Suomenkielisten lehtiartikkelien luokittelu

Mikko Silvonen

Pro gradu -tutkielma
Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Helsinki 30.9.1998
C-1998-58

Sisällys

1 Johdanto

1.1 Mihin luokittelua tarvitaan?

Dokumenttien luokittelu (englanniksi *document classification* tai *text categorization*) on tiedonhaun osa-alue, jonka tarkoituksena on helpottaa oikean tiedon löytymistä lisäämällä dokumentteihin tieto niiden aihepiireistä.

Ennen tietokoneaikakautta luokittelun pääasiallisia käyttäjiä olivat kirjastot ja sanomalehtien leikearkistot. Nykyään näiden rinnalle ovat nousemassa uutistoimistot, hakupalvelut, suuryritykset ja muut suurten tekstiarkistojen ylläpitäjät. Luokiteltavat tekstit voivat olla esimerkiksi uutisia, lehtiartikkeleita, kirjeitä tai yrityksen sisäisiä dokumentteja. Luokittelua on ehdotettu käytettäväksi myös sähköpostin ohjaamiseen oikealle henkilölle viestin sisällön perusteella.

Useimmat yleisessä käytössä olevat hakuohjelmistot (tunnetuimpana esimerkkinä AltaVista[1]) perustuvat nykyään pelkkään **tekstihakuun** (*full-text search*) jossa dokumentin kaikki sanat talletetaan indeksiin hakutermeinä käytettäviksi. Satunnaisen käyttäjän tarpeisiin tekstihaku on yleensä riittävä, mutta ammattimainen käyttäjä voi kaivata mahdollisuutta kohdistaa haku vain tiettyä aihetta käsitteleviin teksteihin. Pelkällä tekstihaulla esim. morfologiaa eli sananmuodostusta käsittelevien kielitieteellisten dokumenttien löytäminen on hankalaa, koska sana *morfologia* on käytössä monella tieteenalalla (kuva 1). Tällaisissa hauissa luokittelusta olisi selvästi hyötyä.

Kuva 1. AltaVista-haku morphology*:*
40 ensimmäistä osumaa 5.1.1998

Aihepiiri	Osumia
biologia	26
kielitiede	4
kuvankäsittely	4
geologia	2
rakennustekniikka	1

[1] `http://www.altavista.digital.com/`

1.2 Luokittelutapoja

Alun perin dokumentteja ovat luokitelleet tehtävään koulutetut **asiantuntijat**. Ihminen ymmärtää lukemansa ja tekee harvoin karkeita virheitä luokittelussa. Käsin luokitteleminen on kuitenkin hidasta. Haittana voi olla myös ihmisen subjektiivisuus: jos luokittelijoita on useita, voi olla vaikeaa omaksua yhtenäisiä luokitteluperiaatteita, ja yhdenkin luokittelijan mieltymykset saattavat muuttua ajan mukana.

Dokumenttikokoelmien kasvaessa on alettu kaivata automaattisia apuneuvoja luokitteluun. Perinteisen tekoälytutkimuksen tarjoama ratkaisu on **asiantuntija-järjestelmä** (*expert system*) eli asiantuntijoiden tietämyksen mallintaminen käsin tehdyksi sääntökokoelmaksi. Tunnetuin dokumentteja luokitteleva asiantuntijajärjestelmä lienee uutistoimisto Reutersissa käytetty CONSTRUE [HAN90]. Hyvin toimivien sääntöjen löytäminen voi kuitenkin osoittautua työlääksi, ja säännöstöillä on tapana paisua laajoiksi ja vaikeasti ylläpidettäviksi. Asiantuntijajärjestelmät ovatkin tyypillisesti vain yhden asiakkaan tarpeisiin räätälöityjä ohjelmistoja, joita on vaikea sovittaa uusiin ongelmiin.

Nopeimman ja joustavimman vaihtoehdon tarjoaa **koneoppiminen** (*machine learning*) eli luokittelusäännöstön tai -mallin automaattinen muodostaminen opetusaineiston avulla. Jos käytössä ei ole valmiiksi luokiteltua aineistoa tai jos dokumenttien väliset yhteydet halutaan selvittää ilman etukäteen määrättyjä luokkia, luokitteluohjelma voi vain pyrkiä liittämään yhteen toisiaan muistuttavat dokumentit. Tällöin puhutaan dokumenttien **ryvästämisestä** eli **klusteroinnista**. Esimerkki tällaisesta **ohjaamattomaan oppimiseen** (*unsupervised learning*) perustuvasta järjestelmästä on Kohosen itseorganisoiva kartta (*self-organizing map*) [Koh95, Hon97].

Ohjatussa oppimisessa (*supervised learning*) ohjelman käytössä on etukäteen luokiteltu opetusaineisto. Tarkoituksena on pyrkiä luomaan aineiston pohjalta malli tai säännöstö, jonka avulla uusia dokumentteja voidaan luokitella. Oma ratkaisuni perustuu ohjattuun oppimiseen.

1.3 Tutkielman tavoite

Tutkielmani tavoitteena oli vertailla dokumenttien luokittelun menetelmiä ja mahdollistaa suomenkielisen luokitteluohjelman prototyypin kehittäminen Sanoma Oy:n lehtiarkiston ja Lingsoft Oy:n käyttöön. Ratkaisun tuli olla mahdollisimman yleispätevä ja joustava, jotta sitä voitaisiin jatkossa soveltaa myös muihin luokitteluongelmiin ja muunkielisiin dokumentteihin. *Helsingin Sanomien* käyttämä luokitus osoittautui poikkeuksellisen laajaksi, joten kirjallisuudesta löytyneet luokittelualgoritmit eivät soveltuneet ongelman ratkaisuun sellaisenaan.

Testasin myös Fred Karlssonin kehittämän FINCG-ohjelman eli suomen kielen rajoitekielioppijäsentimen [Kar98] soveltuvuutta luokiteltavien dokumenttien esikäsittelyyn.

2 Luokitteluongelma

2.1 Helsingin Sanomien arkisto

Jokaisen *Helsingin Sanomien* numeron toimituksellinen aineisto talletetaan tekstitietokantaan, josta toimittajien on helppo tarvittaessa etsiä vanhoja artikkeleita. Tietokantaa voi käyttää perinteisellä tekstihaulla etsimällä artikkeleita niissä esiintyvien sanojen perusteella, mutta järjestelmä sisältää myös vaativammille käyttäjille tarkoitetun **asiasanahaun**, joka perustuu artikkelien luokitteluun.

Asiasanat liitetään artikkeleihin käsityönä. Luokittelijoilla on käytössään ohjelma, jolla voi sekä selata artikkeleita että liittää niihin asiasanoja, mutta artikkelit luetaan yleensä päivän lehdestä eikä tietokoneen näytöltä. Kaikki asiasanat on listattu toisessa ohjelmassa, josta luokittelija voi etsiä oikean vaihtoehdon ja siirtää sen ensimmäisen ohjelman kautta tietokantaan näppäimen painalluksella. Sisällysluetteloita, televisio- ja radio-ohjelmalistoja, taulukoita ja muita lista- tai vakiomuotoisia juttuja ei luokitella [Hjo97].

Koska artikkelien lukeminen vie aikaa ja asiasanoja on paljon, luokittelijoiden työtä helpottaisi ohjelmisto, jonka avulla oikeat asiasanat löytyisivät nopeammin.

2.2 Asiasanasto

Helsingin Sanomien käyttämä luokitus eli **asiasanasto** on hierarkkinen. Asiasanasto on jaettu 17 pääluokkaan, jotka jakautuvat edelleen aliluokkiin. Luokitustasoja on enimmillään kuusi ja luokkia eli **asiasanoja** kaikkiaan yli 11 000. Luokituksessa voidaan käyttää kaikkia tasoja. Tiedonvälitystä käsittelevä yleisluonteinen artikkeli voi siis kuulua luokkaan `tied` eli *tiedonvälitys ja viestintä* ja Pikku Kakkosesta kertova juttu luokkaan `tied-ra-lä-oh-la` eli *lastenohjelmat* (kuva 2). Yhteen artikkeliin voidaan liittää useita asiasanoja.

Kuva 2. Ote HS:n asiasanastosta

Luokka	Selite
tied	tiedonvälitys ja viestintä
...	…
tied-pu	puhelin
tied-ra	radio ja televisio
...	…
tied-ra-la	laitteet ja tekniikka
tied-ra-lä	lähetystoiminta, ohjelmat ja henkilöstö
tied-ra-lä-oh	ohjelmat
...	…
tied-ra-lä-oh-la	lastenohjelmat

2.3 Aineisto

Sanomalehtiartikkeleita luokiteltaessa aineistoa on hyvä olla käytössä vähintään vuoden ajalta, jotta luokitteluohjelma oppisi kaikkiin vuodenaikoihin liittyvät artikkelit, esim. joulu- tai kesälomajutut.

Sain luokitteluohjelman kehittämistä ja testausta varten käyttööni *Helsingin Sanomien* vuoden 1996 toimituksellisen aineiston (jatkossa HS96). Artikkeleita oli 103 577 ja niistä luokiteltuja 52 651 eli noin 51 %. Aineisto sisälsi noin 250 megatavua tekstiä (kuva 3).

Kuva 3. Ote HS96-aineistosta

Asiasanat: täydennyspoliisi, poliisin reservit
Koodit: oike-po-kou, oike-po-vah
Otsikko: Poliisin reservit
Teksti:
POLIISI varautuu poikkeusoloihin siinä missä puolustusvoimatkin. Uuden poliisilain velvoitteiden mukaisesti sisäministeriön poliisiosasto on laatinut ehdotuksen asetuksista ja hallinnollisista määräyksistä, jotka säätelevät täydennyspoliisin toimintaa.

Täydennyspoliisi ei ole poliisin erikoisryhmä, vaan puolustusvoimien reserviläisiin rinnastettava joukko, joka kootaan poikkeusolojen vuoksi poliisin avuksi. Poikkeusoloilla tarkoitetaan valtakunnan turvallisuuden kannalta todella kriittisiä tilanteita. Silloin puhutaan suoranaisesta sodanuhasta. Täydennyspoliisit voidaan kutsua töihin myös vakavan yleistä vaaraa aiheuttavan onnettomuuden tai rajoja koettelevan pakolaistulvan vuoksi. Silloinkin joukon kokoamiseen tarvitaan valtioneuvoston päätös.

...

Luokiteltuihin artikkeleihin oli liitetty keskimäärin 1,8 asiasanaa. Suurin osa asiasanastosta ei ollut aktiivisessa käytössä vuonna 1996: vähintään 5 esiinty-

mää oli 3 959 asiasanalla ja vähintään 10 esiintymää 2 354 asiasanalla. Artikkelit ja asiasanat jakautuivat varsin epätasaisesti eri pääluokkiin (kuva 4).

Kuva 4. HS96-aineiston jakautuminen pääluokkiin

Luokka	Selite	Artikkeleita	Vähintään viidesti esiintyviä asiasanoja
talo	talous	14 958	645
ulko	ulkomaat	13 430	294
harr	harrastukset, vapaa-aika ja urheilu	13 289	380
kult	kulttuuri ja taide	8 418	422
poli	politiikka, valtionhallinto ja -talous	7 595	265
yhte	yhteiskunta, väestö, ihmiset ja terveys	6 766	398
kunn	kunnat, paikkakunnat ja maakunnat	5 244	122
liik	liikenne ja matkailu	4 389	267
koul	koulutus, tiede ja historia	4 023	264
tied	tiedonvälitys ja viestintä	4 014	148
oike	oikeus, poliisi, rikollisuus ja tulli	3 806	197
ympä	ympäristö ja luonto	3 299	189
asun	asunnot ja rakentaminen	1 855	106
onne	onnettomuudet ja pelastus	1 754	79
aatt	aatteet, uskonnot ja kirkot	1 126	68
puol	puolustus ja sodat	1 001	65
aika	aika, sää ja vuodenkulku	713	48

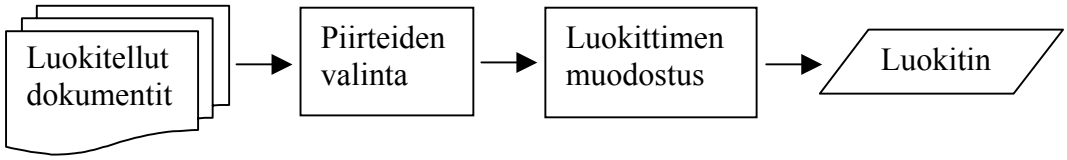
3 Luokittelujärjestelmä

3.1 Yleiskuvaus

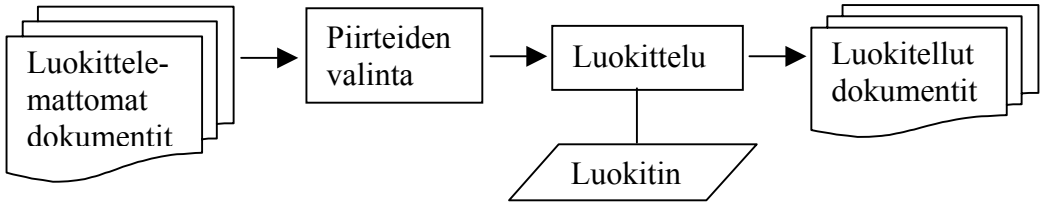
Ihanteellinen luokittelujärjestelmä ymmärtäisi tekstissä esiintyvät käsitteet sekä niiden suhteet toisiinsa ja ympäröivään maailmaan. Ymmärtämisestä ollaan kuitenkin vielä kaukana, joten tekstistä on poimittava mitattavissa olevia suureita ja pyrittävä muodostamaan luokitus niiden pohjalta.

Dokumenttien luokittelu voidaan kuvata ohjattuna koneoppimisongelmana, jossa ohjelma oppii valmiiksi luokiteltujen artikkelien avulla (kuva 5) luokittelemaan itse uutta aineistoa (kuva 6).

Kuva 5. Luokittelun oppiminen



Kuva 6. Luokittelu



Pyrin toteuttamaan kaikki suomen kielen käsittelyyn liittyvät asiat piirteiden valinnassa. Luokittimen muodostus on periaatteessa täysin kielestä riippumaton, mutta sen sijaan se saattaa olla hyvinkin riippuvainen luokitteluongelmasta. Yhdellä luokituksella ja aineistolla parhaiten toimiva algoritmi ja parhaat parametrit eivät välttämättä tuota parasta tulosta toisella ongelmalla.

3.2 Piirteiden valinta

Kieliaineiston monimuotoisuuden vuoksi ensimmäinen vaihe on luokittelussa käytettävien **piirteiden** (*features*) valinta. Piirteiksi voidaan valita esim. kaikki dokumentin sanat tai kaikki *n*:n peräkkäisen sanan yhdistelmät. Luokituksen kannalta vähäarvoiset yleiset sanat voidaan karsia. Jos käytettävissä on lingvististä tietoa dokumentista, piirteitä voivat olla myös esim. perusmuotoon palautetut sanat (*Lipposellakaan* → *Lipponen*), substantiivilausekkeet (*pääministeri Paavo Lipponen*) tai muut syntaktiset rakenteet.

Aktiivisiksi piirteiksi kutsutaan käsiteltävänä olevassa syötedokumentissa esiintyneitä piirteitä.

Kaikki aktiiviset piirteet eivät ole yhtä relevantteja dokumentin luokituksen kannalta. Piirteeseen *f* voidaan liittää sen **voimakkuutta** (*strength*) dokumentissa *d* kuvaava tunnusluku $s(f,d)$. Yksinkertaisin mahdollisuus on valita voimakkuudeksi 0 tai 1 (piirre ei esiinny / esiintyy dokumentissa). Voimakkuus voi olla myös piirteen esiintymiskertojen lukumäärä $n(f,d)$ tai jokin lukumäärän funktio. Myös lingvististä tietoa voidaan käyttää hyväksi esim. antamalla eri lauseenjäsenille (subjekti, objekti jne.) eri voimakkuusluvut. Rakenteisessa (esim. SGML-muotoisessa) dokumentissa voidaan suosia otsikoissa ja muissa tärkeissä osissa esiintyviä piirteitä.

Piirteinä voidaan käyttää myös koko dokumenttia kuvaavia tunnuslukuja, esimerkiksi sanojen määrää, virkkeiden pituutta tai virkerakenteiden monimutkaisuutta. Tällaisista piirteistä voi olla hyötyä varsinkin silloin, kun on selvitettävä myös dokumentin tyylilaji tai luotettavuus eikä pelkkää aihetta [JKa97]. Näitä mahdollisuuksia en kuitenkaan selvittänyt tutkimuksessani.

3.3 Luokittimen muodostaminen

Dokumentin **luokka** (*class, topic*) *c* on sen sisältöä kuvaava koodi tai avainsana. Jokainen dokumentti voi kuulua yhteen tai useampaan luokkaan. Myös luokan puuttuminen on mahdollista. Jos luokat halutaan lajitella paremmuusjärjestykseen, luokituksen osuvuutta voidaan kuvata **pistemäärällä** $S_c(d)$.

Luokitin (*classifier*) saa syötteekseen dokumentin *d* piirrejoukon

$$\{(f, s(f,d)) \mid s(f,d) > 0\},$$

josta se laskee pistemäärän $S_c(d)$ kaikille luokille *c*.

Oppimisohjelma saa syötteekseen joukon valmiiksi luokiteltuja dokumentteja, joista saatavan tiedon perusteella se muodostaa luokittimen.

3.4 Luokittimen päivittäminen

Lehtiartikkeleita luokitteleva ohjelma ei voi käyttää ikuisesti samaa luokitinta: uusia sanoja ja ilmiöitä syntyy jatkuvasti ja entiset poistuvat käytöstä. Siksi luokitinta onkin päivitettävä joko jatkuvasti uusien artikkelien saapuessa tai sopivin väliajoin koottavalla uudella opetusaineistolla.

3.5 Mallin puutteet

Onko piirteiden esiintymiskertojen laskenta riittävän hyvä perusta luokittelulle? Soergel [Soe74] esittää esimerkin mallin heikkoudesta: Heinäkuussa 1967 *Pravda* julkaisi varoituksen kommunistisissa maissa toimiville kansallismielisille liikkeille. Juttu oli suunnattu Romaniaa vastaan, mutta sanaa *Romania* ei esiintynyt siinä lainkaan. Ihminen olisi valinnut luokituksen *Neuvostoliiton ja Romanian suhteet*, mutta piirteiden laskemiseen perustuvalle tietokoneohjelmalle tehtävä olisi ollut mahdoton.

Pravdan artikkeli on tietysti kärjistetty esimerkki, mutta läheskään sataprosenttiseen tulokseen ei nykyisillä menetelmillä päästä. Tulokset olisikin aina annettava ihmisen tarkastettaviksi, jollei likimääräinen luokittelu riitä.

4 Aikaisempi tutkimus

4.1 Yleistä

Internetin ja yritysverkkojen dokumenttikokoelmien nopean kasvun vuoksi dokumenttien luokittelu on kasvattanut suosiotaan tutkimusaiheena, ja uusia tutkimustuloksia julkaistaan jatkuvasti. Valmiita luokitteluohjelmia on kuitenkin toistaiseksi vain vähän ja nekin harvat yleensä yhdelle asiakkaalle räätälöityjä. Kaupallisten luokitteluohjelmien tekijät haluavat usein salata tuotteidensa toimintaperiaatteet.

Ryhmittelin löytämäni algoritmit lähteen [Mit97] mukaan.

4.2 Luokittelualgoritmeja

4.2.1 Bayesiläiset menetelmät

Bayesiläisten menetelmien perusoletuksena on, että opetusaineistosta laskettavien todennäköisyyksien perusteella on mahdollista ennustaa eri luokkien todennäköisyys uusissa dokumenteissa. Oppikirjoissakin esitetty dokumenttien luokittelun perusalgoritmi on **naiivi Bayesin luokitin** [Mit97], joka toimii yleensä melko hyvin ja on helppo toteuttaa.

Lähteessä [Mit97] bayesiläisten menetelmien joukossa esiteltyä **EM-algoritmia** on kokeiltu myös dokumenttien luokittelussa. En kuitenkaan löytänyt vakuuttavaa esimerkkiä sen soveltamisesta todellisen kokoisiin ongelmiin. Lin ja Yamanishin sovelluksessa [LiY97] luokkia oli vain yhdeksän.

4.2.2 Neuraalilaskenta

Neuraalilaskenta sopii hyvin ongelmiin, joissa syötteet ovat monimutkaisia ja sisältävät paljon kohinaa. Neuraalilaskentaan perustuvalla luokitteluohjelmalla on heti alussa käytössään karkea luokittelumalli (verkko tai painovektori). Ohjelma päivittää malliaan jokaisen oikein tai väärin luokittelemansa opetusdokumentin jälkeen, kunnes tulokset ovat riittävän hyviä.

Dokumenttien luokittelussa on kokeiltu lähinnä yksinkertaisia painovektorimenetelmiä. Dagan, Karov ja Roth esittelivät kesäkuussa 1997 Littlestonen **Winnow-algoritmin** muunnelman [Lit88, DKR97], joka soveltui alustavien tutkimustulosten mukaan erinomaisesti dokumenttien luokitteluun.

Monikerroksisten neuraaliverkkojen käytön esteenä ovat käytännön ongelmat: kieliaineistossa on kymmeniä tai satoja tuhansia erilaisia sanoja, joten verkon solmujen määrä paisuisi epäkäytännöllisen suureksi [MLW92]. Pelkkien painovektorimenetelmienkin toteutukset kuluttavat useita megatavuja muistia.

4.2.3 Sääntöjoukkojen oppiminen

Edellä esitellyt menetelmät ovat ”mustia laatikoita”, joiden hyvyyttä voi arvostella vain luokittelutulosten mukaan. Sääntöpohjaisten ratkaisujen hyvä puoli on luettavuus: säännöstö muistuttaa asiantuntijan omaa päättelyä ja voi parhaimmillaan tuottaa myös hänelle hyödyllistä lisätietoa. Sääntöjä voi myös tarvittaessa muokata.

Yksinkertainen sääntöpohjainen algoritmi voi perustua esim. **assosiaatiosääntöihin** [AMS96]. Assosiaatiosääntöjä on kuitenkin käytetty aikaisemmin lähinnä inhimillisen päätöksenteon apuna eikä automaattisesti toimivissa ohjelmissa. Monimutkaisemmasta ratkaisusta sopii esimerkiksi piirteiden yhdistelmiä käyttävä Aptén ja Dameraun **Swap-algoritmi** [ApD94].

4.2.4 Päätöspuut

Päätöspuihin (*decision trees*) perustuvissa algoritmeissa opetusaineistosta kerätty tietämys on talletettu puumuotoon esimerkiksi **if-then**-sääntöinä. Luokittelualgoritmi etenee puussa valitsemalla luokiteltavan dokumentin ominaisuuksia vastaavan haaran. Luokat on talletettu puun lehtiin.

Lewisin ja Ringuetten päätöspuualgoritmi [LeR94] ylsi vain keskinkertaisiin tuloksiin Reutersin aineistolla [DKR97].

4.2.5 Ilmentymäpohjainen oppiminen

Ilmentymäpohjaisessa oppimisessa[1] (*instance-based learning*) opetusdokumentit talletetaan sellaisenaan tai vain hieman esikäsiteltyinä. Kun uusi dokumentti tulee luokiteltavaksi, luokitteluohjelma vertaa sitä talletettuihin dokumentteihin ja valitsee ehdotettavat luokat niistä dokumenteista, jotka muistuttavat eniten uutta dokumenttia. Periaatetta kutsutaan joskus myös **laiskaksi oppimiseksi** (*lazy learning*), koska oppiminen tapahtuu siinä mahdollisimman myöhään.

Masand, Linoff ja Waltz luokittelivat Dow Jonesin uutisia SEEKER-nimisen hakujärjestelmän palauttamien relevanssitietojen avulla ja saavuttivat varsin hyviä tuloksia [MLW92]. Tutkimusryhmän käytössä oli kuitenkin rinnakkaislaskentaa käyttävä supertietokone, ja siinäkin yhden artikkelin käsittely kesti 2 sekuntia.

4.2.6 Muuta kirjallisuutta

Semanttinen disambiguointi [PBW97, Yar95] eli oikean merkityksen valitseminen sanalle kontekstin avulla on sukua dokumenttien luokittelulle: merkityksen löytäminen vaatii yleensä tekstin aihepiirin tai tyylilajin tunnistamista. Aihetta on tutkittu innokkaasti viime aikoina, ja tutkimustuloksista saattaa olla tulevaisuudessa hyötyä myös dokumenttien luokittelussa.

4.3 Luokittelualgoritmien vertailu

Luokittelu ei ole luonteeltaan universaali ongelma, joten tuloksia vertailtaessa on aina kysyttävä, millaisella luokituksella ja aineistolla ne on saavutettu. Useimmat tutkijat ovat käyttäneet opetus- ja testiaineistona artikkelikokoelmaa nimeltä Reuters-21578[2]. Aineisto sisältää 21 578 luokiteltua SGML-muotoista englanninkielistä Reutersin uutista, jotka on jaoteltu 135 luokkaan (kuva 7).

Reutersin aineiston saavuttama asema ei välttämättä ole osoitus sen luokituksen mallikelpoisuudesta tai tyypillisyydestä. Koska luokittelu on tehty talousasioista kiinnostuneiden asiakkaiden näkökulmasta, moni artikkeli on luokiteltu sivu-

[1] Suomennos on omani.
[2] `http://www.research.att.com/~lewis/`

eikä pääaiheen mukaan. Reuters-21578 on kuitenkin paras ilmaiseksi saatavilla oleva luokiteltu tekstikokoelma.

Kuva 7. Esimerkki Reuters-aineistosta

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="5552" NEWID="9">
<DATE>26-FEB-1987 15:17:11.20</DATE>
<TOPICS><D>earn</D></TOPICS>
<TEXT>
<TITLE>CHAMPION PRODUCTS &lt;CH> APPROVES STOCK SPLIT</TITLE>
<DATELINE>ROCHESTER, N.Y., Feb 26 -</DATELINE>
<BODY>
Champion Products Inc said its board of directors approved a
two-for-one stock split of its common shares for shareholders of record
as of April 1, 1987.

The company also said its board voted to recommend to shareholders at
the annual meeting April 23 an increase in the authorized capital stock
from five mln to 25 mln shares.

Reuter
</BODY></TEXT>
</REUTERS>
```

Testiaineiston vähyys haittaa tutkimusta: kun algoritmeja hiotaan ja testataan vuosikausia samoilla artikkeleilla, tulokset eivät välttämättä enää kuvaa algoritmin yleistä hyvyyttä vaan pikemminkin sen toimintaa juuri Reutersin aineistolla.

Luokittelualgoritmeja voidaan verrata esimerkiksi niiden saannin ja tarkkuuden yhtäsuuruuspisteiden avulla (ks. luku 8.1). Kun eri algoritmien tulokset eroavat yleensä toisistaan vain muutamalla prosenttiyksiköllä, liiallisen yleistämisen vaara on suuri (kuva 8).

Kuva 8. Eräiden luokittelualgoritmien saannin ja tarkkuuden yhtäsuuruuspisteet (optimiarvo 100 %) Reutersin aineistolla lähteen [DKR97] mukaan.

Algoritmi	Aptén jako[1]	Lewisin jako
Tasapainotettu Winnow^{+} [Lit88, DKR97]	83,3 %	74,7 %
Ripper [CoS96]	79,6 %	71,9 %
Swap [ApD94]	78,9 %	–
Neuraaliverkko [WPW95]	77,5 %	–
Päätöspuut [LeR94]	–	67,0 %
Bayes [LeR94]	–	65,0 %

[1] Aptén jaossa kaikki opetusdokumentit ovat luokiteltuja, Lewisin jaossa myös luokittelemattomat dokumentit ovat mukana.

Tulosten vertailua vaikeuttaa myös se, että moni tutkija on testannut algoritmejaan vain pienellä osalla Reutersin 135 luokasta. Näiden algoritmien skaalautuvuuteen voi suhtautua kriittisesti. Onko luokkien määrän vähentämisen syynä tulosten kaunistaminen? Toimisivatko algoritmit kohtuullisessa ajassa ja muistitilassa, jos ongelma olisi todellisen kokoinen?

4.4 Luokittelu ja monikielisyys

Englannin kielen valta-asema on hankaloittanut monien tietokoneohjelmistojen sovittamista muille kielille. Tekstinkäsittelyohjelmien oikeinkirjoituksen tarkistus toteutettiin alun perin yksinkertaisilla sanalistoilla, jotka sopivat huonosti suomen kaltaisten taipuvien ja runsaasti yhdyssanoja sisältävien kielten käsittelyyn. Tekstitietokantojen indeksointi- ja hakuohjelmissa sanojen taipuminen on usein unohdettu kokonaan [AlH92, Jär95].

Myös suurin osa luokittelututkimuksesta on toistaiseksi keskittynyt käsittelemään englanninkielisiä dokumentteja. Koska englannin sananmuodostus on poikkeuksellisen yksinkertaista, dokumenttien lingvistinen esikäsittely on usein sivuutettu triviaalina asiana. Suomen kaltaisissa taipuvissa ja runsaasti yhdyssanoja sisältävissä kielissä esikäsittely vaikuttaa kuitenkin olennaisesti luokitteluohjelman toimintaan.

Oma tutkimukseni kuuluu ensimmäisiin yrityksiin soveltaa yleisesti tunnettuja luokittelualgoritmeja muihin kuin englanninkielisiin dokumentteihin.

5 Piirteiden valinta

5.1 Yleistä

Luokitteluohjelman käyttäjä on kiinnostunut **käsitteistä**, mutta ilman lingvististä esikäsittelyä dokumentit ovat vain jono **sanoja**. Luokitteluohjelman pitäisi periaatteessa pystyä poimimaan dokumentista käsitteitä vastaavat **termit**, jotka voivat koostua yhdestä tai useammasta sanasta [Sag90, ApD94].

Luokittelun kannalta relevanttien termien poimiminen on kuitenkin vaikeaa ja vaatii kieliaineiston harvuuden vuoksi suuria opetusaineistoja, jotta terminpoiminnasta olisi todellista hyötyä [Lew92]. Terminpoiminta (jota tarvitaan sekä oppimis- että luokitteluvaiheessa) myös hidastaa luokitteluohjelman toimintaa ja kasvattaa muistinkulutusta. Useimmat englanninkielisiä dokumentteja käsittelevät luokitteluohjelmat tarkastelevatkin vain yksittäisiä sanoja.

Englannin kielelle suunnitellut ratkaisut eivät kuitenkaan aina sovi muille kielille, koska englannin sananmuodostus on poikkeuksellisen yksinkertaista. Suomen kielessä yhdellä sanalla voi olla satoja tai jopa tuhansia muotoja, joten vähintään sananmuotojen normalisointi (esim. perusmuotoon palauttamalla) on tarpeen piirteiden määrän vähentämiseksi.

5.2 Kielenkäsittelymenetelmiä

5.2.1 Kaksitasomalli (TWOL)

Helsingin yliopiston yleisen kielitieteen laitoksessa on kehitetty menetelmiä eri kielten automaattiseen analyysiin. Kimmo Koskenniemen kehittämän **kaksitasomallin** *(two-level model*, TWOL) avulla voidaan kuvata periaatteessa minkä tahansa kielen morfologia eli sananmuodostus [Kos83]. Mallia on sovellettu jo noin 30 kielen kuvaukseen, ja siihen perustuvia sanastoja on käytössä useissa tiedonhaku- ja tekstinkäsittelyohjelmissa.

Englannin kielen sanan *bats* (’lepakot’, ’mailat’, ’lyö’ ym.) morfologinen analyysi:

```
"<bats>"
     "bat" N NOM PL
     "bat" <SVO> <SV>  V PRES SG3 VFIN @+FMAINV
```

(`N` = substantiivi, `V` = verbi, `NOM` = nominatiivi jne. Merkinnät on selitetty lähteessä [Vou96].)

5.2.2 Rajoitekielioppi (CG)

TWOL-malli sopii yksittäisten sanojen käsittelyyn. Kun halutaan ottaa huomioon myös lauseyhteys, TWOL-sanasto tarvitsee tuekseen muita menetelmiä. Fred Karlssonin kehittämä rajoitekielioppi (*Constraint Grammar*, CG) mahdollistaa mm. **morfologisen disambiguoinnin** ja **lauseenjäsenten merkitsemisen** [KVH95, Vou96].

Morfologinen disambiguointi tarkoittaa lauseyhteyteen sopivan morfologisen analyysin valintaa. Esim. englanninkielisessä lauseessa *bats hang from the trees* (’lepakot riippuvat puista’) sanan *bats* substantiivitulkinta on oikea.

Lauseenjäsenet voivat olla perinteisten kielioppikuvauksien mukaisia (subjekti, objekti jne.) tai kulloisenkin sovelluksen mukaan räätälöityjä.

Lauseen *Bats hang from the trees* CG-jäsennys:

```
"<*bats>"
     "bat" <*> N NOM PL @SUBJ
"<hang>"
     "hang" <SVO> <SV> V PRES -SG3 VFIN @+FMAINV
"<from>"
     "from" PREP @ADVL
"<the>"
     "the" <Def> DET CENTRAL ART SG/PL @DN>
"<trees>"
     "tree" N NOM PL @<P
"<$.>"
```

(`@SUBJ` = subjekti, `@+FMAINV` = pääverbi jne. Merkinnät on selitetty lähteessä [Vou96].)

CG-mallia on sovellettu englannin, ruotsin, saksan, suomen, tanskan, norjan, espanjan, suahilin ja baskin kielten kuvauksiin. Englannin kuvaus on toistaiseksi täydellisin.

5.3 FINTWOL ja FINCG

Piirteiden valintaa varten sain käyttööni Fred Karlssonin kehittämän suomen kielen rajoitekielioppijäsentimen eli FINCG:n esiversion [Kar98, Kar85, Kar86]. FINCG sisältää toistaiseksi perusmuotoon palautuksen (FINTWOL, [Kos83, FIN98]) ja morfologisen disambiguoinnin, muttei lauseenjäsenten merkitsemistä.

Lauseen *Poikkeusoloilla tarkoitetaan valtakunnan turvallisuuden kannalta todella kriittisiä tilanteita* FINCG-jäsennys:

```
"<poikkeusoloilla>"
        "poikkeus_olo" N ADE PL up
"<tarkoitetaan>"
        "tarkoittaa" <Ptv-Obj> V PRES PSS PE4
"<valtakunnan>"
        "valta_kunta" N GEN SG
"<turvallisuuden>"
        "turvallisuus" DN-LLINEN DA-UUS N GEN SG
"<kannalta>"
        "kanta" N ABL SG
"<todella>"
        "todella" ADV
"<kriittisiä>"
        "kriittinen" A POS PTV PL
"<tilanteita>"
        "tilanne" N PTV PL
"<$.>"
"<$.>" PUNCT
```

(`N` = substantiivi, `A` = adjektiivi, `V` = verbi jne. Merkinnät on selitetty lähteessä [FIN98].)

5.4 Terminpoiminta

5.4.1 Yleistä

FINCG:n tuottamista analyyseista on mahdollista poimia esim.

- perusmuotoon palautettuja sanoja
  - substantiiveja (*poliisi, Lipponen*)
  - verbejä (*puolustaa*)
  - adjektiiveja (*kriittinen*)
- sanojen yhdysosia (*täydennyspoliisi* → *täydennys* + *poliisi*)

sekä lisäohjelmoinnin jälkeen myös

- substantiivilausekkeita tai muita syntaktisia rakenteita.

5.4.2 Substantiivilausekkeet

Jos FINCG:hen lisättäisiin lauseenjäsenten merkitseminen, sillä voitaisiin poimia tekstistä substantiivilausekkeita eli NP:itä [HaK79, Vou93]. Substantiivilausekkeita on tiedonhakukirjallisuudessa pidetty yleensä hyvinä termiehdokkaina, mutta dokumenttien luokittelussa niihin liittyy useita ongelmia:

- Lähteen [Lew92] mukaan NP-poimintaan perustuvat luokitteluohjelmat eivät ole toimineet englannin kielellä yhtä hyvin kuin pelkkiä yksittäisiä sanoja käyttävät. Syynä on se, että erilaisia NP:itä on valtavasti ja vain pieni osa niistä todellisia termejä. Sama koskee myös suomen kieltä, kuten esimerkistä näkyy (kuva 9). Rakenteeltaan monimutkaisimmat NP:t (esim. *ehdotus asetuksista ja hallinnollisista määräyksistä*) ovat selvästi liian harvinaisia kelvatakseen luokittelun perustaksi, mutta myös yksinkertaisemmista vaihtoehdoista (adjektiiviattribuutti + substantiivi tai genetiiviattribuutti + substantiivi) vain harva toisi todellista lisäarvoa luokitteluun. *Sisäministeriön poliisiosasto* tuntuu termiltä, mutta eroaako sen esiintymistodennäköisyys tiettyyn luokkaan kuuluvissa dokumenteissa ratkaisevasti erillisten sanojen *sisäministeriö* ja *poliisiosasto* todennäköisyyksistä?

- Englannin kielessä on paljon moniosaisia termejä (esim. *noun phrase*, *information retrieval literature*, *term candidate*), mutta suomessa vastaavat termit ovat tyypillisesti yhdyssanoja (*substantiivilauseke*, *tiedonhakukirjallisuus*, *termiehdokas*).

- NP:iden tunnistamista vaikeuttaa suomessa genetiiviattribuutin ja akkusatiiviobjektin samannäköisyys. Esim. lauseessa *Tapasin Lyylin serkun kutsuilla* NP:itä voivat olla joko *Lyylin serkku* + *kutsut* tai *Lyyli* + *serkun kutsut*.

- NP-poiminta kasvattaisi opetusohjelmalle tulevien piirteiden määrän epäkäytännöllisen suureksi. Yksittäistenkin sanojen käsittely vei käytössäni olleen laitteiston suorituskykynsä äärirajoille.

Kuva 9. Luvun 2.3 esimerkkitekstin substantiivilausekkeet ja niiden esiintymiskerrat käsin poimittuina

```
4 poliisi
3 täydennyspoliisi
2 joukko
2 poikkeusolo
2 puolustusvoimat
1 asetus
1 ehdotus asetuksista ja hallinnollisista määräyksistä
1 erikoisryhmä
1 hallinnollinen määräys
1 joukon kokoaminen
1 kriittinen tilanne
1 määräys
1 onnettomuus
1 pakolaistulva
1 poliisilain velvoite
1 poliisilaki
1 poliisin apu
1 poliisin erikoisryhmä
1 poliisin reservi
1 poliisiosasto
1 puolustusvoimien reserviläinen
1 puolustusvoimien reserviläisiin rinnastettava joukko
1 päätös
1 raja
1 rajoja koetteleva pakolaistulva
1 reservi
1 reserviläinen
1 sisäministeriö
1 sisäministeriön poliisiosasto
1 sodanuhka
1 suoranainen sodanuhka
1 tilanne
1 toiminta
1 turvallisuus
1 täydennyspoliisin toiminta
1 uuden poliisilain velvoite
1 uusi poliisilaki
1 vakava yleistä vaaraa aiheuttava onnettomuus
1 valtakunnan turvallisuus
1 valtakunta
1 valtioneuvosto
1 valtioneuvoston päätös
1 velvoite
1 yleinen vaara
1 yleistä vaaraa aiheuttava onnettomuus
```

Edellä esitettyjen ongelmien vuoksi päätin olla toteuttamatta yleistä substantiivilausekkeiden poimintaa luokitteluohjelmassani. Rajoitettu NP-poiminta on kuitenkin jatkotutkimuksen arvoinen aihe: ainakin moniosaisten erisnimien pitäminen yhtenäisinä (esim. *Paavo Lipponen* vs. *Pekka Lipponen*) voisi parantaa hieman tuloksia. Lahtinen [Lah97a, Lah97b] on tutkinut tarkemmin suomen kielen termien rakennetta.

5.4.3 Piirteiden yhdistelmät

Luokitteluohjelma voisi käyttää piirteinä myös useamman kuin yhden piirteen yhdistelmiä. Tätä mahdollisuutta en kuitenkaan tutkinut, sillä piirreyhdistelmien käsittelyä vaivaisivat samat ongelmat kuin substantiivilausekkeiden poimintaa: piirteiden määrä kasvaisi turhan suureksi, ja vain pieni osa yhdistelmistä olisi relevantteja. Monitulkintaisten termien (esim. *hiiri*, 'jyrsijä' tai 'tietokoneen ohjauslaite') käsittelyssä piirteiden yhdistämisestä voisi kuitenkin olla hyötyä (esim. *hiiri* **ja** *juusto*, *hiiri* **ja** *näppäimistö*).

Piirteiden yhdistelmiä ovat tutkineet mm. Apté ja Damerau [ApD94].

5.5 Piirteiden karsinta

Opetusaineistossa liian harvoin esiintyvät piirteet eivät ole hyödyksi luokittelussa: niiden esiintymät saattavat olla ainutkertaisia, eivätkä piirteet välttämättä edes liity dokumentin pääaiheeseen. Kovin yleiset piirteet ovat myös useimmiten hyödyttömiä, koska niitä saattaa esiintyä liian monessa luokassa. Nyrkkisäännöksi annetaankin usein se, että parhaita piirteitä ovat frekvenssiltään keskitasoa olevat.

Kaikkein harvinaisimmat tai yleisimmät piirteet voidaan karsia mekaanisesti frekvenssin perusteella jo esikäsittelyvaiheessa, mutta esiintymiskertojen määrää olennaisempi asia on piirteiden **erottelukyky**. Huonosti erottelevat piirteet voidaan karsia itse opetusalgoritmissa, kunhan algoritmin käytössä on tarpeeksi tietoa piirteiden vaikutuksesta luokitteluun.

Myös lingvististä tietoa voidaan käyttää hyväksi karsimalla ne sanat, joiden merkitys tai tehtävä on liian yleinen, esim. apuverbit tai viikonpäivien ja kuukausien nimet.

Hyvin toteutettu piirteiden karsinta sekä pienentää luokittimen muistintarvetta että parantaa tuloksia vähentämällä aineiston kohinaa.

6 Oppimis- ja luokittelualgoritmit

6.1 Yleistä

Valitsin lähdekirjallisuudesta vertailualgoritmeiksi naiivin Bayesin luokittimen ja kesäkuussa 1997 Reutersin aineistolla parhaiten toimineen tasapainotetun Winnow$^+$:n. Lisäksi laadin itse kolmannen, assosiaatiosääntöihin perustuvan luokittelualgoritmin.

6.2 Hierarkkinen luokittelu

Kaikissa löytämissäni dokumenttien luokittelusta tehdyissä tutkimuksissa luokkia oli korkeintaan toistasataa (usein vain muutama), eivätkä luokat muodostaneet minkäänlaista hierarkiaa. HS:n asiasanaston tuhansia luokkia oli käytännössä mahdotonta käsitellä yhdellä luokittimella, koska muistia olisi tarvittu opetusvaiheessa satoja megatavuja ja valmis luokitinkin olisi kasvanut ainakin kymmenien megatavujen kokoiseksi. Asiasanaston monitasoisen puurakenteen käsittely yhdellä asteikolla olevina rinnakkaisina luokkina ei tuntunut muutenkaan hyvältä ratkaisulta.

Ratkaisin ongelman **hierarkkisella luokittelulla**:

1. Luokitellaan dokumentti **pääluokittimella** yhteen tai useampaan 17 pääluokasta.
2. Valitaan lupaavimmat pääluokat ja luokitellaan dokumentti edelleen aliluokkiin pääluokkakohtaisilla **aliluokittimilla**.
3. Jatketaan samaan tapaan, kunnes haluttu luokittelusyvyys on saavutettu.

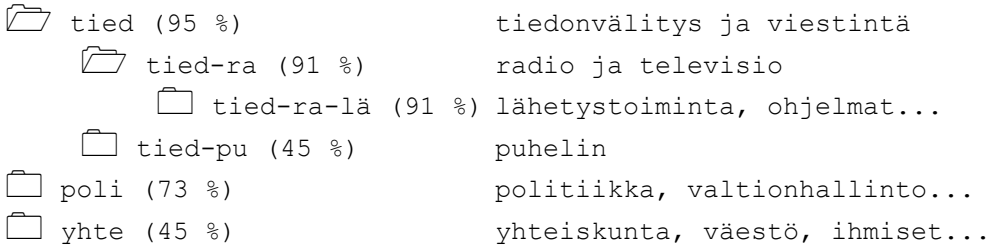
Pääluokittimen muodostamiseen käytetään koko opetusaineistoa. Kunkin aliluokittimen opetusaineistona käytetään vain vastaavaan luokkaan kuuluvia dokumentteja. Esim. aliluokitin `talo` (*talous*) muodostetaan pelkkien talousartikkelien avulla. Pää- ja aliluokittimet voivat käyttää samaa piirrejoukkoa.

Hierarkkinen luokitteluohjelma voi olla joko **automaattinen** tai **interaktiivinen**. Automaattisen ohjelman on löydettävä itse parhaimmat luokat ja oikea syvyys sopivan pisteitysmekanismin ja lopetusehdon avulla esim. lähteessä

[Rij79] kuvattua tapaa mukaillen. Koska prototyypistä on tarkoitus tulla interaktiivinen työväline, en tutkinut tarkemmin automaattista vaihtoehtoa.

Interaktiivisen ohjelman käyttöliittymä voi olla esimerkiksi Windows 95:n Resurssienhallinnan tapainen puunäkymä, jossa ohjelma järjestää luokat paremmuusjärjestykseen ja antaa käyttäjän valita haluamansa haaran ja luokittelusyvyyden (kuva). Lähteessä [Maa95] kuvatun Librarian's Assistant -nimisen ohjelman käyttöliittymä muistuttaa hahmotelmaani (kuva 10).

Kuva 10. Ehdotus luokitteluohjelman käyttöliittymäksi



Koska HS:n asiasanasto sisältää tuhansia luokkia, jokaista haaraa ja tasoa ei kannattane toteuttaa erillisenä luokittimena, vaan osa aliluokittimista voi sisältää kaksi tai mahdollisesti useampiakin tasoja. Esim. aliluokitin `tied` voi sisältää luokat `tied-ra`, `tied-ra-lä` jne. Näin säästetään levytilaa – tosin mahdollisesti luokittelutarkkuuden kustannuksella, kun hierarkkiset luokat joudutaan pakottamaan yhteen tasoon.

6.3 Assosiaatiosäännöt

6.3.1 Taustaa

Tietämyksen muodostaminen tietokannoista (*data mining, knowledge discovery in databases*) on uusi tutkimusalue, jonka peruskäsitteisiin kuuluvat **assosiaatiosäännöt** (*association rules*) [AMS96, Man97, AHK97]. Assosiaatiosääntöjä on tähän mennessä käytetty lähinnä inhimillisen päätöksenteon apuna eikä päättelyohjelmistojen osana, joten niiden soveltuvuus dokumenttien luokitteluun oli avoin ongelma.

Assosiaatiosäännöt ovat muotoa $X \Leftarrow Y$, missä X on seurausosa (tässä tapauksessa luokka) ja Y ehto-osa (eli piirrejoukko). Sääntö tarkoittaa, että jos doku-

mentti sisältää piirrejoukon *Y*, dokumentin luokka on *X*. Käytin säännöstöissäni vain yhden piirteen kokoisia joukkoja (ks. perustelut luvussa 5.4.3).

Jokaiseen sääntöön liittyy kaksi tunnuslukua. **Luottamus** (*confidence*) kertoo, kuinka suuressa osassa piirrejoukon *Y* sisältävistä dokumenteista sääntö $X \Leftarrow Y$ pätee. **Tuki** (*support*) kertoo, kuinka monessa dokumentissa sääntö pätee.

Esimerkiksi sääntö

```
kult <= "konsertti" (72 %, 424)
```

merkitsee, että sanan *konsertti* sisältäneistä artikkeleista 72 % käsitteli kulttuuria ja että sanan *konsertti* sisältäneitä kulttuuriartikkeleita oli 424 kappaletta.

6.3.2 Opetus

Oppimisalgoritmi käy läpi kaikki dokumentit ja laskee, kuinka monessa dokumentissa ja mihin luokkiin yhdistyneenä kukin piirre esiintyy (kuva 11). Algoritmi ei ota huomioon piirteiden voimakkuutta yksittäisen artikkelin sisällä, jottei pitkiä dokumentteja suosittaisi lyhyiden kustannuksella. Laskurimuuttujat *FeatCount* ja *PairCount* ovat merkkijonoilla indeksoitavia assosiatiivisia taulukoita. Määrittelemättömän alkion arvo on 0.

Kuva 11. Assosiaatiosäännöt: oppimisalgoritmi

Classes ← kaikkien luokkien joukko
AllDocs ← kaikkien opetusdokumenttien joukko
Feats ← kaikkien piirteiden joukko
foreach *c* **in** *Classes* **do**
    Docs ← luokkaan *c* kuuluvien dokumenttien joukko
    foreach *D* **in** *Docs* **do**
      F ← dokumentista *D* poimitut piirre-frekvenssiparit (*f*, *s*)
      foreach (*f*, *s*) **in** *F* **do**
        FeatCount[*f*] ← *FeatCount*[*f*] + 1
        PairCount[*c*, *f*] ← *PairCount*[*c*, *f*] + 1
      done
    done
done
foreach *c* **in** *Classes* **do**
    foreach *f* **in** *Feats* **do**
      if (**exists** *PairCount*[*c*, *f*]) **then**
        Rules[*c*, *f*].*conf* ← *PairCount*[*c*, *f*] / *FeatCount*[*f*]
        Rules[*c*, *f*].*supp* ← *PairCount*[*c*, *f*]
      endif
    done
done

6.3.3 Sääntöjen karsinta

Luokittimen pienentämiseksi ja kohinan vähentämiseksi voidaan karsia ne säännöt, joiden luottamus tai tuki ei saavuta tiettyä kynnysarvoa (esim. luottamus 25 % ja tuki 5 dokumenttia).

6.3.4 Luokittelu

Suunnittelin itse sääntöjä käyttävän *ad hoc* -luokittelualgoritmin (kuva 12). Jos dokumentti sisältää piirre-frekvenssiparit $\{(f_1, s_1), \mathrm{K}, (f_n, s_n)\}$ ja k_{cf} on säännön $c \Leftarrow f$ luottamus, algoritmi antaa luokalle *c* pistemäärän

$$S_c = \sum_{i=1}^{n} s_i k_{cf_i} \ .$$

Algoritmi siis painottaa usein esiintyviä piirteitä ja niitä sääntöjä, joilla on suurin luottamus. Piirteiden frekvenssit voidaan nyt ottaa huomioon, koska pisteitettävänä on vain yksi dokumentti kerrallaan eivätkä dokumenttien pituuserot siten haittaa. Eri luokkien esiintymistodennäköisyydet vaikuttavat epäsuorasti pisteitykseen, koska yleisimpiin luokkiin liittyy tavallisesti eniten sääntöjä. Algoritmi ei ota huomioon sääntöjen tukea.

Sääntöpohjaisille ratkaisuille tyypilliseen tapaan pisteitys perustuu pelkkään intuitioon. Algoritmi toimi kuitenkin yllättävän hyvin (ks. luku 8.3.1) ja siten ainakin osoitti, miten pieniä eri luokittelualgoritmien väliset tuloserot ovat.

Kuva 12. Assosiaatiosäännöt: luokittelualgoritmi

Classes ← kaikkien luokkien joukko
Rules ← kaikkien sääntöjen joukko
F ← dokumentista poimitut piirre-frekvenssiparit (*f*, *s*)
foreach *c* **in** *Classes* **do**
    pistemäärä *S*[*c*] ← 0
done
foreach (*f*, *s*) **in** *F* **do**
    foreach *c* **in** *Classes* **do**
        if (**exists** *Rules*[*c*, *f*]) **then**
            S[*c*] ← *S*[*c*] + *s* * *Rules*[*c*, *f*].*conf*
        endif
    done
done
tulosta ne luokat *c*, joilla on suurin pistemäärä *S*[*c*]

6.4 Naiivi Bayesin luokitin

6.4.1 Taustaa

Naiivi Bayesin luokitin (*naive Bayes classifier*) [Mit97] on saavuttanut dokumenttien luokittelussa perusalgoritmin aseman. Algoritmi on helppo toteuttaa, ja sillä on saavutettu varsin hyviä tuloksia dokumenttien luokittelussa.

Algoritmi perustuu todennäköisyyslaskennasta tuttuun **Bayesin kaavaan**. Jos aineistosta poimittu piirre-esiintymäjono on $(a_1,\mathrm{K}\,,a_m)$ ja kunkin luokan esiintymistodennäköisyys aineistossa tunnetaan, luokan c ehdollinen todennäköisyys on

$$P(c \mid a_1,\mathrm{K}\,,a_m) = \frac{P(a_1,\mathrm{K}\,,a_m \mid c)\,P(c)}{P(a_1,\mathrm{K}\,,a_m)}\,.$$

Luokat voidaan siis asettaa paremmuusjärjestykseen vertaamalla pistemääriä

$$S(c \mid a_1,\mathrm{K}\,,a_m) = P(a_1,\mathrm{K}\,,a_m \mid c)\,P(c)\,.$$

Kaikkien todennäköisyyksien $P(a_1,\mathrm{K}\,,a_m \mid c)$ arviointi on kuitenkin käytännössä mahdotonta ilman valtavaa opetusaineistoa. Ongelma ratkeaa kahdella yksinkertaistavalla oletuksella[1]:

1. Oletetaan piirre-esiintymien olevan toisistaan ehdollisesti riippumattomia:

   $$P(a_1,\mathrm{K}\,,a_m \mid c) = \prod_{j=1}^{m} P(a_j \mid c)\,.$$

   Todellisuudessa esiintymät eivät tietenkään ole riippumattomia toisistaan: esimerkiksi sanan *presidentti* esiintymistodennäköisyys luultavasti kasvaa, jos edellinen sana on *tasavallan*.

2. Oletetaan, että piirteen f_i (esim. sanan *presidentti*) esiintymistodennäköisyys on riippumaton sen esiintymispaikasta:

   $P(a_j = f_i \mid c) = P(a_k = f_i \mid c)$ kaikilla i, j, k.

[1] Sana *naiivi* algoritmin nimessä viittaa juuri näihin yksinkertaistuksiin.

Tämäkään oletus ei tarkkaan ottaen pidä paikkaansa: jos aineistona ovat esimerkiksi presidentin saamat kirjeet, sana *presidentti* esiintyy todennäköisesti useammin kirjeen puhutteluosassa kuin sen lopussa.

Näiden kahden välttämättömän yksinkertaistuksen jälkeen riittää, että tunnetaan kunkin luokan esiintymistodennäköisyys opetusaineistossa ja kunkin piirteen esiintymistodennäköisyys kussakin luokassa. Koska piirre-esiintymien järjestyksellä ei ole väliä, ne voidaan yhdistää piirre-frekvenssiparien joukoksi $\{(f_1, s_1), \mathrm{K}\ , (f_n, s_n)\}$. Luokan *c* pistemäärä voidaan laskea kaavalla

$$S(c \mid \{(f_1, s_1), \mathrm{K}\ , (f_n, s_n)\}) = P(c) \prod_{i=1}^{n} P(f_i \mid c)^{s_i} .$$

Jäljellä on vielä yksi ongelma: kaikki luokka-piirreparit eivät yleensä esiinny opetusaineistossa, mutta mikään termeistä $P(f_i \mid c)$ ei voi olla 0. Ratkaisuna on todennäköisyyden estimointi. Jos *m* on kaikkien piirre-esiintymien määrä opetusaineistossa $\{(f_1, s_1), \mathrm{K}\ , (f_n, s_n)\}$ ja $|F|$ kaikkien erilaisten piirteiden lukumäärä, ehdolliselle todennäköisyydelle saadaan arvio

$$P(f_i \mid c) \approx \frac{s_i + 1}{m + |F|} .$$

Aineistossa esiintymättömien luokka-piirreparien lukumäärä s_i on 0, joten ne saavat todennäköisyydekseen $1/(m+|F|)$.

6.4.2 Opetus

Oppimisalgoritmi (kuva 13) laskee luokkien ja piirteiden esiintymistodennäköisyydet luvun 6.4.1 mukaan. Lisäksi algoritmi tallettaa kunkin luokan oletustodennäköisyyden $P_{default}$. Luokitteluohjelma käyttää oletustodennäköisyyttä, jos $P[f \mid c]$ ei ole määritelty jollekin piirteelle f ja luokalle c.

Kuva 13. Naiivi Bayesin luokitin: oppimisalgoritmi

Classes ← kaikkien luokkien joukko
AllDocs ← kaikkien opetusdokumenttien joukko
Feats ← kaikkien piirteiden joukko
foreach *c* **in** *Classes* **do**
    Docs ← luokkaan *c* kuuluvien dokumenttien joukko
    P[*c*] ← |*Docs*| / |*AllDocs*|
    T ← joukkoon *Docs* kuuluvien dokumenttien konkatenaatio
    m ← kaikkien piirre-esiintymien määrä dokumentissa *T*
    foreach *f* **in** *Feats* **do**
        s[*f*] ← piirteen *f* esiintymien määrä dokumentissa *T*
        P[*f* | *c*] ← (*s*[*f*] + 1) / (*m* + |*Feats*|)
    done
    $P_{default}$[*c*] ← 1 / (*m* + |*Feats*|)
done

6.4.3 Piirteiden karsinta

Liian harvoin esiintyvät luokka-piirreparit voidaan karsia luokittimen pienentämiseksi ja kohinan vähentämiseksi.

6.4.4 Luokittelu

Luokittelualgoritmi (kuva 14) poimii dokumentista piirre-frekvenssiparit ja pisteittää kunkin luokan oppimisalgoritmin laskemien todennäköisyyksien avulla.

Kuva 14. Naiivi Bayesin luokitin: luokittelualgoritmi

Classes ← kaikkien luokkien joukko
F ← dokumentista poimitut piirre-frekvenssiparit (*f*, *s*)
foreach *c* **in** *Classes* **do**
    pistemäärä *S*[*c*] ← *P*[*c*]
    foreach (*f*, *s*) **in** *F* **do**
        if (exists *P*[*f* | *c*]) **then**
            S[*c*] ← *S*[*c*] * *P*[*f* | *c*] ^ *s*
        else
            S[*c*] ← *S*[*c*] * $P_{default}$[*c*] ^ *s*
        endif
    done
done
tulosta ne luokat *c*, joilla on suurin pistemäärä *S*[*c*]

6.5 Tasapainotettu Winnow+

6.5.1 Taustaa

Assosiaatiosääntöalgoritmi ja naiivi Bayesin luokitin käyvät aineiston kerran läpi ja muodostavat luokittimen suoraan piirteiden ja luokkien esiintymiskertoja laskemalla. **Tasapainotettu Winnow+** (*Balanced Winnow+*, jatkossa lyhyesti vain Winnow) [Lit88, DKR97] käy läpi aineistoa kierros kierrokselta, kunnes luokitteluvirheitä tapahtuu tarpeeksi vähän.

6.5.2 Opetus

Winnow (kuva 15) muistuttaa klassista painovektorialgoritmia, Rosenblattin perseptronia [Ros58]. Algoritmilla on viisi parametria: **vahvistus- ja heikennysparametrit** (*promotion/demotion parameters*) α ja β ($\alpha > 1,\ 0 < \beta < 1$) sekä kaksi **kynnysarvoa** (*threshold*) θ^+ ja θ^- (esim. $\theta^+ = 1{,}1$ ja $\theta^- = 0{,}9$).

Jokaiseen luokka-piirreparin liittyy kaksi painoa, w^+ ja w^-. Piirteen kokonaispaino on näiden kahden painon erotus. Jos dokumentin piirteiden voimakkuudet ovat (s_1, K, s_n), algoritmi laskee jokaiselle luokalle pistemäärän

$$S_c = \sum_{i=1}^{n} \left(w_{ci}^+ - w_{ci}^- \right) s_i .$$

Jos $S_c > \theta^+$, algoritmi ennustaa dokumentin kuuluvan luokkaan *c*. Jos $S_c < \theta^-$, algoritmi ennustaa, ettei dokumentti kuulu luokkaan *c*. Jos pistemäärä on välillä $[\theta^-, \theta^+]$, luokitus katsotaan aina virheelliseksi. Kahden eri kynnysarvon tarkoituksena on parantaa luokittimen erottelukykyä pakottamalla kaikki epävarmat ehdotukset jompaankumpaan suuntaan.

Painot alustetaan alkuarvoilla, jotka tuottavat lähellä kynnysarvoja olevia pistemääriä (kuva 15). Algoritmi päivittää painovektoria jokaisen virheen jälkeen. Jos algoritmi luokittelee väärin positiivisen esimerkin, kaikkien aktiivisten piirteiden painoja vahvistetaan: $w_i^+ \leftarrow \alpha \cdot w_i^+$ ja $w_i^- \leftarrow \beta \cdot w_i^-$. Jos algoritmi luokittelee väärin negatiivisen esimerkin, painoja heikennetään: $w_i^+ \leftarrow \beta \cdot w_i^+$ ja $w_i^- \leftarrow \alpha \cdot w_i^-$.

Kuva 15. Winnow: oppimisalgoritmi

```
Classes ← kaikkien luokkien joukko
AllDocs ← kaikkien opetusdokumenttien joukko
Feats ← kaikkien piirteiden joukko

/* Alusta painot alkuarvoilla, jotka tuottavat lähellä kynnysarvoja olevia
   pistemääriä */
foreach f in Feats do
        foreach c in Classes do
                θ ← (θ+ + θ−) / 2
                PosW[f, c] ← 2 * θ / (|Feats| / |AllDocs|)
                NegW[f, c] ← θ / (|Feats| / |AllDocs|)
        done
done

do
        foreach D in AllDocs do
                F ← dokumentista D poimitut piirre-voimakkuusparit (f, s)
                foreach c in Classes do
                        pistemäärä S[c] ← 0
                done

                /* Pisteitä kaikki luokat */
                foreach (f, s) in F do
                        foreach c in Classes do
                                S[c] ← S[c] + s * (PosW[f, c] – NegW[f, c])
                        done
                done

                /* Päivitä painot */
                foreach c in Classes do
                        if (c on oikea luokitus and S[c] ≤ θ+ ) then
                                /* Väärin luokiteltu positiivinen esimerkki */
                                foreach (f, s) in F do
                                        PosW[f, c] ← PosW[f, c] * α
                                        NegW[f, c] ← NegW[f, c] * β
                                done
                        else if (c on virheellinen luokitus and S[c] > θ− ) then
                                /* Väärin luokiteltu negatiivinen esimerkki */
                                foreach (f, s) in F do
                                        PosW[f, c] ← PosW[f, c] * β
                                        NegW[f, c] ← NegW[f, c] * α
                                done
                        endif
                done /* foreach c in Classes */
        done /* foreach D in AllDocs */
while (oikeiden luokitusten lkm / kaikkien luokitusten lkm < exit_threshold)
```

6.5.3 Piirteiden karsinta

Kun luokittelua on jatkettu riittävän pitkään, voidaan karsia ne piirteet, jotka vaikuttavat vähiten luokitteluun. Dagan, Karov ja Roth karsivat omassa algoritmissaan kaikki piirteet, joiden painoa oli vahvistettu tai heikennetty enintään yhden askelen verran.

6.5.4 Luokittelu

Luokittelualgoritmi (kuva 16) poimii dokumentista piirre-voimakkuusparit ja laskee kullekin luokalle pistemäärän samalla tavalla kuin oppimisalgoritmissa.

Kuva 16. Winnow: luokittelualgoritmi

Classes ← kaikkien luokkien joukko
F ← dokumentista poimitut piirre-voimakkuusparit (*f*, *s*)
foreach *c* **in** *Classes* **do**
    pistemäärä *S*[*c*] ← 0
    foreach (*f*, *s*) **in** *F* **do**
        S[*c*] ← *S*[*c*] + *s* * (*PosW*[*f*, *c*] – *NegW*[*f*, *c*])
    done
done
tulosta ne luokat *c*, joilla on suurin pistemäärä *S*[*c*]

7 Käytännön toteutus

7.1 Yleistä

Toteutin opetus- ja luokitteluohjelmien prototyypit Perl-ohjelmointikielellä ja muilla Unix-työkaluilla. Perlin hyviä puolia ovat monipuoliset tekstimuotoisen tiedon käsittelyrutiinit ja luokittimen tallettamiseen sopivat assosiatiiviset taulukot. Lopullinen ohjelmisto on kuitenkin tarkoitus toteuttaa tehokkuussyistä C- tai C++-kielellä.

Testauksen nopeuttamiseksi toteutin piirteiden valinnan kokonaan erillisenä oppimis- ja luokitteluohjelmista ja tallensin opetusaineistosta poimimani piirteet etukäteen levylle. Lopullinen luokitteluohjelma sisältää tietysti sekä piirteiden valinnan että luokittelualgoritmin.

Ohjelmien lähdekoodit ovat Lingsoft Oy:n omaisuutta, joten niitä ei ole liitetty tutkielmaan.

7.2 Piirteiden valinta

7.2.1 FINCG

Käytin FINCG:n esiversion (ks. luku 5.3) tuottamia analyyseja sellaisenaan piirteiden valinnan pohjana.

7.2.2 Analyysien esikäsittely

Vaikka en vielä sisällyttänyt substantiivilausekkeiden poimintaa ohjelmaani, muunsin CG-analyysit kuitenkin NP-poiminnalle sopivampaan muotoon, jossa jokainen virke on omalla rivillään ja perusmuotoon palautettuihin sanoihin on liitetty sanaluokkatieto (kuva 17). Tulkitsin partisiippimuodot (esim. *rinnastettava*) verbeiksi.

FINCG:n monitulkintaisiksi jättämille sanoille tulostin kaikki tulkinnat. Tämä ominaisuus vääristää tietysti hieman sanojen frekvenssejä monitulkintaisten sanojen hyväksi.

NP-poimintaa varten tarvittaisiin sanaluokan lisäksi myös tieto kunkin sanan tehtävästä (esim. pääsana, genetiiviattribuutti tai adjektiiviattribuutti), ja monitulkintaisten sanojen eri tulkinnat olisi merkittävä eri tavalla kuin erilliset sanat.

Kuva 17. Luvun 2.3 esimerkkiteksti virkkeisiin jaettuna (`@N` = *substantiivi,* `@A` = *adjektiivi,* `@V` = *verbi,* `@?` = *muu)*

```
poliisi/@N reservi/@N
poliisi/@N varautua/@V poikkeus_olo/@N se/@? mikä/@? puolustus_voima/@N
uusi/@A poliisi_laki/@N velvoite/@N mukainen/@A sisä_ministeriö/@N
   poliisi_osasto/@N olla/@V laatia/@A ehdotus/@N asetus/@N ja/@?
   hallinnollinen/@A määräys/@N joka/@? säädellä/@V säädellä/@A
   täydennys_poliisi/@N toiminta/@N
täydennys_poliisi/@N ei/@V olla/@V poliisi/@N erikois_ryhmä/@N vaan/@?
   puolustus_voima/@N reserviläinen/@N rinnastaa/@V joukko/@N joka/@?
   koota/@V poikkeus_olo/@N vuoksi/@? vuoksi/@N vuo/@N poliisi/@N apu/@N
poikkeus_olo/@N tarkoittaa/@V valta_kunta/@N turvallisuus/@N kanta/@N
   todella/@? kriittinen/@A tilanne/@N
silloin/@? puhua/@V suoranainen/@A sodan_uhka/@N
täydennys_poliisi/@N voida/@V kutsua/@V työ/@N myös/@? vakava/@A
   yleinen/@A vaara/@N aiheuttaa/@V onnettomuus/@N tai/@? raja/@N
   koetella/@A pakolais_tulva/@N vuoksi/@? vuoksi/@N vuo/@N
silloin/@? jouko/@N kokoaminen/@N tarvita/@V valtio_neuvosto/@N
   päätös/@N
```

7.2.3 Terminpoiminta

Esikäsitellystä analyysista voi Unix-työkaluilla helposti poimia esim. substantiivit ja niiden frekvenssit (kuva 18).

Kuva 18. Luvun 2.3 esimerkkitekstin substantiivit ja niiden frekvenssit

```
4    poliisi                    1    reservi
3    täydennys_poliisi          1    raja
3    poikkeus_olo               1    päätös
2    vuoksi                     1    poliisi_osasto
2    vuo                        1    poliisi_laki
2    puolustus_voima            1    pakolais_tulva
1    velvoite                   1    onnettomuus
1    valtio_neuvosto            1    määräys
1    valta_kunta                1    kokoaminen
1    vaara                      1    kanta
1    työ                        1    jouko
1    turvallisuus               1    joukko
1    toiminta                   1    erikois_ryhmä
1    tilanne                    1    ehdotus
1    sodan_uhka                 1    asetus
1    sisä_ministeriö            1    apu
1    reserviläinen
```

Pitempi piirrelista saadaan poimimalla substantiivit, adjektiivit ja verbit (kuva 19). Suomen kielessä erittäin yleiset verbit *olla* ja *ei* saattavat häiritä luokittelualgoritmia, jollei niitä karsita pois.

Kuva 19. Luvun 2.3 esimerkkitekstin substantiivit, adjektiivit ja verbit sekä niiden frekvenssit

```
4    poliisi                    1    reserviläinen
3    täydennys_poliisi          1    reservi
3    poikkeus_olo               1    raja
2    vuoksi                     1    päätös
2    vuo                        1    puhua
2    säädellä                   1    poliisi_osasto
2    puolustus_voima            1    poliisi_laki
2    olla                       1    pakolais_tulva
1    yleinen                    1    onnettomuus
1    voida                      1    määräys
1    velvoite                   1    mukainen
1    varautua                   1    laatia
1    valtio_neuvosto            1    kutsua
1    valta_kunta                1    kriittinen
1    vakava                     1    koota
1    vaara                      1    kokoaminen
1    uusi                       1    koetella
1    työ                        1    kanta
1    turvallisuus               1    jouko
1    toiminta                   1    joukko
1    tilanne                    1    hallinnollinen
1    tarvita                    1    erikois_ryhmä
1    tarkoittaa                 1    ei
1    suoranainen                1    ehdotus
1    sodan_uhka                 1    asetus
1    sisä_ministeriö            1    apu
1    rinnastaa                  1    aiheuttaa
```

Koska yhdyssananrajat on merkitty, myös substantiivien yhdysosien poiminta on mahdollista (kuva 20). Yksi selvä puute on kuitenkin huomattava: FINTWOL ei palauta yhdyssanojen alkuosia perusmuotoon, joten sanan *pakolais-tulva* alkuosa pysyy muodossa *pakolais* eikä yhdisty itsenäisenä esiintyvän sanan *pakolainen* kanssa.

Myös muita puutteita voidaan osoittaa. Esim. sanassa *sisäministeriö* jälkiosa *ministeriö* vaikuttaa luokittelun kannalta selvästi relevantimmalta kuin alkuosa *sisä-*, mutta molemmat saavat tässä saman painon.

Jos sanojen jakamista merkitysyksiköihin haluttaisiin viedä pitemmälle, myös johdokset voitaisiin palauttaa perussanoihinsa, jolloin sana *reserviläinen* yhdistyisi sanan *reservi* kanssa. Tämä vaihtoehto vaatisi kuitenkin lisäyksiä ja muutoksia FINCG:hen.

Kuva 20. Luvun 2.3 esimerkkitekstin substantiivit yhdysosiin jaettuina

```
9    poliisi                    1    reserviläinen
3    täydennys                  1    reservi
3    poikkeus                   1    raja
3    olo                        1    päätös
2    vuoksi                     1    pakolais
2    vuo                        1    osasto
2    voima                      1    onnettomuus
2    puolustus                  1    neuvosto
1    velvoite                   1    määräys
1    valtio                     1    ministeriö
1    valta                      1    laki
1    vaara                      1    kunta
1    uhka                       1    kokoaminen
1    työ                        1    kanta
1    turvallisuus               1    jouko
1    tulva                      1    joukko
1    toiminta                   1    erikois
1    tilanne                    1    ehdotus
1    sodan                      1    asetus
1    sisä                       1    apu
1    ryhmä
```

Edellä esitettyjä poimintatapoja voidaan yhdistellä esimerkiksi niin, että luokittelualgoritmi saa syötteekseen sekä kokonaiset substantiivit että niiden yhdysosat.

7.3 Oppimisohjelmat

7.3.1 Ohjelmien syötemuoto

Kaikki oppimisohjelmat käyttävät samaa syötemuotoa, jossa kustakin artikkelista on lueteltu siihen liitetyt luokat ja artikkelista poimitut piirteet (kuva 21). Poistin opetusaineistosta ne artikkelit, joita ei ollut luokiteltu.

Kuva 21. Esimerkki oppimisohjelman syötteestä

```
ARTICLE 000030423
TOPICS
oike-po-kou
oike-po-vah
TERMS
      5 täydennys_poliisi
      5 poliisi
      4 puolustus_voima
      3 poikkeus_olo
      2 vuoksi
      2 vuo
      2 virka-apu
      2 reservi
      2 poliisi_osasto
      2 poikkeus_tilanne
      ...
ENDARTICLE
```

7.3.2 Assosiaatiosäännöt

Assosiaatiosääntöohjelmalle annetaan poimittavien sääntöjen minimiluottamus ja -tuki komentoriviparametreilla (kuva 22).

Kuva 22. Assosiaatiosääntöohjelman käyttö

```
Usage: assoc-learn [OPTIONS] FILE > CLASSIFIER
--topic-root=TOPIC   build a subclassifier for TOPIC
--topic-levels=N     build a classifier for at most N levels
--confidence=N       confidence threshold (%)
--support=N          support threshold (# of documents)
```

Ohjelmalla voi tuottaa myös halutun syvyisen aliluokittimen halutulle luokalle. Esim. komento

```
assoc-learn --topic-root=talo --topic-levels=2 TIEDOSTO
```

tuottaa kahden tason syvyisen aliluokittimen luokalle `talo` (*talous*).

Ohjelma tulostaa aineistosta löytämänsä säännöt tekstimuodossa (kuva 23).

Kuva 23. Assosiaatiosäännöt: ote HS96-luokittimesta

```
"kult" <= "lyriikka"  (100%, 26)
"kult" <= "tavastia"  (100%, 24)
"kult" <= "ondine"  (100%, 22)
"kult" <= "fuga"  (100%, 21)
"kult" <= "tilaus_teos"  (100%, 20)
"kult" <= "hauru"  (98%, 58)
"kult" <= "iso_puro"  (98%, 57)
```

7.3.3 Naiivi Bayesin luokitin

Myös Bayes-ohjelmassa voidaan karsia piirteitä ja luokkia parametrien avulla (kuva 24). Aliluokitinvalitsimet ovat samat kuin luvussa 7.3.2.

Kuva 24. Bayes-ohjelman käyttö

```
Usage: bayes-learn [OPTIONS] FILE > CLASSIFIER
--topic-root=TOPIC   build a subclassifier for TOPIC
--topic-levels=N     build a classifier for at most N levels
--support=N          discard topic/term pairs with strength < N
--topic-support=N    discard topics with strength < N
--term-support=N     discard features with strength < N
```

Ohjelma tulostaa kunkin luokan esiintymistodennäköisyyden ja kunkin piirteen esiintymistodennäköisyyden kussakin luokassa (kuva 25). Kullekin luokalle tu-

lostetaan lisäksi oletustodennäköisyys (tässä $6{,}12655 \cdot 10^{-6}$), jota käytetään silloin, kun luokan ja piirteen välille ei ole määritelty todennäköisyyttä.

Kuva 25. Naiivi Bayesin luokitin: ote HS96-luokittimesta

```
TOPIC "kult" 1.188521e-01 6.126550e-06
"kult" "ensi-ilta" 5.820223e-04
"kult" "ulf" 4.288585e-05
"kult" "tiedustelu" 3.675930e-05
"kult" "komitea" 1.470372e-04
"kult" "estradi" 6.126550e-05
"kult" "kansan_tanssi" 1.102779e-04
"kult" "teatteri_vaunu" 3.675930e-05
```

7.3.4 Winnow

Winnow-ohjelmalle voidaan antaa komentorivillä kaikki luvussa 6.5.2 mainitut parametrit (kuva 26). Liian harvoin esiintyvät luokat tai piirteet voidaan karsia. Aliluokitinvalitsimet ovat samat kuin luvussa 7.3.2.

Kuva 26. Winnow-ohjelman käyttö

```
Usage: winnow-learn [OPTIONS] [FILE] > CLASSIFIER
--topic-root=TOPIC   build a subclassifier for TOPIC
--topic-levels=N     build a classifier for at most N levels
--alpha=F            promotion parameter
--beta=F             demotion parameter
--threshold=F        threshold
--lower-threshold=F  lower threshold
--upper-threshold=F  upper threshold
--topic-support=N    discard topics with strength < N
--term-support=N     discard features with strength < N
--strength=FUNCTION  linear, binary, trinary or sqrt
```

Ohjelma käyttää oletusarvoisesti piirteiden voimakkuutena niiden lukumääriä. Voimakkuusfunktion voi vaihtaa valitsimella `--strength`. Vaihtoehtoja ovat binaarinen (0 = piirre ei esiinny, 1 = piirre esiintyy dokumentissa) ja kolmiarvoinen (0 = piirre ei esiinny, 1 = piirre esiintyy kerran, 2 = piirre esiintyy useammin kuin kerran) voimakkuusfunktio sekä lukumäärän neliöjuuri.

Ohjelma tulostaa kunkin luokka-piirreparin positiivisen ja negatiivisen painon (kuva 27). Algoritmi tulostaa vain alkuarvoista poikkeavat painot; jollei luokka-piirreparille liity painoa, opetusohjelma käyttää erikseen talletettuja alkuarvoja (`+ default` ja `- default`).

Kuva 27. Winnow: ote HS96-luokittimesta

```
+ default 0.249702099832885
- default 0.124851049916443
+ "kult" "ula" 0.182033
- "kult" "ula" 0.166177
+ "kult" "ulf" 0.17841
- "kult" "ulf" 0.16287
+ "kult" "komitea" 0.211581
- "kult" "komitea" 0.129299
```

7.4 Testiohjelma

Toteutin kaikki luokittelualgoritmit yhdellä testiohjelmalla (kuva 28). Aliluokittimia testattaessa ohjelma ohittaa valitsimen `--topic-root` avulla ne artikkelit, jotka eivät kuulu tutkittavaan luokkaan. Winnow-algoritmin voimakkuusfunktiovalitsin `--strength` on mukana myös testiohjelmassa.

Kuva 28. Testiohjelman käyttö

```
Usage: classify [OPTIONS] [FILE]
--algorithm=NAME     Assoc/Bayes/Winnow
--rule-file=FILE     classifier file
--topic-root=TOPIC   skip articles not belonging to TOPIC
--strength=FUNCTION  linear, binary, trinary or sqrt
```

Ohjelman syötemuoto on sama kuin oppimisohjelmilla, ja se tulostaa jokaiselle artikkelille oikeat luokitukset ja omat ehdotuksensa (kuva 29). Tulostukseen voidaan tarvittaessa lisätä myös kunkin luokan saamat pistemäärät, jos epävarmimpia ehdotuksia halutaan karsia.

Kuva 29. Esimerkki testiohjelman tulostuksesta (T = oikeat luokitukset, G = ohjelman ehdotukset)

```
T yhte-yh-hy yhte-yh-mu yhte-yh-ra
G tied poli yhte koul kult ulko kunn talo aatt oike
T yhte-si-pa-Su ulko-pu-so-sis
G ulko yhte poli talo koul liik ympä kunn kult tied
T ulko-po-vaa ulko-po-pr
G ulko poli aatt yhte tied koul talo kult oike aika
```

Tulosten tilastointia varten oikeiden luokitusten ja ohjelman ehdotusten luokittelusyvyys on vielä asetettava samaksi (kuva 30).

Kuva 30. Esimerkki lopullisesta testiohjelman tulostuksesta (T = oikeat luokitukset, G = ohjelman ehdotukset)

```
T yhte
G tied poli yhte koul kult ulko kunn talo aatt oike
T yhte ulko
G ulko yhte poli talo koul liik ympä kunn kult tied
T ulko
G ulko poli aatt yhte tied koul talo kult oike aika
```

8 Testaus

8.1 Mittausmenetelmät

Luokittelun hyvyyttä arvioidaan tavallisesti kahdella suureella: **Saanti** eli **löytyvyys** (*recall*) on oikeaan osuneiden luokitusehdotusten osuus kaikista oikeista luokituksista. **Tarkkuus** (*precision*) on oikeaan osuneiden luokitusehdotusten osuus kaikista luokitusehdotuksista.

Jos

L on kaikkien oikeiden luokitusten joukko,

E on luokitusehdotusten joukko ja

$L \cap E$ on oikeaan osuneiden ehdotusten joukko,

saanti $r = |L \cap E| / L$ ja tarkkuus $p = |L \cap E| / E$ (kuva 31).

Kuva 31. Esimerkki saannista ja tarkkuudesta

L	*Joulupukki*, *porot*, *Lappi*
E	*Joulupukki*, *Lappi*, *saamelaiset*, *markkinointi*
$L \cap E$	*Joulupukki, Lappi*
saanti	2 / 3 ≈ 67 %
tarkkuus	2 / 4 = 50 %

Luokitusohjelman toiminnan optimoiminen vaatii tasapainoilua saannin ja tarkkuuden välillä. Jos ehdotuslistaan otetaan mukaan myös epävarmoja ehdotuksia, saanti paranee tarkkuuden kustannuksella. Jos ohjelmaa käytetään interaktiivisesti, ehdotuslista voi olla kohtalaisen pitkä, koska käyttäjä voi poimia listasta nopeasti hyvät ehdotukset ja karsia huonot.

Algoritmeja on helpointa asettaa summittaisesti paremmuusjärjestykseen, jos ehdotuslistat katkaistaan aina oikeiden luokitusten listan mittaisiksi. Tällöin saanti on yhtä suuri kuin tarkkuus (tai hieman pienempi, jos ehdotuksia on vähemmän kuin oikeita luokituksia), ja kaksi tunnuslukua on saatu puristettua yhdeksi **saannin ja tarkkuuden yhtäsuuruuspisteeksi** (*recall/precision break-even point*). On tietenkin huomattava, ettei tällainen katkaisutapa ole realistinen,

koska todellisella luokitteluohjelmalla ei ole käytössään tietoa asiantuntijan artikkeliin liittämien luokkien määrästä. Jos ylimääräiset ehdotukset häiritsevät käyttäjää, ohjelma voi karsia esimerkiksi ne ehdotukset, joiden pistemäärä eroaa parhaan ehdotuksen pistemäärästä enemmän kuin tietyn kynnysarvon verran.

Jos seuraavien lukujen taulukoissa ei ole erikseen mainittu saantia ja tarkkuutta, luvut ovat saannin ja tarkkuuden yhtäsuuruuspisteitä.

8.2 Aineiston jako

Jotta kaikki vuodenajat ja viikonpäivät tulisivat edustetuiksi sekä opetuksessa että testauksessa, poimin HS96-aineiston 353 lehdestä aikajärjestyksessä kaksi kolmesta opetukseen ja joka kolmannen testaukseen. Opetusaineistossa oli 236 lehteä (34 978 luokiteltua artikkelia) ja testiaineistossa 117 lehteä (17 673 luokiteltua artikkelia). Luokittelemattomat artikkelit jätin kokonaan pois testauksesta.

Kokeilujen nopeuttamiseksi valitsin opetusaineistosta vielä 79 lehden kokoisen pienen opetusaineiston. Havaitsin kuitenkin pienen opetusaineiston tulosten eroavan yleensä liikaa suuren aineiston tuloksista. Jäljempänä esitettävät tulokset onkin saavutettu täydellisellä opetusaineistolla.

8.3 Pääluokitin

8.3.1 Kolme vertailualgoritmia

Testasin pääluokittimen toimintaa kaikilla kolmella algoritmilla ja erimittaisilla ehdotuslistoilla (kuva 32). Sarakkeen ”=” luvut on saatu katkaisemalla listat oikeiden ehdotusten listan mittaisiksi.

Kuva 32. Pääluokittimen toiminta eri algoritmeilla (r = saanti, p = tarkkuus)

Algoritmi	**Ehdotuslistan pituus**					
	=		**3**		**5**	
	r	*p*	*r*	*p*	*r*	*p*
Assosiaatio-säännöt	67,8 %	67,9 %	82,9 %	35,2 %	91,3 %	23,8 %
Bayes	75,3 %	75,3 %	88,4 %	37,5 %	94,3 %	24,0 %
Winnow	71,8 %	71,9 %	83,9 %	35,6 %	90,9 %	23,1 %

Pyrin valitsemaan parametrit niin, että kaikki algoritmit käyttäisivät suunnilleen saman verran muistia. Käytin tässä vaiheessa piirteiden voimakkuutena niiden esiintymisten lukumäärää. (Winnow-algoritmissa muutkin vaihtoehdot ovat mahdollisia.) Poistin kaikkien algoritmien syötteistä ne luokat ja piirteet, jotka esiintyivät alle 5 kertaa koko aineistossa. Assosiaatiosäännöistä karsin pois ne, joiden luottamus oli alle 25 % tai tuki alle 3 dokumenttia. Bayesin opetusaineistosta karsin pois alle 5 kertaa esiintyneet luokka-piirreparit.

Winnow-algoritmilla käytin vahvistus- ja heikennysparametreja $\alpha = 1{,}05$ ja $\beta = 0{,}95$ sekä kynnysarvoja $\theta^- = 0{,}9$ ja $\theta^+ = 1{,}1$. Parametrien α ja β arvot ovat kompromissi. Kun arvot lähestyvät lukua 1, tulokset paranevat hieman, mutta luokitin kasvaa samalla epäkäytännöllisen suureksi. Alkuarvoissa pysyneitä painoja ei talleteta muistiin, joten painojen liiallinen hienosäätö kasvattaa muistintarvetta.

Kun luokittelu sujui vähintään 90-prosenttisesti oikein, karsin ne piirteet, joiden arvoa oli muutettu enintään yhden askelen verran. Pysäytin opetusajon 95 %:n kohdalla. Tämä raja saavutettiin yleensä jo yhden tai kahden opetuskierroksen jälkeen. Dagan, Karov ja Roth jatkoivat opetusta täysin virheettömään tulokseen asti, mutta HS96-aineistolla näin pitkä ajo huononsi hieman tuloksia.

Assosiaatiosäännöistä oli paljon apua siinä tehtävässä, jota varten ne on kehitettykin: ongelma oli helpompi hahmottaa tarkastelemalla sääntöjä kuin tilastollisten algoritmien tuottamia luokittimia. Assosiaatiosääntöihin perustuva algoritmini ei kuitenkaan toiminut yhtä hyvin kuin Bayes- ja Winnow-algoritmit.

Assosiaatiosääntöalgoritmin puutteita olivat sanojen frekvenssien jättäminen huomiotta opetusvaiheessa sekä luokkien esiintymistodennäköisyyksien ottaminen huomioon vain epäsuorasti sääntöjen määrän kautta. Frekvenssitiedon poisjättö olisi todennäköisesti haitannut vielä enemmän, jos dokumentit olisivat olleet pitempiä. Näiden puutteiden poistaminen olisi kuitenkin tuottanut hyvin paljon naiivia Bayesin luokitinta muistuttavan algoritmin, joten jätin assosiaatiosäännöt pois seuraavista testeistä.

8.3.2 Piirteiden valinta

Testasin piirteiden valinnan vaikutusta pääluokittimen toimintaan Winnow- ja Bayes-algoritmeilla (kuva 33). Poistin aineistosta verbit *olla* ja *ei*, jotka huononsivat Bayes-algoritmin tuloksia muutamalla prosenttiyksiköllä ja Winnow-algoritmin tuloksia peräti 30 prosenttiyksiköllä. Käytin molemmilla algoritmeilla samoja parametreja kuin luvussa 8.3.1.

Kuva 33. Piirteiden valinnan vaikutus tuloksiin

Piirteet	Bayes	Winnow
substantiivit	75,3 %	71,8 %
yhdysosiin jaetut substantiivit	73,9 %	72,2 %
kokonaiset substantiivit ja yhdysosat	75,2 %	73,1 %
substantiivit, adjektiivit ja verbit	74,7 %	72,2 %
yhdysosiin jaetut substantiivit, adjektiivit ja verbit	73,4 %	72,2 %

Molemmat algoritmit toimivat odotusten mukaisesti. Bayes toimii hyvin, kun aineisto sisältää mahdollisimman vähän epärelevantteja piirteitä; Winnow pystyy paremmin valitsemaan aineistosta relevanteimmat piirteet. Suurin osa luokittelua hyödyttävistä piirteistä on substantiiveja, mutta myös niiden yhdysosissa on hyödyllistä tietoa kohinan joukossa. Adjektiiveista ja verbeistä vain pieni osa on relevantteja.

8.3.3 Winnow ja voimakkuusfunktio

Kaikissa edellisissä kokeissa olin käyttänyt piirteiden voimakkuutena niiden esiintymisten lukumäärää. Selvitin, parantaisiko toisenlainen voimakkuusfunktio Winnow-algoritmin tuloksia pääluokittimella. Annoin algoritmille syötteeksi sekä substantiivit että niiden yhdysosat.

Reutersin aineistolla lukumäärän neliöjuuri oli hyvä valinta voimakkuusfunktioksi [DKR97]. Kokeilin myös binaarista (0 = piirre ei esiinny, 1 = piirre esiintyy dokumentissa) ja kolmiarvoista (0 = piirre ei esiinny, 1 = piirre esiintyy kerran, 2 = piirre esiintyy useammin kuin kerran) voimakkuusfunktiota. Havaitsin kuitenkin pelkän piirteiden lukumäärän toimivan parhaiten HS96-aineistolla (kuva 34).

Kuva 34. Voimakkuusfunktion vaikutus Winnow-algoritmin tuloksiin

Voimakkuusfunktio	Winnow
lukumäärä	73,1 %
lukumäärän neliöjuuri	71,8 %
binaarinen (0/1)	71,2 %
kolmiarvoinen (0/1/2)	72,9 %

8.4 Luokitteluhierarkia

HS96:n luokkien runsauden vuoksi kaikkia haaroja ja tasoja ei ollut järkevää toteuttaa erillisinä luokittimina. Pyrin vähentämään luokittimien määrää ja testasin, mitä tapahtuu, jos yksi luokitin sisältää enemmän kuin yhden luokittelutason. Halusin myös selvittää, kuinka hienojakoiseen luokitukseen algoritmeilla olisi mahdollista päästä. Kaikissa seuraavissa taulukoissa luokittimet ovat luokkaan kuuluvien dokumenttien määrän (eivät siis aliluokkien määrän) mukaisessa suuruusjärjestyksessä.

Ensin muodostin kaikille 17 pääluokalle yhden ja kahden tason syvyiset aliluokittimet (kuva 35). Käytin molemmilla algoritmeilla samoja parametreja kuin luvussa 8.3.2. Suurin syy eri luokittimien tulosten välisiin eroihin on aliluokkien määrä (kuva 36). Mitä useampien luokkien väliltä on valittava, sitä huonommin ehdotukset osuvat kohdalleen.

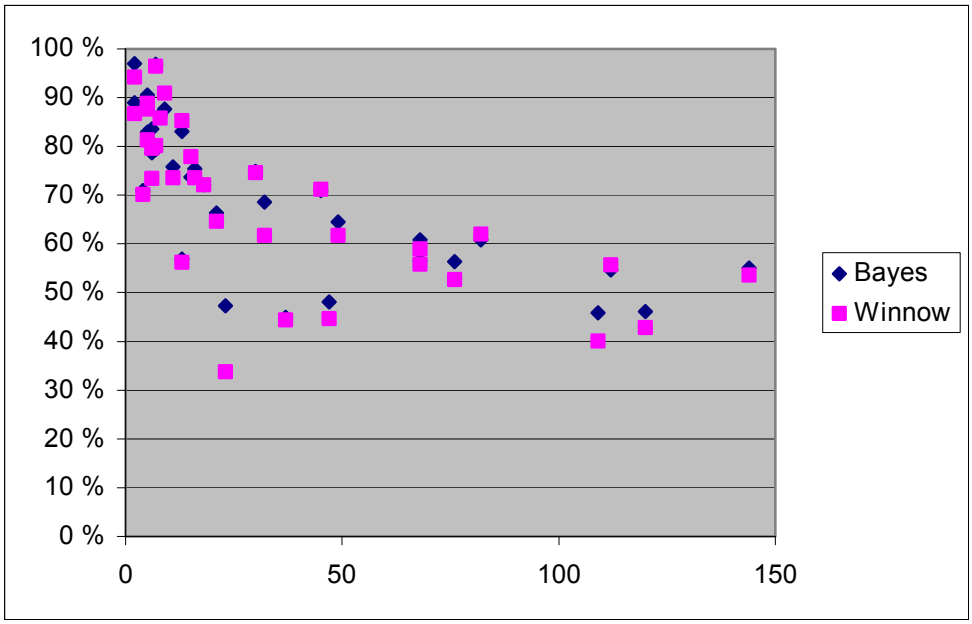
Tutkin tarkemmin niitä luokan `talo` aliluokkia, jotka esiintyivät vähintään viidessä dokumentissa (kuva 37). Tulokset pysyvät vielä suurimmaksi osaksi varsin hyvinä, mutta osassa luokista on liian vähän opetusdokumentteja luotettavan tuloksen saavuttamiseksi.

Muodostin vielä aliluokittimet luokalle `talo-tat` eli *taloustarvikkeet ja raaka-aineet* (kuva 38). Aliluokkia alkoi olla jo niin vähän, että otin mukaan kaikki alemmat tasot enkä vain ensimmäisen ja toisen tason aliluokkia. Opetusdokumenttien vähyys alkoi jo selvästi haitata tuloksia, joten päätin olla jatkamatta syvemmälle hierarkiaan.

Kuva 35. Ylimmän tason aliluokittimien toiminta eri luokilla ja algoritmeilla (lkm = vähintään viidesti esiintyneiden aliluokkien lukumäärä)

Luokka	Luokittimen syvyys					
	1			2		
	lkm	Bayes	Winnow	lkm	Bayes	Winnow
talo	16	75,4 %	73,6 %	144	55,0 %	53,6 %
ulko	18	72,2 %	72,1 %	112	54,7 %	55,7 %
harr	7	96,8 %	96,4 %	30	74,9 %	74,6 %
kult	13	83,1 %	85,3 %	68	60,9 %	59,0 %
poli	15	73,7 %	77,9 %	82	60,9 %	62,0 %
yhte	11	75,8 %	73,6 %	76	56,4 %	52,7 %
kunn	68	56,5 %	55,9 %	109	45,8 %	40,1 %
liik	8	86,4 %	85,8 %	120	46,1 %	42,8 %
koul	5	90,6 %	87,6 %	21	66,3 %	64,7 %
tied	9	87,7 %	90,9 %	45	71,0 %	71,2 %
oike	5	82,9 %	81,4 %	49	64,5 %	61,7 %
ympa	7	79,4 %	80,1 %	32	68,6 %	61,8 %
asun	4	71,0 %	70,2 %	47	48,1 %	44,7 %
onne	2	97,0 %	94,2 %	5	90,2 %	88,8 %
aatt	6	78,7 %	73,5 %	37	44,9 %	44,4 %
puol	2	88,9 %	86,7 %	13	56,9 %	56,3 %
aika	6	83,6 %	79,6 %	23	47,3 %	33,8 %

Kuva 36. Luokkien määrän vaikutus luokittelutuloksiin (ylimmän tason aliluokittimet syvyyksillä 1 ja 2)



Kuva 37. Luokan `talo` *aliluokittimien toiminta eri algoritmeilla (lkm = vähintään viidesti esiintyneiden aliluokkien lukumäärä)*

Luokka	Luokittimen syvyys					
	1			2		
	lkm	Bayes	Winnow	lkm	Bayes	Winnow
talo-tat	16	77,6 %	69,6 %	50	57,2 %	48,9 %
talo-tal	32	71,6 %	70,7 %	56	60,8 %	57,3 %
talo-ra	12	78,0 %	78,4 %	36	58,7 %	49,6 %
talo-ma	3	97,2 %	98,4 %	29	58,7 %	50,0 %
talo-ty	33	62,0 %	58,4 %	51	51,0 %	46,0 %
talo-teo	5	87,2 %	86,8 %	30	52,8 %	47,8 %
talo-ve	3	86,1 %	84,4 %	29	51,6 %	45,4 %
talo-el	3	94,5 %	95,9 %	9	72,2 %	68,2 %
talo-en	5	76,6 %	75,5 %	16	44,6 %	33,7 %
talo-va	5	71,7 %	60,4 %	9	48,1 %	53,7 %
talo-pa	8	51,8 %	34,1 %	10	44,7 %	32,9 %
talo-hi	6	41,7 %	54,2 %	7	39,6 %	52,1 %
talo-ul	8	57,7 %	50,0 %	9	45,3 %	37,7 %

Kuva 38. Luokan `talo-tat` *aliluokittimien toiminta eri algoritmeilla*

Luokka	Aliluokkien lkm	Algoritmi	
		Bayes	Winnow
talo-tat-el	49	35,8 %	24,3 %
talo-tat-ke	12	31,0 %	32,4 %
talo-tat-ra	9	34,1 %	28,0 %
talo-tat-me	5	41,4 %	34,5 %
talo-tat-kä	3	29,2 %	29,2 %
talo-tat-kod	5	22,2 %	25,9 %
talo-tat-na	3	12,5 %	12,5 %
talo-tat-st	2	80,0 %	84,0 %
talo-tat-as	3	48,4 %	41,9 %
talo-tat-pak	3	68,4 %	68,4 %
talo-tat-maat	2	5,9 %	5,9 %
talo-tat-kon	2	10,0 %	10,0 %

8.5 Aika- ja muistivaatimukset

Opetuskoneenani oli Sun SPARCsystem 10 (90 MHz, 256 MB keskusmuistia) ja testikoneena tehokkaampi, mutta opetusajoihin liian vähän virtuaalimuistia sisältävä Linux-palvelin (Pentium II, 300 MHz, 256 MB). FINCG toimii toistaiseksi vain Sun-ympäristössä.

Opetuslaitteistossa yksi Winnow-opetusajokierros kesti täydellä aineistolla noin neljä tuntia ja kulutti enimmillään noin 200 MB keskusmuistia. Naiivin Bayesin luokittimen ja assosiaatiosääntöjen muodostaminen sujui nopeammin ja pienemmässä muistitilassa.

Opetuksen tehokkuutta olennaisempaa on kuitenkin luokitteluohjelman muistintarve ja nopeus. Pääluokitin vaati assosiaatiosäännöillä 35 MB muistia, Bayesillä 29 MB ja Winnow'lla 34 MB.

Suurimmalla pääluokalla `talo` (*talous*) yhden tason syvyinen Bayesin luokitin kulutti 9 MB ja kahden tason syvyinen luokitin 12 MB muistia. Winnow'n vastaavat luvut olivat 11 MB ja 50 MB.

Muistinkulutusta on mahdollista pienentää tietorakenteita tiivistämällä: Perl-kielen assosiatiiviset taulukot eivät todennäköisesti ole optimaalinen talletusmuoto luokittimelle. Luokitinta voi pienentää myös karsimalla lisää piirteitä, jollei tulosten lievä huononeminen haittaa.

Luokitteluohjelman hitain osa on piirteiden valinta (kuva 39). FINCG koostuu nykyisellään monesta kymmenestä erillisestä moduulista. Ohjelmaa siistimällä on mahdollista kasvattaa sen nopeus vähintään muutamaan sataan sanaan sekunnissa. Myös itse luokittelualgoritmit toiminevat nopeammin C:llä toteutettuina.

Kuva 39. Luokittelun nopeus

Ohjelma	**Sun**	**Linux**
piirteiden valinta	90 sanaa/s	–
Bayes	7 dokumenttia/s	35 dokumenttia/s
Winnow	4 dokumenttia/s	18 dokumenttia/s

9 Yhteenveto

Tutkin *Helsingin Sanomien* vuosikerran 1996 artikkelien luokittelemista lehden oman asiasanaston mukaan ja päädyin seuraaviin tuloksiin:

- Naiivi Bayesin luokitin toimi *Helsingin Sanomien* aineistolla yleensä hieman Winnow-algoritmia paremmin, vaikka englanninkielisellä Reutersin aineistolla tulos oli ollut päinvastainen. Suurimpana syynä saattaa olla luokittelutapojen ero: HS:n asiasanasto on tarkoitettu yleiskäyttöiseksi, kun taas Reutersin aineistossa kaikki luokat liittyvät talouselämään. Jos talousasiat ovat vain artikkelin sivuaihe, mutta artikkelin luokka on valittu niiden mukaan, suurin osa artikkelin sanastosta ei ole relevanttia. Aineisto sisältää siis enemmän kohinaa, ja Winnow-algoritmin kohinansietokyky on parempi kuin Bayesin.
- Suomenkielisten lehtiartikkelien luokittelu vaatii sanojen taipumisen vuoksi lingvistisen esikäsittelyn. Luokittelupiirteiksi riittävät perusmuotoon palautetut substantiivit. Winnow-algoritmille on hyötyä myös substantiivien yhdysosista.
- HS96-aineistossa piirteiden lukumäärä oli paras valinta Winnow-algoritmin voimakkuusfunktioksi, vaikka Reutersin aineistolla lukumäärän neliöjuuri oli toiminut paremmin.
- Jollei luokittelun laadusta haluta tinkiä, luokitteluohjelman muistinkulutus pysyy käytännöllisissä rajoissa, jos luokkia on enintään muutamia kymmeniä.
- *Helsingin Sanomien* asiasanaston kaikkien haarojen toteuttaminen automaattisella luokittelulla ei ole mahdollista: pienimmistä aliluokista ei kerry riittävästi opetusaineistoa toimivan luokittimen muodostamiseksi. Ainakin kolmen tason syvyyteen on kuitenkin mahdollista päästä.

Seuraavat kysymykset kaipaavat jatkotutkimusta:

- Miten hyvin käyttämäni menetelmät sopivat muihin kieliin kuin suomeen ja englantiin?

- Olisiko algoritmien paremmuusjärjestys sama toisenlaisella aineistolla (esim. teknisillä dokumenteilla tai Usenetin uutisryhmäartikkeleilla) tai luokituksella (esim. jonkin erityisalan omalla asiasanastolla)?
- Olisiko metatiedosta (artikkelin sijainti lehdessä, kirjoittaja, sanojen lukumäärä jne.) hyötyä luokittelussa?
- Paranisivatko tulokset rajoitetulla substantiivilausekkeiden poiminnalla (esim. pitämällä erisnimet yhtenäisinä)?
- Toimisiko jokin parametrien yhdistelmä vielä paremmin kuin kokeilemani vaihtoehdot?
- Miten luokittimen tuottama ehdotuslista tulisi katkaista, jotta epävarmimmat ehdotukset jäisivät pois häiritsemästä käyttäjää?
- Miten usein luokitinta pitää päivittää? Kuinka nopeasti sanomalehden käyttämä sanasto muuttuu?

Muutaman prosenttiyksikön parannus saantiin tai tarkkuuteen ei kuitenkaan loppujen lopuksi vaikuta olennaisesti ohjelman käytettävyyteen. Tärkeintä on saada aikaan toimiva käyttöliittymä, joka nopeuttaa selvästi luokittelijan työtä.

Viitteet

[AMS96] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A. I., ”Fast Discovery of Association Rules”, s. 307–328 teoksessa *Advances in Knowledge Discovery and Data Mining*. Toim. U. M. Fayyad *et al*. AAAI Press / The MIT Press. Menlo Park, CA 1996.

[AHK97] Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A. I., *Applying Data Mining Techniques in Text Analysis* [online]. Helsingin yliopisto, tietojenkäsittelytieteen laitos 1997. [Viitattu 8.1.1998.] Saatavilla PostScript-muodossa:
`http://www.cs.helsinki.fi/~oheinone/publications/`.

[AlH92] Alkula, R., Honkela, T., *Tekstin tallennus- ja hakumenetelmien kehittäminen suomen kielen tulkintaohjelmien avulla*. FULLTEXT-projektin loppuraportti. VTT. Espoo 1992.

[ApD94] Apté, C., Damerau, F., ”Automated Learning of Decision Rules for Text Categorization”, *ACM Transactions on Information Systems* **12** (3), July 1994, s. 233–251.

[CoS96] Cohen, W. W., Singer, Y., ”Context-sensitive learning methods for text categorization”, s. 307–315 teoksessa *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 1996.

[DKR97] Dagan, I., Karov, Y., Roth, D., *Mistake-Driven Learning in Text Categorization* [online]. Computation and Language E-Print Archive, Los Alamos National Laboratory, 1997. [Viitattu 8.1.1998.] Saatavilla PostScript-muodossa: `http://xxx.lanl.gov/cmp-lg/` (artikkeli cmp-lg/9706006).

[FIN98] *FINTWOL: Finnish Morphological Analyser* [online]. Lingsoft Oy. [Viitattu 23.5.1998.] Saatavilla www-muodossa:
`http://www.lingsoft.fi/doc/fintwol/`.

[HaK79] Hakulinen, A., Karlsson, F., *Nykysuomen lauseoppia*. Suomalaisen Kirjallisuuden Seura. Jyväskylä 1979.

[HAN90] Hayes, P. J., Andersen, P. M., Nirenburg, I. B., Schmandt, L. M., ”TCS: A shell for context-based text categorization”, s. 320–326 teoksessa *Proceedings of the 6th Conference on Artificial Intelligence Applications (CAIA)*. IEEE. Los Alamitos, CA 1990.

[Hjo97] Hjorth, T. *et al*, *Arkistojuttujen valmisteluohjeet*. Sanoma Oy:n sisäinen dokumentti Windows-ohjetiedostona. 28.8.1997.

[Hon97] Honkela, T., *Self-Organizing Maps in Natural Language Processing*. Väitöskirja. Teknillinen korkeakoulu, neuroverkkojen tutkimusyksikkö. 1997.

[Jär95] Järvelin, K., *Tekstitiedonhaku tietokannoista*. Suomen ATK-Kustannus. Jyväskylä 1995.

[JKa97] Karlgren, J., *Non-Topic Information Retrieval*. Helsingin yliopiston yleisen kielitieteen laitoksen seminaari. 12.11.1997.

[Kar85] Karlsson, F., ”Parsing Finnish in terms of Process Grammar”, s. 137–176 teoksessa *Computational Morphosyntax: Report on Research 1981-1984*. Helsingin yliopisto, yleisen kielitieteen laitos. 1985.

[Kar86] Karlsson, F., ”Process Grammar”, s. 162–171 teoksessa *Papers from the Ninth Scandinavian Conference of Linguistics*. Toim. Ö. Dahl. Stockholms universitet, Institutionen för lingvistik. 1986.

[Kar98] Karlsson, F., *FINCG: Morphological Disambiguation of Finnish* (tulossa).

[KVH95] Karlsson, F., Voutilainen, A., Heikkilä, J., Anttila, A, *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter. Berlin / New York 1995.

[Koh95] Kohonen, T., *Self-Organizing Maps*. Springer. Berlin, Heidelberg 1995.

[Kos83] Koskenniemi, K., *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Väitöskirja. Helsingin yliopisto, yleisen kielitieteen laitos. 1983.

[Lah97a] Lahtinen, T., *Framework for Extracting Index Terms*. Helsingin yliopisto, yleisen kielitieteen laitos. 1997.

[Lah97b] Lahtinen, T., *An index term corpus as a means to develop an automatic indexer*. Helsingin yliopisto, yleisen kielitieteen laitos. 1997.

[Lew92] Lewis, D., ”An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task”, s. 37–50 teoksessa *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. New York 1992.

[LeR94] Lewis, D., Ringuette, M., ”A comparison of two learning algorithms for text categorization”, teoksessa *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*. University of Las Vegas. 1994.

[LiY97] Li, H., Yamanishi, K., *Document Classification Using a Finite Mixture Model* [online]. Computation and Language E-Print Archive, Los Alamos National Laboratory, 1997. [Viitattu 8.1.1998.] Saatavilla PostScript-muodossa: `http://xxx.lanl.gov/cmp-lg/` (artikkeli cmp-lg/9705005).

[Lit88] Littlestone, N., ”Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm”. *Machine Learning* **2** (4), 1988, s. 285–318.

[Maa95] Maarek, Y. S., *Organizing documents to support browsing in digital libraries* [online]. 37th Allerton Institute, 1995. [Viitattu 8.1.1998.] Saatavilla www-muodossa:
`http://edfu.lis.uiuc.edu/allerton/95/s4/maarek.html`.

[Man97] Mannila, H., *A tutorial on data mining* [online]. Helsingin yliopisto, tietojenkäsittelytieteen laitos. [Viitattu 8.1.1998.] Saatavilla www-muodossa:
`http://www.cs.helsinki.fi/research/fdk/publications/`.

[MLW92] Masand, B., Linoff, G., Waltz, D., ”Classifying News Stories using Memory Based Reasoning”, s. 59–65 teoksessa *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. New York 1992.

[Mit97] Mitchell, T. M., *Machine Learning*. McGraw-Hill. New York 1997.

[PBW97] Pedersen, T., Bruce, R., Wiebe, J., *Sequential Model Selection for Word Sense Disambiguation* [online]. Computation and Language E-Print Archive, Los Alamos National Laboratory, 1997. [Viitattu 8.1.1998.] Saatavilla PostScript-muodossa: `http://xxx.lanl.gov/cmp-lg/` (artikkeli cmp-lg/9702008).

[Rij79] van Rijsbergen, C. J., *Information Retrieval* [online]. University of Glasgow, 1979. [Viitattu 8.1.1998.] Saatavilla www-muodossa: `http://www.dcs.glasgow.ac.uk/Keith/Preface.html`.

[Ros58] Rosenblatt, F., ”The perceptron: A probabilistic model for information storage and organization in the brain”. *Psychological Review* **65**, 1958, s. 386–408. Uusintapainos teoksessa *Neurocomputing: Foundations of Research*. Toim. Anderson, J. A., Rosenfeld, E. MIT Press. 1988.

[Sag90] Sager, J. C., *A Practical Course in Terminology Processing*. John Benjamins Publishing Company. Amsterdam/Philadelphia 1990.

[Soe74] Soergel, D., *Indexing Languages and Thesauri: Construction and Maintenance*. Melville Publishing Company. USA 1974.

[WPW95] Wiener, E., Pedersen, J., Weigend, A., ”A neural network approach to topic spotting”, teoksessa *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*. University of Las Vegas. 1995.

[Vou93] Voutilainen, A., ”NPtool, a detector of English noun phrases” teoksessa *Proceedings of Workshop on Very Large Corpora*. Ohio State University. 1993. Saatavilla myös www-muodossa: `http://www.lingsoft.fi/doc/nptool/intro/`.

[Vou96] Voutilainen, A., *A Short Introduction to ENGCG* [online]. Lingsoft Oy 1996. [Viitattu 24.3.1998.] Saatavilla www-muodossa: `http://www.lingsoft.fi/doc/engcg/intro/`.

[Yar95] Yarowsky, D., ”Unsupervised Word Sense Disambiguation”, teoksessa *Proceedings of the 33rd Annual Meeting of the ACL*. 1995.

Muuta kirjallisuutta

[Har95] Harman, D. K. (toim.), *Overview of the Third Text Retrieval Conference (TREC-3)*. National Institute of Standards and Technology. Washington 1995.

[HeH96] Hearst, M., Hirsh, H. (toim.), *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, March 25-27 [online]. [Viitattu 13.1.1998.] Saatavilla PostScript-muodossa: `http://www.parc.xerox.com/istl/projects/mlia/`.

[HYR96] Hodges, J., Yie, S., Reighart, R., Boggess, L., ”An automated system that assists in the generation of document indexes”. *Natural Language Engineering* **2** (2), 1996, s. 137–160.

[LaC96] Larkey, L. S., Croft, W. B., ”Combining Classifiers in Text Categorization”. *Proceedings of SIGIR 96*.

[MiA95] Milton, J. S., Arnold, J. C., *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*. Third Edition. McGraw-Hill. New York 1995.

[RiL94] Riloff, E., Lehnert, W., ”Information Extraction as a Basis for High-Precision Text Classification”. *ACM Transactions on Information Systems* **12** (3), July 1994, s. 296–333.

[Soe85] Soergel, D., *Organizing Information: Principles of Data Base and Retrieval Systems*. Academic Press. USA 1985.

[WeK91] Weiss, S. M., Kulikowski, C. A., *Computer Systems That Learn*. Morgan Kaufmann Publishers. San Mateo, CA 1991.

[YSA88] *Yleinen suomalainen asiasanasto*. Kokeilupainos. Helsingin yliopiston kirjasto. Helsinki 1988.