Ei	On	Ei	On	**Spam**
S_{14}	On	On	On	On	Ei	Ei	On	Ei	Ei	**Spam**
S_{15}	On	Ei	On	Ei	Ei	On	On	Ei	On	**Spam**

S_{16}	On	On	On	Ei	Ei	Ei	On	Ei	Ei	**Spam**

Taulukon 11 pohjalta voidaan huomata, että kahdeksan sähköposteista kuuluu luokkaan *Muu* ja loput kahdeksan luokkaan *Spam*. Kaavan 13 mukaan taulukon sisältämä informaatio on:

$$I = -\frac{8}{16}\log_2\left(\frac{8}{16}\right) - \frac{8}{16}\log_2\left(\frac{8}{16}\right) = 1$$

Ensimmäinen osa $-\frac{8}{16}\log_2\left(\frac{8}{16}\right)$ edustaa *Spam* suhdetta kaikkiin posteihin. Jälkimmäinen osa puolestaan edustaa luokan *Muu* sähköpostien osuutta kaikkiin posteihin. Datasta saatava informaatio on tuolloin yksi (1). Tämä johtuu siitä, että molempien luokkien osuus annetusta materiaalista on puolet. Todellisessa aineistossa näin ei välttämättä ole ja luokkien osuudet posteista voivat olla jotakin muuta. Työhön on kuitenkin tarkoituksellisesti valittu tasainen jakauma kahden luokan välille alkutilanteessa. Tuolloin luokkajako myös maksimoi epävarmuuden päätöksen tekemiseksi. Tarkoituksena on sanoa, että luokkien kokoa voidaan myös hyödyntää tehtäessä ositusta. Taulukoissa 12 ja 13 esitetään sanojen puuttuminen ja esiintyminen luokissa *Muu* ja *Spam*. Taulukkojen tehtävänä on tiivistää aineistoa asioiden havainnollistamiseksi.

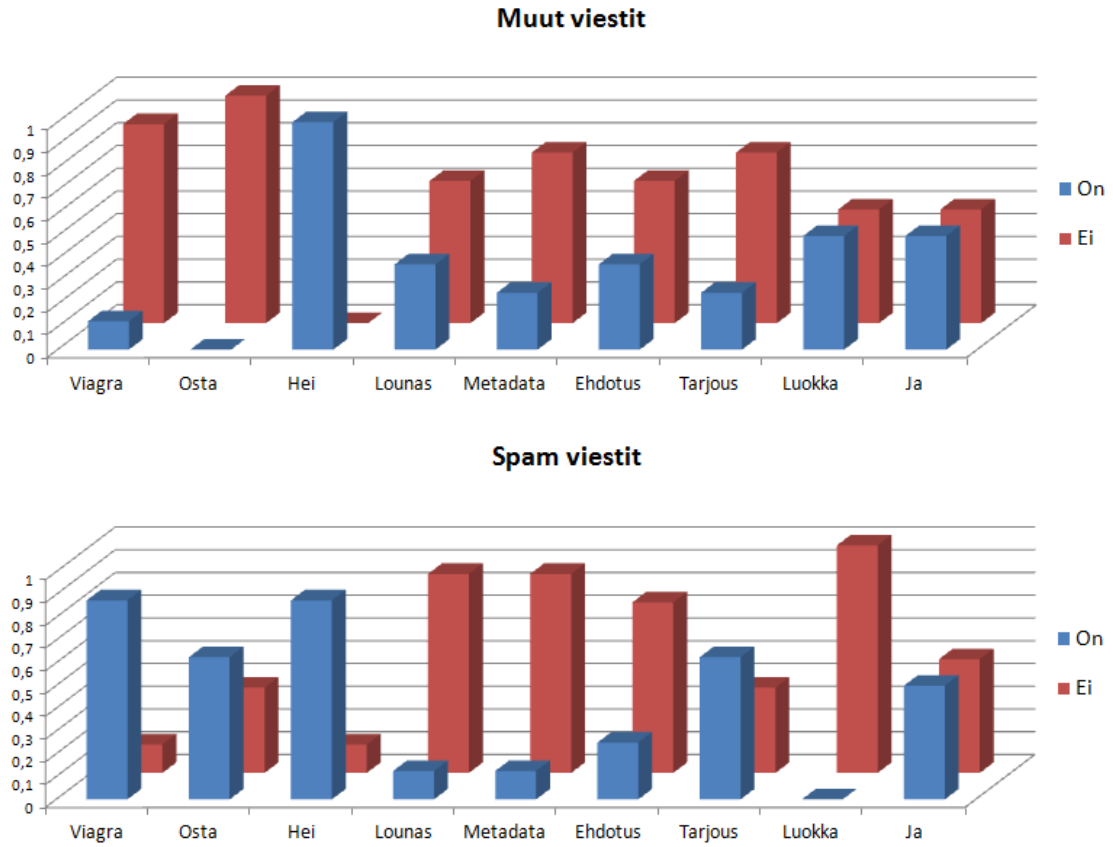
Taulukko 12. *Sanan puuttuminen tai esiintyminen luokassa Muu*

		Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**
Muu	**On**	1	0	8	3	2	3	2	4	4
Muu	**Ei**	7	8	0	5	6	5	6	4	4

Taulukko 13. *Sanan puuttuminen tai esiintyminen luokassa Spam*

		Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**
Spam	**On**	7	5	7	1	1	2	5	0	4
Spam	**Ei**	1	3	1	7	7	6	3	8	4

Taulukkoa 13 voidaan tulkita seuraavasti: Sana *Viagra* esiintyy seitsemässä sähköpostissa, mutta se puuttuu yhdestä sähköpostista. Vastaavasti Taulukossa 13 nähdään sanan *Viagra* löytyvän vain yhdestä luokkaan *Muu* kuuluvasta sähköpostista. Taulukkoihin 12 ja 13 liittyvät kuvaajat esitetään kuvassa 20. Tämän avulla on yksinkertaisempaa tehdä havaintoja luokiteltavasta aineistosta.



Kuva 20. *Sanojen suhteet Muut- ja Spam- luokissa*

Kaavioita tarkastelemalla voidaan havaita seuraavaa:

1. *Viagra* esiintyy usein luokassa *Spam*
2. *Luokka* ei esiinny koskaan luokassa *Spam*
3. *Viagra* esiintyy harvoin luokassa *Muut*
4. *Hei* esiintyy aina luokassa *Muut*
5. *Ja* sana esiintyy tasaisesti molemmissa luokissa

Johtopäätöksien tekemiseksi aineistosta tulee määrittää kaikille sanoista informaatio ehdoilla *On* ja *Ei* (Kaava 13). Nämä arvot annetaan taulukossa 14.

Taulukko 14. *Attribuuttien informaatio parametreilla On ja Ei*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**
On	0,54	0,00	1,00	0,81	0,92	0,97	0,86	0,00	1,00
Ei	0,54	0,85	0,00	0,98	1,00	0,99	0,92	0,92	1,00

Taulukossa 14 sanalle *Viagra* ehdolla (*On*), informaatio on määritetty seuraavasti. Tiedetään, että sana on löydettävissä seitsemästä luokkaan *Spam* kuuluvasta sähköpostista ja yhdestä muusta sähköpostista. Kokonaisuudessaan se on löydettävissä yhteensä kahdeksasta sähköpostiviestistä. Nämä arvot on katsottu taulukoista 12 ja 13.

$$I(On) = -\left(\frac{7}{8}\right)\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

Vastaava arvo voidaan laskea sanan *Viagra* puuttumiselle seuraavasti:

$$I(Ei) = -\left(\frac{1}{8}\right)\log_2\left(\frac{1}{8}\right) - \frac{7}{8}\log_2\left(\frac{7}{8}\right) \approx 0.54$$

Taulukossa 14 esitetyt arvot on määritelty edellä kuvatulla tavalla. Sanan sisältämä informaatio voidaan määrittää laskemalla sen *Ei* ja *On* -tilanteissa tarjoaman informaation avulla. Näitä lukuja painotetaan suhteilla, jossa sana esiintyy kummassakin luokista. Tuolloin avaimen *Viagra* kahden alipuun sisältämä informaatio on:

$$\mathrm{I(Viagra)} = (\tfrac{7+1}{16} * 0{,}54) + (\tfrac{1+7}{16} * 0{,}54) \approx 0{,}54$$

Annetussa lausekkeessa ensimmäinen osa $\frac{7+1}{16}$*0,54 tulee siitä, että sana löytyy seitsemästä *Spam* ja yhdestä *Muu* -luokan viestistä. Kaikkiaan ensimmäisessä alipuussa on 16 viestiä ennen ositusta (taulukko 11). Tämän jälkeen se on kerrottu (On) tuottamalla informaatiolla. Seuraavassa osassa laskua vastaava arvo määritellään sanan *Viagra* puuttumiselle. Informaation saannin määrittämiseksi sanan tuoma informaatio vähennetään taulukon 11 informaatiosta (kaava 14). Informaation saanti sanalle *Viagra* on näin ollen:

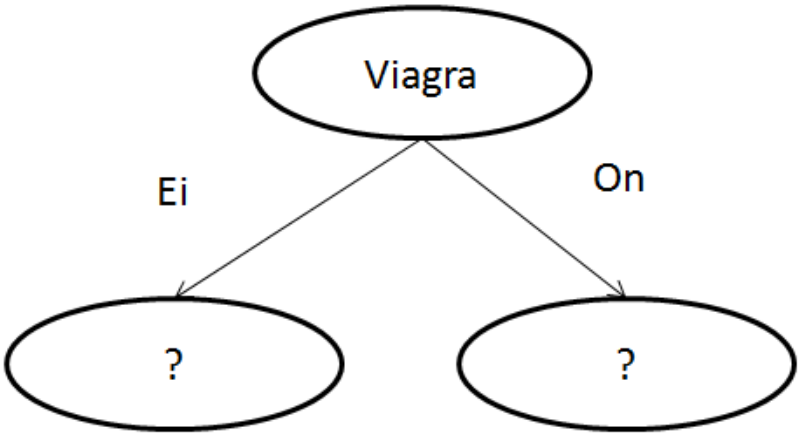
$$\mathrm{Gain(Viagra)} = 1 - 0{,}54 = 0{,}46$$

Kaikille avaimille saadut tulokset esitetään taulukossa 15.

Taulukko 15. *Informaation saanti*

Mahdolliset alipuut	**Informaatio ennen jakoa**	**Informaatio jaon jälkeen**	**Informaation saanti**
Viagra	1,00	0,54	0,46
Osta	1,00	0,58	0,42
Hei	1,00	0,93	0,07
Lounas	1,00	0,94	0,06
Metadata	1,00	0,98	0,02
Ehdotus	1,00	0,99	0,01
Tarjous	1,00	0,89	0,11
Luokka	1,00	0,69	0,31
Ja	1,00	1,00	0,00

Taulukon 15 pohjalta voidaan huomata, että *Viagra* vähentää epävarmuutta kaikista eniten. Tämä johtuu siitä, että sen informaation saanti on suurin. Tuolloin se valitaan jakoperusteeksi. Näin ollen voidaan muodostaa kaksi uutta alipuuta (kuva 21)



Kuva 21 Ositus Viagran perusteella

Induktion seuraavassa vaiheessa tulee tutkia kumpikin ehdoista *Ei* ja *On* muodostamalla uudet alipuut. Tämä tapahtuu eliminoimalla sana *Viagra* eli käyttämällä sanaa jakoperusteena. Tässä vaiheessa tutkitaan myös sisältääkö kumpikaan alijoukoista vain yhtä luokkaa, jos näin on, niin ositusta ei jatketa. Taulukossa 16 esitetään ne viesteistä, joissa *Viagra* on. Taulukossa 17 esitetään ne viestit, joissa kyseinen sana ei esiinny.

Taulukko 16. *Alipuu posteista, joista sana Viagra löytyy*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_4	On	Ei	On	Ei	Ei	Ei	Ei	Ei	Ei	**Muu**
S_9	On	Ei	On	Ei	Ei	Ei	Ei	Ei	On	**Spam**
S_{10}	On	Ei	Ei	Ei	Ei	Ei	On	Ei	Ei	**Spam**
S_{11}	On	On	On	Ei	On	On	Ei	Ei	On	**Spam**
S_{13}	On	On	On	Ei	Ei	Ei	On	Ei	On	**Spam**
S_{14}	On	On	On	On	Ei	Ei	On	Ei	Ei	**Spam**
S_{15}	On	Ei	On	Ei	Ei	On	On	Ei	On	**Spam**
S_{16}	On	On	On	Ei	Ei	Ei	On	Ei	Ei	**Spam**

Taulukko 17. *Alipuu posteista, joista sanaa Viagra ei löydy*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_1	Ei	Ei	On	On	Ei	On	Ei	On	On	**Muu**
S_2	Ei	Ei	On	On	Ei	Ei	Ei	On	Ei	**Muu**
S_3	Ei	Ei	On	Ei	Ei	On	On	Ei	On	**Muu**
S_5	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_6	Ei	Ei	On	On	On	On	On	Ei	Ei	**Muu**
S_7	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_8	Ei	Ei	On	Ei	On	Ei	Ei	Ei	Ei	**Muu**
S_{12}	Ei	On	On	Ei	Ei	Ei	Ei	Ei	Ei	**Spam**

Taulukkojen perusteella voidaan huomata, että kummastakin alipuusta löytyy kahdeksan postia. Lisäksi kumpikaan alipuista ei ole yksiselitteinen luokan suhteen. Tästä syystä induktiota tulee jatkaa molempien alipuiden osalta. Tämän perusteella voidaan tehdä myös uudet johtopäätökset odotusarvoista molemmissa alipuissa. Taulukko 16

sisältää vain yhden luokan *Muu* viestin ja taulukko 17 sisältää yhden luokan *Spam* viestin. Odotusarvo sähköpostin kuulumiseen luokkaan *Spam* sanan Viagra esiintyessä on näin ollen $\frac{7}{8}$. Sanan puuttuessa odotusarvo luokan *Spam* jäsenyydelle on $\frac{1}{8}$. Vastaavat odotusarvot ovat luokalle *Muu*, kun *Viagra* puuttuu $\frac{7}{8}$ ja sanan löytyessä $\frac{1}{8}$. Koska kumpikaan alijoukoista ei ole yksiselitteinen, ositusta tulee jatkaa molempien alipuiden osalta. Työssä seuraavaksi ositusta alipuussa (Ei). Tästä alipuusta päätökset saadaan seuraavassa osituksessa. Alipuulle *Viagra* (Ei) tulee laskea sen sisältämä informaatio. Tämä tapahtuu hyödyntämällä taulukkoa 17.

$$I = -\frac{1}{8}\log_2\left(\frac{1}{8}\right) - \frac{7}{8}\log_2\left(\frac{7}{8}\right) \approx 0{,}5$$

Yhtälön arvot tulevat siitä, että alipuussa on yksi *Spam* ja seitsemän luokkaan *Muut* kuuluvaa sähköpostia. Taulukon 17 pohjalta laaditut sanojen esiintymiset tai puuttumiset luokittain esitetään taulukoissa 18 ja 19.

Taulukko 18. *Luokan Spam jäsenten sanat ehdolla, että Viagra puuttuu*

		Osta	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**
Spam	**On**	1	1	0	0	0	0	0	0
Spam	**Ei**	0	0	1	1	1	1	1	1

Taulukko 19. *Luokan Muu jäsenten sanat ehdolla, että Viagra puuttuu*

		Osta	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**
Muu	**On**	0	7	3	2	3	2	4	4
Muu	**Ei**	7	0	4	5	4	5	3	3

Taulukoissa 18 ja 19 on nähtävissä, että sana *Osta* vaikuttaa luokkajakoon voimakkaasti. Tämä johtuu siitä, että se esiintyy ainoastaan luokan *Spam* viesteissä (taulukko 17). Se ei sisälly kertaakaan viestiin, joka kuuluisi luokkaan *Muut*. Avaimen *Osta* informaatio määritellään ehdoilla *On* ja *Ei* seuraavasti:

$$I(On) = -\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \frac{0}{8}\log_2\left(\frac{0}{8}\right) = 0$$
$$I(Ei) = -\left(\frac{0}{7}\right)\log_2\left(\frac{0}{7}\right) - \frac{7}{7}\log_2\left(\frac{7}{7}\right) = 0$$

Kahteen muodostettavaan alipuuhun jäävä informaatio on näin ollen:

$$\frac{1}{8}I(On) + \frac{7}{8}I(Ei) = 0$$

Taulukossa 20 esitetään kaikkien attribuuttien tuoma informaatio:

Taulukko 20. *Alipuun Ei Viagra jäljellä olevien sanojen informaatio*

Mahdolliset alipuut	**Informaatio ennen jako**	**Informaatio jaon jälkeen**	**Informaation saanti**
Osta	0,5	0,00	0,50
Hei	0,5	0,54	-0,04
Lounas	0,5	0,45	0,05
Metadata	0,5	0,49	0,02
Ehdotus	0,5	0,45	0,05
Tarjous	0,5	0,49	0,02
Luokka	0,5	0,41	0,10
Ja	0,5	0,41	0,10

Taulukon 20 pohjalta *Osta* valitaan jakavaksi tekijäksi. Seuraavaksi ominaisuudesta *Osta* muodostetaan kaksi uutta alipuuta *On* ja *Ei*. Niiden perusteella on helppoa huomata, ettei alipuita tarvitse jakaa uudelleen luokan määrittämiseksi.

Taulukko 21. *Alipuu, jossa ei esiinny sana Viagra tai Osta*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_1	Ei	Ei	On	On	Ei	On	Ei	On	On	**Muu**
S_2	Ei	Ei	On	On	Ei	Ei	Ei	On	Ei	**Muu**
S_3	Ei	Ei	On	Ei	Ei	On	On	Ei	On	**Muu**
S_5	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_6	Ei	Ei	On	On	On	On	On	Ei	Ei	**Muu**
S_7	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_8	Ei	Ei	On	Ei	On	Ei	Ei	Ei	Ei	**Muu**

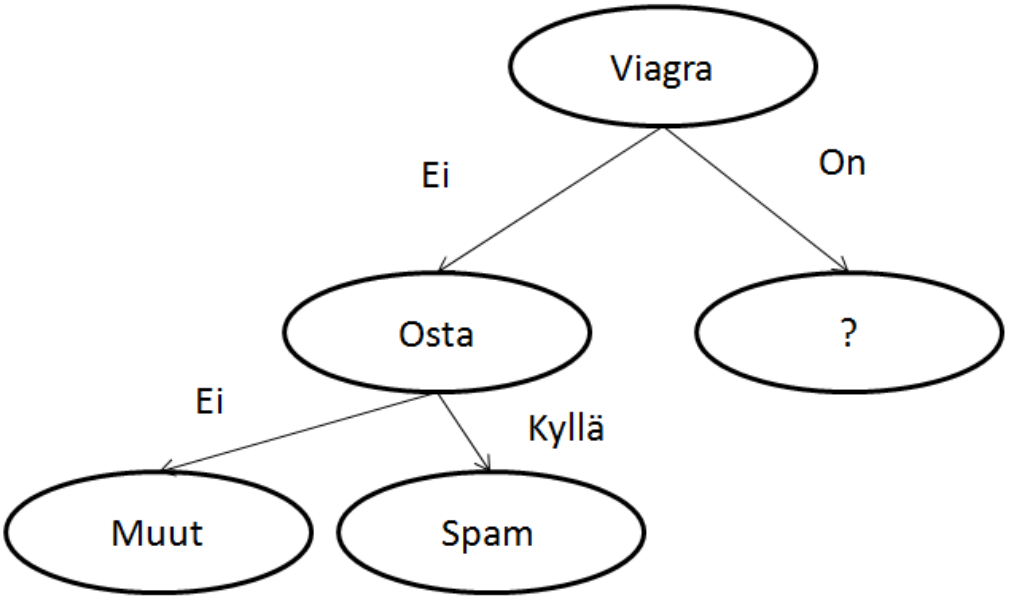
Taulukko 22. *Alipuu sisältäen sanan Osta, muttei sanaa Viagra*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_{12}	Ei	On	On	Ei	Ei	Ei	Ei	Ei	Ei	**Spam**

Tämän perusteella saadaan seuraavat loogiset lauseet:

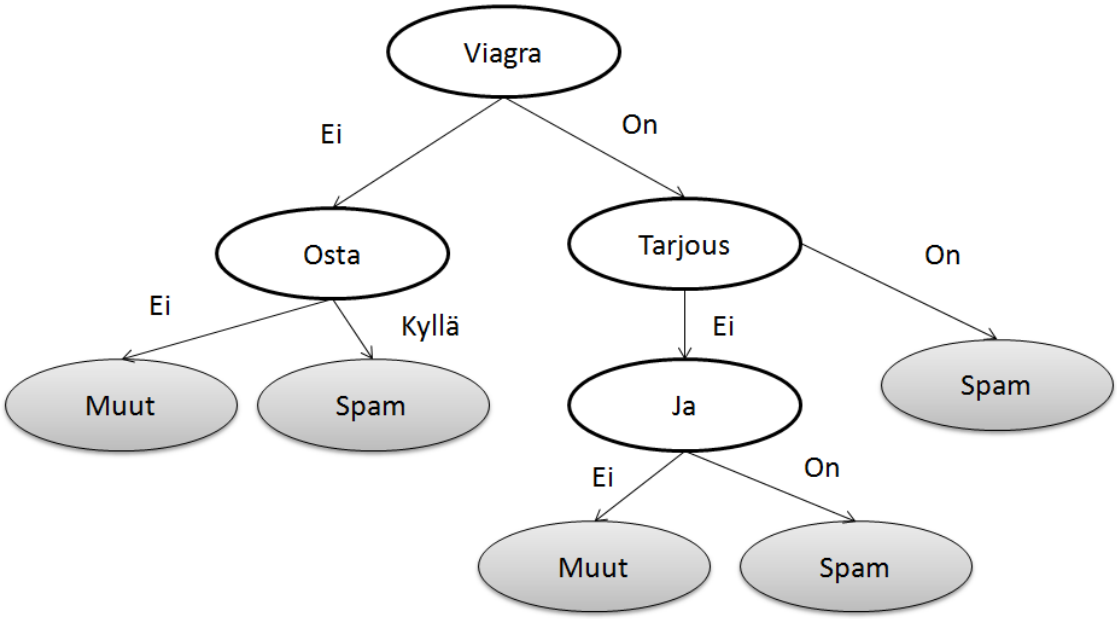
1. Jos Viagra ei ja Osta on niin Spam
2. Jos Viagra ei ja Osta ei niin Muut

Tämä tilanne on esitetty kuvassa 22.



Kuva 22. *Ositus Viagran ja Osta perusteella*

Työssä esimerkkiä ei jatketa tarkoituksellisesti *Viagra* sanan haaran *(On*) osalta. Varsinainen lopputulos on kuitenkin esitetty kuvassa 23.



Kuva 23. *Lopullinen päätöspuu taulukon 11 aineistolle*

Testaamalla saatua puuta (kuva 23) voidaan huomata sen luokittelevan alkuperäisen taulukon aineiston täysin oikein.

Lisää päätöspuihin liittyviä esimerkkejä lähteenä käytetystä kirjasta Introduction to Data Mining with Case Studies. Kirjasta löytyy esimerkki gini-indeksin hyödyntämisestä sivuilta 123-128. Gini-indeksin hyödyntämistä työn yhteydessä ei ole nähty tarpeelliseksi esitellä.

3.3.3 Bayesialainen päättely

Bayesialainen päättely (engl. bayesian) on yksi suosituimmista kirjallisuudessa esitellyistä luokittelumenetelmistä. Se perustuu Thomas Baysin (1702-1761) kehittämään teoriaan hypoteesin testaamisesta. Menetelmän alkuperäistä tarkoituksesta kirjallisuus esittää vain lähinnä arvauksia. (Gupta 2012, ss. 131) Kyseistä menetelmää on mahdollista hyödyntää myös tekstin luokittelussa. Tuolloin puhutaan naiivista Bayesian päättelystä (engl. *naive bayesian*) . Teorian osalta työssä pyritään johdattelemaan lukijaa ajatuksiin, mitä naiivi Bayesilainen päättely tarkoittaa ja mistä se on peräisin. Varsinainen Bayesian teoria on esitetty kaavassa 23.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{23}$$

Annettua kaavaa 23 voidaan hyödyntää arvioitaessa, mihin luokkaan jokin dokumentti mahdollisesti kuuluu. Tuolloin testataan kutakin luokkaa hypoteesina ja valitaan paras hypoteesi. Kysymys on mallin rakentamisesta jokaiselle testattavista hypoteeseista eli luokista. Kaavassa 23 P(c|d) tarkoittaa luokan (c) todennäköisyyttä dokumentille (d). Tästä käytetään myös nimitystä posteriori todennäköisyys. Merkinnällä P(c) tarkoitetaan luokan todennäköisyyttä ja P(d) dokumentin todennäköisyyttä. Molemmista todennäköisyyksistä käytetään nimitystä priorit. (Manning et al. 2009, s. 265) Naiiviudella tekstin luokittelussa tarkoitetaan perusoletusta, jonka mukaan kaikki arvioivat ominaisuudet eli sanat ovat riippumattomia toisistaan. Tämä on syynä, miksi menetelmää kutsutaan naiiviksi Bayesian päättelyksi. Menetelmän toisen keskeisen oletuksen mukaan sanojen sijainnilla ei ole merkitystä. Todellisuudessa kumpikaan näistä oletuksista ei päde. Näistä rajoituksista huolimatta, Bayesian päättelyn on usein todettu toimivan varsin hyvin. (Manning et al. 2009, s. 266) Menetelmä on yksi suosituimmista algoritmeista tekstin luokittelussa. Tähän syynä on menetelmän yksinkertaisuus. Tekstin luokittelussa käytetyt Bayesian menetelmät jaetaan yleensä kirjallisuudessa kahteen eri malliin, jotka ovat Bernoullin malli (engl. *multivariate bernoulli model*) ja Multinomialimalli (engl. *multinomial model*). (Zhai 2012, ss. 182) Termeille ei työn yhteydessä löytynyt virallista suomennosta, joten niihin viitataan nimillä Bernoullin malli ja Multinomialimalli. Menetelmien keskeisin ero liittyy dokumenttivektorin esitykseen. Bernoullin mallissa dokumentissa esiintyvät sanat esitetään binäärivektorina ja Multinomialimallissa otetaan huomioon myös sanojen esiintymistiheys. Mallit jakavat suuren osan perusoletuksista. Lyhyesti sanottuna tekstin luokittelussa pyritään löytämään se hypoteesi, joka maksimoi $P(c|d)$ arvon. (Manning et al. 2009, ss. 265) Tämä tarkoittaa samalla sitä, että vertaamme testattavaa hypoteesia kaikkiin olemassa oleviin malleihin luokista. Tämä voidaan lausua kaavan 24 avulla.

$$C_{MAP} = \max_{c \epsilon C} P(c|d) \tag{24}$$

Bayesian teorian mukaan kaava 24 voidaan lausua toisessa muodossa seuraavasti (kaava 25).

$$C_{MAP} = \max_{c \epsilon C} \frac{P(d|c)P(c)}{P(d)} \quad (25)$$

Arvioinnissa käytetään aina samaa dokumenttia, jolloin varsinaisesti dokumentin prioria P(d) ei ole tapana huomioida. Tämän perusteella voidaan tehdä seuraava oletus $P(c|d) \propto P(d|c)P(c)$. Se tarkoittaa, että dokumentin todennäköisyyden oletetaan olevan vakio. Tämä perustuu siihen, että käytetty jakaja on kaikille arvioitaville hypoteeseille sama. Tuolloin isoimpaan arvoon liittyvä vaatimus ei kuitenkaan muutu, vaikka dokumentin prioria ei huomioida lainkaan. Dokumentin priorin oletetaan olevan jokin mielivaltainen vakio (k), joka on kaikille arvioitaville luokille sama. Tätä merkitään yhtälössä yleisesti $\propto$ symbolilla. Olkoon se yksi, mikä ei muuta alkuperäistä oletusta. Tuolloin naiivi Bayesian luokittelu voidaan kirjoittaa kaavan 26 muodossa.

$$C_{NB} = \max_{c \epsilon C} P(d|c)P(c) \quad (26)$$

Varsinaisesti dokumentti (d) koostuu joukosta sanoja eli testattavia ominaisuuksia (w_1,..,w_k). Kaava 26 tulee kirjoittaa sellaisessa muodossa, että se huomioi dokumenttivektoriin liittyvät ominaisuudet eli avaimet. Lähteenä käytetyssä kirjallisuudessa avaimista käytetään yleisesti merkintää (w), joten työssä ei ole katsottu tarpeelliseksi muuttaa kaavoja käyttämään aikaisemmin käytettyä merkintää (t). Kaava 26 voidaan kirjoittaa vektorimuodossa kaavan 27 mukaan:

$$C_{NB} = \max_{c \epsilon C} P(w_1, w_2, \ldots, w_k|c)P(c) \quad (27)$$

Kaava 27 pitää sisällään useita testattavia ominaisuuksia eli avaimia. Tuolloin hypoteesin vahvuus määritellään kertomalla kaikkien todennäköisyydet keskenään. Arvoa painotetaan luokan odotusarvolla. Yksittäiselle testattavalle hypoteesille malli voidaan kirjoittaa kaavan 28 muodossa.

$$P(c|d) \propto \acute{P}(c) \prod_{1 \leq k \leq n_d} \acute{P}(w_k|c) \quad (28)$$

Koska varsinaisia todennäköisyyksiä ei tunneta, käytetään opetusmateriaalista saatuja odotusarvoja. Merkinnällä $\acute{P}$ tarkoitetaan aineiston pohjalta esitettyä arviota todennäköisyydestä. Yleinen tapa on hyödyntää MLE:tä (engl. *maximum likelihood estimates*). Asiaa käsitellään tarkemmin kaavojen 30 ja 31 yhteydessä. Kaava 28 on yksittäiselle luokalle laskettu malli. Varsinainen päätös luokasta saadaan kaavalla 29. Tuolloin valitaan vahvin hypoteesi.

$$C_{Map} = \max_{c \epsilon C} \acute{P}(c) \prod_{1 \leq k \leq n_d} \acute{P}(w_k|c) \quad (29)$$

(Manning et al. 2009, ss. 258-266). Bernoullin mallissa käsitellään myös sanojen puuttumiset. Tämä tulee ottaa huomioon käytettäessä kaavaa 29. Multinomialimallissa jokainen kohdattu todiste eli avain käsitellään uutena todisteena luokan puolesta. Tämä tarkoittaa, että sama todiste voi esiintyä useampaan kertaan tutkittavien todisteiden joukossa. Lisäksi huomattavaa on, että tietokoneille erittäin pienien lukujen esittäminen voi olla ongelmallista. Tästä syystä kaava 29 esitetään joskus kaavan 30 muodossa.

$$C_{Map} = \max_{c \epsilon C} \left[\log \acute{P}(c) + \sum_{1 \leq k \leq n_d} \acute{P}(w_k|c)\right] \quad (30)$$

Se on johdettu kaavan 30 muotoon hyödyntäen tietämystä, että $\log(xy) = \log(x) + \log(y)$. Tuolloin kertolaskut voidaan korvata yhteenlaskulla. Edelleen korkein logaritminen arvo on todennäköisin malli eli muutos ei muuta varsinaisesti kaavan 29 ajatusta (Manning et al. 2009, ss. 258).

Maximum likelihood estimates

Naiiviin Bayesian päättelyyn käytetään MLE-arvoja. Nämä arvot ovat koulutusmateriaalista kerättyjä odotusarvoja. Tässä mielessä kirjallisuudessa käytetty nimitys todennäköisyys (engl. *probability*) on harhaanjohtava. Kysymys on enemmänkin odotusarvosta. Työssä MLE-arvojen valintaa käsitellään Bernoullinmallin yhteydessä. Tämän jälkeen selvitetään muutoksia, jotka liittyvät Multinomialimallin hienosäätöihin. MLE:n avulla luokan P(Cj) odotusarvo voidaan määritellä kaavan 31 mukaisesti.

$$P(c_j) = \frac{N_c}{N} \quad (31)$$

Kaavassa 31 (Nc) on kaikkien dokumenttien määrä, jotka kuuluvat luokkaan (cj). Merkinnällä (N) tarkoitetaan kaikkien dokumenttien määrä, jotka kuuluvat koulutusmateriaaliin. Kaavan 31 mukaan arvioidaan vain luokkien priorit. Kaavaa 31 ei kuitenkaan voida käyttää avainten arviointiin. Tämä johtuu mahdollisuudesta, ettei avainta esiinny lainkaan tutkittavassa dokumentissa. Asian huomioiminen on tärkeää, koska mikä tahansa luku kerrottuna nollalla on nolla (kaava 28 ja 29). Toisin sanoen emme halua väittää, etteikö kohde voi olla luokan jäsen ominaisuuden eli avaimen puuttuessa. Tekstin luokittelussa kyseinen tilanne on hyvin todennäköinen. Se voidaan ratkaista estämällä nolla-arvot. Tuolloin yhtälössä käytetään *smoothing*-tekniikkaa. Tämä on esitetty kaavassa 32.

$$P(W|C_j) = \frac{W_{cw}+1}{(\sum_{w\prime \in W} W_{cw\prime})+B'} \quad (32)$$

Kaavassa 32 on kirjallisuudessa esitetty *laplace smoothing*-tekniikka. Kaavassa merkinnällä (W_{cw}) tarkoitetaan sitä, montako kertaa avain (w) esiintyy luokassa (c). Arvoon lisätään aina pehmennys eli +1, jolla estetään MLE:n nolla arvot. Tämän lisäksi yhtälöön on tapana lisätä (B'). Se on kaikkien uniikkien sanojen määrä korpuksessa eli sarakkeiden lukumäärä dokumenttitermimatriisissa. Sen avulla estämme myös arvon ole-

masta yksi. Arvo yksi on ongelmallinen erityisesti Bernoullin mallissa, koska myös puuttuvat sanat arvioidaan. Asiaa voidaan perustella Bernoullin periaatteella käänteisten arvojen määrittämisellä, joka esitetään kaavassa 33. Puolestaan $\sum_{w\prime\in W} W_{cw\prime}$ on luokkaan (c) kuuluvien dokumenttien sanojen määrä. Esitellyn *laplace smoothing* tekniikan lisäksi on olemassa myös useita muitakin vaihtoehtoja ongelman ratkaisemiseksi. *Smoothing*-tekniikoista löytyy lisää informaatiota Chen & Goodman artikkelista. (Manning et al. 2009, ss. 259-260; Chen & Goodman 1996) Sanan puuttumiseen luokan jäsenestä sovelletaan yleisesti Bernoullin periaatetta.

$$\acute{P}(\overline{w}|c) = 1 - \acute{P}(w_k|c) \tag{33}$$

(Weisstein 2013) . Tästä syystä haluamme myös estää muuttujan arvioidun MLE:n $\acute{P}(w_k|c)$ arvon yksi, koska se johtaisi arvoon nolla. Se on myös perustelu, miksi arvo ei saa olla yksi.

Multinomialimalli eroaa Bernoullin mallista siinä, ettei se huomioi sanojen puuttumisia. Lisäksi se ottaa huomioon myös sanojen esiintymiset useampaan kertaan saman dokumentin sisällä. Näin ollen kaavaa 33 ei sovelleta Multinomialimallissa. Mallin käyttäminen edellyttää myös muutamia muutoksia kaavan 28, 29 ja 30 osalta. Kaavassa 32 (W_{cw}) tulee ottaa huomioon myös sanojen useat esiintymiset dokumenteissa. Tuolloin arvioidaan, montako kertaa kyseinen sana kuuluu luokkaan (c). Lisäksi kaavassa 32 käytetyn $\sum_{w\prime\in W} W_{cw\prime}$ arvon tulee huomioida sanojen useat esiintymiset luokan dokumenteissa. Se lasketaan summaamalla luokan kaikkiin dokumentteihin liittyvät sanat yhteen (Manning et al. 2009, ss. 262-270). Kaavan 28 osalta merkittävin muutos on, ettei avaimen puuttumista huomioida. Lisäksi kaavassa 28 tulee huomioida, kuinka usein etsittävä ominaisuus esiintyy arviotavassa kohteessa. Tuolloin kaava 28 voidaan kirjoittaa kaavan 34 muodossa.

$$C_{Map} = \max_{c\epsilon C} \acute{P}(c) \prod_{1\le k\le n_d} [\acute{P}(w_k|c)]^X \tag{34}$$

Kaavassa 34 (x) tarkoittaa sitä, montako kertaa tutkittava kohde sisältää ominaisuuden eli avaimen w_k. (Manning et al. 2009, s. 270, 243).

3.3.4 Bayesialainen päättely käytännössä

Tavoitteena on esittää lukijalle, miten Bayesian teoriaa voidaan hyödyntää dokumentin luokittelussa. Tarkoituksena on esittää, miten Bernoullin- ja Multinomialimallia voidaan hyödyntää tekstin luokitteluun.

Aloitetaan tarkastelemalla Bernoullin mallia (engl. *multivariate bernoulli model)* ja sitä, miten sen avulla luokittelu voidaan tehdä. Täydennetään esimerkkiä uudella kuvitteellisella tekstillä, joka halutaan luokitella. Tämän tekstin sisältö on "*Tarjous Osta Viagra*". Taulukossa 23 on esitetty kahteen luokkaan liittyvien sanojen esiintymiset dokumenteittain. Lisäksi taulukossa esitetään myös luokiteltava teksti viimeisellä rivillä.

Taulukko 23. *Koulutusmateriaali ja luokiteltava sähköposti*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_1	Ei	Ei	On	On	Ei	On	Ei	On	On	**Muu**
S_2	Ei	Ei	On	On	Ei	Ei	Ei	On	Ei	**Muu**
S_3	Ei	Ei	On	Ei	Ei	On	On	Ei	On	**Muu**
S_4	On	Ei	On	Ei	Ei	Ei	Ei	Ei	Ei	**Muu**
S_5	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_6	Ei	Ei	On	On	On	On	On	Ei	Ei	**Muu**
S_7	Ei	Ei	On	Ei	Ei	Ei	Ei	On	On	**Muu**
S_8	Ei	Ei	On	Ei	On	Ei	Ei	Ei	Ei	**Muu**
S_9	On	Ei	On	Ei	Ei	Ei	Ei	Ei	On	**Spam**
S_{10}	On	Ei	Ei	Ei	Ei	Ei	On	Ei	Ei	**Spam**
S_{11}	On	On	On	Ei	On	On	Ei	Ei	On	**Spam**
S_{12}	Ei	On	On	Ei	Ei	Ei	Ei	Ei	Ei	**Spam**
S_{13}	On	On	On	Ei	Ei	Ei	On	Ei	On	**Spam**
S_{14}	On	On	On	On	Ei	Ei	On	Ei	Ei	**Spam**
S_{15}	On	Ei	On	Ei	Ei	On	On	Ei	On	**Spam**
S_{16}	On	On	On	Ei	Ei	Ei	On	Ei	Ei	**Spam**
Testi	On	On	Ei	Ei	Ei	Ei	On	Ei	Ei	**?**

Taulukon 23 pohjalta voidaan määrittää kummankin luokan todennäköisyys MLE-menetelmällä. Tämä tapahtuu kaavan 31 mukaan jakamalla luokkaan kuuluvien dokumenttien määrä kaikkien dokumenttien määrällä. Arvo tulee määrittää molemmille arvioitavista hypoteeseista, jotka ovat dokumentti kuuluu luokkaan *Spam* tai *Muu*. Tuolloin priorit taulukossa 23 esitetyille luokille ovat:

$$P(Spam) = \frac{8}{16} = 0{,}5$$

$$P(Muu) = \frac{8}{16} = 0{,}5$$

MLE-arvojen määrittämiseksi jokaiselle sanoista tarvitaan sanojen esiintymiset luokittain. Taulukon 23 pohjalta laadittu tiiviste sanojen esiintymisistä luokittain annetaan taulukossa 24.

Taulukko 24. *Monessako dokumentista sana voidaan löytää luokittain*

	Viagra (w_1)	**Osta (w_2)**	**Hei (w_3)**	**Lounas (w_4)**	**Metadata (w_5)**	**Ehdotus (w_6)**	**Tarjous (w_7)**	**Luokka (w_8)**	**Ja (w_9)**
Spam	7	5	7	1	1	2	5	0	4
Muu	1	0	8	3	2	3	2	4	4

Taulukon 24 pohjalta voidaan tehdä seuraavat johtopäätökset:

1. Yhteensä korpukseen kuuluu 9 kpl sanoja ($w_{1\text{-}9}$) eli B' = 9

2. Luokkaan *Muu* kuuluvissa dokumenteissa esiintyy sanat *Viagra*, *Hei*, *Lounas*, *Metadata*, *Ehdotus*, *Tarjous*, *Luokka* ja *Ja*. Tuolloin kyseiseen luokkaan kuuluvien sanojen määrä on kahdeksan eli $\sum_{w' \in W} W_{Muuw'} = 8$
3. Luokkaa *Spam* sisältää sanat *Viagra*, Osta, *Hei*, *Lounas*, *Metadata*, *Ehdotus*, *Tarjous*, ja *Ja*. Tuolloin kyseiseen luokkaan kuuluvien sanojen määrä on myös kahdeksan eli $\sum_{w' \in W} W_{Spamw'} = 8$

Taulukossa 25 annetaan MLE-arvot, jotka on määritetty kaavan 32 mukaan. Taulukossa esitetään myös ominaisuuden puuttuminen sarakkeessa $1 - \acute{P}(w_k|c)$ (kaava 33). Lisäksi taulukossa on nähtävillä myös logaritmiset arvot kaavan 30 todentamiseksi.

Taulukko 25. *Aineistosta saadut arviot ehdollisista todennäköisyyksistä*

Sana (w)	**Luokka (c)**	$\acute{P}(\boldsymbol{w}\|\boldsymbol{c})$	$\log_2(\acute{P}(\boldsymbol{w_k}\|\boldsymbol{c}))$	$1 - \acute{P}(\boldsymbol{w_k}\|\boldsymbol{c})$	$\log_2(1 - \acute{P}(\boldsymbol{w_k}\|\boldsymbol{c}))$
Viagra	Spam	0,47	-1,09	0,53	-0,92
Osta	Spam	0,35	-1,50	0,65	-0,63
Hei	Spam	0,47	-1,09	0,53	-0,92
Lounas	Spam	0,12	-3,09	0,88	-0,18
Metadata	Spam	0,12	-3,09	0,88	-0,18
Ehdotus	Spam	0,18	-2,50	0,82	-0,28
Tarjous	Spam	0,35	-1,50	0,65	-0,63
Luokka	Spam	0,06	-4,09	0,94	-0,09
Ja	Spam	0,29	-1,77	0,71	-0,50
Viagra	Muu	0,12	-3,09	0,88	-0,18
Osta	Muu	0,06	-4,09	0,94	-0,09
Hei	Muu	0,53	-0,92	0,47	-1,09
Lounas	Muu	0,24	-2,09	0,76	-0,39
Metadata	Muu	0,18	-2,50	0,82	-0,28
Ehdotus	Muu	0,24	-2,09	0,76	-0,39
Tarjous	Muu	0,18	-2,50	0,82	-0,28
Luokka	Muu	0,29	-1,77	0,71	-0,50
Ja	Muu	0,29	-1,77	0,71	-0,50

Esimerkinomaisesti $\acute{P}(Viagra|Spam)$ eli sanan *Viagra* kohtaaminen luokassa *Spam* on määritetty seuraavasti:

$$\acute{P}(Viagra|Spam) = \frac{7+1}{8+9} \approx 0{,}47$$

Arvo tulee siitä, että *Viagra* on löydettävissä seitsemästä dokumentista, jotka kuuluvat luokkaan *Spam*. Annettu +1 on pehmennys (engl. *smoothing*). *Spam* luokka sisältää yhteensä kahdeksan eri sanaa ja koko korpukseen kuuluu yhdeksän eri sanaa. Todennäköisyys, ettei sanaa kohdata luokan dokumentissa on määritetty kaavan 33 mukaisesti.

$$1 - 0{,}47 = 0{,}53.$$

Loput arvoista on määritelty vastaavalla tavalla kuten taulukossa 24. Bernoullin mallissa huomioidaan sanan puuttuminen ja löytyminen luokasta. Tuolloin hypoteesi sille, että dokumentti *Testi* kuuluu luokkaan *Spam*, kirjoitetaan seuraavasti:

$$P(Testi|Spam) \propto \text{P}(Spam) * \acute{\text{P}}(Viagra|Spam) * \acute{\text{P}}(Osta|Spam) * \acute{\text{P}}(Tarjous|Spam) * (1 - \acute{\text{P}}(Hei|Spam)) * (1 - \acute{\text{P}}(Lounas|Spam)) * (1 - \acute{\text{P}}(Metadata|Spam)) * (1 - \acute{\text{P}}(Ehdotus|Spam)) * (1 - \acute{\text{P}}(Luokka|Spam)) * (1 - \acute{\text{P}}(Ja|Spam))$$

Lopulliset arvot voidaan laskea kertomalla priori avaimiin liittyvillä odotusarvoilla. Hypoteesiin liittyvillä odotusarvoilla, jotka on poimittu taulukosta 25.

$$P(Testi|Spam) \propto 0{,}5 * 0{,}47 * 0{,}35 * 0{,}35 * 0{,}53 * 0{,}88 * 0{,}88 * 0{,}82 * 0{,}94 * 0{,}71 \approx 0{,}00647$$

Vastaava arvo luokalle *Muu* voidaan määrittää seuraavasti:

$$P(Testi|Muu) \propto 0{,}5 * 0{,}12 * 0{,}06 * 0{,}18 * 0{,}47 * 0{,}76 * 0{,}82 * 0{,}76 * 0{,}71 * 0{,}71 \approx 0{,}00007$$

Koska $P(Testi|Spam) > P(Testi|Muu)$, on kysymyksessä *Spam-viesti*. On syytä huomata, että kyse on erittäin pienestä luvusta. Tämä johtuu siitä, että arvioivia ominaisuuksia on hyvin paljon. Tästä syystä järkevämpää on käsitellä tämän tyyppisiä lukuja logaritmisella asteikolla. Sen avulla voidaan välttää pieniin lukuihin liittyvät pyöristysongelmat. Tässä tapauksessa luokittelu voidaan tehdä kaavan 30 avulla. Tuolloin myös käytetyt priorit luokille tulee muuttaa samalle asteikolle. Annetun esimerkin tapauksessa $P(Spam) = P(Muu) = \log_2(0{,}5) = -1$. Tämä johtuu siitä, että molemmat luokat olivat yhtä todennäköisiä aineiston perusteella. Molempien luokkien priorit ovat (-1). Tuolloin luokittelu tehdään kaavan 30 avulla seuraavasti:

$$P(Testi|Spam) = (-1) + (-1{,}09) + (-1{,}05) + (-1{,}50) + (-0{,}92) + (-0{,}18) + (-0{,}18) + (-0{,}28) + (-0{,}09) + (-0{,}5) \approx -7{,}24$$

$$P(Testi|Muu) = (-1) + (-3{,}09) + (-4{,}09) + (-2{,}50) + (-1{,}09) + (-0{,}39) + (-0{,}28) + (-0{,}39) + (-0{,}5) + (-0{,}5) \approx -13{,}82$$

Logaritmisella asteikolla voidaan tehdä sama johtopäätös, että kysymys on *Spam-viestistä*. Perustelu asialle on, että $-7{,}24 > -13{,}82$.

Multinomialimalli

Multinomialimalli tunnetaan myös sanojen esiintymistiheydet. Merkittävin muutos Bernoullin malliin on MLE-arvojen laskeminen dokumenttien sisältämille sanoille. Täydentämällä edellistä esimerkkiä sanojen esiintymistiheyksillä luokittelu voidaan

tehdä Multinomialimallilla. Koska emme muuta luokkien *Spam* tai *Muu* suhdetta, priorit ovat samat molemmille arvioitavista hypoteeseista. Näin ollen niitä ei tarvitse määrittää uudelleen. Taulukossa 26 on esitetty kuvitteelliset sanojen esiintymistiheydet luokittain.

Tauluko 26. *Dokumenttitermimatriisi sanojen esiintymistiheyksin*

	Viagra	**Osta**	**Hei**	**Lounas**	**Metadata**	**Ehdotus**	**Tarjous**	**Luokka**	**Ja**	
S_1	0	0	3	2	0	2	0	2	1	**Muu**
S_2	0	0	2	3	0	0	0	2	0	**Muu**
S_3	0	0	1	0	0	1	2	0	2	**Muu**
S_4	1	0	4	0	0	0	0	0	0	**Muu**
S_5	0	0	1	0	0	0	0	1	1	**Muu**
S_6	0	0	1	2	5	2	1	0	0	**Muu**
S_7	0	0	2	0	0	0	0	3	3	**Muu**
S_8	0	0	1	0	2	0	0	0	0	**Muu**
S_9	2	0	1	0	0	0	0	0	1	**Spam**
S_{10}	4	0	0	0	0	0	1	0	0	**Spam**
S_{11}	2	2	1	0	1	3	0	0	2	**Spam**
S_{12}	0	1	1	0	0	0	0	0	0	**Spam**
S_{13}	2	1	1	0	0	0	2	0	1	**Spam**
S_{14}	4	4	1	1	0	0	3	0	0	**Spam**
S_{15}	2	0	1	0	0	2	1	0	1	**Spam**
S_{16}	2	3	1	0	0	0	2	0	0	**Spam**
Testi	1	1	0	0	0	0	1	0	0	**?**

Taulukon 26 mukaan sähköposteissa $S_{1\text{-}16}$ on yhteensä 110 sanaa. Ne ovat jakautuneet seuraavasti: *Spam* luokassa on 57 sanaa ja loput 53 sanaa sijaitsevat luokassa *Muut*. Sanojen jakaumat ovat nyt uudet $\sum_{w' \in W} W_{cw'}$ arvot. Tiiviste sanojen esiintymisestä luokittain on annettu taulukossa 27.

Taulukko 27. *Sanojen yhteenlasketut määrät luokittain*

	Viagra (w_1)	**Osta (w_2)**	**Hei (w_3)**	**Lounas (w_4)**	**Metadata (w_5)**	**Ehdotus (w_6)**	**Tarjous (w_7)**	**Luokka (w_8)**	**Ja (w_9)**
Spam	18	11	7	1	1	5	9	0	5
Muu	1	0	15	7	7	5	3	8	7

MLE-arvot voidaan määrittää edelleen sanoille kaavan 32 mukaan. Tuolloin (W_{cw}) on arvioitavan sanan yhteenlaskettu esiintymismäärä luokkaan kuuluvissa dokumenteissa. MLE-arvo sanan *Viagra* kohtaamiseksi luokassa *Spam* saadaan $\acute{P}(Viagra|Spam) = \frac{18+1}{57+9} \approx 0{,}28788$. Taulukkojen 26 ja 27 pohjalta määritellyt MLE-arvot esitetään taulukossa 28. Logaritmista menetelmää ei tässä yhteydessä ole tarkoitus todentaa uudelleen.

Taulukko 28. *MLE -arvot*

$\acute{P}(Viagra\|Spam)$	0,28788
$\acute{P}(Osta\|Spam)$	0,18182
$\acute{P}(Hei\|Spam)$	0,12121
$\acute{P}(Lounas\|Spam)$	0,03030
$\acute{P}(Metadata\|Spam)$	0,03030
$\acute{P}(Ehdotus\|Spam)$	0,09091
$\acute{P}(Tarjous\|Spam)$	0,15152
$\acute{P}(Luokka\|Spam)$	0,01515
$\acute{P}(Ja\|Spam)$	0,09091
$\acute{P}(Viagra\|Muu)$	0,03226
$\acute{P}(Osta\|Muu)$	0,01613
$\acute{P}(Hei\|Muu)$	0,25806
$\acute{P}(Lounas\|Muu)$	0,12903
$\acute{P}(Metadata\|Muu)$	0,12903
$\acute{P}(Ehdotus\|Muu)$	0,09677
$\acute{P}(Tarjous\|Muu)$	0,06452
$\acute{P}(Luokka\|Muu)$	0,14516
$\acute{P}(Ja\|Muu)$	0,12903

Nämä arvot esitetään tarkoituksella viiden desimaalin tarkkuudella, sillä saatava tulos on hyvin pieni luku. On hyvä muistaa, ettei Multinomialimallissa huomioida sanojen puuttumisia. Tuolloin käytetyssä esimerkissä hypoteesit voidaan kirjoittaa seuraavaan muotoon.

$$P(Testi|Spam) \propto \acute{P}(Spam) * \acute{P}(Viagra|Spam) * \acute{P}(Osta|Spam) * \acute{P}(Tarjous|Spam)$$

$$P(Testi|Muu) \propto \acute{P}(Spam) * \acute{P}(Viagra|Muu) * \acute{P}(Osta|Muu) * \acute{P}(Tarjous|Muu)$$

On hyvä huomata, että *Testi* dokumentissa jokainen sana esiintyi vain kerran. Jos sana esiintyisi useamman kerran samassa dokumentissa, sitä kohdeltaisiin samanlaisena todisteena kuin muitakin sanoja. Tuolloin esiintymiskerrat voidaan esittää eksponentin avulla (kaava 34).

$$P(Testi|Spam) \propto 0{,}5 * 0{,}28788 * 0{,}18182 * 0{,}15152 \approx 0{,}003965$$

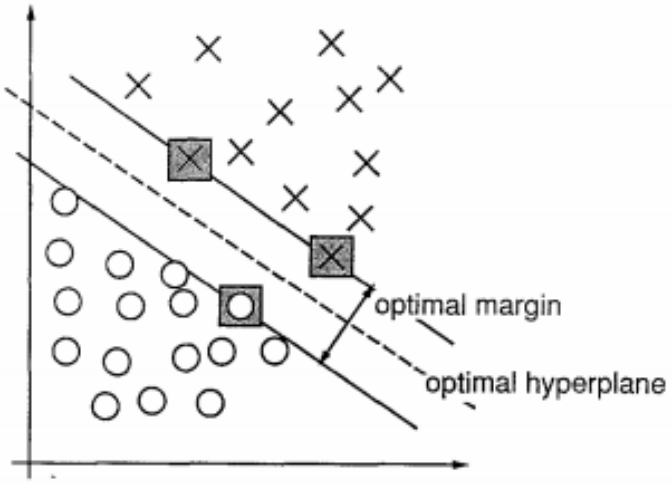
$$P(Testi|Muu) \propto 0{,}5 * 0{,}03226 * 0{,}01613 * 0{,}06452 \approx 0{,}00002$$

On varsin selvää, että $P(Testi|Spam) > P(Testi|Muu)$. Tämän perusteella hypoteesi Spam valitaan luokitteluksi.

Annettu esimerkki oli kooltaan hyvin rajoittunut. On hyvä huomata, että todellisuudessa arvioitavia ominaisuuksia olisi hyvin paljon enemmän. Annettujen esimerkkien perusteella voitiin havaita, että tulos oli odotusten mukainen kaikissa tapauksissa. Bayesian päättelyssä on näin ollen kysymys todennäköisimmän hypoteesin valinnasta luokiteltavalle kohteelle.

3.3.5 Yhteenveto luokittelumenetelmistä

Kirjoittajan näkemyksen mukaan kirjallisuudessa esitellään varsin hyvin toimivia luokittelualgoritmeja. Ne näyttäisivät toimivan myös varsin hyvin monissa eri käyttökohteissa kuten roskapostisuodattimissa. Työn puitteissa käsitellyt menetelmät olivat yksinkertaisimpia kirjallisuudessa esitetyistä menetelmistä. Ne soveltuvat tästä huolimatta hyvin useisiin eri käyttökohteisiin. On hyvä tietää, että esiteltyjen menetelmien lisäksi on olemassa myös vektoreihin perustuvia luokittelumenetelmiä. Työn puitteissa niitä ei ole mahdollista käsitellä tarkasti. Yksi vektorimenetelmistä on SVM (engl. *support Vector Machines*). Se arvioidaan usein yhdeksi tehokkaimmista luokittelualgoritmeista. Lyhyesti käsiteltynä SVM-menetelmässä ideana on esittää ominaisuudet vektoriavaruudessa. Tämän jälkeen luokkien ja niissä esiintyvien ominaisuuksien välille sovitetaan hypertaso (engl. *hyperplane*). Kyseinen taso toimii sitten luokkia ja sen ominaisuuksia erottavana rajana (Cortes & Vapnik 1995, ss. 273-274; István 2005).



Kuva 24. Luokittelu vektoriavaruudessa (Cortes & Vapnik 1995, s. 275)

Tästä syystä vektorimalli on työväline myös tekstin esittämiseksi vektoriavaruudessa. Todellisuudessa varsinkin vektoreihin perustuvia luokittelualgoritmeja on hankalaa esittää kuviolla, sillä varsinaisia ulottuvuuksia avaruudessa on paljon. SVM-menetelmät osana luokittelua ovatkin hyvin mielenkiintoinen aihealue. Se on erityisen hyvä jatkotutkimusaihe tälle työlle. SVM:n läpikäyminen edellyttää jo itsessään diplomityön pituista tarkastelua, joten työn puitteissa sitä ei ole mahdollista käsitellä. Menetelmästä on saatavilla lisää informaatiota Cortes ja Vapnikin artikkelissa.

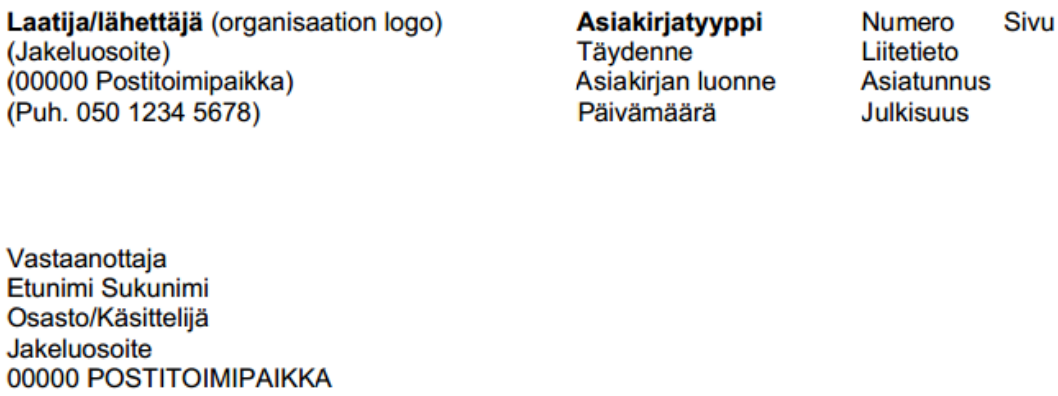
Johtopäätös edellä esitetyistä menetelmistä on, että ne ovat hyviä perustyökaluja todisteiden arvioimiseksi ja luokittelun ennustamiseksi historiaan perustuen. Käsiteltyä Bayesian teoriaa voidaan hyödyntää myös muilla osa-alueilla. Kysymyksessä on yleiskäyttöinen työväline todisteiden arvioimiseksi. Menetelmää voidaan käyttää useissa eri sovelluksissa, joista yksi käyttökohde on kontekstisidonnaisten kirjoitusvirheiden ja nimien tunnistus.

3.4 Avainsanojen poiminta

Tavoitteena on tutustuttaa lukija tekniikoihin, joiden avulla dokumenteista voidaan poimia avainsanoja. Tarkoituksena on tutkia kolmea perustekniikkaa, joita ovat säännölliset lausekkeet, sanan sijainti ja sanalistat. Kohta on jaettu kahteen osaan, joista ensimmäinen käsittelee säännöllisiä ilmaisuja ja toinen käsittelee sanojen sijaintiin sekä listoihin perustuvaa informaation hakua.

Säännölliset lausekkeet (engl. *regular expression, regexp*) mahdollistavat muodoltaan säännöllisten ilmaisujen etsimisen merkkijonoista. Säännöllisellä ilmaisulla tarkoitetaan sellaista merkkijonoa, jonka muoto voidaan määritellä tarkasti (SC18-9878-04 2012, ss. 155-156). Niitä ovat esimerkiksi henkilötunnukset, sähköpostiosoitteet ja y-tunnukset. Luonnollisella kielellä kirjoitetut merkkijonot eivät yleensä ole säännöllisiä. Hyvänä esimerkkinä luonnollisen kielen monimutkaisuudesta ovat katujen nimet, jotka sisältävät kadun, tien tai raitin osana nimeä. Tämä ei kuitenkaan pidä aina paikkaansa, koska on olemassa poikkeuksia: *Erottaja* ja *Kolmas linja*. Luonnollisella kielellä ilmaistujen konseptien poiminta tekstistä edellyttääkin kehittyneempien menetelmien hyödyntämistä tai sanakirjan hyödyntämistä. Näitä menetelmiä esiteltiin työssä osana konseptien tunnistusta. Kyseisestä rajoituksesta huolimatta useimmat tekstiä käsittelevät järjestelmät perustuvat Regular expressionin käyttöön. Tekstin jakaminen avaimiin voidaan tehdä yksinkertaisesti Regular expressionin avulla. Menetelmänä se on yksinkertainen tapa etsiä, pilkkoa, korvata ja täsmätä säännöllisiä merkkijonoja. Tekniikkana se perustuu merkkijonojen kuvaamiseen erityisinä formaaleina ilmaisuina (Goyvaerts 2009). Työn tekijän kokemuksesta säännöllisyys on useimmille ihmisille varsin tuntematon käsite. Asiaa voidaan tarkastella y-tunnuksen avulla. Patentti- ja rekisterihallituksen mukaan "*Tunnuksessa on 7 numeroa, väliviiva ja tarkistusmerkki eli se on muotoa 1234567-1*" (Patentti- ja rekisterihallitus 2010). Tämä voidaan kuvata Regular expressionilla seuraavalla tavalla: (\b[0-9]{7}-[0-9]\b). Annetussa lausekkeessa \b tarkoittaa sanarajaa, johon kuuluu välilyönti, piste tai pilkku. Merkintä [0-9] tarkoittaa ryhmää, johon sisältyvät kaikki numerot nollan ja yhdeksän väliltä. Aaltosulje ilmoittaa, montako peräkkäistä ryhmään kuuluvaa symbolia tulee esiintyä peräkkäin. Annettu (-) tarkoittaa tavuviivaa seitsemän esitetyn numeron jälkeen. Lopussa oleva ryhmä tarkoittaa yksittäistä numeroa nollan ja yhdeksän väliltä, jota seuraa sanaraja. Annettu lauseke täsmää tuolloin seuraavaan merkkijonoon 1111111-1. Se ei täsmää kuitenkaan merkkijonoon FI1111111-1, koska numerosarja ei ala sanarajalla. Toisin sanottuna Regular expression on kieli, jonka avulla voidaan muodostaa merkkijonoryhmiä (Oracle 2013).

Eräs esitetty keino avaimien tunnistamiseen on hyödyntää dokumentin vakiintunutta rakennetta. Asiakirjamallissa tunnistetietojen voidaan olettaa sijaitsevan tietyssä järjestyksessä (kuva 25).Voimme tuolloin hyödyntää tätä järjestystä tekstissä navigoimiseen ja informaation poimintaan (SC18-9878-04 2012, s. 155).



Kuva 25. *Asiakirjamalli (BBM. 2013) mukaan*

Luonnollisen kielen ongelma on, ettei mikään pakota dokumentteja noudattamaan tiettyä mallia. Toinen ongelma on, ettei kaikissa tapauksissa avaimien määrä pysy vakiona. Teksti voi sisältää moniosaisia konsepteja kuten *Hong Kong*. Tuolloin poimittavan käsitteen sijainti voi muuttua. Tämä johtuu siitä, että järjestelmä hyödyntää avaimia konseptien asemasta. Yksittäinen konsepti voi taas koostua useammasta avaimesta, joten avaimellinen indeksi voi muuttua. Koska järjestelmällä ei ole ymmärrystä avaimen riippuvuuksista muihin avaimiin, tekstin sisältämät konseptit voivat aiheuttaa ongelmia sijainnin määrittämiseksi. Edellä kuvattu haaste voidaan ratkaista osittain hyödyntämällä vakiomuotoisia avaimia. (SC18-9878-04 2012, s. 155) Tuolloin poimittava informaatio voidaan pyrkiä määrittämään suhteessa vakiomuotoiseen avaimeen. Tiedämme esimerkiksi, että avainta *Vastaanottaja* seuraa nimi (kuvan 25). Sen perusteella voimme olettaa, että kaksi seuraavaa avainta liittyy henkilön nimeen. Lähestymistapaan liittyy kuitenkin haaste, joka syntyy tekstin lukusuunnasta. Yleisen oletuksen mukaan sanat tallennetaan vasemmalta oikealle. Tämä oletus on ehdottoman väärä, kuten kuvasta voidaan huomata. Oletusta noudatetaan usein myös tiedostoissa, jolloin sijainnin määrittäminen on hankalaa. Kuvassa 25 tekstin *Asiakirjatyypin* alapuolella olevat tunnisteet ovat haasteellisia paikantaa. Tämä johtuu siitä faktasta, että asiakirjatyyppiä seuraa annetussa mallissa numero ja sivu. Niitä ei kuitenkaan ole pakko antaa, jolloin avainten paikat voivat muuttua. Avainten sijaintien hyödyntäminen sopii ratkaisuiksi rajattuihin käyttökohteisiin, jossa sijainti voidaan vakioida.

Sanalistat ovat hieman edellä kuvattua parempi työkalu avainsanojen poimintaan. Menetelmässä ideana on luoda lista avaimista, jotka tunnistetaan vertailemalla sanakirjaa eli sanalistaa. Sanalistan käyttöön voidaan myös liittää vaatimus avaimen sijainnista tekstissä. (SC18-9878-04 2012, ss. 174-180). Tuolloin esimerkiksi vakiomuotoinen tunniste on yksinkertaista rajata dokumentin alusta. Menetelmä edellyttää

kuitenkin, että tunnistettavat sanat voidaan määrittää. Se tarkoittaa, ettei vaihtoehtojen määrä voi olla rajaton. Tämä on kuitenkin ongelma, koska näitä listoja tulee päivittää.

4 INTEGRAATIORATKAISU

Luvun tavoitteena on kuvata, miten ja mihin *IBM Content Classification Modulea* voidaan hyödyntää. Tuotteen esittely tehdään hyödyntäen erästä projektia, jossa se integroitiin Elinarin tekemään Office Integraatio -ratkaisuun. Luvussa esitetään tutkimuksessa käytetylle projektille asetetut tavoitteet, tuotteen käyttöönotto ja siitä saatuja kokemuksia. Tarkoitus ei ole esittää integroinnissa käytettyä ohjelmakoodia, vaikka tutkittavaan projektiin sisältyi myös toteutusta. Asiakaskohtaisia yksityiskohtia ei myöskään ole tarkoitus esitellä.

4.1 Projektin tavoitteet

Kohta 4.1 kuvaa tutkimuskohteena käytetyn projektin ja sille asetut tavoitteet. Työssä tarkastellaan näiden tavoitteiden täyttymistä osana toteutettua ratkaisua.

Projektin tarkoitus
Office integraation tarjoama lisäarvo on säästää sähköpostien ja eri dokumenttien käsittelyjärjestelmään tallentamiseen kuluvaa työaikaa. Työssä käsiteltävän projektin tavoitteena oli täydentää aikaisempaa Office Integraatio -ratkaisua avainsanojen poiminnalla ja luokittelulla. Projektin motivaationa oli selvittää teknologian kypsyyttä tallennuksessa käytetyn metadatan tuottamiseksi. Tästä on hyötyä sähköpostitse saapuvien liitteiden käsittelyssä. Projektin työtuloksena syntyi (POC) -tyyppinen (engl. *Proof of Concept)* ratkaisu, jonka avulla asiakasorganisaatio pystyy arvioimaan tuotteen toimivuutta.

Projektissa toteutettava käyttötapaus oli seuraava:
Vahinkokäsittelijä aloittaa asiakirjan tai sähköpostin käsittelyn, jolloin siitä pyydetään luokittelutietoja. Luokittelija poimii kohteesta avainsanat ja luokan. Kohteesta saadut avainsanat ja luokittelu esitetään vahinkokäsittelijälle käyttöliittymässä. Vahinkokäsittelijä tarkistaa saadut kentät ja tallentaa asiakirjan vahinkokäsittelyjärjestelmään.

Toiminnalliset vaatimukset olivat seuraavat:
1. Luokittelija ehdottaa sähköpostin liitteille ja muille tiedostoille luokittelua perustuen tiedostojen sisältöön. Sähköpostissa automaattinen luokittelu valitaan liitekohtaisesti.
2. Sähköpostin liitteistä ja muista tiedostoista tulee automaattisesti tunnistaa avaimia, perustuen esitysmuotoon. Avainsanoja ehdotetaan OFI-lomakkeella arvoiksi asiakirjakohtaisiin metadatakenttiin. Arvo valitaan lomakkeelle automaattisesti, jos tunnis-

tettuja arvoja on vain yksi. Jos arvoja on useita, käyttäjän tulee valita arvo alasvetovalikosta.
3. Automaattiseen tunnistukseen liittyvät tietokenttämuodot ovat päivämäärä, henkilötunnus, y-tunnus, rekisteritunnus, asiakastunnus, vakuutustunnus, vahinkotunnus ja icd-koodi.
4. Tuettavia tiedostomuotoja automaattisessa tunnistuksessa ja luokittelussa ovat doc, docx, xls, xlsx, rtf, txt, html, xml ja pdf.
5. Käyttäjä voi tarkastella luokittelun antamia tietoja käyttöliittymän erillisessä ikkunassa.

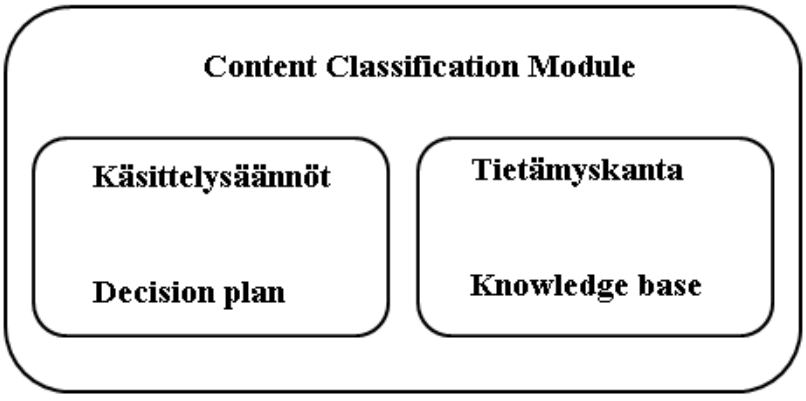
Ei toiminnalliset vaatimukset

Content Classification Modulen integroiminen osaksi tutkittavaa tuotetta päätettiin tehdä erillistä verkkoon asennettavaa luokittelupalvelinta hyödyntäen. Tavoitteena oli välttää työasemiin ylimääräisten ohjelmistojen asentamista. Samalla muutosten tekeminen sääntöihin on yksinkertaisempaa asiakkaan kannalta. (Mähönen & Suominen 2012)

4.1 Content Classification Module

Kohdassa esitellään *Content Classification Moduleen* liittyvää terminologiaa. Tavoitteena on perehtyä tuotteen toimintaperiaatteisiin ennen tuotteen esittelyä.

Content Classification Module on IBM:n kehittämä ohjelmisto luokitteluun ja avainsanojen tuottamiseen. Varsinainen tuotteen toiminta perustuu kahteen keskeiseen käsitteeseen, jotka ovat tietämyskanta (engl. *knowledge base, KB*) ja käsittelysäännöt (engl. *decision plan*) (kuva 26).



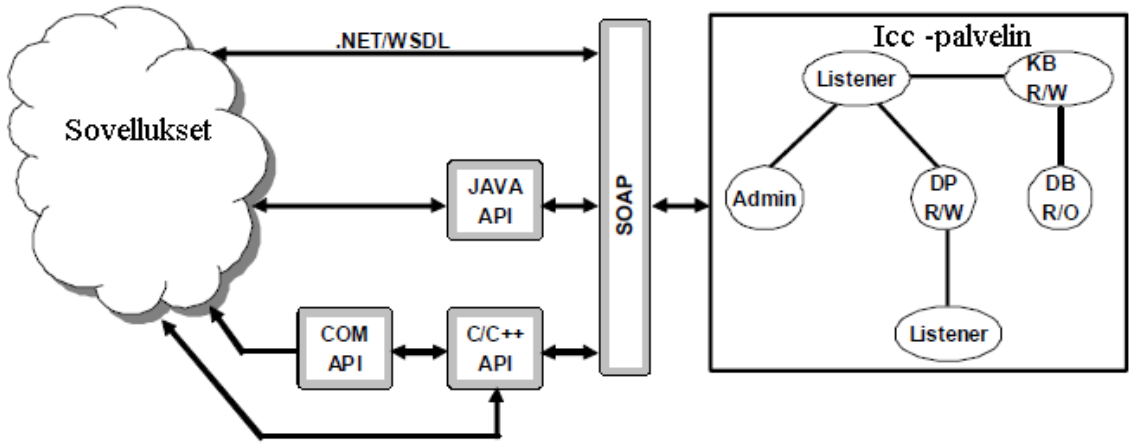
Kuva 26. *Content Classification Modulen osat*

Tietämyskannan eli KB:n tehtävänä on tuottaa tekstille luokittelu. Annetulla käännöksellä työssä ei haluta kuitenkaan väittää, että kysymys olisi filosofisessa mielessä tietämyksestä. Kysymys on tekstin luokittelusta tilastollisin menetelmin (engl. *statistical pattern matching*). Se tarkoittaa, että luokan päättely tapahtuu ilman ohjelmakoodiin kiinteästi rakennettuja luokittelusääntöjä (engl. *rule based classification*). Luokittelupäätöksen tekemiseksi tuote vertaa luokiteltavan dokumentin sisältämää semanttista informaatiota (engl. *semantic information*) luokkamalleihin (Zhu et al. 2009, s. 25). Kir-

joittajan tiedossa ei ole luokittelualgoritmia, jota IBM käyttää tässä tuotteessa. Tästä algoritmista ei kirjoittaja halua myöskään esittää arvailuja.

Käsittelysääntöjen (engl. *decision plan, DP)* avulla ohjelmistossa voidaan suorittaa erilaisia toimenpiteitä, joihin voi kuulua dokumentin siirtäminen tietovarastoon, avainsanojen etsintä ja luokittelutietojen hakeminen. Käsittelysäännöt koostuvat kahdesta konseptista, jotka ovat ehdot (engl. *trigger*) ja toimenpiteet (engl. *actions*). Ehtojen (engl. *trigger*) avulla ohjelmistossa voidaan valita suoritettavat käsittelysäännöt. Valinnassa voidaan hyödyntää tekstin sisältämiä avaimia, tuotettua luokittelua tai muita ehtoja. (Zhu et al. 2009, s. 21-26) Käsittelysääntöjen eriyttäminen omaksi komponentiksi perustuu ideaan yksinkertaistaa sitä käyttävien sovellusten ylläpitoa. Konseptin ansiosta säännöt sijaitsevat luokittelupalvelimella, jolloin niiden päivittäminen onnistuu ilman niitä käyttävien sovellusten muokkaamista. Tästä on erityisesti avainsanojen kanssa, koska sääntöihin voidaan tarvita muutoksia ja tarkennuksia ajoittain.

Tuote itsessään on työväline, joka on suunniteltu luokittelun ja avainsanojen tuottamiseen. Se pitää integroida osaksi toista sovellusta tai tietovarastoa. Osaksi käytettyä tietovarastoa se voidaan integroida yksinkertaisesti valmiin integraatioratkaisun avulla. Tähän IBM tarjoaa valmiin ratkaisun omille järjestelmilleen, kuten *Filenet* ja *Content Manager* (Zhu et al. 2009, s. 17). Tuolloin asiakirja voidaan tallentaa suoraan sisällönhallintajärjestelmään tuotetuin avainsanoin, eikä tuotteen käyttöönottamiseksi tarvita ohjelmistokehitystä. Tuote voidaan myös integroida osaksi jotain muuta sovellusta. Tästä syystä ohjelmistossa on Api-rajapinta (engl. *Application programming interface*), jonka avulla sen käyttötarkoitusta voidaan laajentaa. Sen avulla tuote voidaan integroida myös tietovarastoihin, jotka eivät ole IBM:n valmistamia. Kuvassa 27 on kuvattu tuotteen tarjoamat rajapinnat ja arkkitehtuuri.



Kuva 27. *Tuote ja siihen liitettävät sovellukset (Zhu et al. 2009, s. 49)*

Content Classification Module sisältää varsinaisen palvelinohjelmiston lisäksi kaksi keskeistä muuta työkalua, jotka ovat *Management console* ja *Classification Workbench*. *Management Console* toimii hallintatyökaluna ja sen avulla luokittelupalvelimelle voidaan tuoda uusia käsittelysääntö- ja tietämyskantaprojekteja. Sen avulla voidaan myös

päivittää olemassa olevia projekteja ja asettaa lukuisia muita asetuksia. *Classification Workbench*-työkalulla voidaan perustaa uusia tietämyskantoja ja muokata käsittelysääntöjä graafisesti. Ohjelmiston keskeisin komponentti on kuitenkin luokittelupalvelin (engl. *classification server*). Se pitää sisällään käytetyt käsittelysäännöt ja tietämyskannat. Luokittelupalvelin on itsessään verkkorajapinta, johon eri sovellukset kytkeytyvät. Tarkemmin se jakautuu *Listeneriin* ja *Admin* -prosesseihin. Näistä *Listener*-prosessi toimii tiedostojen sisääntulokanavana luokittelupalveluun. *Admin*-prosessin tehtävänä on hallita järjestelmään tulevia muutospyyntöjä. Ylläpitopyyntöihin kuuluvat toimenpiteet, joita voidaan suorittaa *Management consolesta*. (Zhu et al. 2009, ss. 31-32)

4.2 Käyttöönotto

Kohdan tarkoitus on kuvata kevään 2012 aikana toteutettua integraatioratkaisua ja siihen liittyviä työvaiheita. Työssä kerrotaan *Content Classification Modulen* käyttöönotosta ja siihen liittyvistä työvaiheista. Tämä tapahtuu hyödyntäen asiakasprojektista saatuja kokemuksia.

Tuotteen käyttöönottamiseksi tarvittiin seuraavat työvaiheet:

1. Tuotteen asennus
2. Tietämyskannan määrittäminen ja käyttöönotto
3. Käsittelysääntöjen määrittely ja käyttöönotto
4. Integroiminen osaksi OFIv3:a

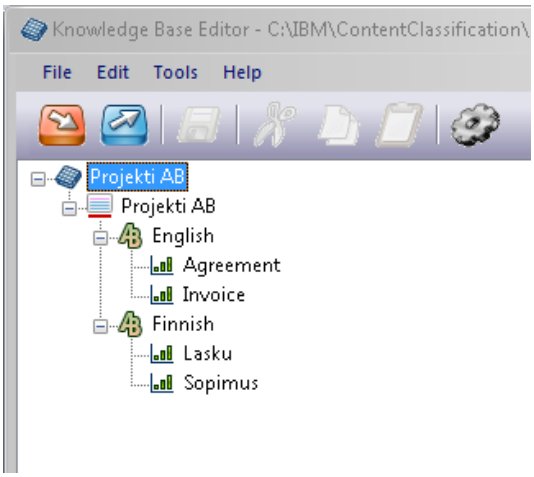
Kohdan sisältö on jäsennetty seuraavasti: ensin esitellään, miten tuotteen tietämyskanta eli luokittelu määritellään. Tämän jälkeen tutustutaan käsittelysääntöihin. Lopuksi tarkoituksena on esitellä, mitä tuotteen integroiminen toiseen ohjelmaan edellyttää. Tuotteen asennusta ei työn puitteissa ole tarkoitus kuitenkaan käsitellä.

4.2.1 Tietämyskanta

Tietämyskanta eli KB on ohjelman oppiva komponentti. Se oppii sille annetusta palautteesta (engl. *feedback*) ja sen avulla saadaan tekstiä koskeva luokittelu. Komponentin avulla ei tuoteta avainsanoja, vaan sen tehtävänä on vastata kysymykseen, mitä luokkaa luokiteltava teksti esittää. (Zhu et al. 2009, ss. 21-23) KB:n määrittelemiseksi ja käyttöönottamiseksi tarvitaan seuraavat työvaiheet:

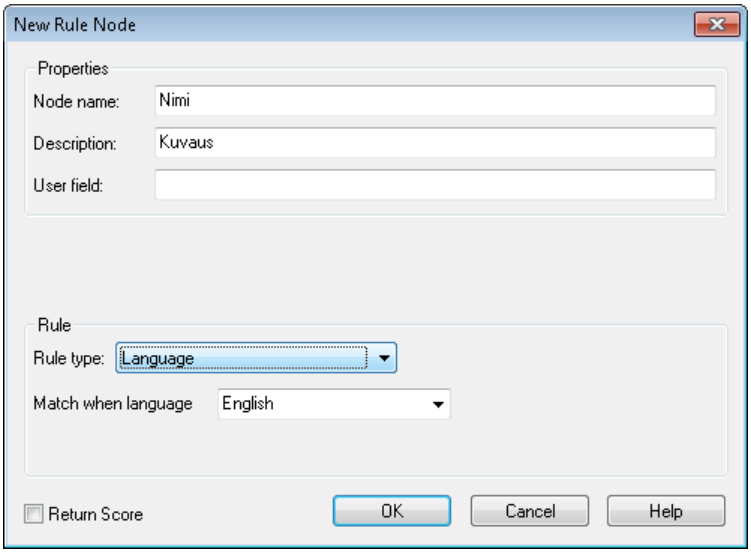
1. Luokkien määrittely
2. Kieliasetuksien määrittely
3. Koulutus

KB:n rakentaminen tapahtuu määrittelemällä ensin luokkahierarkia, joka tapahtuu erillisellä työkalulla nimeltä *Knowledge Base Editor* (kuva 28). Työkalu on löydettävissä *Classification Workbench* -työkalusta valikosta *tools*.



Kuva 28. *Luokat ja säännöt Knowledge Base Editorissa*

Työkalu pitää sisällään kaksi keskeistä käsitettä, jotka ovat *rule nodet* ja *learning nodet*. RN eli *rule node* on ehto, jonka perusteella käytettävissä olevat luokat valitaan arviointiin. Konseptin avulla luokista voidaan rajata käytettäväksi vain osaa tunnistetun kielen perusteella (kuva 29).

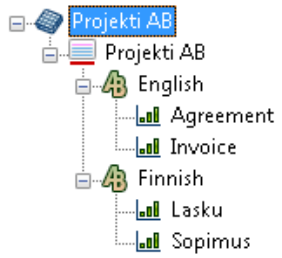


Kuva 29. *RN määrittäminen*

Ohjelmassa on käytettävissä seuraavat ehdot luokkien valintaan:

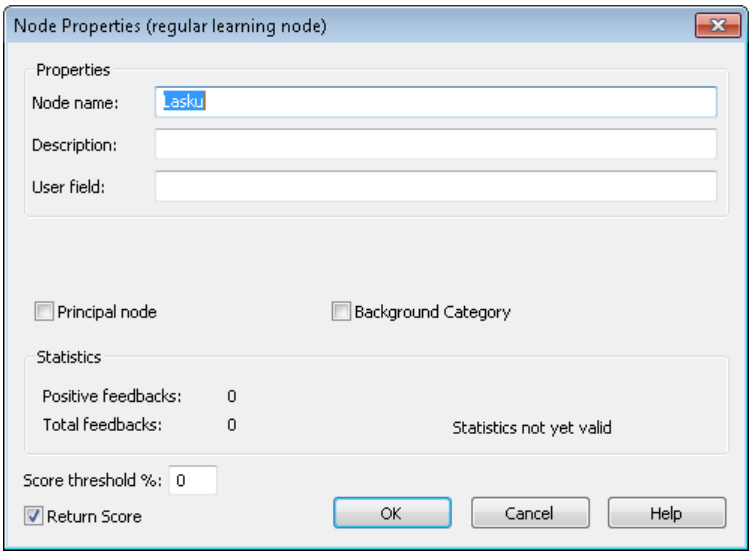
1. Sisällytä aina (engl. *always*)
2. Sisällytä, jos mikään aikaisempi ehto ei toteutunut (engl. *otherwise*)
3. Sisällytä, kun luokiteltava kohde ei sisällä tekstiä (*engl. empty item*)
4. Valitse, kun kohde sisältää tekstiä (engl. *not empty item*)
5. Valitse, kun kirjoituskieli on (engl. *language*)
6. Valitse, kun kentän arvo on (engl. *field contains*)
7. Muu sääntö (engl. *expression*). Toiminto on varattu IBM:n valmistamille mukautetuille suodattimille.

RN mahdollistaa myös sääntöryhmien laatimisen (engl. *hierarchical knowledge bases*), missä yhtä ehtoa seuraa toinen ehto. Konseptin käytölle on tarvetta, jos esimerkiksi järjestelmän pitää osata käsitellä monikielistä aineistoa (kuva 30). (SC18-9878-04 2012, s. 125-128)



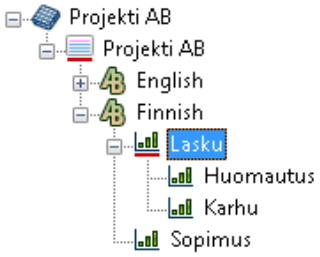
Kuva 30. *Kaksi sääntöryhmää*

LN eli *learning node* on luokka, joka dokumentille voidaan valita luokittelussa. LN tulee kuulua aina RN:ään. *Node namella* tarkoitetaan luokan nimeä. Luokan nimi voi olla esimerkiksi lasku (kuva 31).



Kuva 31. *Luokan perustaminen*

Score treshold -parametrilla voidaan määrittää kynnys, jonka ylitettyä luokittelija ottaa kyseisen luokan mukaan tuloksiin. *Principal node* -parametria käytetään luomaan tilastollinen hierarkia alipuussa (kuva 32).

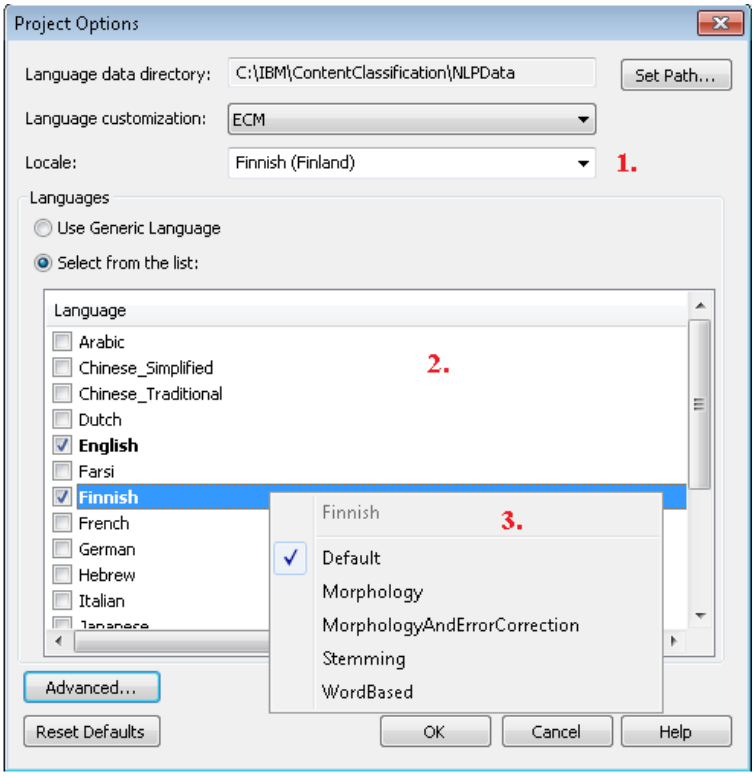


Kuva 32. *Principal Node*

Principal nodea tarvitaan, kun kaksi luokkaa jakavat tilastollisesti paljon samoja ominaisuuksia. Tuolloin asettamalla kyseinen asetus päälle voidaan parantaa suorituskykyä. Tämä tarkoittaa yhden tai useamman alaluokan määrittämistä (kuva 32). *Background Category* kertoo järjestelmälle, että järjestelmään saapuvasta datasta suurin osa on irrelevanttia. Tämä asetus valitaan luokalle, joka pitää sisällään irrelevantit dokumentit. Asetusta ei valita luokalle, johon relevantit dokumentit kuuluvat. Roskapostien suodatuksessa asetus valittaisiin normaaleille poisteille, koska ollaan kiinnostuneita löytämään ainoastaan roskapostit. *Return Score* -asetus puolestaan kertoo, onko kyseinen luokka luokittelujärjestelmän käytettävissä vai ei. (Zhu et al. 2009, s. 143; SC18-9878-04 2012, ss.130-131).

Kieliasetukset

Ennen koulutusmateriaalin antamista järjestelmään tulee määrittää kieli- ja lokalisaatioasetukset. Kieleen liittyviä asetuksia voi muuttaa projektin asetuksista (kuva 33).



Kuva 33. *Projektin asetukset*

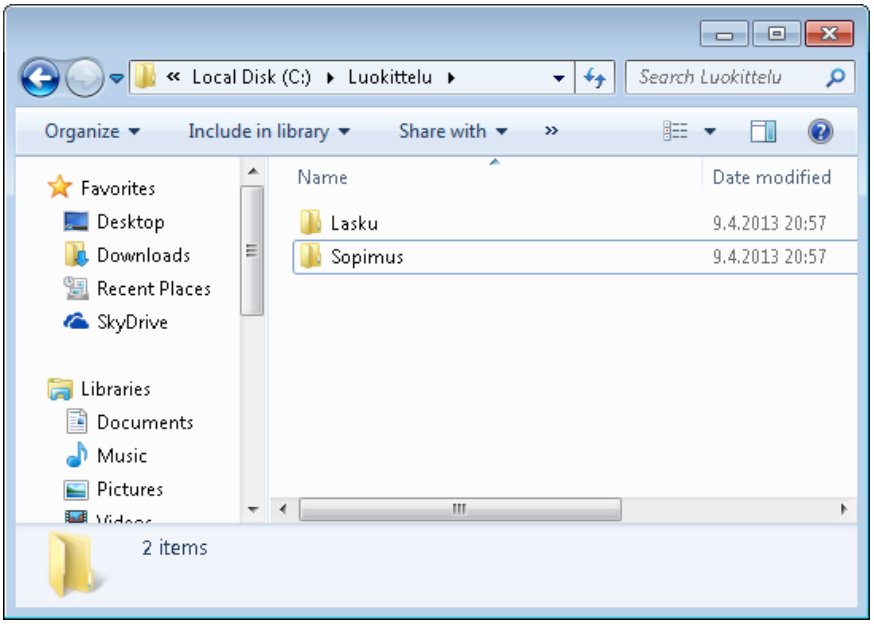
1. *Locale*-asetuksella voidaan määrittää, miten päivämärät, valuutta ja numerot esitetään *Classification Workbench* -työkalussa.
2. Listassa on esitetty luokittelussa virallisesti tuetut kielet, joille ohjelmassa on omat käsittelysäännöt. *Generic Language* -asetuksella voidaan prosessoida kieliä, jotka eivät ole virallisesti tuettu.
3. Asetuksella voidaan vaihtaa tekstin käsittelytapa. Oletuksena järjestelmä käyttää asetusta *Stemming*. Morfologisia sääntöjä voidaan hyödyntää, jos halutaan saavuttaa paras mahdollinen tarkkuus. Prosessina sanat johdetaan perusmuotoon morfologisin

säännöin. *MorphologyAndErrorcorrection* asetus pyrkii korjaamaan tekstissä esiintyviä virheitä, mutta edellyttää kuitenkin sanalistan määrittelyä. *WordBased* asetuksesta mainitaan, että sitä tulisi käyttää vain, kun luokiteltavan tekstin kieli ei ole tuettu. Aasialaisten kielten kanssa asetusta tulisi käyttää kaikkien muiden paitsi korean kielen osalta.

Koulutusmateriaalin käsittely

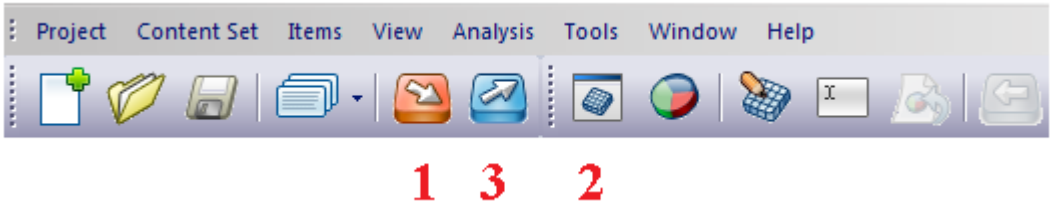
Luokittelujärjestelmälle koulutusmateriaalin antamiseksi ohjelmisto tarjoaa kaksi keskeistä lähestymistapaa. Nämä ovat materiaalin käsittely *Classification Workbench* -ohjelmassa ja palautteen antaminen luokittelua käyttävästä ohjelmasta. Tutkimuskohteessa järjestelmän opettaminen hoidettiin OFI:n avulla. Työssä on kuitenkin tarkoituksena esittää, miten palautteen käsittely onnistuu *Classification Workbench* -ohjelmassa.

Opetusvaiheessa järjestelmään tuodaan sellaisia dokumentteja, jotka luokittelijan halutaan tunnistavan automaattisesti (Zhu et al. 2009, 124). Jos koulutusmateriaali tuodaan levyltä, dokumentit voidaan ryhmitellä hakemistoihin luokittain (kuva 34). Tuolloin *Classification Workbench* -työkalu osaa asettaa valmiiksi oikean luokan jokaiselle tuotavista dokumenteista.



Kuva 34. *Koulutusmateriaali levyasemalla*

Yleinen sääntö koulutusmateriaalin suhteen on, mitä enemmän malleja luokan jäsenistä on, sitä paremmin luokittelu toimii. Valitun materiaalin tulisi olla kuitenkin relevanttia luokan sisällön näkökulmasta. Lisäksi materiaalin luokitteluun ei saa vaikuttaa ulkoiset tekijät, joita ei selviä materiaalista. Tällä tarkoitetaan, että luokkien sisältö tulee erota sanastollisesti toisistaan. Uuden koulutusmateriaalin tuominen tapahtuu painikkeella 1.



Kuva 35. *Materiaalin tuominen järjestelmään*

Kun koulutusmateriaali on tuotu järjestelmään, se voidaan käsitellä painikkeella 2 (kuva 35). Tämän jälkeen projekti on valmis käyttöönotettavaksi ja se voidaan viedä luokittelupalveluun painikkeella 3. (Zhu et al. 2009, ss. 59-76) *Content Classification Modules-sa* luokittelupäätöksen tekeminen perustuu tutkittavan kohteen vertaamiseen annettuun koulutusmateriaaliin. Oppiminen tarkoittaa luokittelijan näkökulmasta esimerkkien keräämistä kuhunkin luokkaan kuuluvista dokumenteista. Näistä dokumenteista johdetaan matemaattisesti luokittelusäännöt, joiden perusteella luokka voidaan valita testattavalle kohteelle.

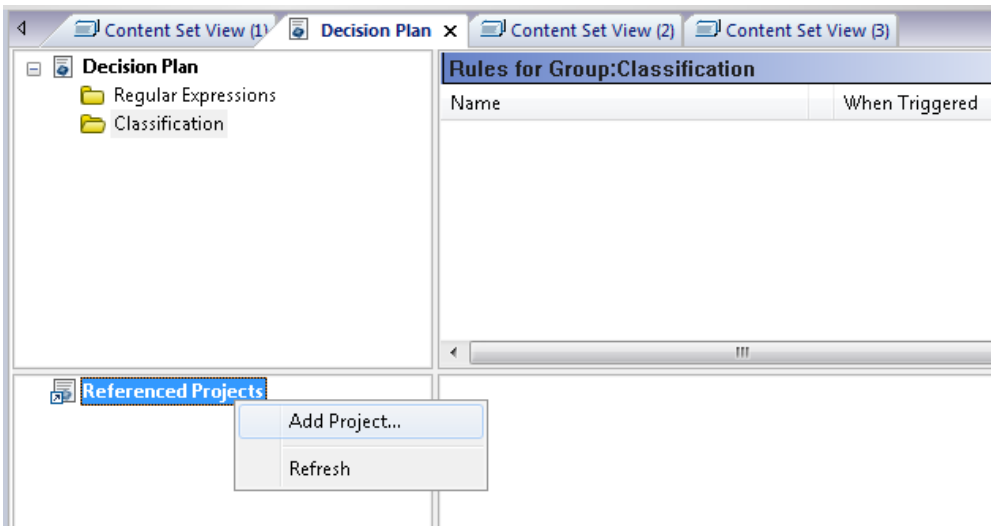
4.2.2 Käsittelysäännöt

Käsittelysääntöjen eli DP:n avulla ohjelmassa voidaan pyytää luokittelua, etsiä avainsanoja ja siirtää tiedosto tietovarastoon. Käsittelysäännöt pitävät sisällään kaksi keskeistä käsitettä, jotka ovat varsinaiset toimenpiteet (engl. *action)* ja ehdot (engl. *trigger*). Ne muodostavat yhdessä säännön (engl. *rule*), joka voi sisältää useita ehtoja ja toimenpiteitä. Se tarkoittaa, ettei käsittelysääntö ole rajattu ainoastaan yhden toimenpiteen suorittamiseen.

DP:n määrittelyyn tarvitaan seuraavat työvaiheet:

1. Viittaaminen tietämyskantaan
2. Kenttämääritykset
3. Käsittelysääntöjen määrittely
4. Testaus ja käyttöönotto

Viittaaminen tietämyskantaan
Luokittelun aktivoimiseksi käytetty tietämyskanta eli KB tulee linkittää osaksi käsittelysääntöjä. Tämä tapahtuu valitsemalla *Referenced Projects* (kuva 37).



Kuva 36. *Viittaaminen KB:hen*

Järjestelmässä on mahdollista käyttöönottaa yksi tai useampi yhtäaikainen luokittelumalli. Tuolloin järjestelmään voidaan ottaa mukaan useampi tietämyskanta.

Kenttämääritykset

Kentät toimivat ohjelmassa dataa sisältävinä muuttujina (kuva 37). Niitä käytetään tuotettujen avainsanojen ja luokittelun tallentamiseen. Esitettyjä kenttiä voidaan käyttää myös luokittelussa arvioitavan tekstisisällön valintaan.

Field Definitions

Name	Data Type	Content Type	Hide
Puhelinnumerot	string		
Body	string	Body	
Categories	classifica...		
FileName	string		
ICM_OriginalFileSize	number		
ICM_Creation_Date	string		
FilePath	string		
FileModifiedTime	string		
ICM_ExtractedTextSize	number		
Y-tunnukset	string		
Luokka	string		

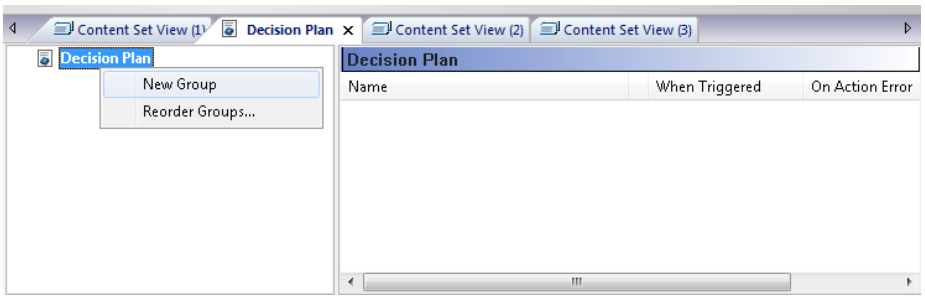
Kuva 37. *Kenttämääritykset* (engl. *field definitions*)

Järjestelmässä tulee olla ainakin yksi kenttä, jota käytetään luokitteluperusteena. Oletuksena tämä kenttä on *Body*, johon käsiteltävän asiakirjan sisältö puretaan. Tarvittaessa myös muita kenttiä voidaan asettaa luokitteluperusteeksi. Tuolloin kyseiselle kentälle tulee määrittää *Content Type*. Luokittelussa käytetyn kentän tulee sisältää kuitenkin merkityksellistä tekstiä. Muille kuin luokitteluperusteena käytettäville kentille *Content Type* tulee jättää tyhjäksi. Avainsanojen osalta järjestelmään tulisi perustaa omat kentät, josta ne voidaan poimia. Tietotyyppejä käytettävissä on kolme. *String*, johon voidaan tallentaa merkkijonoja, kuten poimittu y-tunnus. *Number*, johon voidaan tallentaa numeerinen arvo, kuten puhelinnumero. *Classification*-tietotyyppiä käytetään luokittelutietojen esittämiseen. Tämä kenttä sisältää myös muuta informaatiota, jotka liittyvät tekstin

luokitteluun. (SC18-9878-04 2012, ss. 52-53) Kuvassa 37 esitellyistä kentistä järjestelmä perustaa automaattisesti itse *Categories*, *Filename*, ICM_*OrginalFileSize*, *ICM_Creation_Date*, *FilePath*, *FileModifiedTime* ja ICM_ExtractedText_Size.

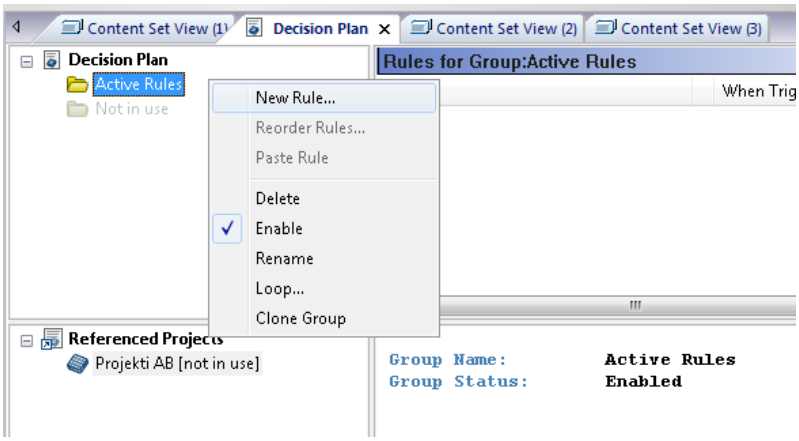
Käsittelysääntöjen määrittely

Käsittelysääntöjen tulee kuulua aina sääntöryhmään (eng. *rule group*) (kuva 38). Järjestelmässä voi olla useita ryhmiä ja niitä voidaan käyttää sääntöjen jakamiseen loogisiksi kokonaisuuksiksi.



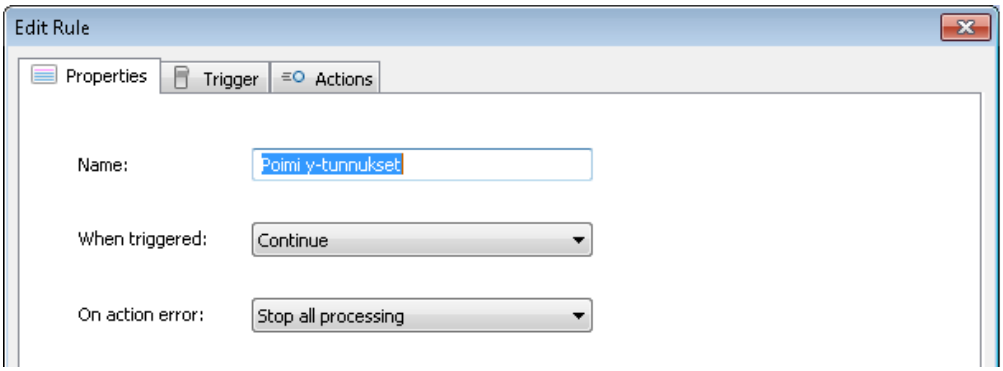
Kuva 38. *Sääntöryhmän perustaminen*

Käytössä olevat käsittelysäännöt voidaan poistaa käytöstä tai ottaa käyttöön ryhmäkohtaisesti (kuva 39). Uusien käsittelysääntöjen perustaminen ryhmään tapahtuu valitsemalla *New Rule* -toiminto (kuva 39).



Kuva 39. *Uuden käsittelysäännön luominen*

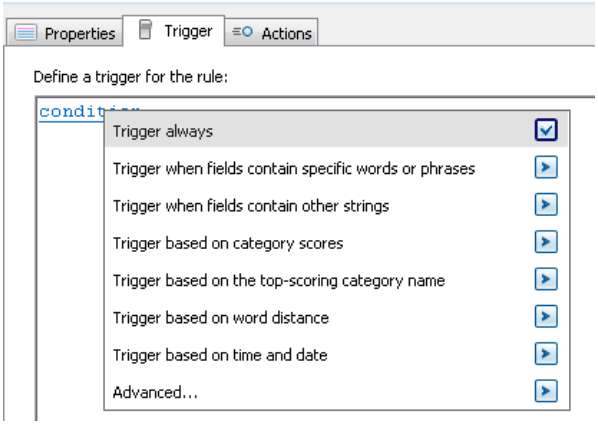
Luodulle käsittelysäännölle tulee asettaa nimi (kuva 39). Sen lisäksi sille voidaan valita, mitä ohjelma tekee, kun sääntöön liittyvät toimenpiteet on käsitelty (engl. *when triggered*). Lisäksi voidaan märittää, miten virhetilanteessa toimitaan (engl. *on action error*).



Kuva 40. *Uuden säännön perustaminen*

Kahdella edellä mainitulla asetuksella voidaan tarvittaessa keskeyttää sääntöjen käsittely, joko ryhmässä tai kokonaan. Virhetilanteessa myös yksittäisen säännön käsittely voidaan keskeyttää.

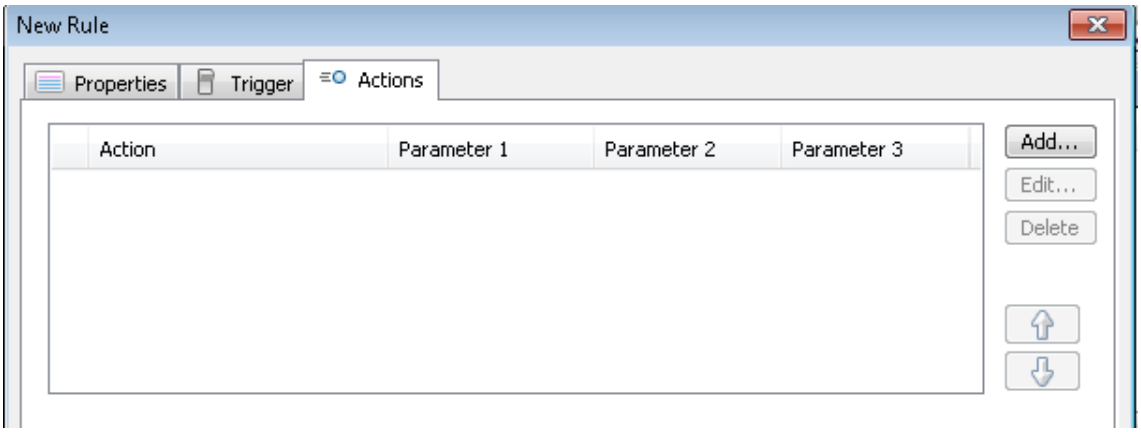
Välilehdeltä *Trigger* löytyvät käsittelysääntöön liittyvä ehtolauseke (kuva 39). Sen avulla ohjelmassa voidaan määrittää, millä ehdolla toimenpiteet valitaan suoritettavaksi.



Kuva 41. *Ehtojen määrittely*

Ehtoina järjestelmässä voidaan käyttää saatua luokittelua tai sen sisältämiä avaimia. Käytettyjä ehtoja voi olla myös useita, jolloin ne ketjutetaan toisiinsa *ja*- ja *tai* -operaatioiden avulla.

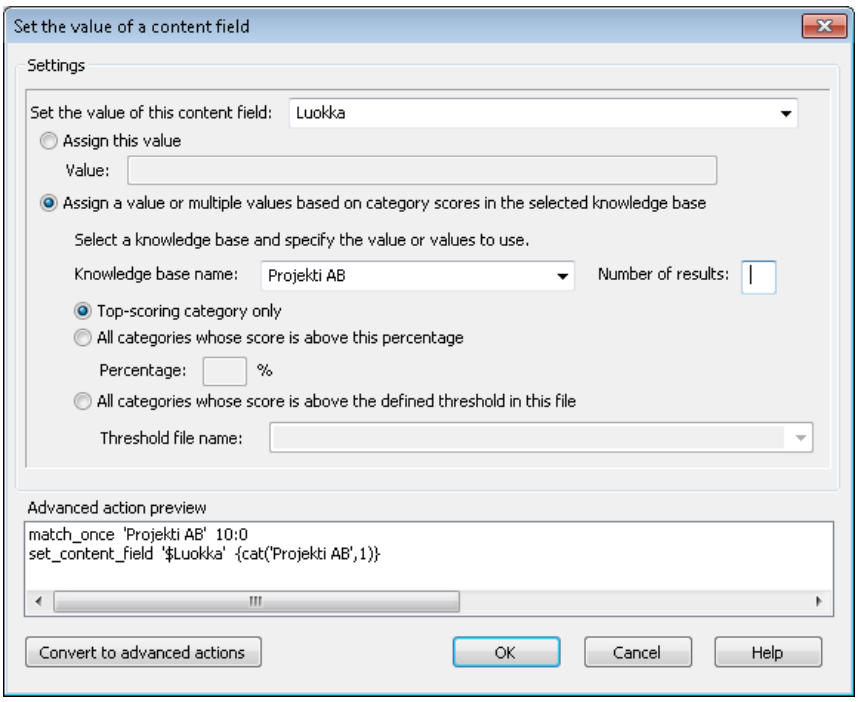
Actions-välilehdellä säännöistä voidaan määrittää ohjelman suorittamia toimenpiteitä, kun annettu ehto toteutuu. Uusia toimenpiteitä voidaan lisätä toiminnolla *Add* (kuva 42).



Kuva 42. *Käsittelysääntöön liittyvät toimenpiteet*

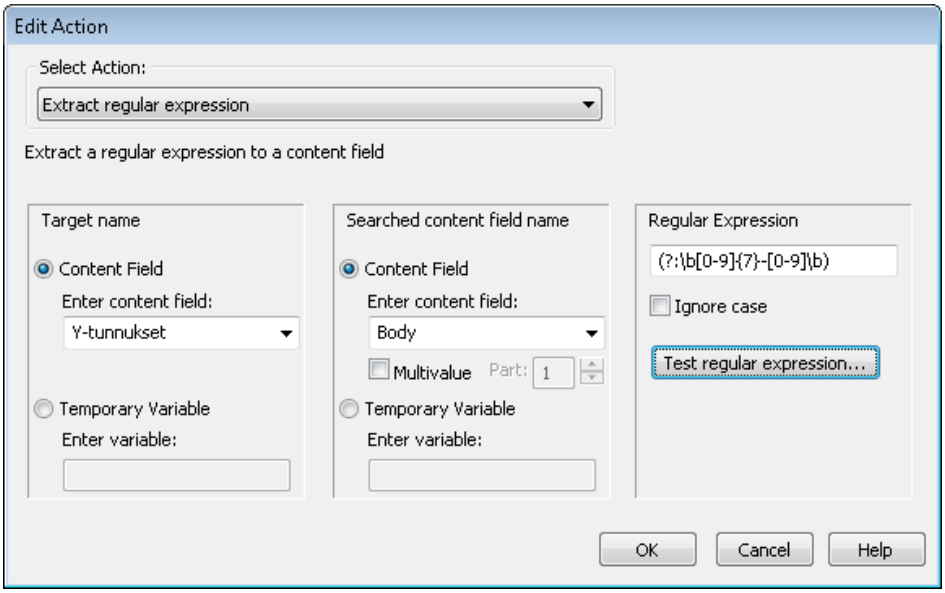
Seuraavaksi tarkoituksena on esitellä muutamia ohjelman tarjoamista toiminnoista avainsanojen ja luokittelun tuottamiseen. Tarkoituksena ei ole käydä ohjelman kaikkia toimintoja läpi. Rajaukseksi on valittu luokittelun hakeminen ja avainsanojen poimintaan käytetyt säännölliset lausekkeet ja sanalistat. Tämän lisäksi ohjelma tarjoaa useita muita vaihtoehtoja, joita ei ole tarkoitus käsitellä.

Luokan arvo voidaan määrittää kenttään luokka valitsemalla *Set the value of a content field* (kuva 43). Tämän jälkeen valitaan kenttä (engl. *content field*), johon luokan nimi kirjoitetaan. Tämän lisäksi määritellään, mitä tietämyskantaa käytetään luokan valintaan. Tarjottujen luokitteluvaihtoehtojen määrää voidaan rajata vain parhaiten sopivaan luokkaan tai palauttaa kaikki tietyn sopivuusprosentin ylittävät luokat.



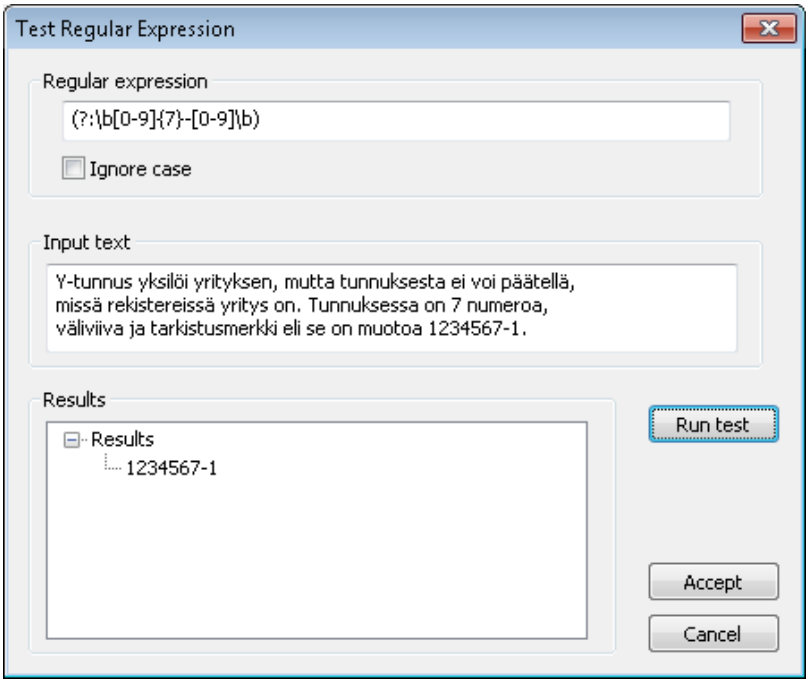
Kuva 43. *Luokittelun pyytäminen tietämyskannasta*

Määrämuotoisten sanojen tunnistamiseen käytettävissä on vaihtoehto *Extract regular expression* (kuva 44).



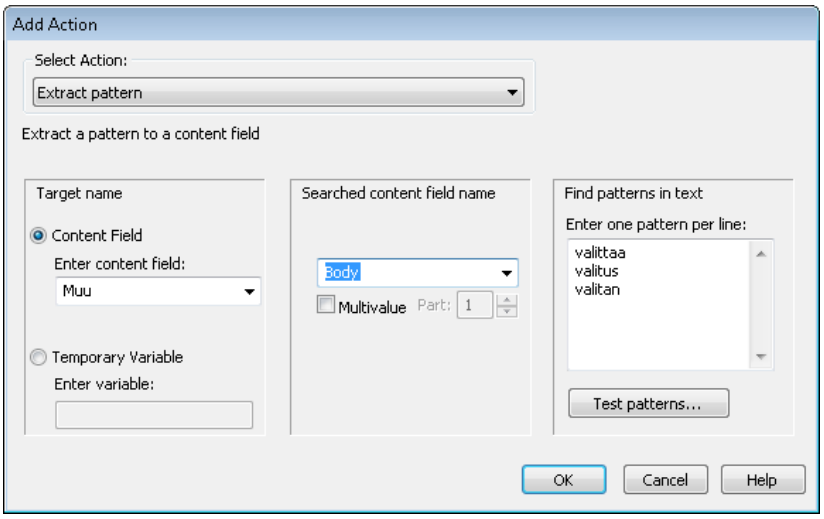
Kuva 44. *Extract regular expression*

Säännöllisten lausekkeiden testaamiseen ohjelma tarjoaa myös oman testipenkin (kuva 45). Tuolloin käytettyä lauseketta voidaan kehittää hyödyntäen mallia poimittavasta kohteesta.



Kuva 45. *Y-tunnuksen poiminta tekstistä*

Sanalistojen käsittelyyn voidaan käyttää toimintoa *Extract Pattern*. Sen avulla voidaan laatia yksinkertainen lista etsittävistä sanoista (kuva 46). Nämä tulee erottaa rivinvaihdolla toisistaan.



Kuva 46. *Sanalista*

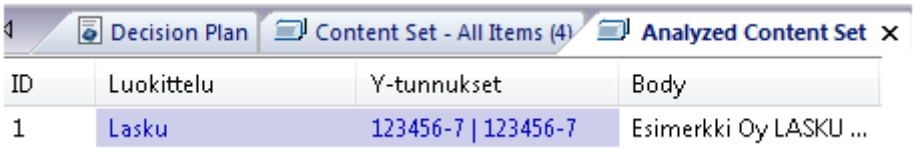
Näiden lisäksi ohjelmistosta löytyy joukko muita toimintoja avainsanojen poimintaan. Näihin kuuluu esimerkiksi avaimen sijainti tekstissä. Työn yhteydessä näitä menetelmiä ei ole katsottu tarpeelliseksi käsitellä.

Testaus ja käyttöönotto

Käsittelysäännöt voidaan testata ennen käyttöönottoa. Tätä varten järjestelmään tarvitsee tuoda testimateriaalia. Se voidaan tehdä painikkeella 1 (kuva 47).

Kuva 47. *Testaus ja käyttöönotto*

Järjestelmään tuodun testimateriaalin analysointi tapahtuu painikkeella 2. Tämän jälkeen järjestelmä kysyy esitettävät kentät, jonka jälkeen saaduista tuloksista muodostetaan raportti (kuva 48).

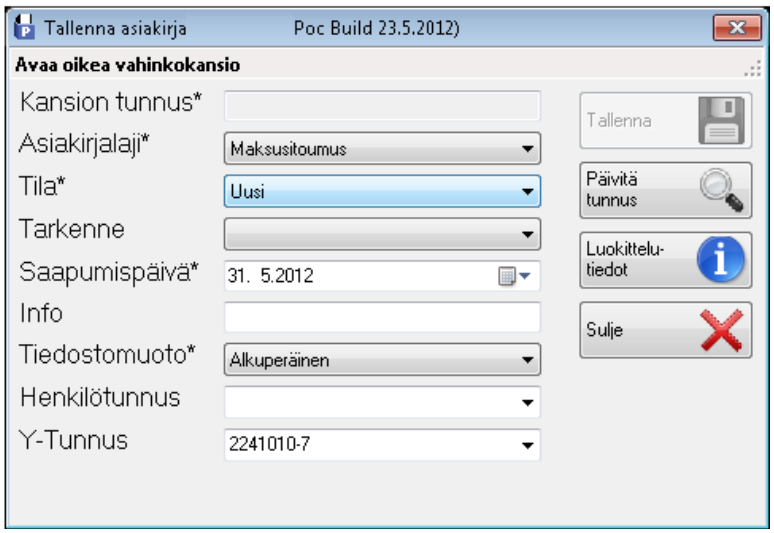


Kuva 48. *Poimitut avainsanat ja luokittelu*

Järjestelmän testaamisen jälkeen säännöt ovat valmiit käyttöönotettavaksi luokittelupalvelimella. Tämä voidaan tehdä kuvassa 47 näkyvällä painikkeella 3. Tämän jälkeen järjestelmä on valmis käyttöönotettavaksi sitä käytettävässä ohjelmistossa. (Zhu et al. 2009, ss. 87-97)

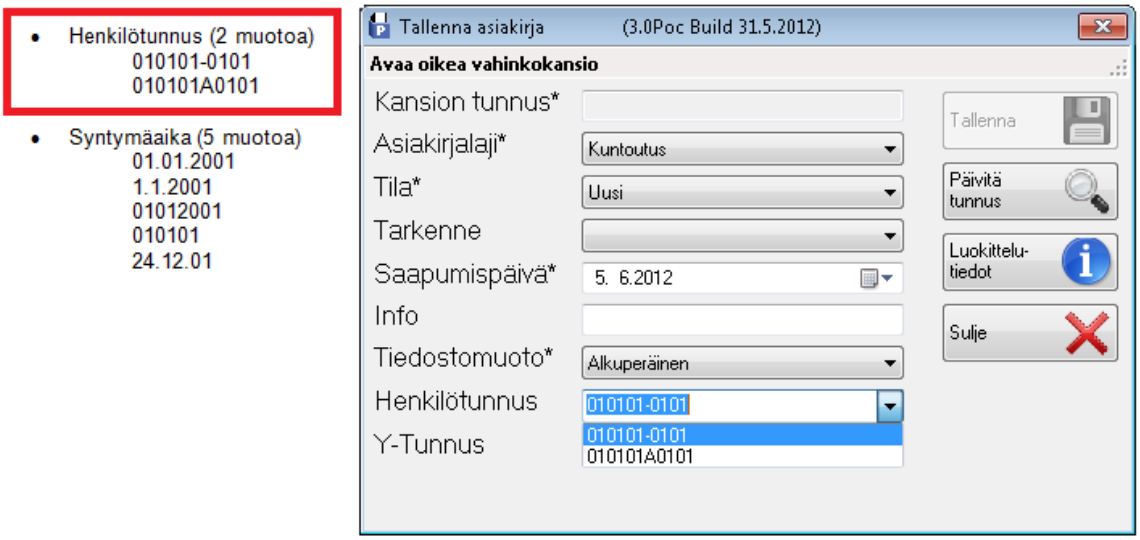
4.2.3 Office Integraatio -ratkaisu

Office Integraatio -ratkaisuun eli OFIv3:een liitettiin luokittelu ja avainsanojen poiminta *WebService*-rajapintaa hyödyntäen. Projektissa tavoitteena oli vähentää metadatan kirjaamiseen kuluvaa aikaa paikantamalla osa metadatasta automaattisesti. Projektissa tämä tarkoitti asiakirjalajin valintaa perustuen tilastolliseen luokitteluun. Avainsanojen osalta tavoitteena oli poimia ja esittää ne automaattisesti käyttöliittymässä (kuva 49).



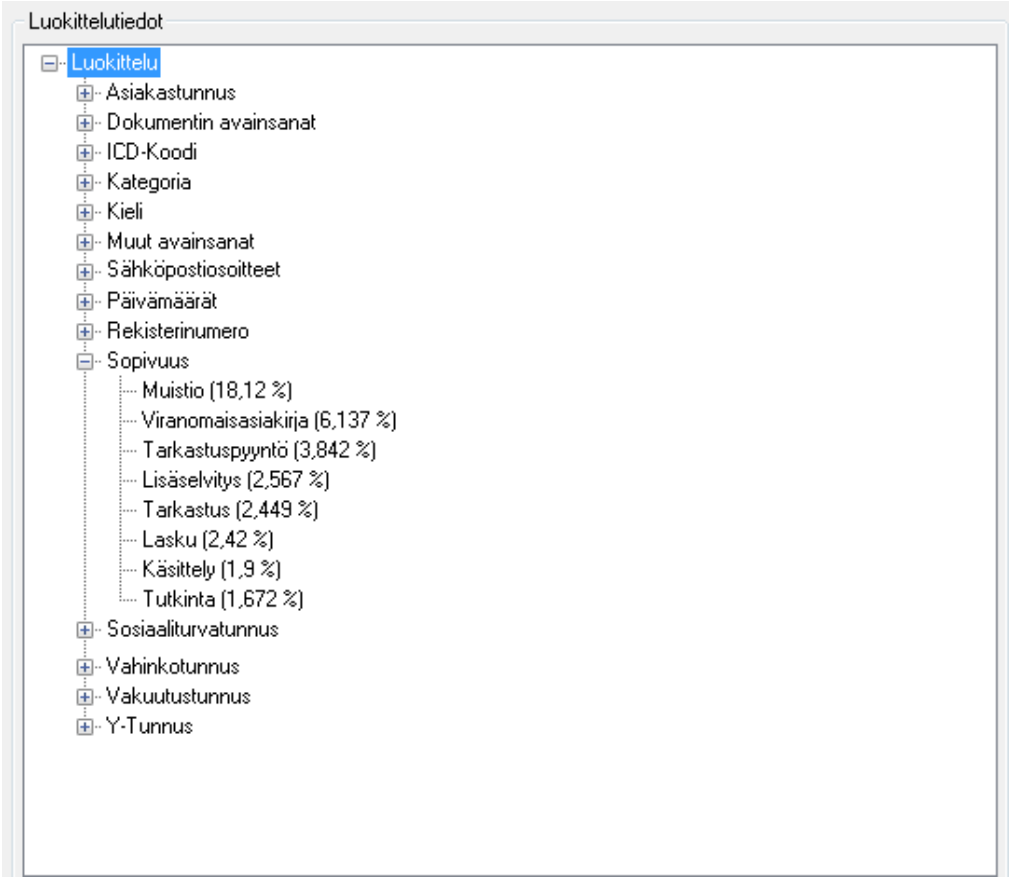
Kuva 49. *Office Integraatio Micrsoft Wordissa*

Avainsanojen osalta havaittiin, että asiakirjat voivat sisältää useita sopivia vaihtoehtoja, jotka voidaan löytää tekstistä (kuva 50). Tässä tilanteessa on vaikea tehdä päätös, mikä avainsanoista tulisi valita tallennettavan kentän arvoksi.



Kuva 50. *Avainsanat*

Ohjelmistoon lisättiin myös luokittelutiedot-painike, jonka avulla käyttäjät voivat tarkastella asiakirjasta kaikkia poimittuja avainsanoja ja sopivuutta luokkaan (kuva 51).



Kuva 51. *Poimitut metadata ja luokittelu*

Kokonaisuudessaan projektissa tutkittiin integraatiota Microsoft Word, Excel, PowerPoint ja Outlook -sovelluksista. Kokonaisuudessaan saadun palautteen perusteella integraatio Office -tuotteiden kanssa onnistui hyvin. Kaikki avainsanojen poimintaa liittyvistä vaatimuksista täyttyivät ja järjestelmä kykeni tuottamaan luokittelun arvioitaville dokumenteille.

4.3 Yhteenveto

Kirjoittajan näkemyksen mukaan tilastollinen luokittelu toimii varsin hyvin ottaen huomioon kieleen liittyvät monimutkaiset haasteet. Avainsanojen poiminnassa havaittiin haasteita, koska sopivia vaihtoehtoja oli useita. Kokonaisuudessaan projektista saatu palaute oli hyvää ja järjestelmää halutaan kehittää edelleen. Projektin aikana havaittiin seuraavat haasteet:

1. Jos poimittuja avainsanoja on useita, käsittelijä ei tiedä, mikä avainsana valitaan.
2. Excelin osalta päivämääriä ei kyetty tunnistamaan. Tämä johtui ohjelman sisäisestä tallennusmuodosta.
3. Osa Pdf-tiedostoista saapuu kuvamuodossa, jolloin järjestelmä ei kykene purkamaan niiden sisältämää informaatiota tekstiksi.
4. Luokittelupäätös ei voi riippua sellaisista ulkopuolisista tekijöistä, jotka eivät selviä dokumentin sisältämästä tekstistä.

Projektista saadut kokemukset olivat yleisesti rohkaisevia ja luokittelua ja avainsanojen poiminnan käyttöönottoa pohditaan laajemmin myös muihin kohdeyrityksessä käytettyihin tuotteisiin.

Luokittelujärjestelmän kannalta tutkittavassa kohteessa esiintyvät avaimet ratkaisevat, mihin luokkaan luokiteltava kohde kuuluu. Näiden avainten valinta perustuu tilastollisiin menetelmiin. Tilastollisten menetelmien tehtävänä on etsiä sellaiset avaimet, jotka erottavat eri luokkia toisistaan. Luokittelujärjestelmässä oppiminen perustuu näin ollen esimerkkien keräämiseen. Annettujen esimerkkien perusteella määritellään automaattisesti tilastollinen malli, jota käytetään luokan valintaan. Tämä edellyttää avainten esikäsittelyä NLP-menetelmillä, jotta avainten vertailu olisi mahdollista. Varsinainen luokittelupäätös tehdään käyttäen luokittelualgoritmia. Kirjoittaja ei kuitenkaan työn yhteydessä tarkoituksellisesti halua ottaa kantaa ohjelmistossa käytettyyn luokittelualgoritmiin, sillä IBM ei ole tätä ilmoittanut. Käytetyssä algoritmissa on kuitenkin kysymys tekstissä esiintyvien avainten vertailemisesta luokkaan tyypillisesti kuuluviin avaimiin.

Avainsanojen poiminta perustuu kiinteiden sääntöjen määrittelyyn. Näiden sääntöjen perusteella voidaan valita sellaiset merkkijonot, jotka täyttävät tietyn määritellyn kriteerin. Kriteereinä merkkijonojen poimintaan voidaan käyttää esimerkiksi merkkijonon sijaintia tai muotoa. Avainsanojen poiminta on tehokas tapa poimia merkkijonoja, joiden muoto tai sijainti voidaan rajata tekstissä. Tähän ohjelmistossa on mahdollista käyttää Regular expression -kieltä, sanakirjaa ja avainten sijaintia.

5 YHTEENVETO JA JOHTOPÄÄTÖKSET

Työn tavoitteet olivat:

A. Avata luokitteluun ja tiedonhakuun liittyvät keskeiset käsitteet lukijalle
B. Selvittää tiedonhakua konseptitasolla ja käytännössä
C. Esitellä toimintamalleja ja periaatteita, joita tekstin luokittelemiseen tarvitaan
D. Etsiä millaisia rajoituksia tiedonhaku- ja luokittelutekniikoihin liittyy
E. Selvittää tilastollista luokittelua ja avainsanojen poimintaa käytännön ympäristössä

Yhteenveto työn tuloksista

Työssä käsitellään tekstin luokitteluun liittyviä keskeisiä konsepteja niin käytännön kuin teorian avulla. Työssä analysoitiin myös erityyppisten datan rakenteiden vaikutusta informaation löytämiseksi tietovarastosta. Työssä selvitettiin laajasti, mitä luokittelulla, metadatalla ja tiedonhaulla oikeastaan tarkoitetaan. Yksi näkökohta oli myös menetelmien kehitys. Sen perusteella voitiin huomata, että nykyiset menetelmät hyödyntävät jo muinaisissa tietovarastoissa käytettyjä periaatteita. Nämä periaatteet ovat kehittyneet länsimaisiin tieteisiin pitkän ajan kuluessa. Suurin syy menetelmien kehittymiseen näyttää olleen motivaatio hyödyntää saatavilla olevaa informaatiota tehokkaammin. Tiedonhaun teorialla työssä oli tarkoituksena perustella ja esittää, miksi tarvitsemme luokittelua ja avainsanoja osana tietovarastojen hallintaa. Tavoitteena oli selvittää kahden eri menetelmän eroja informaation paikantamiseksi. Käsitellyt menetelmät olivat informaation hakeminen rakenteettomista aineistosta tiedonhaun menetelmin ja sisältöä kuvaavan metadatan käyttö osana tietovarastoja. Työssä tiedonhaun menetelmiä käsiteltiin niin konsepteina, kuin käytännön esimerkkien avulla. Niiden yhteydessä käsiteltiin kahta keskeisintä rajausmenetelmää, jotka olivat vektori- ja Boolen hakumalli. Rakenteen hyödyntämiseen perustuvaa metadataa käsiteltiin tietovarastojen hallinnan näkökulmasta. Samassa yhteydessä arvioitiin myös metadatan ja luokittelun välistä yhteyttä. Molempien menetelmien käyttöönoton huomattiin olevan keskeisessä asemassa informaation paikantamiseksi nykytietovarastoista. Samalla huomattiin, että metadata antaa varsin rajalliset mahdollisuudet informaation hakemiseksi dokumentin sisältä. Tästä syystä myös rakenteettomia menetelmiä tarvitaan osana sisällönhallintaa. Yhdessä nämä kaksi menetelmää muodostavat organisaation kannalta kokonaisvaltaisen ratkaisun sisällönhallintaan. Tekstin tilastollisen luokittelun osalta työssä selvitettiin työvaiheita ja menetelmiä. Työssä käsiteltiin kahta merkittävää konseptia, jotka olivat NLP-menetelmät ja luokittelualgoritmit. NLP-menetelmien voitiin huomata ratkaisevan haasteita, jotka liittyvät merkkijonojen vertailuun tietokoneen näkökulmasta. NLP-menetelmät ovat välttämättömiä, koska tietokone ei kykene yhdistämään kahta eritavoin kirjoitettua merkki-

jonoa automaattisesti toisiinsa. Luokittelumenetelmien osalta työssä tutkittiin päätöspuita ja naiivia Bayesian -päättelyä. Molemmat algoritmit todennettiin hyödyntäen teoriaa ja esimerkkejä. Työn aikana luokittelujärjestelmistä löydettiin kolme merkittävää rajoitusta. Ensimmäinen rajoitus oli, ettei tekstin luokittelijalla ole ihmiselle ominaisia abstraktiotasoja. Tällä työssä viitataan teoriaan sanoihin liitettävistä prototyypeistä. Samalla se kertoo, että luokittelujärjestelmämme sisältää useita eri tasoja. Tietokoneen mallissa näitä tasoja on vain yksi, koska avainta ei voida yhdistää korkeammalla abstraktiotasolla oleviin konsepteihin. Tämän rajoituksen poistaminen voisi kirjoittajan mukaan tuoda tietokoneen lähemmäksi todellisuutta. Tuolloin voisimme käyttää prototyyppiä nimeltä fyysinen objekti viittamaan todellisiin käsitteisiin, kuten tuoli, pöytä, työkalu tai sänky. Tämän tyyppisten rajoituksen poistaminen voisi auttaa myös hakujärjestelmiä. Toinen havaittu rajoitus oli, että aineisto tulee voida jakaa klustereihin. Se tarkoittaa, että dokumentit tulee voida erottaa toisistaan niissä käytettyjen sanojen perusteella. Luokittelualgoritmit eivät yleisesti myöskään ottaneet huomioon sanojen sijaintia. Tietyissä tapauksissa merkitsevimmät sanat löytyvät kuitenkin dokumentin alusta. Näistä haasteista huolimatta johtopäätös on, että algoritmit toimivat varsin hyvin. Työssä tutkittiin luokittelua käytännössä *IBM Content Classification Modulen* avulla. Tuotteen tutkimisessa hyödynnettiin asiakasprojektista saatua kokemusta ja siitä saatua palautetta. Projektista saatu palaute oli myös rohkaisevaa ja asiakas haluaa sitoutua järjestelmän jatkokehitykseen. Projektin aikana havaittiin luokittelun toimivan myös käytännössä. Avainsanoihin voitiin huomata kuitenkin liittyvän rajoituksia. Projektin yhteydessä havaitut rajoitukset olivat:

1. Jos poimittuja avainsanoja on useita, käsittelijä ei tiedä, mikä avainsana valitaan.
2. Excelin osalta päivämääriä ei kyetty tunnistamaan. Tämä johtui ohjelman sisäisestä tallennusmuodosta.
3. Osa Pdf-tiedostoista saapuu kuvamuodossa, jolloin järjestelmä ei kykene purkamaan niiden sisältämää informaatiota tekstiksi.
4. Luokittelupäätös ei voi riippua sellaisista ulkopuolisista tekijöistä, jotka eivät selviä dokumentin sisältämästä tekstistä.

Avainsanojen osalta kirjoittajan suositus on esittää lause tai muutama edeltävä sana, josta avainsana on poimittu. Tuolloin käsittelijä saa lisää informaatiota oikean avainsanan valintaan. Toinen vaihtoehtoehto on ehdottaa useimmiten dokumentissa käytettyä avainsanaa. Tästä huolimatta konteksti, jossa avainsana esitetään, voi olla tarpeellista esittää. Excelin ja Openofficen käsittelemien tiedostojen xlsx-tiedostojen osalta päivämäärään liittyvät rajoitukset voidaan purkaa erillisellä ohjelmakoodilla. Kyseessä on avoin tiedostomuoto, jonka sisältö on XML:ää. Pdf- ja kuvatiedostojen osalta suositus on käyttää OCR-tunnistusta tekstille (engl. *optical character recognition*). OCR-tunnistuksen osalta tiedostoille tulisi suorittaa myös virheenkorjaus tuloksena saatuun tekstiin, sillä varsinkin käsinkirjoitetun tekstin osalta voi syntyä huomattavia kirjoitusvirheitä. Näiden tiedostojen osalta tulisi myös miettiä, mikäli niitä voidaan hyödyntää koulutusmateriaalina virheistä johtuen. Tätä aihealuetta ei kuitenkaan työn yhteydessä ollut mahdollista tutkia tarkemmin.

Tulevaisuus

AI-tekniikoiden (engl. *artificial intelligence*) tutkiminen on keskeisessä roolissa, jotta tulevaisuuden tietovarastoja voitaisiin hyödyntää entistä tehokkaammin. Aihealuetta koskeva tutkimus on nostanut myös mielenkiintoisen kysymyksen, mikä todellisuudessa erottaa ihmisen ajattelun koneista. Tätä asiaa on pohtinut matemaatikko Alan Turning artikkelissaan *Can machines think?* 1950-luvulla. Turning kehitti erityisen testin, jonka tarkoitus on mitata ihmisen ja koneen välistä eroa. Turningin testi (TT) perustuu ideaan esittää kahdelle testiin osallistuvalle kohteelle vapaasti eri kysymyksiä. Esitettyjen kysymyksien perusteella kysymysten esittäjälle pitäisi paljastua, onko kyseessä kone vai ihminen. Koneen tavoite on matkia ihmistä niin, ettei testin tekijä pysty selvittämään kysymysten perusteella vastausta tähän kysymykseen. Turningin testin läpäiseminen voidaan nähdä AI-tekniikoiden lopulliseksi tavoitteeksi. Testiä ei kuitenkaan ole kyetty tähän päivään mennessä ratkaisemaan (Saygin 2000, ss. 465-512). Tilastollisten menetelmien hyödyntämisestä tiedonlouhinnassa on saatavilla innostavia esimerkkejä. Tästä hyvänä esimerkkinä toimii IBM:n kehittämä Watson, joka voitti ihmiset Jeopardy-tietovisassa (IBM A 2012).

Havaitut jatkotutkimuskohteet

Jos ihminen luokittelee asioita, niin abstraktiotasoja on useita. Mallintamalla osa ihmisen abstraktiotasoista voitaisiin kehittää myös luokittelu- ja hakujärjestelmien kyvykkyyttä. Tällä tavoin tietokone osaisi yhdistää esimerkiksi merkkijonon *auto* merkkijonoon *ajoneuvo*. Tämän rajoituksen poistaminen voisi tuoda nykyiset järjestelmät lähemmäksi todellisuutta. Aihealue on mielenkiintoinen ja se edellyttää lisää tutkimusta.

Nimien, katujen ja paikkojen tunnistamiseen käytettyjä ohjelmistoja ei löytynyt suomen kielelle. Tämän tyyppisistä ohjelmistoista voisi olla hyötyä useissa organisaatioissa. Samalle teknologialle olisi käyttöä myös useiden tietovarastojen yhteydessä, kun halutaan vastata kysymyksiin kuka ja missä. Kirjoittaja aikoo jatkaa aihealueen tutkimista mahdollisesti tulevien projektien yhteydessä.

LÄHTEET

Abiteboul, S., Hull, R. & Victor, V. 1994. Foundations of Databases, 685 p.

Aggarwal, C. & Zhai C. 2012. Mining Text Data. Springer, 533 p.

Alaterä, A., Halttunen, K. & Sormunen, E. 2005. Sisällönkuvailu: luokitus ja indeksointi. [WWW]. [Viitattu 5.10.2012]. Saatavissa: http://oppimateriaalit.internetix.fi/fi/avoimet/0viestinta/informaatiotutkimus/tiedon_organisoinnin/luku6/

Arkistolaitos. 2012. Keskeisiä käsitteitä. [WWW]. [Viitattu 1.8.2012]. Saatavissa: http://www.arkisto.fi/normit/arkistolaitoksen-suositus-arkistonmuodostussuunnitelmaan-termit/

Bargmeyer, B. & Gillman, D. 2012. Metadata Standards and Metadata Registeries. [WWW]. [Viitattu 3.7.2012]. Saatavissa: http://www.bls.gov/ore/pdf/st000010.pdf
BBM. 2013. Asiakirjamalli. [WWW]. [Viitattu 20.3.2013]. Saatavissa: http://newsletter.bbm.fi/Default.aspx?tabid=2906

Blumberg, R. & Atre, S. 2003. The Problem with Unstructured Data. [WWW] . [Viitattu 1.7.2012]. Saatavissa: http://soquelgroup.com/Articles/dmreview_0203_problem.pdf

Borthwick, A. 1999. A Maximum Entropy Approach to Named Entity Recognition. New York University, Computer Science Department. 105 p.

Bunescu, R. & Mooney, R. 2005. Statistical Relational Learning for Natural Language Information Extraction. MIT Press. 19 p.

Bush, V. 1979. As We May Think. ACM SIGPC Notes 1(1979)4, pp. 36-44.

Butler, L. 2012. The Top 5 Mistaken Beliefs About Content Management [WWW]. [Viitattu 25.8.2012]. Saatavissa: http://www.ecmforhighereducation.com/2012/01/30/the-top-5-mistaken-beliefs-about-content-management/

Chen, S. & Goodman, J. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. Proceedings of the 34th annual meeting on Association for Computational Linguistics, USA. pp. 310-318

Cohen, H. & Lefebvre, C. 2005. Handbook of Categorization in Cognitive Science.

Cortes, C. & Vapnik, V. 1995. Support-Vector Networks. Machine Learning. Volume 20, Issue 3. pp. 273-297.

Davis, M. 2003. [WWW]. Cambridge. Cognition and Brain Sciences Unit. 30.10.2003. [Viitattu 1.3.2013]. Saatavissa: http://www.mrc-cbu.cam.ac.uk/people/matt.davis/Cmabrigde/

Edghill, E. 1994. On Interpretation. [WWW]. [Viitattu 1.7.2012]. Saatavissa: http://classics.mit.edu/Aristotle/interpretation.html

Edmunds, A. & Morris, A. 2000. The problem of information overload in business organisations: a review of the literature. International Journal of Information Management pp. 17-28

Farrance, R. 2012. Timeline: 50 Years of Hard Drives. [WWW]. [Viitattu 23.9.2012]. Saatavissa: http://www.pcworld.com/article/127105/article.html

Feather, J. 1998. In The information society: A study of continuity and change.

Fisher, M. & Sheth, A. 2012. Semantic Enterprise Content Management. [WWW]. [Viitattu 28.7.2012]. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.195.9032&rep=rep1&type=pdf

Francis, W. 1973. The Beginnings of Indexing and Abstracting: Some notes towards a history of Indexing and Abstracting in Antiquity and the middle Ages. The Indexer. Vol. 8. No. 4. pp 193-198

Gilliland, A. Setting the Stage. J. 2008. Introduction to Metadata. In: Gill, T., Whalen, M. & Woodley, M. Vol. 3, Paul Getty Trust. pp. 1-19

Golding, A. 1995. A Bayesian hybrid method for context-sensitive spelling correction. Association for Computational Linguistics. pp 39-53

Goyvaerts, J. 2009. Tutorial. [WWW]. [Viitattu 20.3.2013]. Saatavissa: http://www.regular-expressions.info/

Gupta, G. 2012. Introduction to Data Mining with Case Studies. Second Edition. New Delhi. PHO Learning Private Limited. 489 p.

HGST Data Sheet Ultrastar 7K4000. 2012. [WWW]. [Viitattu 23.9.2012]. Saatavissa: http://www.hgst.com/tech/techlib.nsf/techdocs/FD3F376DC2ECCE68882579D40082C393/$file/US7K4000_ds.pdf

Huijsmans, D. & Dionysius, P. 2001. Extended Performance Graphs for Cluster Retrieval. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of The 2001 IEEE Computer Society Conference on. Leiden Institute of Advanced Computer Science. pp. 26-31.

IBM 2012 a. Faqs. [WWW]. [Viitattu 23.9.2012]. Saatavissa: http://www.research.ibm.com/deepqa/faq.shtml

IBM. 2012 b. Glosary. [WWW]. [Viitattu 10.9.2012]. Saatavissa: http://pic.dhe.ibm.com/infocenter/classify/v8r8/index.jsp?topic=%2Fcom.ibm.classify.pic.dhe.ibm.com%2Finfocenter%2Fclassify%2Fv8r8%2Ftopic%2Fcom.ibm.classify.common.doc%2Fbnrgl000.htm

IBM. 2009. Classification Module: Make It Work for You. IBM. 439 p.

ISO704. 2009. Terminology work - Principles and methods. [WWW]. [Viitattu: 28.7.2012]. Saatavissa: ftp://ftp.omg.org/pub/sbvr-rtf/ISOStandards/ISO704_2009.pdf

István, P. 2005. Text Categorization and Support Vector Machines. [WWW]. [Viitattu 30.3.2013]. Saatavissa: http://conf.uni-obuda.hu/mtn2005/Pilaszy.pdf

Kapitzke, C. 2001. Ceremony and cybraryDigital libraries and the dialectic of place and space. Social Alternatives. [WWW]. [Viitattu 12.9.2012]. Saatavissa: http://eprints.qut.edu.au/43995/

Klein, D. & Manning, C. 2003. Maxent Models, Conditional Estimation, and Optimization. [WWW]. [viitattu 4.2.2013]. Saatavissa: http://nlp.cs.berkeley.edu/tutorials/maxent-tutorial-slides-6.pdf

Korenius, T., Laurikkala, J., Järvelin, K, & Juhola, M. 2004. Stemming and lemmatization in the clustering of finnish text documents. CIKM '04 Proceedings of the thirteenth ACM international conference on Information and knowledge management, CIKM '04, pp. 625-633 .

Kotimaisten kielten keskus. 2012 a. Kirjoitetun suomen kielen sanojen taajuuksia. [WWW]. [Viitattu 12.12.2012]. Saatavissa: http://kaino.kotus.fi/sanat/taajuuslista/parole.php

Kotimaisten kielten keskus. 2012 b. Kielitoimisto. [WWW]. [Viitattu 4.1.2013]. Saatavissa: http://www.kotus.fi/index.phtml?s=110

Kummamuru, K. 2012. Text Mining & Question Answering. [WWW]. [Viitattu 12.3.2013]. Saatavissa: http://krishnarajpm.com/bigdata/krishna.pdf

Kuronen, T. 1998. Hajautettu dokumenttien hallinta. [WWW]. [Viitattu 22.8.2012]. Saatavissa: http://herkules.oulu.fi/isbn951425242X/html/book1.html

L 25.4.2003/325. Arvonlisävero laki.

Liddy, E. 2001. System and method for classifying text. [WWW]. [Viitattu 9.9.2012]. Saatavissa: http://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub

Luojola, T. 2006. Kielitieteellisen aineiston kvantitatiiviset analyysimenetelmät [WWW]. [Viitattu 9.3.2013]. Helsingin yliopisto, yleisen kielitieteen laitos. Saatavissa: http://www.ling.helsinki.fi/~fkarlsso/methods/kvant_men.pdf

Maimon & Rokach 2010. Data Mining and knowledge discovery handbook. Springer. 1035 p.

Manktelow, M. 2012. History of Taxonomy. [WWW]. [Viitattu 1.10.2012]. Saatavissa:http://atbi.eu/summerschool/files/summerschool/Manktelow_Syllabus.pdf

Manning, C. & Schütze, H. 1999. Foundations of Statistical Natural Language Processing. London, England. The MIT Press. 680 p.

Manning, C. 2013. Information Extraction and Named Entity Recognition. [WWW]. [Viitattu 22.3.2013]. Saatavissa: http://www.stanford.edu/class/cs124/lec/Information_Extraction_and_Named_Entity_Recognition.pdf

Manning, C., Raghavan, P. & Schütze, H. 2008. Introduction to Information Retrieval. 483 p.

Mähönen, M. & Suominen, M. 2012. Asiakasvaatimusmääritelmä. Ei julkisesti saatavilla.

Nicola, M, & John, J. 2003. Xml parsing: A threat to database performance. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM). pp. 175–178.

Niiniluoto, I. 1996. Informaatio, tieto ja yhteiskunta: Filosofinen käsiteanalyysi. 5. täydennetty painos. 136 p.

Niso. 2001. Understanding Metadata. [WWW]. [Viitattu 3.7.2012]. Saatavissa: http://www.niso.org/publications/press/UnderstandingMetadata.pdf

NRSI. 2001. Character set encoding basics. [WWW]. [Viitattu 1.9.2012]. Saatavissa: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=IWS-Chapter03#79e846db

Nurmi, T. 2004. Julkishallinnon nimet [WWW]. [viitattu 1.2.2013]. Saatavissa: http://webcgi.oulu.fi/oykk/abc/kielenhuolto/oikeinkirjoitus/alkukirjain/julkishallinnon_nimet/

Oracle 2012. Object-Oriented Programming Concepts. [WWW]. [Viitattu 1.7.2012]. Saatavissa: http://docs.oracle.com/javase/tutorial/java/concepts/index.html

Oracle. 2013. Lesson: Regular Expressions. [WWW]. [Viitattu 20.3.2013]. Saatavissa: http://docs.oracle.com/javase/tutorial/essential/regex/index.html

Patentti- ja rekisterihallitus. 2010. Yritys- ja yhteisötietojärjestelmä (YTJ) ja Y-tunnus. [WWW]. [Viitattu 20.3.2013]. Saatavissa: http://www.prh.fi/fi/kaupparekisteri/yleista/ytj.html

PcMag. 2012. Definition of: Boolean logic. [WWW]. [Viitattu 4.10.2012]. Saatavissa: http://www.pcmag.com/encyclopedia_term/0,1237,t=Boolean+logic&i=38836,00.asp

Pellegrin, P. 1986. Aristole's Classification Of Animals. University of California Press. 229 p.

Rawlinson, G. 2007. The Significance of Letter Position in Word Recognition. IEEE A&E Systems Magazine, pp. 26-29.

Rayner, K., White, S. Johnson, R. Liversedge, S. 2006. Raeding wrods with jubmled letetrs: There’s a cost. Psychological Science. Vol 17. pp. 192-193.

Rosch, E. & Lloyd, B. 1978. Cognition and categorization. 1st edition. Lawrence Erlbaum. 336 p.

Rouse, R. 2012. Definition Database. [WWW]. [Viitattu 22.8.2012]. Saatavissa: http://searchsqlserver.techtarget.com/definition/database

Rud, O. 2009. Business Intelligence Success Factors., Wiley & SAS Business Series. 283 p.
Rybinski, H. 1987. On First-Order-Logic Databases. ACM Transactions on Database Systems.12(1987)3, pp. 325 - 349.

Salminen A. 2005. Metatiedot organisaatioiden sisällönhallinnassa. [WWW]. [Viitattu 22.8.2012]. Saatavissa: http://users.jyu.fi/~airi/papers/Metatietoartikkeli-2005.pdf

Salton, G., Fox, E. & Wu, H. 1983. Extended Boolean Information Retrieval. Communications of the ACM 26(1983)12, pp. 1022-1036.

Sapir, E. 1921. Language: An introduction to the study of speech. Mariner Books. 252 p.

Savolainen, E, Finn Lectura. 2001. Muoto-oppi eli morfologia [WWW]. [Viitattu 2.2.2013]. Saatavissa: http://www.finnlectura.fi/verkko/esittely/sivu2.htm

Saygin, A., Cicekli, I. & Akman, V. 2000. Turing Test: 50 years later. Minds and Machines. 2000. Vol. 10, No. 4. pp. 463–518.

SC18-9878-04. 2012. Configuration Guide. Ibm. 222 p.

Shilakes B. & Tylman, J. 2012. Enterprise Information Portals. Merrill Lynch. [WWW]. [Viitattu 3.7.2012]. Saatavissa: http://ikt.hia.no/perep/eip_ind.pdf

Singhal, A. 2001. Modern Information Retrieval: A Brief Overview. [WWW]. [Viitattu 13.9.2012]. Saatavissa: http://singhal.info/ieee2001.pdf

Sint, R., Schaffert, S., Stroka, S. & Ferstl, R. 2012. Combining Unstructured, Fully Structured and Semi-Structured Information in Semantic Wikis. [WWW]. [Viitattu 22.8.2012]. Saatavissa: http://ceur-ws.org/Vol-464/paper-14.pdf

Smith, B. 2000. Ontology. [WWW]. [Viitattu 29.7.2012]. Saatavissa: http://ontology.buffalo.edu/smith/articles/ontology_pic.pdf

Smith, R. 2012. Aristotle's Logic. [WWW]. [Viitattu 29.7.2012]. Saatavissa: http://plato.stanford.edu/entries/aristotle-logic/

The Lexiteria. 2010. English Word Frequency 2010. [WWW]. [Viitattu 9.3.2013]. Saatavissa: http://www.lexiteria.com/word_frequency/english_word_frequency_list.html

Toutanova, K. & Manning, C. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. Association for Computational Linguistics, Volume 13, pp. 63-70.

Vatanen, T., Väyrynen, J. & Virpioja, S. 2010. Language Identification of Short Text Segments with N-gram Models. Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, pp. 3423-3430.

Verner M. & Baillet, A. Adrien Baillet. 1968. Baillet and his rules for an alphabetical subject catalog, Library Quarterly 38 (3) (1968) pp. 217-230.

W3Schools. 2012 a. Why Use XML Schemas. [WWW]. [Viitattu 10.9.2012]. Saatavissa: http://www.w3schools.com/schema/schema_why.asp

W3Schools. 2012 b. Introduction to DTD. [WWW]. [Viitattu 10.9.2012]. Saatavissa: http://www.w3schools.com/dtd/dtd_intro.asp

Walker, C. 2002. Positive Atheism's Big List of Thomas Jefferson Quotations. [WWW]. [Viitattu 11.3.2013]. Saatavissa: http://www.positiveatheism.org/hist/quotes/jefferson.htm

Weglarz, G. 2004. Two Worlds of Data – Unstructured and Structured. [WWW]. [Viitattu 3.7.2012]. Saatavissa: http://www.information-management.com/issues/20040901/1009161-1.html

Weisstein, Eric W. 2013. Bernoulli Distribution. [WWW]. [Viitattu 28.3.2013]. Saatavissa: http://mathworld.wolfram.com/BernoulliDistribution.html

Welty, C. & Jenkins, J. Formal ontology for subject. 1999. Data & Knowledge Engineering 31. pp. 155-181.

Yang, Y. & Pedersen, J. 1997. A Comparative Study on Feature Selection In Text Categorization. Morgan Kaufmann Publishers. pp. 412-420

Zhu, J., Alon, T., Arkus, G., Duran, R., Haber, M., Liebke, R., Morreale, F., Roth, I. & Sumano, A. 2011. Metadata Management with IBM InfoSphere Information Server, IBM. 434 p.

Zhu, W., Barron S., Gallotti, M., Gupta, V., Wang, X., Magdalen, J. & Singer, J. 2009.

Ölvecký, T. 2005. N-Gram Based Statistics Aimed at Language Identification. April 2005. IIT.SRC. Bratislava, pp. 1-17.