OULUN YLIOPISTO
UNIVERSITY of OULU

TIETOTEKNIIKAN OSASTO

Ari Mustonen

PÄÄOSIN OHJAAMATON SANASTON POIMINTA RAKENTEETTOMASTA TEKSTISTÄ

Diplomityö
Tietotekniikan koulutusohjelma
Helmikuu 2014

Mustonen A. (2014) Pääosin ohjaamaton sanaston poiminta rakenteettomasta tekstistä. Oulun yliopisto, tietotekniikan osasto, 64 s.

TIIVISTELMÄ

Sanaston kattavuudella on suuri merkitys monille luonnollista kieltä käsitteleville algoritmeille. Sanaston puute vaikeuttaa tällaisten algoritmien soveltamista esimerkiksi vähemmistökieliin liittyviin ongelmiin. Sanastojen tuottaminen ja laajentaminen perinteisin menetelmin on työlästä ja kallista, joten on tarve kehittää automaattisia, yleiskäyttöisiä ja kieliriippumattomia sanaston kerääjiä.

Automaattisia sanaston kerääjiä on olemassa muutamia, mutta niiden yleiskäyttöisyyttä, laatua ja soveltamista useille kielille voidaan vielä parantaa. Kehittyneeseen sanaston keräämiseen voidaan soveltaa uusimpia ohjaamattomia sanojen erottelun, morfologian induktion, ja sanaluokan induktion menetelmiä. Monet kiinnostavimmista menetelmistä hyödyntävät Bayesin menetelmää.

Tässä diplomityössä toteutettiin pääosin ohjaamaton, useaa kieltä tukeva sanaston kerääjä. Se otti syötteenä merkitsemättömän korpuksen ja tuotti listan sanoja ja niiden sanaluokkia. Järjestelmän kaikki tärkeimmät osat pohjautuivat ei-parametriseen Bayesin menetelmään: sanojen erottelu ja morfologian induktio toteutettiin hierarkkisella Pitman-Yor-prosesseilla ja sanaluokan induktio Pitman-Yor-prosessin mikstuurimallilla.

Toteutus saavutti 16%:n tarkkuuden suomenkielisten sanojen perusmuotojen poiminnassa, kun sanaluokkatietoa ei huomioitu. Sanojen perusmuotojen ja sanaluokkien yhdistelmien poiminnassa tarkkuus oli 3%:a. Toiminnan arvioitiin olevan samaa tasoa englannilla ja japanilla.

Ratkaisun eri aliosien suorituskyvyt olivat heikkoja vastaavien osien uusimpiin toteutuksiin verrattuna. Etenkin morfologian ja sanaluokan induktion suorituskykyä voitaisiin kehittää huomattavasti. Parempia tuloksia voitaisiin saavuttaa myös sulauttamalla järjestelmän aliosia tiiviimmin yhteen.

Avainsanat: luonnollisen kielen käsittely, sanaston poiminta, morfologian induktio, sanojen erottelu, sanaluokan induktio, Bayesin menetelmä

Mustonen A. (2014) Mostly-unsupervised lexicon acquisition from unstructured text. University of Oulu, Department of Computer Science and Engineering, 64 p.

ABSTRACT

The coverage of the lexicon has great implications on the performance of a number of natural language processing algorithms. Insufficient vocabulary complicates the application of these algorithms on problems that involve, for example, minority languages. Producing and extending lexicons with traditional means is both slow and expensive so there is a need to develop automatic, generic purpose, language independent lexicon acquisition systems.

Automatic lexicon acquisition systems exist in small numbers, but there is room to improve their flexibility, quality and applicability to multiple languages. Advanced lexicon acquisition systems can be developed by applying the state-of-the-art methods from word segmentation, morphology induction, and part-of-speech induction to the problem. Many of the most interesting methods are based on the Bayesian approach.

In this Master's thesis, a mostly-unsupervised, multilingual lexicon acquisition system was developed. It accepted as an input an unannotated corpus and generated a list of words and their part-of-speech tags. All the important parts of the system relied on non-parametric Bayesian methods: word segmentation and morphology induction used nested Pitman-Yor processes and part-of-speech induction used a Pitman-Yor process mixture model.

The implementation achieved 16% precision in the acquisition task of base forms of Finnish words without part-of-speech tags. With finnish base form and part-of-speech tag combinations, the precision was 3%. The results were estimated to be of the same quality in English and Japanese.

The individual parts of the system had poor performance compared to the state-of-the-art. Especially morfology and part-of-speech induction could be improved significantly. Better results could also be improved by integrating the parts of the system more deeply with each other.

Keywords: natural language processing, lexicon acquisition, morphology induction, word segmentation, part-of-speech induction, Bayesian method

SISÄLLYSLUETTELO

ALKULAUSE

Haluan kiittää tämän diplomityön valvojaa, Mika Rautiaista, hänen luonnollisen kielen käsittelyyn liittyvästä asiantuntemuksestaan. Hänen alkuvaiheessa tarjoamillaan neuvoilla ja opastuksella sekä loppuvaihessa antamallaan palautteella oli suuri positiivinen vaikutus työn onnistumisen ja laadukkuuden kannalta.

Lisäksi haluan kiittää Smartifik Oy:tä ja sen henkilökuntaa kokonaisuudessaan erinomaisten puitteiden tarjoamisesta tämän työn tekemiselle.

Oulussa 4.2.2014

Ari Mustonen

LYHENTEIDEN JA MERKKIEN SELITYKSET

LKK	Luonnollisen kielen käsittely
LSA	Latentti semanttinen analyysi
MKP	Minimaalinen kuvauksen pituus
MH	Metropolis-Hastings
DP	Dirichlettiprosessi
PYP	Pitman-Yor-prosessi
HPYP	Hierarkkinen Pitman-Yor-prosessi
MI	Morfologian induktio
SLI	Sanaluokan induktio
TVK	Tukivektorikone
ESK	Ehdollinen satunnaiskenttä
MPM	Markovin piilomalli

1. JOHDANTO

Ihmisten välinen viestintä ja keräämä tieto on suurelta osin esitetty jollakin meille kaikille tutuista ja helposti lähestyttävistä luonnollisista kielistä. Esimerkiksi kirjat, lehdet, verkkosivut, sähköpostit, pikaviestit ja sosiaalinen media koostuvat valtaosin luonnollisen kielen tekstistä. Koska tällaista tekstiä on valtaisasti ja sitä hyödynnetään kauttaaltaan niin tieteessä, liike-elämässä kuin yksityiskäytössäkin, on sen tietokoneavusteinen käsittely, LKK (luonnollisen kielen käsittely), jo pitkään ollut suuren kiinnostuksen kohde. LKK-menetelmiä pyritään kehittämään entistä yleiskäyttöisemmiksi ja tarkemmiksi, jotta niitä pystyttäisiin soveltamaan uusiin ja entistä haastavampiin ongelmiin.

Sanastotiedolla on merkittävä rooli LKK:ssa, sillä monien LKK-menetelmien tarkkuus riippuu niiden käyttämän sanaston kattavuudesta [1]. Kun lauseita esimerkiksi jäsennetään niin, että niille etsitään pääverbi sekä siihen liittyvät lauseenjäsenet, lauseessa kohdatut tuntemattomat sanat kasvattavat virheellisen jäsennyksen riskiä [1, 2], sillä jäsennysalgoritmin käytettävissä olevan tiedon määrä on pienempi kuin silloin, kun kaikki sanat ovat tunnettuja. Jos tehtävänä on ratkaista lauseen sanojen perusmuodot, tehtävä helpottuu, kun tiedetään hyvin kattavasti, mitä sanoja on olemassa ja miten ne taipuvat. Jos tehtävänä on tunnistaa tekstistä erisnimiä, voidaan tehtävää helpottaa hyödyntämällä kattavaa listaa tunnetuista nimistä. Jotta parhaita tuloksia voitaisiin saavuttaa näiden kaltaisissa tehtävissä, tulisi ratkaisumentelmien käytettävissä olevan sanaston olla mahdollisimman kattava. Tämän vuoksi kattavien sanastojen tuottaminen hyödyttää LKK-sovelluksia.

Vaikka sanastoja on tuotettu jo pitkään, niiden laajuus ei vieläkään ole riittävä kaikille käytännön sovelluksille [3]. Esimerkiksi kattavaa alakohtaista tai vähemmistökielille suunnattua sanastoa ei ole aina helposti saatavilla, ja niiden tuottaminen sekä ylläpito asiantuntijavoimin on vaivalloista ja usein ekonomisesti kannattamatonta. Lisäksi olemassa olevien sanastojen lisenssiehdot voivat vaikeuttaa niiden yhdistämistä tai hyödyntämistä tietyissä ympäristöissä. Myös kieli muuttuu ajan myötä, sillä esimerkiksi uusia teknologia-alan sanoja kehittyy nopeaa tahtia uusien keksintöjen ja tuotteiden julkistusten myötä. Siksi sanastoja pitäisi myös päivittää jatkuvasti, jos niiden halutaan pysyvän kattavina. Koska sanalistojen kirjoittaminen ja päivittäminen käsin on epäkäytännöllistä, on sanaston poiminta olemassa olevista tekstiaineistoista joko osittain tai kokonaan automaattisesti huokutteleva ajatus LKK-sovellusten kehitystä ajatellen.

LKK-järjestelmien sanastotiedon poimintaan on kehitetty useita erilaisia ohjattuja menetelmiä, kuten esimerkiksi valmiiksi merkattujen korpusten pohjalta sanastoa oppivia menetelmiä, mutta kaikkia edellä mainittuja ongelmia ei ole niilläkään pystytty ratkaisemaan. Esimerkiksi merkattuihin korpuksiin pohjautuvien menetelmien ongelmia ovat, että vain noin 20 tai 30:lle kielelle kaikista noin 6000:sta maailman kielestä on kerätty kattava käsin merkattu korpus [4], kattavat korpuksetkaan eivät välttämättä sisällä niin paljon tietoa kuin olisi toivottua [5], ja uusien korpusten tuottamien on hyvin vaivalloista ja kallista [5]. Jotta sanastoja voitaisiin käytännössä tuottaa laajalle kielivalikoimalle ja laajalla kattavuudella, pitäisi tehtävää automatisoida vieläkin enemmän.

Sanaston poimintaan on jo kehitetty joitakin automaattisia menetelmiä. Toisin kuin ohjatut menetelmät, ne eivät vaadi opetusta asiantuntijoilta, vaan ne oppivat poimimaan sanat automaattisesti merkkaamattomasta korpuksesta analysoimalla esimerkiksi sen tilastollisia ominaisuuksia. Näillä menetelmillä on saatu lupaavia tuloksia, mutta ne

eivät vielä täysin vastaa esimerkiksi kaikkiin monikielisyyden asettamiin haasteisiin. Jos näiden menetelmien jäljelle jäävät ongelmat voitaisiin ratkaista, LKK-menetelmien vaatima sanastotieto saataisiin kerättyä helposti ja edullisesti riippumatta käytetystä kielestä tai muista tekstin erityispiirteistä. Tämä antaisi edelleen paremmat edellytykset LKK-menetelmien käytön nopeammalle laajentamiselle suurempaa tarkkuutta vaativiin, erityissanastoa käsitteleviin ja vähän palveltuja kieliryhmiä koskeviin ongelmiin.

Käytännöllisen sanastotietoa poimivan menetelmän pitäisi pystyä keräämään hyödyllistä ja johdonmukaisessa muodossa esitettyä tietoa korpuksessa esiintyvistä sanoista. Monien yleisesti käytettyjen LKK-sovellusten kannalta hyödyllistä tietoa ovat sanojen perusmuodot ja sanaluokat. Näiden kerääminen olisi täten yksi tehtävä, joka pitäisi pystyä ratkaisemaan. Menetelmän pitäisi myös vaatia mahdollisimman vähän asiantuntijoiden opetusta tai avustusta. Ideaalisesti, sen tulisi toimia täysin automaattisesti ja ilman mitään esitietoa, mutta käytännössä jo pääosin ohjaamaton menetelmäkin helpottaisi sanaston poimintaa merkittävästi. Ideaalisen menetelmän pitäisi myös toimia usealla kielellä tai olla ainakin helposti laajennettavissa tukemaan niitä.

Tässä diplomityössä keskityttiin edellä kuvatun kaltaisten, usealla kielellä toimivien, vähän asiantuntijoiden avustusta vaativien sanastoa poimivien menetelmien tarkasteluun ja kehittämiseen. Työssä rajoituttiin kehittämään järjestelmää, joka ottaa syötteenä merkkaamattoman korpuksen, josta se johtaa sanalistan, jossa sanat ovat perusmuodossa ja merkattu sanaluokkatiedolla. Koska ohjaamattoman monikielisen sanaston poimijan toteuttaminen on erittäin vaativa tehtävä, jota ei realistisesti pystytä ratkaisemaan diplomityölle tarkoitetussa ajassa, tyydyttiin tässä työssä rakentamaan peruspuitteet tällaiselle järjestelmälle. Tulosten parantaminen korkealaatuiselle tasolle jätettiin tulevalle kehitykselle.

Työn käytännön osassa toteutettiin sanaston poimija C++- ja Python-kielillä. Toteutuksessa hyödynnettiin Boost-kirjastoa ja sen todennäköisyysmoduulia sekä Pythonin peruskirjastoja. Muilta osin ratkaisu koostui kokonaan uudesta koodista. Algoritmitasolla ratkaisu hyödynsi jo olemassa olevia ei-parametrisia Bayesin menetelmiä. Täysin uusia algoritmejä ei esitetty, mutta ratkaisussa käytettyjen menetelmien keskinäisen yhteistyön merkitystä sanaston poiminnan toimivuuden kannalta pohdittiin ja testattiin alustavasti.

Työn rakenne on seuraavanlainen: Luvussa 2 esitellään työhön läheisesti liittyvät LKK:n perusteet ja perusmenetelmät. Luvussa 3 tehdään katsaus uusimpiin olemassa oleviin sanaston poiminnan ratkaisuihin. Luvussa 4 tarkastellaan sanaston poimijoiden hyödyntämiä yksittäisiä menetelmiä. Luvussa 5 esitellään arviointimenetelmät, joilla sanaston poiminnan ja sen menetelmien suorituskykyä voidaan tarkastella. Luvussa 6 esitellään taustatietojen pohjalta kehitelty sovellus ja kaikki sen olennaiset toimintaperiaatteet. Luvussa 7 verrataan toteutetun sovelluksen suorituskykyä uusimpiin vastaaviin tekniikoihin sekä olemassa oleviin sanastoihin. Luvussa 8 pohditaan mahdollisia parannuksia ja tulevia kehityssuuntia sovellukseen ja ongelmaan liittyen. Luvussa 9 tehdään yhteenveto koko työstä.

2. LUONNOLLISEN KIELEN KÄSITTELYN PERUSMENETELMÄT

Lähtökohtaisesti sanaston kerääminen on yksi LKK-menetelmien sovellus, ja täten sillä on vahva teoreettinen kytkös luonnollisen kielen tutkimukseen. Luonnollisen kielen käytännön toiminnan ymmärtäminen on erityisen tärkeää monikieliselle sanaston poimijalle, sillä kielispesifisten ad-hoc-ratkaisujen käyttö ei ole enää käytännöllistä. Tämän vahvan riippuvuuden vuoksi on luonnollista lähteä tarkastelemaan ongelmaa luonnollisen kielen toiminnan kannalta.

Luonnollinen kieli esitetään tietokoneelle sopivassa muodossa kielimallien avulla. Kielimalleja on kehitetty lukuisia, ja monet niistä poikkeavat toisistaan hyvinkin paljon eri ongelmiin, kuten monikielisyyteen, sovellettavuuden suhteen. Kielimallin valinta vaikuttaa myös siihen, millaisia menetelmiä toteutuksessa voidaan käyttää, joten eri kielimallit on hyödyllistä tuntea hyvin.

Bayesin menetelmä on yksi oppivien LKK-sovellusten kielimalleissa hyödynnetty menetelmä, jota sovelletaan myös tämän työn ratkaisussa sen ohjaamattomiin ongelmiin soveltuvuuden vuoksi. Bayesin menetelmä on todennäköisyysteoriaan perustuva ratkaisu, joka tekee päätelmiä ilmiöiden ennakoitujen todennäköisyyksien ja tehtyjen havaintojen todennäköisyyksien perusteella.

Bayesin menetelmistä tässä työssä keskitytään vielä edelleen niiden ei-parametriseen luokkaan. Ei-parametristen Bayesin menetelmien erityispiirre on, että ne eivät tee oletuksia järjestelmän parametreista, vaan ne johdetaan automaattisesti. Ei-parametriset Bayesin menetelmät ovat saavuttaneet laajaa suosiota uusimmissa ohjaamattomissa LKK-menetelmissä niiden joustavuuden ja antamien hyvien tulosten vuoksi.

2.1. Luonnollinen kieli

Universaalia ja yksikäsitteistä listaa kielissä esiintyvistä rakenteista ja ominaisuuksista ei ole onnistuttu kehittämään. Monien kielten toiminta on hyvin spesifistä. Useasti kielten piirteet ovat sellaisia, ettei niille ole tarkkoja vastineita muissa kielissä, ja vastineiden ollessa olemassa niissä voi silti olla joitakin eroja, joiden vuoksi niitä ei voida suoraan yhdistää yhdeksi universaaliksi ominaisuudeksi. [6]

Vaikka yksiselitteistä universaalia mallia kielistä ei ole löydetty, niiden piirteiden yleisyyttä ja luokittelua yleensä on tutkittu laajasti lingvistisen typologian alalla. Tunnettujen kielten piirteiden, niiden luokittelun ja luokittelun selitysten pohjalta on esitetty sääntöjä, niin sanottuja kielellisiä universaaleja, jotka pätevät pääosalle kielistä ja asettavat rajoitteita kielissä esiintyville muunnelmille. [7, s. 3-8]

Kuten seuraavissa osioissa tullaan huomaamaan, on perusteltua olettaa, että kaikki kirjoitettu luonnollinen kieli on segmentoitavissa sanoiksi, ja sanat ovat edelleen segmentoitavissa yhteen tai useampaan morfeemiin kaikilla kielillä. Lisäksi sanat voidaan luokitella sanaluokkiin niiden käyttäytymisen perusteella.

2.1.1. Sanat

Käytännössä järkevänä oletuksena voidaan pitää sitä, että luonnollisen kielen teksti koostuu sanoista. Vaikka joidenkin kielten, kuten kiinan, kirjoitusjärjestelmät eivät

suoranaisesti erottele sanoja toisistaan, on sanan käsite siltikin olemassa. [8]

Sanaa pidetäänkin yhtenä universaalina luonnollisen kielen elementtinä. Kieliopillisesti sanan määritelmä on, että se koostuu yhdestä tai useammasta elementistä. Kaikki yhden sanan elementit esiintyvät samassa kohtaa lausetta, niillä on määrätty esiintymisjärjestys ja selkeä merkitys. [9, s. 1-2]

Sanoja voidaan käsitellä suppeasti ortografisina yksikköinä, eli yksittäisinä sanoina, tai laajemmin leksikaalisina yksiköinä, jotka voivat koostua useista ortografisista yksiköistä. Esimerkiksi suomen kielen ilmaus "suuna päänä" yhdistää kaksi ortografista yksikköä yhdeksi leksikaaliseksi yksiköksi, jolla on kokonaan uusi merkitys. [9, s. 2]

Ortografinen yksikkö voidaan edelleen jakaa pienempiin osiin. Yksi esimerkki näistä elementeistä ovat yhdyssanan osat. Yhdyssanoja esiintyy kaikentyyppisissä kielissä, joskin tietyn tyyppisissä kielissä harvemmin kuin toisissa. [9, s. 24-35]

Toinen ortografisen yksikön osa on morfeemi, joihin lukeutuvat esimerkiksi kokonaiset sanat, sanan vartalo, etuliitteet ja päätteet. [9, s. 3-9]

2.1.2. Morfologia

Morfologia tutkii sitä, miten sanat jakautuvat osiin eli morfeemeihin ja miten nämä osat käyttäytyvät [10, s. 5]. Morfologiat voidaan luokitella morfeemien käyttäytymisen perusteella kolmeen eri pääluokkaan: isolatiiviseen, agglutinatiiviseen ja fuusionaaliseen morfologiaan. [10, s. 5]

Isolatiivisissa kielissä, kuten vietnamissa, jokainen sana koostuu vain yhdestä morfeemista. Agglutinatiivisissa kielissä sanat voivat koostua useista erillisistä morfeemeista, jotka ovat selkeästi erotettavissa toisistaan. Fuusionaalisissa kielissä morfeemit voivat yhdistyä yksiköiksi, joista niitä ei voi enää selkeästi erottaa. [9, s. 3-9]

Agglutinatiivisissa ja fuusionaalisissa kielissä morfologia lisää sanoihin joko etu- tai jälkiliitteitä tai molempia. Liitteiden yhdistäminen voi tapahtua monella eri tavalla, joista ainakin kahdeksan tunnetaan. [7, s. 120-155].

Yleensä morfeemeilla merkitään sanojen semanttisia eli merkityksellisiä rooleja, mutta tämä ei pidä aina paikkaansa. Joillakin morfeemeilla ei välttämättä ole ollenkaan semanttista roolia, ja joillakin se voi vaihdella riippuen kontekstista. [10, s. 5]

Morfologian käyttö voi määrällisesti vaihdella huomattavan paljon saman tyyppistä morfologiaakin käyttävien kielten välillä. Esimerkiksi englannissa morfologia on varsin rajoittunutta [11] verrattuna japaniin ja suomeen, vaikka näiden kaikkien morfologia on agglutinatiivista.

2.1.3. Sanaluokat

Sanat on perinteisesti luokiteltu sanaluokkiin. Tämä on perusteltu valinta siinä mielessä, että niillä on tärkeä rooli kielioppisääntöjen esittämisen kannalta. [12, s. 109]

Sanaluokkien olemassaoloa yleisesti voidaan pitää mahdollisesti kielellisenä universaalina, mutta selkeitä kaikille kielille päteviä luokittelukriteerejä ei ole olemassa. Yksittäisenkin kielen kohdalla luokittelu voi perustua monien eri morfologisiin, lauseopillisiin ja semanttisiin piirteisiin, joita ei ole helppoa yleistää useille kielille. [12, s. 109-113]

Sanojen luokittelun vaikeus huomataan helposti jo tässä työssä käsiteltävää kolmea kieltä tarkastelemalla, sillä näillä kielillä ei ole selkeitä yhteisiä sanaluokkia, ja se, miten sanat luokitellaan kullekin kielelle spesifisiin sanaluokkiin vaihtelee huomattavasti.

Suomessa sanat on perinteisesti luokiteltu morfologisen käyttäytymisen perusteella nomineihin, verbeihin sekä taipumattomiin sanoihin. Nominit jakautuvat edelleen alaluokkiin, jotka koostuvat substantiiveista, adjektiiveista, pronomineista ja numeraaleista. Taipumattomien sanojen alaluokat taaskin ovat adverbit, adpositiot ja partikkelit. [13, s. 264]

Englannin perinteiset sanaluokat taaskin ovat substantiivi, adjektiivi, verbi, adverbi ja prepositio. [11, s. 29-60] Erona suomeen on esimerkiksi se, että adjektiivit ja substantiivit eivät jaa keskenään taivutusmuotoja, joten niitä ei voida yhdistää yhdeksi nominin luokaksi. Lisäksi suomessa ei ole esimerkiksi prepositioita ollenkaan.

Japanissa taaskin voidaan katsoa olevan viisi sanaluokkaa: verbi, adjektiivi, kopula, substantiivi ja partikkelit. [14, s. 316-355] Näiden sanaluokkien erikoisuuksia ovat muun muassa se, että adjektiivit taipuvat aikamuodoissa ja substantiivit eivät taivu ollenkaan. Nämä piirteet eroavat merkittävästi suomesta ja englannista. Lisäksi japanin uniikille kopula-sanaluokalle on vaikea löytää vastinetta mistään muustakaan kielestä.

Näiden tietojen pohjalta on perusteltua olettaa, että sanaluokat ovat olemassa, mutta niiden lukumäärä ja luokittelukriteerit ovat kielispesifiset. Universaali sanaluokkatietoa käsittelevä järjestelmä voi siis tehdä hyvin vähän oletuksia niistä.

2.2. Kielen mallintaminen

Jotta luonnollisen kielen tekstiä pystyttäisiin hyödyntämään koneellisesti, se on esitettävä tietokoneelle sopivassa muodossa. Kielen mallintaminen on sekä haastava että erittäin tärkeä ongelma LKK-sovelluksissa, minkä vuoksi on kehitetty useita erilaisia kielimalleja.

Sopivan kielimallin valinta on erityisen tärkeää yleiskäyttöisissä ja kieliriippumattomissa sovellutuksissa, sillä mallin rajoitukset voivat heikentää ratkaisun universaalisuutta ja tulosten laatua. Tämän vuoksi on syytä käydä läpi tämän työn kannalta olennaisimmat kielimallit ja pohtia sitä, kuinka hyvin ne soveltuvat kieliriippumattomaan sanaston oppimiseen.

Merkkijonometriikat ovat ortografiaan eli kirjoitusasuun perustuvia menetelmiä, joilla mitataan merkkijonojen samankaltaisuutta. Niitä voidaan hyödyntää esimerkiksi sanojen samankaltaisuuden arvioinnissa merkkijonotasolla, minkä vuoksi niillä on sovellutuksia kielimalleissa.

N-grammit ovat vakiopituisia sekvenssejä, jotka on kerätty näytteistämällä N kappaletta peräkkäisiä sanoja, merkkejä tai muita elementtejä. Niitä voidaan käyttää merkkijonojen tai sanasekvenssien käyttäytymisen estimointiin. Paikallinen n-grammi on n-grammin laajennus, joka huomioi elementtien sisällön lisäksi niiden paikat näytteistyssekvenssissä.

LSA:lla (latentti semanttinen analyysi) voidaan estimoida tekstien piirteitä kun tiedetään, mitä sanoja niissä esiintyy, tai toisaalta yksittäisten sanojen piirteitä sen perusteella, mitä muita sanoja niiden lähiympäristössä esiintyy.

MKP (minimaalinen kuvauksen pituus) on ohjaamattomasti opetettava kielimalli, joka etsii sille annetusta syötteestä säännöllisyyksiä ratkaisemalla sille mahdollisimman tiiviin esitystavan. Suorien sovelluskohteiden lisäksi MKP:n periaatetta pienestä esitystavasta hyödynnetään laajasti esimerkiksi muissa kielimalleissa, sillä ne ovat osoittautuneet tehokkaiksi ja intuitiivisiksi.

Probabilistiset mallit kuvaavat kieltä sen tilastollisten ominaisuuksien avulla. Tyypillisesti probabilistisia mallien tietoja käsitellään tilastollisilla induktiomenetelmillä, kuten esimerkiksi Bayesin menetelmällä.

2.2.1. Merkkijonometriikat

Monet ohjaamatonta oppimista hyödyntävät LKK-sovellukset hyödyntävät niin sanottua ortografista tietoa, eli käytännössä ne arvioivat, kuinka samanlaisia sanat ovat merkkijonotasolla. Tällainen merkkijonotason vertaaminen voidaan toteuttaa sitä varten kehitetyillä merkkijonometriikoilla.

Merkkijonometriikkojen keskeinen käsite on muokkausetäisyys, joka ilmoittaa, kuinka monen muokkauksen päässä merkkijonot ovat toisistaan. Nolla muokkausta tarkoittaa, että merkkijonot ovat samat, ja tätä suuremmat luvut kertovat merkkijonojen erilaisuuden kasvusta.

Tunnetuimpia merkkijonometriikoita ovat Hamming- ja Levenshtein-etäisyys.

2.2.1.1. Hamming-etäisyys

Hamming-etäisyys mittaa kahden samanpituisen sekvenssin eroa sen perusteella, kuinka monta niiden samoissa kohdissa olevista symboleista on samoja [15]. Esimerkiksi sanojen "villasukka" ja "villakoira" Hamming-etäisyys on 4, sillä osat "sukk" ja "koir" sisältävät 4 toisistaan eroavaa merkkiä.

Hamming-etäisyys voidaan laskea nopeasti O(N) ajassa. Se ei kuitenkaan arvioi erityisen hyvin luonnollisen kielen merkkijonoille tyypillisiä poikkeamia, sillä se mittaa pelkästään merkkien vaihtoja toisiin. Esimerkiksi kirjoitusvirheille tyypillisiä merkkien lisäyksiä ja poistoja ei tueta ollenkaan, sillä pituus ei saa muuttua.

Hamming-etäisyys on kuitenkin hyödyllinen referenssinä muille merkkijonometriikoille, jotka käytännössä noudattavat usein samoja periaatteita. Esimerkiksi Levenshtein-etäisyys voidaan ajatella Hamming-etäisyyden laajennukseksi.

2.2.1.2. Levenshtein-etäisyys

Levenshtein-etäisyys on Hamming-etäisyyden laajennus, jonka muokkaukset ovat merkin korvaaminen, lisääminen ja poistaminen. Se on Damerau-Levenshtein-etäisyyden [16, 17] yksinkertaistus, josta on poistettu muokkaus kahden vierekkäisen merkin paikkojen vaihtamiselle keskenään.

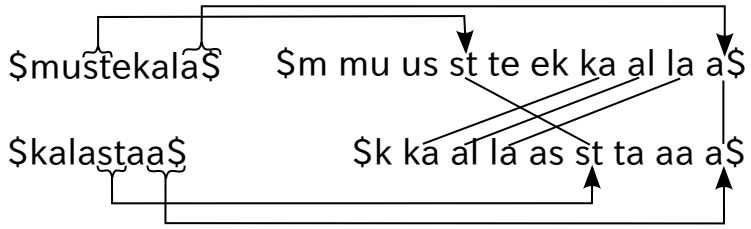
Levenshtein-etäisyyttä on käytetty LKK-sovelluksissa, sillä sen käyttämä metriikka on intuitiivinen ja se pystyy esittämään käytännössä riittävällä tarkkuudella esimerkiksi yleisimmät kirjoitusvirheet. Myös Damerau-Levenshtein-etäisyyden vierekkäisten merkkien vaihto voidaan mallintaa kahdella Levenshtein-etäisyyden sallimalla muokkauksella.

2.2.2. N-grammit

Merkkijonotason n-grammit ovat vakiopituisia alimerkkijonoja, joiden merkit on poimittu alkuperäisestä merkkijonosta liu'uttamalla n merkin pituista ikkunaa sen yli. [18, 19]. Niitä voidaan käyttää esimerkiksi sanojen ortografiapohjaiseen vertailuun tai indeksointiin.

Yksinkertainen tapa verrata kahta merkkijonoa n-grammien avulla on laskea, kuinka monta yhteistä n-grammia niillä on, sillä samanlaisilla merkkijonoilla on tyypillisesti yhteisiä n-grammeja. Yhteisten n-grammien lukumäärästä voidaan päätellä pienin muokkausetäisyys, joka sanoilla voi olla. [18]

N-grammien poimimista merkkijonoista ja sanojen vertaaminen niiden n-grammien avulla on havainnollistettu kuvassa 2.1.



Kuva 2.1: Esimerkki n-grammien poimimisesta ja sanojen vertailusta niiden avulla. Sanojen bigrammit on poimittu liu'uttamalla kahden merkin mittaista ikkunaa niiden ylitse. Tämän jälkeen on etsitty sanojen yhteiset n-grammit, joita löytyi yhteensä viisi. Suhteutettuna sanojen pituuteen, tämän perusteella voidaan arvioida sanojen muistuttavan toisiaan osittain.

Käytännössä n-grammien etuna on, että ne ovat yksinkertaisia toteuttaa ja niillä voidaan löytää sanoja silloinkin, kun muokkausetäisyys on suuri. Toisaalta n-grammien ongelmana on, että sovelluksen tarpeisiin nähden liiankin erilaiset sanat saatetaan tulkita samanlaisiksi etenkin sen vuoksi, että n-grammin paikkaa merkkijonossa ei huomioida ollenkaan.

Merkkitason esityksen lisäksi n-grammeja käytetään yleisesti myös sanatasolla, jolloin tarkastellaan merkkien sekvenssien sijaan sanojen sekvenssejä. Tätä esitystapaa itsessään voidaan käyttää kielimallina, sillä sanatason n-grammien yleisyyden avulla voidaan arvioida esimerkiksi sitä, esiintyvätkö kaksi sanaa yleensä vierekkäin.

2.2.3. Latentti semanttinen analyysi

LSA [20] on menetelmä, jolla voidaan analysoida, millaisissa konteksteissa yksittäiset sanat esiintyvät. Tätä tietoa voidaan hyödyntää esimerkiksi sanojen vertailussa toisiinsa tai pidempien tekstien luokittelussa. Yksi tyypillisimmistä sovelluksista LSA:lle on asiakirjojen luokittelu eri aihealueisiin sen mukaan, mitä sanoja ne sisältävät. [21]

LSA:n toiminta pohjautuu siihen, että luokilteltavista teksteistä kerätään sanojen esiintymistiheydet tai entropialla painotetut esiintymistiheydet, ja ne esitetään suorakulmaisena matriisina. Tämän matriisin dimensionaalisuutta pienennetään matemaattisesti singulaariarvohajotelman avulla, ja se palautetaan alkuperäistä matriisia vastaavaan muotoon lineaarikombinaatioiden avulla. Näin saadun matriisin informaatiosisältö on pienentynyt, mutta se approksimoi silti alkuperäistä matriisia. Käytännössä matriisissa olleet sanatiheyden ovat kasvaneet ja pienentyneet sen mukaan, onko sama sana esiintynyt muissa matriisin sarakkeissa. [20, 21]

Lineaarialgebrallisen esityksen lisäksi LSA:sta on kehitetty probabilistisia muunnelmia [22, 23], joten sitä voidaan hyödyntää muiden probabilististen menetelmien puitteissa.

LSA on siinä mielessä universaali menetelmä, että se ei tee mitään oletuksia esimerkiksi sanajärjestyksestä, morfologiasta tai muista kieliopillisista seikoista. Sen sijaan se käyttää syötteenä raakaa sanoihin pilkottua tekstiä. [20]

2.2.4. Minimaalinen kuvauksen pituus

MKP kuuluu ohjaamattomasti opetettavien kielimallien ryhmään. Sen yksinkertaistettu toimintaperiaate on löytää annetulle korpukselle sellainen koodikirjasta ja siihen viittaavista indekseistä koostuva esitys, jonka koko on minimaalinen [24][25, s. 375-378].

MKP voidaan itse asiassa lukea myös probabilististen mallien luokkaan, sillä tilastollisen kompressiokriteerin hyödyntäminen on yleistä käytännössä. Lausekkeessa (2.1) on annettu esimerkki siitä, miten tilastollinen kriteeri $P(.)$ voidaan sisällyttää MKP:n kaavaan. [26][27, s. 21]

$$\begin{aligned} MKP(C) &= \arg\min_{M} \left[\text{len}(M) + \text{len}(\text{encoding}_{\text{M}}(C))\right] \\ &= \arg\min_{M} \left[\text{len}(M) + \log_2 \frac{1}{P(C|M)}\right] \\ C &= \text{Korpus.} \\ M &= \text{Kuvaus.} \\ P(C|M) &= \text{Korpuksen todennäköisyys kuvauksen perusteella.} \end{aligned} \tag{2.1}$$

MKP:ta on sovellettu esimerkiksi sanojen erottelussa, tekstien segmentoinnissa [28] ja MI:ssä. Lisäksi sillä on periaatteellisia yhtäläisyyksiä muihin menetelmiin, kuten esimerkiksi Bayesin menetelmään perustuvaan tekstin segmentointiin.

2.2.5. Probabilistiset mallit

Probabilistisissa eli tilastollisissa kielimalleissa kieltä mallinnetaan sen tilastollisten ominaisuuksien, kuten esimerkiksi merkkien, sanojen tai sanaparien esiintymistodennäköisyyksien perusteella. Näiden todennäköisyyksien perusteella pyritään sitten tekemään päätelmiä liittyen esimerkiksi jonkin virkkeen tulkintaan tai kielen piirteisiin yleensä.

Tyypillinen probabilististen kielimallien käytännön esiintymismuoto ovat Monte-Carlo-algoritmit eli satunnaisnäytteistävät algoritmit. Niiden perusperiaate on, että kun probabilistisesta mallista otetaan riittävän paljon satunnaisia näytteitä, näiden näytteiden jakaumaa tarkastelemalla voidaan arvioida eri hypoteesien todennäköisyyttä. Jos hypoteesia arvioiva todennäköisyysmalli on oikea tai sitä muokataan oikeaan suuntaan näytteitä otettaessa, malli konvergoituu kohti ratkaisua.

Probabilististen kielimallien ja Monte-Carlo-algoritmien yhdistelmä on saavuttanut laajaa suosiota ongelmissa, joissa tiedon epäsäännöllisyydet tai sääntöjen monimutkaisuus tekevät muiden algoritmien ohjelmoinnista haastavaa. Universaali kielen mallintaminen on juuri tällainen ongelma, sillä kielten säännöt ovat usein monimutkaisia ja sisältävät paljon erilaisia poikkeustapauksia. Tämän vuoksi näitä menetelmiä on sovellettu myös uusimmissa sanaston poimintaan liittyvissä ratkaisuissa, kuten tullaan myöhemiin tässä työssä huomaamaan.

Kieliriippumattomassa ja ohjaamattomassa kielen mallintamisessa erityistä suosiota ovat saavuttaneet Bayesin menetelmään perustuvat kielimallit, jotka kuuluvat juuri probabilististen Monte-Carlo-menetelmien luokkaan. Koska Bayesin menetelmän merkitys tähän työhön liittyen on huomattava, se ja siihen liittyvien Monte-Carlo-

algoritmien toiminta tullaan esittelemään myöhemmin tässä työssä sekä niitä hyödyntävien käytännön sovellusten kautta että tarkemmin teoreettiselta kannalta.

2.3. Bayesin menetelmä

Bayesin menetelmä on todennäköisyysteoriaan perustuva tapa arvioida havaintojen perusteella sitä, kuinka todennäköinen jokin ilmiö on.

Bayesin menetelmällä on havaittu olevan monia samoja piirteitä kuin ihmisten kognitiivisella järjestelmällä. Samoin kuin ihminen, Bayesin menetelmä pystyy tekemään päätelmiä rajoittuneen ja kohinaisen tiedon pohjalta, ja se pystyy inkrementaalisesti korjaamaan aikaisemmin oppimaansa tietoa uuden syötteen valossa. Muun muassa näiden piirteiden ansioista sitä on sovellettu lupaavin tuloksin moniin eri ongelmiin, kuten visuaalisten näkymien tunnistukseen, semanttisiin muisteihin sekä luonnollisen kielen prosessointiin ja oppimiseen. [29, s. 1-4]

Bayesin menetelmä perustuu Bayesin induktioon eli päättelyyn. Bayesin induktio arvioi ilmiön todennäköisyyttä tilastopohjaisesti Bayesin säännön avulla, suosien hypoteesejä, jotka ovat todennäköisiä sekä havaintojen että ennakko-odotusten perusteella.

Ongelmat, joissa Bayesin menetelmää tyypillisesti käytetään ovat harvoin analyyttisesti ratkaistavissa. Tämän vuoksi ne ratkaistaan yleensä Monte-Carlo eli satunnaisnäytteistävillä menetelmillä, kuten MH (Metropolis-Hastings) -näytteistys tai Gibbsin näytteistys.

Bayesin menetelmiä ratkaistavia ongelmia mallinnetaan usein generoivilla malleilla. Ne kuvaavat luonnossa esiintyvien prosessien käyttäytymistä tilastojakaumien ja probabilististen prosessien avulla.

2.3.1. Bayesin induktio

Bayesin induktion perustana toimii Bayesin sääntö, joka on annettu lausekkeessa (2.2) [30].

$$P(a|b) = \frac{P(b|a)P(b)}{P(a)} \tag{2.2}$$

Koneoppimisessa Bayesin säännöllä voidaan estimoida havaitun datan perusteella sitä, millainen prosessi kyseisen datan tuotti. Perusajatuksena on, että datalle selitystä etsivä agentti valitsee jonkin hypoteesin sille, millainen dataa generoiva prosessi on, ja estimoi Bayesin lauseella sitä, kuinka todennäköisesti hypoteesi oli oikea. Lausekkeessa (2.3) on annettu Bayesin lause ja sen jäsenten selitykset, kun sitä sovelletaan koneoppimiseen. [29, s. 5-7]

$$P(h|d) = \frac{P(d|h)P(h)}{P(d)} \tag{2.3}$$

$P(h|d) =$ Hypoteesin uskottavuus havaintojen perusteella

$P(d|h) =$ Havainnon todennäköisyys hypoteesin perusteella

$P(h) =$ Hypoteesin priori uskottavuus

$P(d) =$ Havainnon todennäköisyys

Selkokielisesti ilmaistuna, Bayesin mallin mukainen oppija suosii sellaisia hypoteesejä, joiden mukaan tehty havainto on todennäköinen ja hypotetisoitu prosessi itsessään on todennäköinen. Tämä seuraa siitä, että hypoteesin uskottavuus on suuri vain silloin, jos molemmat jakoviivan yläpuolella olevista todennäköisyyksistä ovat suuria. Jos havainto on epätodennäköinen hypoteesin perusteella, $P(d|h)$ on pieni, ja jos hypotetisoitu prosessi itsessään on epätodennököinen, $P(h)$ on pieni. [29, s. 7]

Kielimallina Bayesin lausetta voidaan soveltaa esimerkiksi niin, että havainnot koostuvat yksittäisistä merkeistä tai sanoista, ja hypoteesinä toimii jokin sovelluksen kannalta perusteltu kuvaus, kuten MKP, jonka perusteella $P(h)$:ta voidaan arvioida. Data ajetaan Bayesin induktion lävitse tyypillisesti useita kertoja, minkä seurauksena Bayesin induktio löytää iteratiivisesti sellaiset parametrit, joilla sovelluksen käyttämä kuvaus kielestä selittää datan mahdollisimman hyvin.

Bayesin kielimallin yksi merkittävä vahvuus on, että priorin tiedon tyypille tai määrälle ei aseteta rajoituksia. Bayesin mallia voidaankin soveltaa hyödyntämään niin monimutkaista kielihypoteesejä kuin MKP:n kaltaisia yksinkertaisia periaatteita. [31] Myös asiantuntijoiden tuottamaa tietoa voidaan käyttää priorina, joten Bayesin kielimalli voidaan myös muokata osittain ohjatuksi, jos tämä on sovelluksen kannalta perusteltua.

2.3.2. Induktio näytteistämällä

Valitettavasti Bayesin induktiolle on harvoin olemassa analyyttistä ratkaisua. Tämän vuoksi Bayesin induktio ratkaistaan käytännössä satunnaisnäytteistävillä eli markovinketju Monte-Carlo -algoritmeilla. Tällaisia algoritmeja ovat MH-näytteistys ja Gibbsin näytteistys. [32, 33]

MH-näytteistys [34, 35] perustuu siihen, että prosessille generoidaan probabilistisesti seuraava tila, johon siirrytään, jos lausekkeessa (2.4) annettu MH-siirtymisehto täyttyy. Jos ehto ei toteudu, siirto hylätään ja toinen tilakandidaatti generoidaan. [32]

$$\alpha(x,y) = \begin{cases} \min\left[\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right] & \text{jos } \pi(x)q(x,y) > 0 \\ 1 & \text{jos } \pi(x)q(x,y) \leq 0 \end{cases} \tag{2.4}$$

$$\begin{aligned} \pi(x) &= \text{Nykyisen tilan todennäköisyys} \\ \pi(y) &= \text{Seuraavan tilan todennäköisyys} \\ q(x,y) &= \text{Todennäköisyys siirtyä nykyisestä tilasta seuraavaan} \\ q(y,x) &= \text{Todennäköisyys siirtyä seuraavasta tilasta nykyiseen} \end{aligned}$$

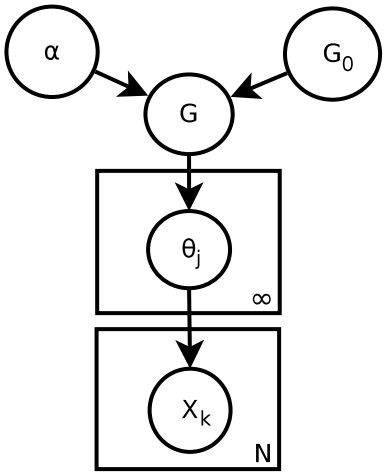
Gibbsin näytteistys [36] on lohkotetun MH-näytteistyksen tärkeä erikoistapaus, jossa lohko on yhden parametrin kokoinen. [32] Gibbsin näytteistyksen ideana on, että yksittäinen näyte poistetaan mallista, ja sille valitaan uusi luokka kaikkien jäljelle jäävien näytteiden perusteella. [33]

Gibbsin näytteistystä voidaan esimerkiksi hyödyntää sanojen erotteluun. Jos tunnetaan jokin Bayesiin pohjautuva algoritmi, joka pystyy erottelemaan yksittäisen tekstikappaleen sanat sen perusteella, miten kaikkien muiden korpuksen tekstikappaleiden sanat on eroteltu, voimme ratkaista koko korpuksen sanojen erottelun Gibbsin näytteistyksellä. Jos alussa ei tunneta mitään esitietoa kappaleiden segmentoinneista, erotellaan ensimmäisen kappaleen sanat käytännössä täysin satunnaisesti. Toinen kappale segmentoidaan ensimmäiseen segmentointiin nojaten, kolmas ensimmäiseen ja toiseen nojaten, ja niin edelleen. Kun kaikki kappaleet on segmentoitu kerran, aletaan

niitä segmentoimaan uudelleen niin, että jokaisen kappaleen segmentoinnissa hyödynnetään kaikkien muiden kappaleiden segmentointeja. Kun näin jatketaan riittävän kauan, konvergoituu jokaisen yksittäisen kappaleen segmentointi kohti ratkaisua, joka on todennäköinen kaikkien muiden segmentointien perusteella. Toisin sanoen, näytteistämällä segmentointeja uudelleen riittävän kauan, konvergoituvat kaikkien kappaleiden segmentoinnit kohti jotain ratkaisua, jota käytetty sanojen erottelun algoritmi pitää todennäköisenä.

2.3.3. Mallintaminen

Bayesin mallilla ratkaistavia ongelmia voidaan mallintaa generatiivisella mallilla, joka kuvaa sitä, millaisesta prosessista näytteet voisivat olla peräisin tilastollisessa mielessä. Lausekkeessa (2.5) ja kuvassa 2.2 on esitetty esimerkki tällaisesta mallista. [33]



Kuva 2.2: Tässä generatiivisessa mallissa on N kappaletta näytteitä X_k, joista jokainen kuuluu yhteen mahdollisesti äärettömän monesta luokasta j. Jokaisella luokalla j on parametri θ_j, joka on generoitu prosessista G, jolla on parametri α ja perusjakauma G_0.

$$
\begin{aligned}
G &\sim DP(G_0, \alpha) \\
\theta &\sim G \\
x_k &\sim \theta_k \\
G_0 &= \text{Perusjakauma.} \\
G &= \text{Dirichlettiprosessi.} \\
\theta &= \text{Piilotettu parametri.} \\
X_k &= \text{Havaittu näyte.}
\end{aligned}
\tag{2.5}
$$

2.4. Ei-parametriset Bayesin menetelmät

Ei-parametriset Bayesin menetelmät ovat Bayesin menetelmien alaluokka. Ne eivät tee oletuksia tarkasteltavan prosessin parametreista, kuten esimerkiksi sanojen lukumäärästä, vaan ne opitaan automaattisesti iteraation yhteydessä.

DP (Dirichlettiprosessi) on keskeisessä roolissa ei-parametrisissa Bayesin menetelmissä, sillä sen avulla pystytään mallintamaan ja ratkaisemaan monia käytännön ongelmia.

PYP (Pitman-Yor prosessi) on DP:n kaksiparametrinen yleistys, joka on saavuttanut suosiota monissa uusimmista LKK-menetelmissä.

DP:eistä ja PYP:eistä voidaan rakentaa hierarkisia prosesseja, joiden avulla pystytään mallintamaan monimutkaisempia ongelmia.

2.4.1. Dirichlettiprosessi

DP [37] on diskreetti todennäköisyysjakauma, jonka näytteet ovat todennäköisyysjakaumia. DP:n diskreettisyys tarkoittaa sitä, että sen todennäköisyys generoida sama näyte kaksi kertaa on suurempi kuin nolla, toisin kuin jatkuvilla jakaumilla. Todennäköisyysjakaumien generointi tarkoittaa sitä, että jokainen DP:stä otettu näyte on jokin sovelluskohtainen todennäköisyysjakauma. DP:n generoimat jakaumat voivat olla esimerkiksi multinomiaalijakaumia, jota voidaan näytteistää edelleen tavallisen todennäköisyysjakauman tavoin. [38, 39, 40, 41]

Käytännössä DP:n diskreettisyys toteutuu siten, että siitä otettu näyte on sama kuin jokin aikaisemmin otettu näyte sitä suuremmalla todennäköisyydellä, mitä useammin tämä näyte on otettu aikaisemmin. Lisäksi DP generoi kokonaan uusia näytteitä sitä suuremmalla todennäköisyydellä, mitä suurempi sen hajautumiskerroin α on ja mitä vähemmän näytteitä on generoitu. Toisin sanoen, DP:llä on klusterointiominaisuus. [38, 39, 40, 41]

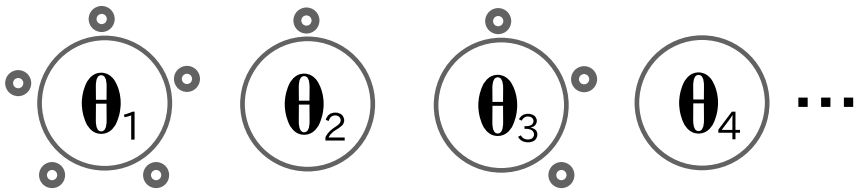
DP on matemaattisesti muotoa $DP(G_0, \alpha)$, missä G_0 on sen perusjakauma ja α hajautumiskerroin. [37] G_0 määrää sen, minkä tyyppisiä todennäköisyysjakaumia DP generoi. Jos se on esimerkiksi dirichlettijakauma, DP:stä otetut näytteet ovat multinomiaalijakaumia. Hajautumiskerroin vaikuttaa siihen, kuinka todennäköisesti prosessista otettu näyte on kokonaan uusi: mitä suurempi se on, sitä todennäköisemmin DP generoi kokonaan uuden näytteen. [38, 39, 40, 41]

DP:stä on esitetty useita erilaisia formalisointeja: kiinalaisen ravintolan prosessi [42], Pólyan uurnan prosessi [43] ja tikunkatkaisuprosessi [44]. Pólyan uurnan prosessi esittää klusterin valinnan eri tavalla kuin kiinalaisen ravintolan prosessi, mutta matemaattisesti menetelmät ovat identtiset. Tikunkatkaisuprosessi eroaa näistä kahdesta, sillä se ei generoi klustereita yksittäisille näytteille, vaan tuottaa suoraan klusterien suhteelliset koot.

Kiinalaisen ravintolan prosessissa on ääretön määrä klustereita, joihin lisätään näytteitä yksi kerrallaan. [45, s. 58-62] Ensimmäistä näytettä lisätessä klustereiden lukumäärä on nolla, joten näyte lisätään uuteen klusteriin, ja klusterille otetaan arvo G_0:sta. Seuraava näyte lisätään samaan klusteriin todennäköisyydellä $1/(1 + \alpha)$ tai uuteen klusteriin todennäköisyydellä $\alpha/(1 + \alpha)$. Kun näytteiden lisäämistä jatketaan, prosessin tila muodostuu kuvan 2.3 kaltaiseksi, ja seuraavan näyte klusteroidaan lausekkeen (2.6) antamien todennäköisyyksien mukaisesti. [38, 39, 40, 41]

$$P(c_n = k|c_{1:n-1}) = \frac{N_k}{n-1+\alpha}, 1 \leq k \leq K \tag{2.6}$$
$$P(c_n = K+1|c_{1:n-1}) = \frac{\alpha}{n-1+\alpha}$$

$n =$ Lisättävän näytteen järjestysnumero.
$k =$ Mikä tahansa ei-tyhjä klusteri.
$K =$ Ei-tyhjien klusterien lukumäärä.
$c_n =$ Näytteen n klusteri.
$c_{1:n-1} =$ Kaikkien edellisten näytteiden paikat klustereissa.
$N_k =$ Näytteiden lukumäärä klusterissa k.
$\alpha =$ Hajautumiskerroin.



Kuva 2.3: Esimerkki kiinalaisen ravintolan prosessista. Isot ympyrät ovat klustereita ja pienet pisteet niihin kuuluvia näytteitä. θ:t ovat klustereiden parametreja, jotka on näytteistetty DP:n G_0:sta.

Käytännössä DP:tä voidaan soveltaa, kun tarkasteltava ongelma koostuu näytteistä tai muista vastaavista yksiköistä, joilla on joitakin tunnettuja piirteitä sekä jokin tuntematon ominaisuus, joka halutaan selvittää. Tällöin näytteet voidaan lisätä DP:hen käyttäen hyväksi prosessin klusterointitodennäköisyyttä $P(x)$ sekä jotain sovelluskohtaista todennäköisyyttä $F(x, \theta_k)$, joka kertoo, kuinka hyvin näytteen tunnetut piirteet vastaavat klusterin parametreja. Näiden tietojen pohjalta näytteet voidaan luokitella klustereihin esimerkiksi Gibbsin näytteistyksen avulla. [46]

Klusterien induktion lisäksi myös DP:n hyperparametrit voidaan laskea automaattisesti Gibbsin näytteistyksen yhteydessä. [47] Tällöin saadaan aikaan prosessi, joka toimii täysin automaattisesti ilman yhtään käsin asetettavaa parametria.

2.4.2. Pitman-Yor prosessi

DP:sta on yleistetty Poisson-jakaumaa hyödyntävä PYP [48], joka lisää DP:n klusterointitodennäköisyyteen uuden muuttujan. Sen on todettu soveltuvan erityisen hyvin luonnollisen kielen mallinnukseen, sillä se mallintaa potenssilain jakaumia DP:tä paremmin [49].

PYP on matemaattisesti muotoa $PYP(G_0, \alpha, d)$, missä G_0 ja α ovat samat kuin DP:ssä ja d säätelee klusterien kokojen jakauman hännän pituutta. Parametrin arvolla $d = 0$ PYP palautuu DP:ksi.

PYP:n klusterin generoinnin todennäköisyyden kaava [49] on annettu lausekkeessa (2.7).

$$P(c_n = k|c_{1:n-1}) = \frac{N_k - d}{n - 1 + \alpha}, 1 \leq k \leq K \tag{2.7}$$
$$P(c_n = K + 1|c_{1:n-1}) = \frac{\alpha + d \cdot K}{n - 1 + \alpha}$$

$n =$ Lisättävän näytteen järjestysnumero.
$k =$ Mikä tahansa ei-tyhjä klusteri.
$K =$ Ei-tyhjien klusterien lukumäärä.
$c_n =$ Näytteen n klusteri.
$c_{1:n-1} =$ Kaikkien edellisten näytteiden paikat klustereissa.
$N_k =$ Näytteiden lukumäärä klusterissa k.
$d =$ Hyperparametri.
$\alpha =$ Hyperparametri.

Käytännössä PYP:iin pohjautuva induktio voidaan DP:n tavoin toteuttaa esimerkiksi Gibbsin näytteistyksen avulla [49]. Lisäksi PYP:n hyperparametrien induktion Gibbsin näytteistyksen yhteydessä on mahdollista [50].

2.4.3. Hierarkkiset prosessit

Kun DP:n tai PYP:n perusjakaumaksi G_0 asetetaan toinen samanlainen prosessi, kutsutaan näiden prosessien kokonaisuutta hierarkiseksi prosessiksi. Hierarkisessa prosessissa jokainen ylemmän tason klusteri on yhdistetty erilliseen alemman tason prosessiin niin, että alemman tason prosessin induktio kohdistetaan pelkästään siihen liittyvän ylemmän tason klusterin näytteisiin [51].

Yleensä hierarkkisia prosesseja hyödynnetään sellaisissa ongelmissa, joiden ratkaisemisessa voidaan hyödyntää eri tasoilla tehtyjä yleistyksiä. Näistä prosesseista etenkin HPYP (hierarkkinen Pitman-Yor prosessi) on saanut laajaa huomiota, ja sitä on käytetty esimerkiksi uusimmissa n-grammikielimalleissa niin, että hierarkian eri tason kuvaavat eri pituisia n-grammikonteksteja. HPYP:n todennäköisyys on esitetty lausekkeessa (2.8) [49, 52].

$$P(w|h) = \frac{c(w|h) - dt_{hw}}{\theta + c(h)} + \frac{\theta + dt_h}{\theta + c(h)} P(w|h') \tag{2.8}$$

$P(w|h) =$ Todennäköisyys.
$P(w|h') =$ Pohjatodennäköisyys.
$c(w|h) =$ Näytteiden lukumäärä klusterissa h, 0 jos h:ta ei ole olemassa.
$t_{hw} =$ 1 jos klusteri h on olemassa, 0 jos h:ta ei ole olemassa.
$c(h) =$ Näytteiden lukumäärä.
$t_h =$ Klusterien lukumäärä.
$d =$ Hyperparametri.
$\theta =$ Hyperparametri.

Hierarkkisista prosesseista on kehitetty useita erilaisia variaatioita, jotka käyttävät tai yhdistävät erilaisia prosesseja tai induktiomenetelmiä. Lisäksi prosessien väliin voi-

daan lisätä erillisiä adaptorikerroksia [53, 54], joilla prosessilta toiselle siirtyvää tietoa voidaan muuntaa eri muotoon induktion tehostamiseksi.

3. KATSAUS SANASTON POIMINNAN TUTKIMUKSEEN

Ohjaamatonta sanaston oppimista ja poimintaa on tutkittu aikaisemmin kohtuullisessa määrin, vaikkakin kokonaisia monikielisiä järjestelmiä on vielä melko vähän. Uusimpia menetelmiä tarkastelemalla voidaan siltikin oppia paljon sanaston poimijoiden perusrakenteesta ja eri tekniikoiden toimivuudesta.

Ohjaamaton sanaston poiminta on kokonaisuudessaan varsin laaja tehtävä, mikä on oletettavasti syynä siihen, ettei sen ratkaisemiseksi ole esitetty suurta määrää ratkaisuja. Muutamia on kuitenkin olemassa, kuten Schonen [55] tohtorin väitöskirjassa esitelty järjestelmä ja Goldwaterin [27] väitöskirjassa esitelty Bayesin menetelmään pohjautuva järjestelmä.

Schonen ja Goldwaterin menetelmien pohjalta voidaan perustellusti esitellä kieliriippumattoman sanaston poimijan rakenne. Näiden menetelmien pohjalta voidaan todeta, että yksi perusteltu lähestymistapa sanaston keräämiseen on jakaa ongelma kolmeen osaan: sanojen erottelu, MI (morfologian induktio) ja SLI (sanaluokan induktio). Tässä työssä nojaudutaan pitkälti samaan rakenteeseen, kuin mitä nämä kaksi uusinta sanaston poimijaakin hyödyntävät.

3.1. Schonen menetelmä

Schone esitteli tohtorin väitöskirjassaan ohjaamattoman ja pääosin esitiedosta vapaan menetelmän sanaston poimijan. Menetelmä koostui kolmesta välivaiheesta: sanojen erottelu, MI ja SLI. [55]

Aluksi Schonen menetelmä erotteli sanat toisistaan. Erottelu voitiin tehdä joko kiinteiden sanavälimerkkien perusteella tai ilman sanavälejä erillisen algoritmin avulla. Yksittäisten sanojen tunnistamisen lisäksi järjestelmä käytti erillistä algoritmia usean sanan sanaliittojen tunnistamiseen. [55]

Erotellut sanat käsiteltiin seuraavaksi monivaiheisella MI-menetelmällä. Induktion ensimmäisessä vaiheessa sanoille tunnistettiin mahdolliset ortografiset etuliitteet ja päätteet. Toisessa vaiheessa niistä valittiin relevantit lantentin kontekstitiedon sekä etuliitteiden ja päätteiden yleisyyksien perusteella. [55]

SLI:n Schone toteutti klusteroimalla aluksi sanat käyttäen klusterointipiirteinä sekä niiden morfologiaa että ympäröiviä sanoja. Schonen klusterointimenetelmä on varsin monimutkainen, sillä se tuottaa aluksi sanoille kolme eri tyyppistä klusteria hyödyntäen eri morfologisen ja kontekstipohjaisen tiedon yhdistelmiä. Tämän jälkeen nämä klusterit yhdistetään erillisessä vaiheessa ja merkitään itse sanaluokkatiedolla vielä erillisellä probabilistisella menetelmällä. [55]

Käytännön toteutuksen kannalta Schonen lähestymistavassa on ongelmallista se, että sen eri osat käyttävät hyvin erilaisia tekniikoita toisiinsa verrattuna. Tämä lisää järjestelmän monimutkaisuutta eikä mahdollista helppoa tiedon jakamista järjestelmän eri osien välillä, mikä on ongelmallista jatkokehityksen kannalta.

Schonen menetelmän julkaisun jälkeen aiheeseen liittyvien osaongelmien piirissä on tapahtunut paljon jatkokehitystä. Esimerkiksi sanojen erotteluun, MI:hin ja SLI:hin on esitelty monia lupaavia menetelmiä, joita voitaisiin hyödyntää sanaston poiminnassa.

3.2. Goldwaterin menetelmä

Goldwater esitteli tohtorin väitöskirjassaan Bayesin menetelmään perustuvan sanaston poimijan. Järjestelmässä oli kaksi vaihetta, sanojen erottelu ja MI, jotka toimivat toisistaan erillään. Lisäksi tutkimuksessa esiteltiin, kuinka nämä kaksi menetelmää voitaisiin yhdistää, mutta varsinainen käytännön työ tämän tekemiseksi oli jätetty myöhempää tutkimusta varten. [27]

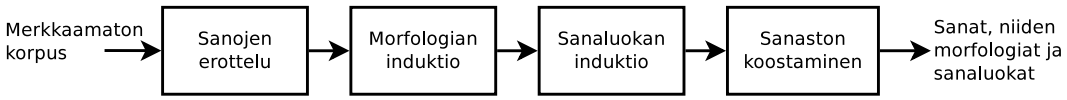
Menetelmässä hyödynnettiin joustavaa kaksiosaista kehystä, jota pystyttiin pienin muutoksin soveltamaan molempiin aliongelmiin. Goldwaterin kehys koostui yhdestä generaattorista, joka generoi satunnaisesti korpusta muistuttavia näytteitä, sekä yhdestä adaptorista, joka muunsi nämä näytteet Bayesin menetelmän hyödyntämään muotoon. Adaptorina toimi käytännössä aina jokin Bayesin prosessi. [27]

Goldwaterin testeissä käyttämä käytännön toteutus ei ollut täysin ohjaamaton, sillä esimerkiksi sanojen morfologisia luokkia ei johdettu automaattisesti. Goldwater esitti kuitenkin teoreettisesti, että järjestelmä voitaisiin laajentaa kokonaan ohjaamattomaksi. [27]

Goldwaterin menetelmä ei sisältänyt erillistä SLI:tä, mutta se johti sanoille morfologisia luokkia MI:n yhteydessä. [27] Periaatteessa yhdistetty järjestelmä pystyisi siis johtamaan pääasiallisesti samat tiedot kuin Schonen menetelmäkin, olettaen tarkastellun kielen sanaluokkien olevan suoraan yhteydessä sanojen morfologiseen käyttäytymiseen.

3.3. Sanaston poimijan rakenne

Kattavan sanaston poimijan toteutuksen voidaan katsoa sisältävän neljä päävaihetta, jotka ovat sanojen erottelu, MI, SLI ja sanaston koostaminen. Yksinkertaisimmillaan tällainen järjestelmä toimii niin, että se ottaa syötteenä merkkijonoja ja erottelee niistä sanat. Erotellut sanat annetaan MI:lle, joka johtaa niistä morfologian. Seuraavaksi sanoille johdetaan sanaluokat morfologian, esiintymiskontekstin tai molempien perusteella. Lopuksi kerätty tieto koostetaan sitä käyttävän sovelluksen vaatimaan muotoon. Kuvassa 3.1 on havainnollistettu tätä rakennetta graafisesti.



Kuva 3.1: Tyypillisen sanaston poimijan välivaiheet, joiden avulla merkitsemätön korpus voidaan muuntaa koneellisesti hyödynnettäväksi sanastoksi.

Toteutustavasta riippuen järjestelmän vaiheita voidaan mahdollisesti yhdistää, kuten esimerkiksi Goldwaterin menetelmässä. Hyvin monipuolisesti mallinnettu Bayesin prosessi voisi mahdollisesti yhdistää sekä sanojen erottelun, MI:n että SLI:n yhdeksi suureksi prosessiksi. [27]

4. SANASTON POIMIJAN OSAT JA MENETELMÄT

Kuten luvussa 3 todettiin, sanaston poiminta voidaan jakaa kolmeen pääosaan: sanojen erottelu, MI ja SLI. Tässä luvussa tarkastellaan sitä, miten kukin näistä yksittäisistä menetelmistä voidaan toteuttaa ohjaamattomassa ympäristössä.

Sanojen erottelu on perinteinen LKK-tehtävä, jossa yksittäiset sanat eristetään merkkijonosta. Se on käytännössä välttämätön osa sanaston poimintaa, sillä sanat on luonnollisesti pystyttävä tunnistamaan, jotta ne voitaisiin poimia. Sanojen erottelua varten on olemassa useita menetelmiä, mutta kaikki niistä eivät sovellu hyvin monikielisiin tai osittain ohjaamattomiin tehtäviin.

MI on toinen paljon tutkittu LKK-tehtävä. Siinä kielen morfologia, eli sanojen taipumisen käyttäytyminen, johdetaan automaattisesti kielestä tehtyjen havaintojen perusteella. Tämä tehtävä on erityisen tärkeä monikielisille sanaston kerääjille, sillä sanojen taivutusmuodoilla on tärkeä rooli monissa kielissä.

SLI on ohjaamaton ja esitiedosta vapaa variantti toisesta paljon tutkitusta LKK-tehtävästä, sanaluokan merkinnästä. Kuten sanaluokan merkintäkin, sen tehtävänä on tunnistaa, kuuluuko tarkasteltava sana esimerkiksi substatiivien tai verbien sanaluokkaan, mutta ilman esitietoa siitä, mitä sanaluokkia on olemassa ja mitä sanoja niihin kuuluu. SLI:n tai sanaluokan merkinnän voidaan katsoa olevan yksi olennaisista sanaston kerääjän tehtävistä, sillä sanaluokkatietoa hyödynnetään laajasti LKK-sovelluksissa.

4.1. Sanojen erottelu

Sanojen erottelu voi vaikuttaa yksinkertaiselta ongelmalta esimerkiksi germaanisia kieliä tai suomea tuntevalle, sillä näissä kielissä sanat voidaan poimia melko yksinkertaisilla sanavälejä ja välimerkkejä hyödyntävillä säännöillä. Tämä ei kuitenkaan ole mahdollista kaikkien kielten kohdalla [25, s. 372], sillä esimerkiksi kiina, japani ja thai ovat kieliä, joissa ei käytetä sanavälejä erottamaan sanoja. Koska on olemassa kieliä, joissa ei käytetä sanavälejä, selvästikin universaali menetelmä ei voi perustua niiden käyttöön; ei ainakaan yksistään.

Sanojen erottelua universaalina ongelmana voidaan havainnollistaa poistamalla suomalaisesta tekstistä välilyönnit. Kun näin tehdään, on helppo havaita, että ihmiselle ei juurikaan tuota ongelmia palauttaa sanavälejä esimerkiksi tekstiin "tämäonpieniaskelihmiselle,muttajättiharppauskokoihmiskunnalle."Koska ihminen pystyy tähän helposti, voidaan oletettavasti kehittää myös algoritmi, joka pystyy siihen.

Sanojen erotteluun tarkoitetut algoritmit voidaan luokitella karkeasti kahteen ryhmään sen mukaan, vaativatko ne käsin merkittyä opetustietoa toimiakseen. Ohjatut menetelmät vaativat tätä, mutta ohjaamattomat menetelmät pystyvät oppimaan segmentoinnin ilman merkittyä esitietoa.

Ohjaamattomat menetelmät poikkeavat ohjatuista niin, etteivät ne käytä asiantuntijoiden merkitsemätöntä opetusmateriaalia, vaan johtavat opetusvaiheessa tilastollisen mallinsa puhtaasti merkitsemättömästä korpuksesta. Tämän lähestymistavan etuna on, että hidas ja kallis korpusten merkintä voidaan välttää, minkä ansioista menetelmiä voidaan helposti soveltaa uusiin kieliin.

Ohjaamattomat sanojen erottelijat ovat käytännössä myös esitiedosta vapaita, sillä sanojen erottelussa ei ole sanavälien paikkojen lisäksi paljoa muuta tietoa, joka on helposti hyödynnettävissä. Menetelmät, jotka hyödyntävät muuta tietoa tai hyvin pientä

määrää sanavälitietoa on perinteisesti luokiteltu ohjatuiksi tai osittain ohjatuiksi.

Uusimmista ohjaamattomista menetelmistä MKP:hen ja transitiotodennäköisyyteen perustuvilla menetelmillä on saavutettu hyviä tuloksia.

Bayesin menetelmään perustuva ohjaamaton sanojen erottelu on saanut viime aikoina erityisen paljon huomiota, ja lukuisia siihen perustuvia algoritmeja on kehitetty lupaavin tuloksin.

4.1.1. Ohjatut menetelmät

Ohjattujen menetelmien kielimalli opetetaan merkityn korpuksen avulla tai jotenkin muuten asiantuntijoiden tietoja hyväksi käyttäen, minkä jälkeen sitä käytetään itse segmentointiongelman ratkaisuun. Pääperiaatteena on, että opetusvaiheessa menetelmälle annetaan sekä tekstin tilastolliset ominaisuudet että sanavälien paikat, ja se johtaa säännöt sille, mitkä tekstin piirteet aiheuttavat sanavälin.

Yleensä ohjatut menetelmät mallintavat kieltä n-grammeilla. N-grammin pituuden kasvattaminen parantaa tämän luokan menetelmien tarkkuutta, joten jopa 11 merkin pituisten n-grammien käyttö voi olla kannattavaa[56]. Kehittyneemmissä malleissa käytetään myös muita piirteitä, mutta hyvä malli voi päästä yli 90%:n [56] tarkkuuteen pelkästään n-grammeilla.

TVK (tukivektorikone) ja ESK (ehdollinen satunnaiskenttä) ovat eräitä tehokkaiksi todettuja ohjatun sanojen erottelun menetelmiä.

4.1.1.1. Tukivektorikone

TVK [57] on binäärinen luokittelija, joka luokittelee korkeadimensionaalisen syötevektorin kahteen luokkaan jakamalla syöteavaruuden kahtia hypertason avulla. TVK opetetaan antamalla sille esimerkkisyötteitä, joiden on merkitty kuuluvan jompaankumpaan kahdesta luokasta. Näiden tietojen avulla TVK:lle voidaan opettaa parametrit, jotka määräävät luokat erottavan hypertason orientaation. Nämä paramaterit määräävät siten myös sen, kumpaan luokkaan itse prosessoitavan datan syötteet luokitellaan. [58]

Sanojen erottelussa TVK:n syötteinä voidaan käyttää esimerkiksi tarkasteltavaa kohtaa edeltävien ja seuraavien n-grammien vektoriesityksiä. Tuloksena saadaan, onko kyseisessä kohdassa sanaväliä vai ei.

TVK:lla on saavutettu esimerkiksi 92%:n tarkkus ja 88%:n saanti thain kielisten sanojen erottelussa. [56]

4.1.1.2. Ehdollinen satunnaiskenttä

ESK on probabilistinen malli, jota voidaan hyödyntää sekvenssien tai graafien merkitsemiseen. Se on sukua MPM:lle (markovin piilomalli), mutta mahdollistaa sitä joustavamman piirteiden hyödyntämisen parametreinä. Piirteet voivat olla mitä tahansa funktioita ja niitä voidaan ottaa mistä tahansa sekvenssin elementistä, mukaan lukien seuraavat elementit. [59]

ESK on yksi parhaimmista ohjatuista menetelmistä sanojen erotteluun. [56] Se saavutti esimerkiksi noin 95%:n [60] saannin ja tarkkuuden thain kielisten sanojen erottelussa ja noin 97%:n japanin kielisten sanojen erottelussa [61].

4.1.2. Minimaalinen kuvauksen pituus

MKP:hen perustuva sanojen erottelu kuuluu kompressiopohjaisten menetelmien joukkoon. Kompressiopohjaiset menetelmät etsivät niille annetulle korpukselle minimaalisen koodikirjaan perustuvan esityksen ja johtavat sanavälien paikat koodikirjasta.

MKP:tä on hyödynnetty esimerkiksi ohjaamattomassa sanojen erottelijassa, joka oppi erotteluun käytettyjen sanojen vartaloiden ja päätteiden koodikirjan. Menetelmä pystyi johtamaan yleisimpiä englannin ja turkin päätteitä, mutta sen tarkkuutta itse segmentointitehtävässä ei oltu ilmoitettu. [62]

4.1.3. Transitiotodennäköisyysmalli

Transitiotodennäköisyyspohjaiset menetelmät arvioivat sanavälien paikan todennäköisyyttä ympäröivien tekstisegmenttien yleisyyksien perusteella. Niiden toiminta pohjautuu siihen, että merkkien sekvenssit ovat yleensä ennustettavampia sanojen sisällä kuin niiden katkaisukohtien ylitse. [63, 64]

Yksi menetelmä on arvioida transitiotodennäköisyyttä n-grammien yleisyyksien avulla. Sanaväli voidaan havaita esimerkiksi silloin, kun tarkasteltavan kohdan molemmin puolin esiintyvien n-grammien todennäköisyys ylittää tietyn kynnysarvon tai muodostaa paikallisen maksimin. Tämä menetelmä vaati erittäin vähän käsin merkittyä opetusmateriaalia, ja sitä voidaan pitää kieliriippumattomana, koska se käyttää hyödyksi vain merkkitason tietoa. [65]

4.1.4. Bayesiin pohjautuvat menetelmät

Bayesin menetelmä soveltuu hyvin sanojen erotteluun, sillä se tuottaa yleensä parempia tuloksia kuin transitiotodennäköisyyksiin pohjautuvat menetelmät. [64, 66, 67] Toisin kuin transitiotodennäköisyyspohjaiset oppijat, Bayesin sanojen erottelijan toiminta perustuu lähtökohtaisesti itse sanojen oppimiseen sanavälien oppimisen sijaan. [67]

Bayesin segmentoinnin toimintaperiaate on segmentoida syötteitä probabilistisesti sillä tavalla, että kunkin segmentoinnin todennäköisyys on sitä suurempi, mitä enemmän aikaisemmin arvattuja sanoja se sisältää. Kun näytteitä on segmentoitu uudelleen riittävän monta kertaa, prosessi konvergoituu kohti oikeaa vastausta, olettaen että käytetty todennäköisyysmalli on sopiva.

Koska Bayesin sanojen erottelia oppii sanoja todennäköisyysmallin avulla, sen perustoimintaperiaate muistuttaa huomattavasti MKP:tä. Joissakin menetelmissä MKP:tä hyödynnetään myös suoraan yhtenä Bayesin induktion käyttämistä todennäköisyysmalleista. [68]

Tyypillisesti Bayesin segmentoijat hyödyntävät dynaamista ohjelmointia, kuten esimerkiksi Viterbi-algoritmia, segmentoinnin todennäköisyyksien ratkaisemisessa. [68, 67] Tällöin kaikkien eri segmentointien todennäköisyydet lasketaan rekursiivisesti, ja jo ratkaistujen segmenttien todennäköisyyksiä hyödynnetään muiden segmenttien todennäköisyyden laskemisessa.

Mochihashin [52] käyttämä menetelmä on yksi uusimmista Bayesiin pohjautuvista menetelmistä. Se mallintaa hierarkkisilla Bayesin prosesseilla [27, 31, 66] interpoloitua n-grammikielimallia [50], jonka pohjalta segmentoinnin todennäköisyydet lasketaan. Hierarkkisten prosessien avulla on mahdollista laskea todennäköisyydet yksittäisille merkeille, niiden n-grammeille, yksittäisille sanoille ja sanojen n-grammeille.

Koko merkkijonon segmentointien todennäköisyydet voidaan laskea dynaamisella ohjelmoinnilla, ja näiden todennäköisyyksien avulla voidaan näytteistää segmentointi tarkastellulle tekstilohkolle. Kun tekstilohkojen segmentointeja on satunnaisnäytteistetty uudelleen tarpeeksi monta kertaa, prosessi konvergoituu kohti ratkaisua. [52]

4.2. Morfologian induktio

MI:n tehtävänä on oppia, millaisia taivutussääntöjä tarkastellussa kielessä on olemassa. Perinteisesti MI:n tutkimus on keskittynyt ohjaamattomiin menetelmiin, jotka ovat ihmisten kielen oppimisen tutkimuksen kannalta mielenkiintoisimpia.

MI-menetelmät voidaan jakaa kahteen luokkaan sen mukaan, tarvitsevatko ne esitietoa vai pystyvätkö ne johtamaan morfologian ilman mitään esitietoa. Esitietopohjaiset menetelmät hyödyntävät esimerkiksi tietoa sanaluokista.

Esitiedoista vapaissa menetelmissä eräs lähestymistapa on hyödyntää sanojen ortografiaa olettamalla muokkausetäisyydeltään läheiset merkkijonot saman sanan taivutusmuodoksi. Toinen menetelmä on verrata sanojen kontekstia esimerkiksi LSA:n avulla ja yhdistää sanoja, jotka esiintyvät samanlaisessa kontekstissa.

MKP on suosittu esitiedosta vapaa menetelmä. Se löytää ihmisille tuttuja morfeemeja etsimällä segmentoinnin, joka periaatteessa minimoi vartaloiden ja liitteiden lukumäärän.

Bayesin menetelmään pohjautuvat menetelmät ovat yksi viime aikoina paljon huomiota saanut algoritmien ryhmä. MKP:n tavoin ne etsivät usein esiintyviä vartaloita ja päätteitä, mutta segmentoinnin uskottavuuden estimoinnissa voidaan soveltaa MKP:n kompressiokriteerin sijaan hyvin monia erilaisia todennäköisyyspohjaisia kriteerejä.

4.2.1. Esitietopohjaiset menetelmät

Yleensä esitietoa hyödyntävät MI-menetelmät hyödyntävät sanojen sanaluokkatietoa. Käytännössä ne ottavat syötteenä taivutettuja sanoja sekä sanaluokkamerkkauksen jokaista sanaa kohti. Näiden tietojen avulla ne pyrkivät johtamaan kullekin sanaluokalle tyypillisen morfologian ja edelleen yksittäisten sanojen morfologian.

Yksi esimerkki tällaisista mentelmistä on Gaussierin [69] esittämä algoritmi. Se ratkaisee sanaluokille tyypilliset päätteet etsimällä samaan sanaluokkaan ja samoilla kirjaimilla alkavia sanoja ja tarkastelemalla niiden päätteitä. Toinen samanlainen menetelmä on Neuvelin and Fulopin [70] algoritmi, joka tunnistaa sanaluokilla merkitystä teksistä sanojen etuliitteitä ja päätteitä merkkijonometriikan avulla. [71]

Näitä menetelmiä ei voida kuitenkaan suoraan hyödyntää ohjaamattomassa sanaston poimijassa, sillä sanaluokkatietoa ei ole saatavilla MI:tä ratkaistaessa.

4.2.2. Ortografia ja semantiikka

Yksi menetelmä käytti sekä merkkijonometriikkaa että sanojen esiintymisympäristöjä morfologisesti riippuvaisten sanojen löytämiseen. Tämä menetelmä poimi sanoja merkitsemättömästä tokenisoidusta korpuksesta ja muodosti niistä kaksi listaa, joista toinen määritteli sanojen merkkijonotason samankaltaisuuden ja toinen esiintymisympäristön samankaltaisuuden. Näiden listojen avulla poimitut sanat pisteytettiin pareittain sen mukaan, kuinka samanlaisia ne olivat ortografian ja semantiikan suhteen. Suuren pistemäärän saaneiden sanojen oletettiin olevan morfologisesti riippuvaisia. [72]

Universaalisuuden kannalta edellä kuvatulla menetelmällä on se toivottu piirre, ettei se oleta morfologian perustuvan etu- tai jälkiliitteisiin. Kun käytetään merkkijonometriikkaa liitteiden etsimisen sijaan, voidaan tukea muunkinlaista morfologiaa, kuten esimerkiksi englannin woman/women sanaa. [72]. Lisäksi universaalisuuden kannalta hyvä ominaisuus on, ettei tämä menetelmä tarvitse sanojen erottelua lukuun ottamatta mitään esitietoa kielestä.

Pelkästään ortografiaa ja semantiikkaa hyödyntämällä ei kuitenkaan saada virheettömiä tuloksia. Erilaisia virheitä voidaan eliminoida esimerkiksi ottamalla huomioon etu- ja jälkiliitteiden yleisyydet, paikallinen syntaktinen ympäristö ja aikaisemmin löydetyt morfologiset yhteydet. [73]

4.2.3. Minimaalinen kuvauksen pituus

MKP:tä käytetään MI:hin melko pitkälti samalla tavalla kuin sanojen erottelussakin. Tällöin MKP pyrkii minimoimaan sekä sanojen vartaloiden että mahdollisten etuliitteiden ja päätteiden kuvausten pituudet.

Yksi mielenkiintoinen esimerkki MKP:n hyödyntämisestä on Snoverin [74] esittelemä järjestelmä. Se käytti tavallisen vartaloiden ja päätteiden minimoinnin lisäksi tietoa siitä, mitkä morfeemit esiintyivät tietyn näköisten vartaloiden kanssa. [71]

Goldsmith käytti myös MKP:tä ohjaamattomaan MI:hin tunnetussa Linquistica-sovelluksessaan. Hänen menetelmässään sanat koostuivat vartalosta sekä mahdollisesti tyhjästä päätteestä. Kuvauksen pituuden laskemisessa käytettiin samoin hyväksi tietoa siitä, mitkä morfeemit yleensä esiintyivät samassa vartalossa. Täten menetelmä hyödynsi periaatteessa myös yksinkertaista tietoa sanaluokista. [75, 26]

4.2.4. Bayesiin pohjautuvat menetelmät

Yleensä Bayesiin pohjautuva MI hyödyntää sanojen ortografista tietoa segmentoimalla yksittäisten sanojen merkkijonoja probabilistisesti morfeemeihin. Morfeemien todennäköisyyksiä mallinnetaan yleensä Bayesin prosesseilla, jotka suosivat usein havaittuja morfeemikandidaatteja harvinaisten sijaan.

Yksi segmentointiin nojaava menetelmä on Goldwaterin käyttämä adaptori-generaattori -kehykseen perustuva menetelmä. Se käytti ei-parametrisia Bayesin prosesseja sanojen vartaloiden ja morfeemien todennäköisyyksien mallintamiseen ja segmentoi tämän mallin avulla sanat vartaloihin ja päätteisiin. Päätteiden todennäköisyyden mallinnuksessa hyödynnettiin myös tietoa siitä, kuinka usein pääte oli esiintynyt sanan vartalon yhteydessä. [27]

Johnson yleisti Goldwaterin kehyksestä edelleen hierarkkisiin Bayesin prosesseihin pohjautuvat adaptorikieliopit, joita hän hyödynsi MI:ssä. Adaptorikieliopeilla pystytään segmentoimaan merkkijonoja hierarkkisiksi puurakenteeksi, minkä ansiosta niillä voidaan ratkaista sekä sanojen erotteluun että MI:hin liittyviä ongelmia. [31, 54]

Bayesin menetelmä mahdollistaa pelkän segmentoinnin lisäksi myös muun tyyppisen tiedon oppimisen. Sillä pystytään oppimaan yksinkertaisen segmentoinnin lisäksi esimerkiksi vartalonmuunnoksen sääntöjä [76], joilla voidaan parantaa muun muassa suomen astevaihtelun tai englannin vartalon viimeisen merkin toiston tai poiston mallinnusta.

4.3. Sanaluokan induktio

SLI:ssä sanoille johdetaan sanaluokat ilman, että tiedetään yhdenkään sanan luokkaa etukäteen. Siinä voidaan käyttää korkeintaan esitietoa siitä, mitä sanoja ja mahdollisesti sanaluokkia on olemassa.

Koska esitietoa sanaluokista ei ole olemassa, SLI-menetelmät tyytyvät klusteroimaan samalla tavalla käyttäytyviä sanoja nimettömiin luokkiin, jotka pyritään saamaan vastaamaan todellisia sanaluokkia mahdollisimman hyvin. Käytännön sovellusten vaatima metatieto, kuten sanaluokkien nimet, on käytännössä lisättävä manuaalisesti jälkikäteen.

Sanaluokan klusterointimenetelmät voidaan jakaa karkeasti kahteen luokkaan: distributionaaliseen ja ominaisuuspohjaiseen klusterointiin. [55] Distributionaalinen klusterointi käyttää sanojen kontekstia klusterointikriteerinä. Ominaisuuspohjainen klusterointi hyödyntää yksittäisten sanojen piirteitä, yleensä morfologiaa [77, 28].

SLI:hin on kehitetty useita Bayesin menetelmään perustuvia algoritmeja. Useimmat niistä hyödyntävät distributionaalista tietoa, mutta myös morfologiaa on hyödynnetty joissakin menetelmissä.

4.3.1. Distributionaalinen klusterointi

MPM (piilotettu markovinmalli) on yksi suosituimmista distributionaalisista SLI-menetelmistä. MPM:ään pohjautuvat menetelmät analysoivat sanojen ympäristöä ja päättelevät sen perusteella, mihin sanaluokkaan ne todennäköisesti kuuluvat.

Vanhin MPM:ään pohjautuva menetelmä on Brownin algoritmi, joka mallintaa tekstiä yksinkertaisesti sanojen bigrammeilla. Se valitsee aluksi kaikille sanoille uniikin klusterin ja tämän jälkeen iteratiivisesti yhdistää klusterita sen mukaan, kuinka paljon operaatio pienentää bigrammisanamallin kompleksisuutta. [78, 77]

Pitkään parhaimpana algoritmina pidetty Clarkin menetelmä hyödyntää niin ikään distributionaalista tietoa samanlaisen MPM:n muodossa. Todennäköisin syy Clarkin menetelmän tehokkuuteen on se, että se käytti sekä distributionaalista bigrammi-MPM:ää että ominaisuuspohjaista tietoa. [28, 77]

4.3.2. Ominaisuuspohjainen klusterointi

Pelkän distributionaalisen tiedon hyödyntämisessä on se ongelma, että harvoin esiintyviä sanoja ei pystytä klusteroimaan luotettavasti, sillä niistä ei ole saatavilla riittävästi tietoa. Nämä sanat ovat merkittäviä koko induktioprosessin kannalta, joten ominaisuuspohjaisen tiedon hyödyntäminen voi parantaa menetelmän suorituskykyä huomattavasti. [28]

Tyypillisesti ominaisuuspohjaisessa klusteroinnissa hyödynnetään sanojen ortografisia piirteitä. Tällaisina piirteinä voidaan käyttää esimerkiksi sanaan kuuluvia merkkijonotason n-grammeja tai mahdollisesti sanojen morfeemeja, jos ne on pystytty jotenkin ratkaisemaan.

Clark hyödynsi hybridimenetelmässään sanojen ortografisia ominaisuuksia, suosien klusterointia, jossa samassa klusterissa oli samanlaisia sanoja. Sanojen samanlaisuutta arvioitiin probabilistisesti merkkijonotason MPM:n avulla. [28, 77]

4.3.3. Bayesiin pohjautuvat menetelmät

Suurin osa perinteisistä Bayesiin pohjautuvista SLI-menetelmistä käyttivät pelkästään MPM:ää klusterointikriteerinä. [77] Tyypillisen tällaisen menetelmän toimintaperiaate oli johtaa sanaluokat trigrammi MPM:stä esimerkiksi Dirichlet-multinomiaalijakaumaan pohjautuvan induktion avulla. [79, 77]

Pelkästään distributionaalista tietoa hyödyntävät Bayesin menetelmät eivät ole kuitenkaan saavuttaneet yhtä hyviä tuloksia kuin Clarkin ja Brownin menetelmät. Uusimmat sekä distributionaalista että ominaisuuspohjaista tietoa hyödyntävät Bayesiin pohjautuvat menetelmät ovat kuitenkin päässeet samalle tasolle niiden kanssa. [77]

Ensimmäisiä tällaisia menetelmiä oleva algoritmi käytti MPM:n lisäksi merkkijonotason ominaisuuksia klusterointikriteereinä. Merkkijonotason ominaisuuksista menetelmä hyödyntää merkkien trigrammeja sekä tietoa isoista alkukirjaimista. Merkkien trigrammeilla voidaan karkeasti mallintaa sitä, miten morfologia ja sanaluokat liittyvät toisiinsa. [77]

Blunsomin tätä ideaa edelleen kehittänyt menetelmä käytti myös trigrammi MPM:ää, mutta tasoitti sitä hierarkisilla Bayesin prosesseilla. Tasoituksen ansiosta bigrammi- ja unigrammitietoa voitiin hyödyntää silloin, kun trigrammi- tai bigrammitietoa ei ollut olemassa tarpeeksi. Sanatason trigrammien lisäksi menetelmä hyödynsi merkkitason bigrammeja, joiden tarkoitus oli niin ikään mallintaa morfologiaa. [80]

5. SANASTON POIMINNAN ARVIOINTI

Sanaston poiminnan eri ratkaisumenetelmien lisäksi on syytä perehtyä siihen, miten niillä saatuja tuloksia voidaan arvioida ja verrata keskenään. Näin saadaan arvokasta tietoa järjestelmän sekä sen osien suorituskyvystä ja siten viitteitä mahdollisista tulevan kehityksen suunnista.

Poimitun sanaston määrän ja laadun arviointi on pääasiallinen tapa, jolla sanaston poimijoita voidaan arvioida. Koska sanaston poiminta koostuu useista aliongelmista, jotka ovat itsessään haastavia, on lisäksi hyödyllistä tarkastella myös niiden toimintaa erikseen.

Sanojen erottelun toiminnalla on merkittävä vaikutus koko järjestelmän toimintaan ja siten tulosten laatuun, sillä järjestelmä ei luonnollisesti löydä yhtä paljon oikeita sanoja, jos niiden erottelu ei toimi. Tämän vuoksi on perusteltua arvioida sanojen erottelua erikseen.

MI:tä arvioitaessa voidaan saada tietoa siitä, tunnistaako järjestelmä sanojen rakenteet oikein ja pystyykö se yhdistämään saman sanan eri taivutusmuodot. Koska morfologiatietoa voidaan hyödyntää sekä SLI:ssä että lopullisessa sanastossa, on tämän tiedon oikeellisuuden arviointi erikseen niin ikään perusteltua.

SLI:n arviointi antaa suoraa tietoa siitä, kuinka hyvin kerätyn sanaston sanaluokat vastaavat todellisuutta. Jos sanaluokkatietoa halutaan kerätä, on myös sen toimintaa siis syytä arvioida.

5.1. Poimitun sanaston arviointi

Sanaston poimintaa voidaan arvioida vertaamalla poimittua sanastoa asiantuntijan merkitsemään oikeaan sanastoon. Tällöin voidaan käyttää perinteisiä arviointimenetelmiä, kuten tarkkuutta, saantia ja F-mittaa [81].

Tarkkuus kuvaa sitä, kuinka suuri osuus kaikista kerätyistä näytteistä on oikein luokiteltuja. Saanti taaskin kertoo sen, kuinka paljon oikein luokiteltuja näytteitä oli suhteessa suurimpaan mahdolliseen määrään oikeita näytteitä, joita oltaisiin voitu löytää. F-mitta yhdistää nämä kaksi lausekkeen (5.1) esittämällä tavalla. [81]

$$F_\alpha = \frac{PR}{(1-\alpha)P + \alpha R}, 0 \leq \alpha \leq 1 \tag{5.1}$$

$$P = \frac{C}{C+S+I}$$

$$R = \frac{C}{C+S+D}$$

$\alpha =$ Saannin painotuskerroin.

$P =$ Tarkkuus.

$R =$ Saanti.

$C =$ Oikein luokiteltujen oikeiden sanojen lukumäärä.

$S =$ Väärin luokiteltujen oikeiden sanojen lukumäärä.

$D =$ Puuttuvien lukumäärä.

$I =$ Ylimääräisten lukumäärä.

Kun keskitytään poimimaan sanoja ja niiden sanaluokkia, lausekkeen (5.1) muuttu-

jille voidaan antaa seuraavat selkokieliset merkitykset:

- C: Löydetyt oikeat sanat, joiden sanaluokka on oikein.
- S: Löydetyt oikeat sanat, joiden sanaluokka on väärin.
- D: Oikeat sanat, joita ei löydetty.
- I: Ylimääräiset sanat, jotka löydettiin.

Oletetaan esimerkin vuoksi, että tällainen sanoja poimiva ja niiden sanaluokkia merkitsevä järjestelmä on käsitellyt tekstin, jossa on yhteensä 100 eri sanaa. Se on löytänyt $C = 40$ sanaa, joiden sanaluokat se on merkannut oikein, ja $S = 30$ sanaa, joiden sanaluokat se on merkannut väärin. Lisäksi se on löytänyt $I = 10$ virheellistä sanaa, joita ei ollut tekstissä, ja $D = 30$ sanaa on jäänyt löytämättä. Tällöin lausekkeen (5.1) mukaan järjestelmän tarkkuus $P = 40/80$ ja saanti $R = 40/100$. Tästä voidaan edelleen laskea järjestelmän F-mitan arvo, joka on tässä tapauksessa $F_{0.5} = 0.444$.

Sanaston laadun lisäksi tämän työn kannalta kiinnostava arviointikriteeri on myös se, kuinka paljon asiantuntijan tekemää työtä sanaston poiminta vaatii. Tätä voidaan arvioida esimerkiksi mittaamalla työhön kuluva aika ja suhteuttamalla se saatujen tulosten määrään.

5.2. Sanojen erottelun arviointi

Sanojen erottelua voidaan niin ikään arvioida vertaamalla järjestelmän tuottamaa segmentointia asiantuntijan tuottamaan oikeaan segmentointiin. Arviointimenetelminä voidaan myös käyttää tarkkuutta ja saantia samaan tapaan kuin sanaston arvioinnissa.

Sanojen erottelussa tarkkuus ja saanti käsittelevät pelkästään sitä, olivatko hypoteettisen segmentoinnin yksiköt osa todellista segmentointia vaiko eivät, sillä ongelmassa ei ole luokitteluosaa ollenkaan. Täten tarkkuus ja saanti voidaan laskea lausekkeen (5.2) mukaisesti. [67]

$$
\begin{aligned}
P &= \frac{C}{C+I} \\
R &= \frac{C}{C+D} \\
P &= \text{Tarkkuus.} \\
R &= \text{Saanti.} \\
C &= \text{Oikeiden segmenttien lukumäärä.} \\
D &= \text{Puuttuvien segmenttien lukumäärä.} \\
I &= \text{Ylimääräisten segmenttien lukumäärä.}
\end{aligned}
\tag{5.2}
$$

Tarkkuus ja saanti voidaan esittää sanavälien, tokenien tai leksikaalisten yksiköiden tasolla. Ensimmäisessä näistä tapauksista tarkastellaan sitä, oliko sanaväli samassa paikkaa kuin oikeassa segmentoinnissa. Toisessa verrataan eroteltujen sanojen sekvenssejä oikeisiin sanojen segmentointeihin. Kolmannessa verrataan segmentointien yksilöllisiä sanoja huomioimatta sitä, missä tai kuinka monesti ne esiintyivät. [67]

5.3. Morfologian induktion arviointi

Perinteisesti MI:tä on tyydytty arvioimaan tarkastelemalla asiantuntijoiden voimin, kuinka järkeviä järjestelmän löytämät yleisimmät päätteet ovat. [28] Tämä menetelmä on siinä mielessä käytännöllinen, että se huomioi morfologian epämääräisyyden ja on melko vaivaton asiantuntijoille. Toisaalta se ei suoraan kerro sitä, kuinka paljon vääriä segmentointeja on todellisuudessa tehty.

MI:n tarkemmassa arvioinnissa ongelmallista on se, että morfeemien segmentointiin ei ole aina yksiselitteistä vastausta. Esimerkiksi sana "kuninkaalle" voitaisiin jonkin morfologiamallin mukaan segmentoida muotoon "kunin+kaalle" ja toisen mukaan "kuningas-gas+kaalle". Tämän vuoksi tuloksien oikeellisuutta on usein hankalaa analysoida objektiivisesti. Tätä ongelmaa voidaan kuitenkin pienentää löyhentämällä kriteeriä, jolla segmentointeja verrataan. [82]

Yksi käyttökelpoinen menetelmä on tarkastella sitä, miten saman sanan eri taivutusmuotojen vartalo segmentoidaan. Tämän kriteerin mukaan sana katsotaan olevan oikein segmentoitu, jos sen vartalo on sama kuin muissa saman sanan taivutusmuotojen segmentoinneissa. [82] Segmentointien epämääräisyyden hallinnan lisäksi menetelmällä on se käytännöllinen piirre, että asiantuntijoiden ei tarvitse merkata korpuksen morfologiaa. Merkintöjen tarvitsee pelkästään kertoa, mitkä sanat ovat toistensa eri taivutusmuotoja.

5.4. Sanaluokan induktion arviointi

SLI:n arvioinnista tekee hankalaa se, että tulokset ovat merkitsemättömiä klustereita. Tämä vuoksi niitä on vaikeaa verrata käsin merkatun korpuksen sanaluokkiin. Lisäksi tuotettujen klusterien lukumäärä voi poiketa käsin merkatun korpuksesta, minkä vuoksi induktiomenetelmä voi tuottaa eri tarkkuustason luokittelun, joka on silti järkevä. [77]

SLI:n toimivuutta voidaan kuitenkin arvioida esimerkiksi tarkastelemalla järjestelmän löytämän klusteroinnin ja asiantuntijan oikeaksi katsoman luokittelun välistä konditionaalista entropiaa. [28] Entropiapohjaista V-mittaa [83] on suositeltu tähän tarkoitukseen, sillä se on vähemmän riippuvainen löydettyjen ja todellisten klusterien lukumäärästä kuin monet muut menetelmät. [77]

V-mitta V_β mittaa esitetyn klusteroinnin K homogeenisyyttä h ja saantia c suhteessa oikeaan luokitteluun C. Täysin homogeenisessa klusteroinnissa yksi klusteri sisältää vain yhden luokan näytteitä, ja saanniltaan täydellisessä klusteroinnissa kaikki yhden luokan näytteet ovat samassa klusterissa. V-mitta yhdistää nämä kaksi mittaa lausekkeen (5.3) esittämällä tavalla. [83]

$$V_\beta = \frac{(1+\beta)hc}{\beta h + c} \tag{5.3}$$

$$h = \begin{cases} 1 & \text{jos } H(C|K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{jos } H(C|K) > 0 \end{cases}$$

$$c = \begin{cases} 1 & \text{jos } H(K|C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{jos } H(K|C) > 0 \end{cases}$$

$V_\beta =$ V-mitta.
$\beta =$ Saannin painotuskerroin.
$h =$ Homogenisyys.
$c =$ Saanti.
$C =$ Oikea luokittelu.
$K =$ Esitetty klusterointi.
$H =$ Entropiafunktio.

6. RATKAISUN KUVAUS

Tässä työssä toteutettiin ohjelmisto, joka poimii koneellisesti hyödynnettävää sanastoa merkkaamattomasta korpuksesta. Ratkaisu pohjautuu ei-parametriseen Bayesin menetelmään, joka mahdollistaa koostamista lukuun ottamatta kaikkien sanaston poiminnan vaiheiden lähes täysin ohjaamattoman toteutuksen.

Ratkaisu toteutettiin C++- ja Python-ohjelmointikielillä. Kaikki ratkaisun pääkomponenentit sekä valtaosa niiden hyödyntämästä taustakoodista tuotettiin osana tätä diplomityötä. Käytännössä ainoat ratkaisun osat, jotka eivät kuulu tämän diplomityön rajattuun skooppiin, ovat Boost-kirjaston todennäköisyyksien laskentaan liittyvä toiminnallisuus sekä C++- ja Python-ohjelmointikielten peruskirjastot.

Toteutetun ohjelmiston ohjaamaton osa, joka on sen keskeisin osa, koostuu neljästä pääosasta: sanaston kerääjän pääohjelma, sanojen erottelija, MI ja SLI. Lisäksi ohjelmistoon kuuluu erillinen työkalu sanaston koostamiseen.

6.1. Toteutusympäristö

Ratkaisun ohjaamaton osa toteuttiin C++-ohjelmointikielellä, ja siinä hyödynnettin Boost-kirjaston perustietorakenteita sekä sen todennäköisyysmoduulia. Boostin todennäköisyysmoduulia käytettiin alfa-, beta-, gamma- ja Poisson-jakautuneiden näytteiden generointiin. Muilta osin ratkaisun ohjaamaton osa oli kokonaan uutta koodia.

Sanaston koostaja toteutettin Python-ohjelmointikielellä. Pythonin integroitujen kirjastojen lisäksi ei käytetty muita kirjastoja. Sanaston koostajan käyttöliittymä toteutettiin Pythonilla yksinkertaisena komentokehotepohjaisena ratkaisuna.

Ohjelmisto suoritettiin vuonna 2013 hankitulla laitteistolla, jonka suorittimessa oli 6 ydintä, ja jossa oli keskusmuistia 16 gigatavua. Ohjelmisto oli toteutettu yksisäikeisenä, mutta järjestelmän moniprosessointitehoa hyödynnettiin ajamalla useita testejä eri kielillä ja asetuksilla rinnakkain.

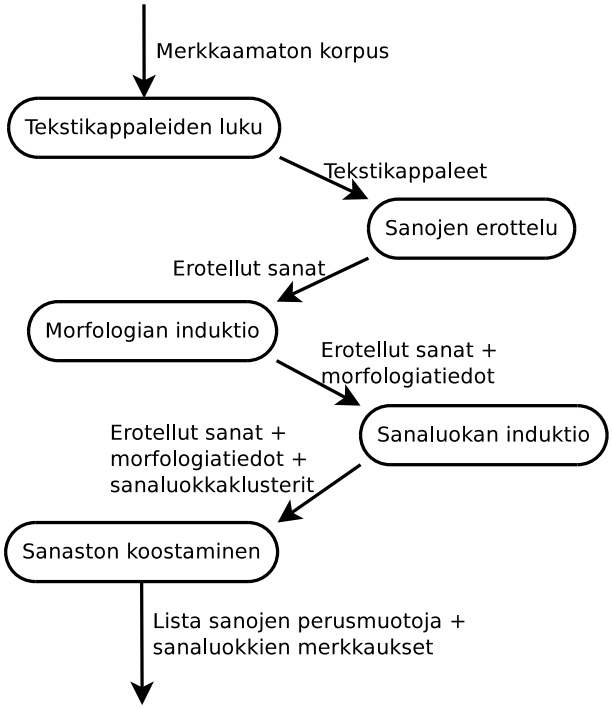
Tällä kokoonpanolla yksittäisten sanaston poimijan komponenttien testiajot kestivän kukin muutamia tunteja testistä riippuen. Koko sanaston poimijan ohjaamattoman osan testiajon annettiin iteroida noin kaksi päivää ennen tulosten keräämistä.

6.2. Toimintaperiaate

Järjestelmän korkeimman tason toimintaperiaate on, että sille annetaan syötteenä merkkaamaton korpus, jonka se prosessoi, ja josta se tuottaa koneellisesti hyödynnettävän sanaston. Kerätty sanasto sisältää listan perusmuotoisia sanoja sekä kunkin sanan sanaluokan.

Tarkemmalla tasolla datavuo kulkee niin, että sanojen erottelija ottaa syötteenä pääohjelman lukemia tekstikappaleita, joista se erottelee sanat. Erotellut sanat syötetään yksi kerrallaan MI:hin, joka johtaa niille morfologiasegmentoinnin. Tämän jälkeen sanat morfologiatietoineen syötetään yksi kerrallaan SLI:hin, joka johtaa niille sanaluokat. Kun näitä neljää komponenttia on ajettu niin kauan, että ratkaisu on oletettavasti löytynyt, niiden tuottama tieto annetaan erilliselle sanaston koostajalle, joka koostaa asiantuntijan avustuksella niistä koneellisesti hyödynnettävän sanaston. Tätä on havainnollistettu kuvassa 6.1.

Järjestelmän ohjaamaton osa pohjautuu ei-parametrisiin Bayesin menetelmiin, ja se



Kuva 6.1: Ohjelmiston datavuo. Tekstiä luetaan tiedostosta yksi kappale kerrallaan, ja jokaisesta kappaleesta erotellaan sanat. Erotellut sanat syötetään MI:lle, ja johdetut morfologiasegmentoidut sanat syötetään edelleen SLI:lle. Lopuksi morfologiasegmentoidut sanat ja niiden sanaluokat syötetään koostamistyökalulle, joka generoi asiantuntijan avustuksella sanoille niiden perusmuodot ja antaa sanaluokille käyttösovelluksen vaatimat merkkaukset.

toimii lähes kokonaan ilman ohjausta tai esitietoa. Osa järjestelmän parametreista, kuten merkistökohtaiset sanojen pituuksien odotusarvot, jätettiin käsin asetettaviksi toteutuksen yksinkertaisuuden vuoksi, mutta teoriassa nekin oltaisiin voitu johtaa Bayesin induktion yhteydessä.

Ratkaisun ohjaamattoman osan korkean tason toimintaperiaate on se, että sanavälejä, sanojen päätteitä ja sanaluokkia näytteistetään satunnaisesti. Aluksi näytteistys on täysin satunnaista, sillä esitietoa ei ole saatavilla, mutta näytteistyksen edetessä järjestelmä alkaa suosia aikaisemmin havaittuja sanavälien paikkoja, sanojen päätteitä ja sanaluokkia PYP:n klusterointiominaisuuden vuoksi.

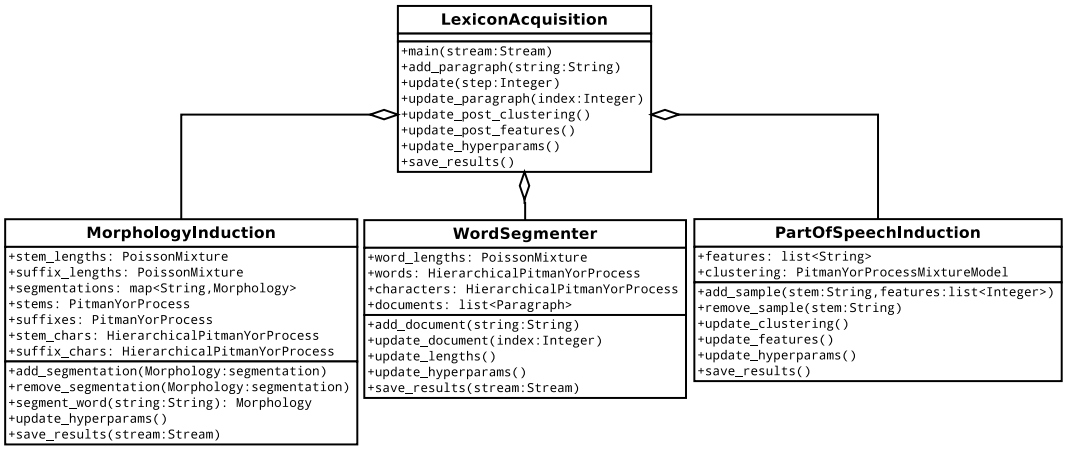
Käytännössä sanojen erottelu, MI ja SLI toteutetaan yhden suuren Gibbsin näytteistyssilmukan avulla. Tässä silmukassa tekstikappaleille näytteistetään yksi kerrallaan sanojen erottelun, MI:n ja SLI:n tiedot niin, että näytteistettävän kappaleen mahdollinen aikaisempi tieto poistetaan prosessista ja sille lasketaan uudet tiedot. Sanojen erottelijan löytämiä sanoja lisätään hierarkkisesti MI- ja SLI-prosesseihin sitä mukaa, kun niitä löydetään, ja poistetaan niistä kun sanojen erottelu näytteistetään uudelleen. Perustoimintaperiaate on siis sama, kuin kappalessa 2.3.2 esitetyssä Gibbsin näytteistyksen havaintoesimerkissä.

Järjestelmän ohjaamaton osa tukee myös asiantuntijoiden merkkaaman sanojen erottelu- ja morfologiatiedon hyödyntämistä, mikä käytännössä toteutettiin niin, että asiantuntijat voivat käsin merkata kappaleiden segmentointeja erilliseen tekstitiedostoon. Asiantuntijoiden merkkausten hyödyntäminen on toteutettu yksinkertaisesti niin, että opetusnäytteitä ei näytteistetä koskaan uudelleen. Tätä ominaisuutta ei kuitenkaan

käsitellä tässä työssä sen syvällisemmin, sillä painopiste on ohjaamattomassa sanaston poiminnassa. Käytännön sovelluksissa asiantuntijoiden merkkauksista voi kuitenkin olla hyötyä, sillä niillä voidaan nopeuttaa induktioprosessia tai ohjata sitä käytännön sovelluksen vaatiman konvention suuntaan.

6.3. Arkkitehtuurikuvaus

Ohjelmisto koostuu pääohjelmaa edustavasta luokasta sekä kolmesta muusta sen ohjaamasta luokasta, kuten on esitetty kuvan 6.2 korkean tason luokkadiagrammissa.



Kuva 6.2: Ohjelmisto koostuu neljästä osasta: pääohjelma sekä sen ohjaamat sanojen erottelu, MI ja SLI.

Sanaston poimijan pääohjelman tehtävä on lukea syötteitä tiedostosta ja ohjata muita luokkia niiden käsittelyssä. Muista luokista kukin suorittaa yksittäisen tehtävän, eli joko sanojen erottelun, MI:n tai SLI:n.

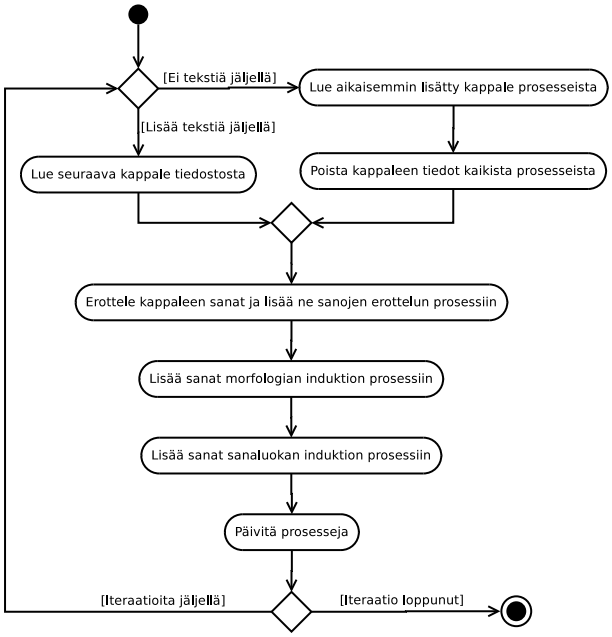
6.3.1. Sanaston poiminta

Sovelluksen pääohjelma lukee ensimmäisen iteraation aikana tekstiä kappale kerrallaan tiedostosta ja syöttää sen sanojen erottelijalle, MI:lle ja SLI:lle. Tietyin väliajoin se päivittää myös näiden prosessien parametreja ja kirjoittaa tulokset tiedostoon.

Kun ensimmäinen iteraatio on suoritettu, alkaa sovellus prosessoida vanhoja näytteitä uudelleen. Tällöin vanha tekstikappale luetaan muistista ja siitä aikaisemmin erotellut sanat poistetaan sanojen erottelu-, MI- ja SLI-prosesseista. Tämän jälkeen kappale prosessoidaan uudelleen aivan kuin se lisättäisiin viimeisenä prosessiin.

Sanaston poiminnan kulku on esitetty graafisesti kuvan 6.3 aktiviteettidiagrammissa.

Tekstikappaleiden päivityksen lisäksi pääohjelma ohjasti MI:tä ja SLI:tä päivittämään prosessiensa tilat tietyin väliajoin sekä päivitti niiden parametreja. MI:n tilan päivitys käsitti vartaloiden ja päätteiden pituuksien muuttamista MI-siirroilla. SLI:n tilan päivitys käsitti vartaloiden ja klusterien näytteistämisen uudelleen. Testeissä MI:n ja SLI:n tilat laskettiin uudelleen 1000 tekstikappaleen segmentoinnin välein, ja kaikkien prosessien parametrit päivitettiin aina uudelleen heti tämän jälkeen.



Kuva 6.3: Sanaston poiminnan aktiviteettidiagrammi. Tekstiä luetaan aluksi kappale kerrallaan tiedostosta, ja kappaleet segmentoidaan ja lisätään sanaston poiminnan, MI:n ja SLI:n prosesseihin. Tämän jälkeen kappaleita prosessoidaan uudelleen yksi kerrallaan. Uudelleen prosessoitava kappale poistetaan aluksi kaikista näistä kolmesta prosessista, minkä jälkeen se segmentoidaan uudelleen ja sen uusi segmentointi lisätään takaisin kaikkiin kolmeen prosessiin.

6.3.2. Sanojen erottelu

Sanojen erotteluun käytettiin muunnelmaa Mochihashin algoritmin [52] bigrammiversiosta, joka paloittelee syötteen sanoihin aloittaen merkkijonon lopusta.

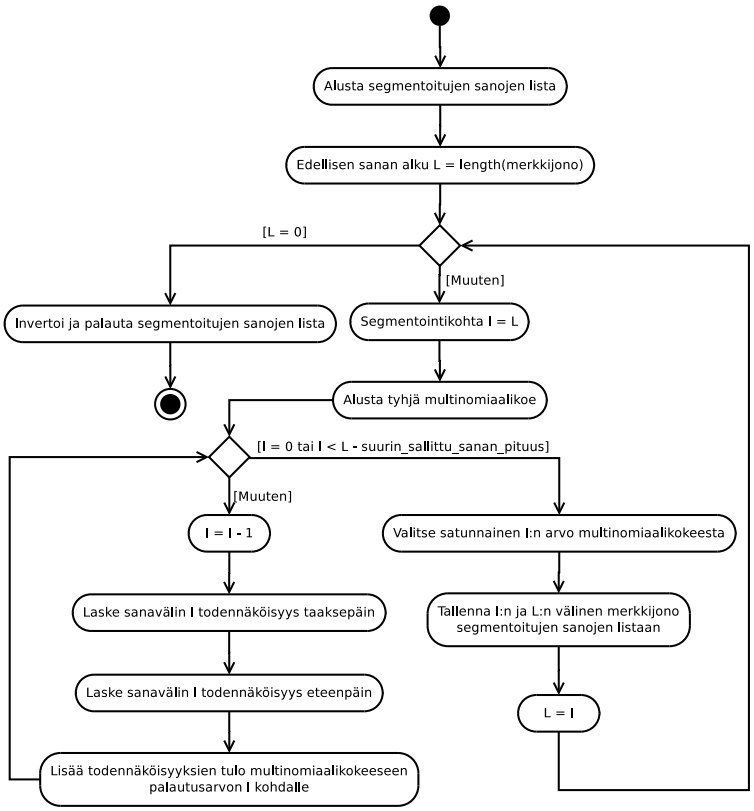
Segmentointitehtävän alkaessa alustetaan tyhjä lista eroteltuja sanoja varten sekä asetetaan laskurin L arvo vastaamaan merkkijonon loppukohtaan. Sanoja erotellaan, kunnes $L = 0$ eli erotellut sanat kattavat koko merkkijonon.

Yksittäinen sana voidaan periaatteessa segmentoida niin, että lasketaan seuraavan sanavälin todennäköisyys jokaiselle mahdolliselle kohdalle välillä $[0, L - 1]$. Näistä eri sanaväleistä valitaan yksi todennäköisyydellä painotetun satunnaisvalinnan eli multinomialikokeen avulla.

Käytännössä kuitenkin sanan pituus rajataan johonkin maksimiarvoon, jota enempää segmentointivaihtoehtoja ei tarkastella. Tämä rajoitus tarvitaan, jotta algoritmi ei hidastuisi kohtuuttomasti. Tämä arvo asetettiin nopean testattavuuden vuoksi tässä työssä 15:een englannilla ja japanilla. Suomella sanan pituutena käytettiin 20:tä. Englannille ja etenkin suomelle nämä arvot ovat selvästi liian pieniä. Käytännössä arvon muuttaminen suuremmaksi on kuitenkin triviaalia, ja yksittäisen ajon kesto voi olla vielä siedettävä myös 30 merkin pituiset sanat sallittaessa. Toteutusympäristössä suoritetussa testiajossa esimerkiksi 500 tekstikappaleen sanojen erottelun yksi iteraatio kesti 20:n merkin maksimisananpituudella noin 1 minuutin ja 30:n merkin maksimisananpituudella noin 2.5 minuuttia.

Sanojen erottelun toimintaa on havainnollistettu kuvan 6.4 aktiviteettidiagrammissa.

Seuraavissa osioissa kerrotaan, miten segmentoinnin todennäköisyys laskettiin. Mochihashin menetelmän tavoin, yksittäisen sanan todennäköisyys laskettiin merkki-



Kuva 6.4: Merkkijonon segmentoinnin aktiviteettidiagrammi. Sanoja eroteltiin yksitellen lopusta alkaen, kunnes enempää sanoja ei ollut jäljellä. Yksittäinen sana segmentoitiin laskemalla sen kaikkien kandidaattisegmentointien todennäköisyydet ja valitsemalla niistä yksi probabilistisesti.

jonotason HPYP:stä, ja sanojen bigrammien todennäköisyydet laskettiin tarkastellusta sanasta sekä eteen- että taaksepäin. Lopullinen todennäköisyys saatiin yhdistämällä nämä kolme todennäköisyyttä.

6.3.2.1. Sanan todennäköisyys

Sanan todennäköisyys laskettiin merkkijonotason HPYP:n, sanatason HPYP:n ja morfologiasegmentointitodennäköisyyden avulla. Morfologiatiedon hyödyntämistä lukuun ottamatta ratkaisu oli käytännössä identtinen Mochihashin [52] menetelmän kanssa.

Aluksi laskettiin tarkastellun merkkijonon todennäköisyys merkkijonotason HPYP:stä. Merkkijonoprosessi sisälsi kaikki aikaisemmin segmentoitujen sanojen merkit 15-grammeina. Merkkijonon yksittäisten 15-grammien todennäköisyydet laskettiin HPYP:n todennäköisyyden kaavalla, joka annettiin taustan lausekkeessa 2.8. Koko merkkijonon todennäköisyys laskettiin Mochihashin menetelmän muunnelmalla, jonka kaava on annettu lausekkeessa (6.1).

$$P_{str}(c_1...c_k) = \frac{P(c_1...c_k, k|\Theta_C)P_{corr}(k|c_1...c_k)}{P(k, \Theta_C)} \tag{6.1}$$

$$P_{str}(c_1...c_k, k|\Theta_C) = \prod_{i=1}^{k} P(c_i|c_1...c_{i-1})$$

$P_{str}(c_1...c_k) =$ Koko merkkijonon todennäköisyys.

$P_{str}(c_1...c_k, k|\Theta_C) =$ Merkkitason todennäköisyys.

$P_{corr}(k|c_1...c_k) =$ Poisson-korjaus sanan pituudelle.

$P(k, \Theta_C) =$ Sanan pituuden todennäköisyys.

$k =$ Merkkijonon pituus.

$\Theta_C =$ Merkkien HPYP:n tila.

Todennäköisyyden laskemisessa käytetty sanan pituuden Poisson-korjaus laskettiin sanan merkistön perusteella. Jokainen sanan merkki luokiteltiin Unicode-merkistön avulla johonkin merkkiluokkaan, joille oli kaikille käsin approksimoitu jokin sanan pituuden odotusarvo. Kullekin merkkijonossa esiintyvälle merkkiluokalle laskettiin Poisson-todennäköisyys sanan pituuden ja merkkiluokan pituuden odotusarvon avulla. Näiden todennäköisyyksien tuloa käytettiin Poisson-korjaukseen sellaisenaan. Tässä toteutuksessa käytetyt merkkiluokat olivat länsimaalaiset merkit, välimerkit, hiragana, katakana, kanji ja kaikki muut Unicode-merkistön merkit.

Kaavassa käytetty sanan pituuden todennäköisyys laskettiin näytteistämällä merkkiprosessista suuri määrä satunnaisia sanoja ja laskemalla pituuksien todennäköisyys niiden jakaumasta. Pituuksien todennäköisyyksien uudelleennäytteistys suoritettiin määräajoin prosessien parametrien päivityksen yhteydessä, sillä niiden jatkuva päivitys ei olisi ollut käytännöllistä merkkiprosessin sisällön jatkuvan muuttumisen takia.

Merkkijonotason todennäköisyyden jälkeen laskettiin sanatason bigrammien todennäköisyys samanlaisesta HPYP:stä, johon oli tallennettu merkkien sijaan sanoja.

6.3.2.2. Sanojen bigrammin todennäköisyys

Sanatason bigrammin todennäköisyys laskettiin suoraan sanaprosessin HPYP:n todennäköisyydestä. Sanaprosessi sisälsi aikaisemmin johdettujen sanojen bigrammeja samaan tapaan kuin merkkijonoprosessikin sisälsi n-grammeja, ja bigrammin todennäköisyys laskettiin suoraan HPYP:n todennäköisyydestä. HPYP:n perustodennäköisyytenä kuitenkin käytettiin vakion sijasta edellä laskettua merkkijonon todennäköisyyttä. Tämä on esitetty matemaattisesti lausekkeessa (6.2).

$$P_{bigram}(w_1, w_2) = P_{HPYP}(w_1, w_2|\Theta_W, P_{str}(w_1)) \tag{6.2}$$

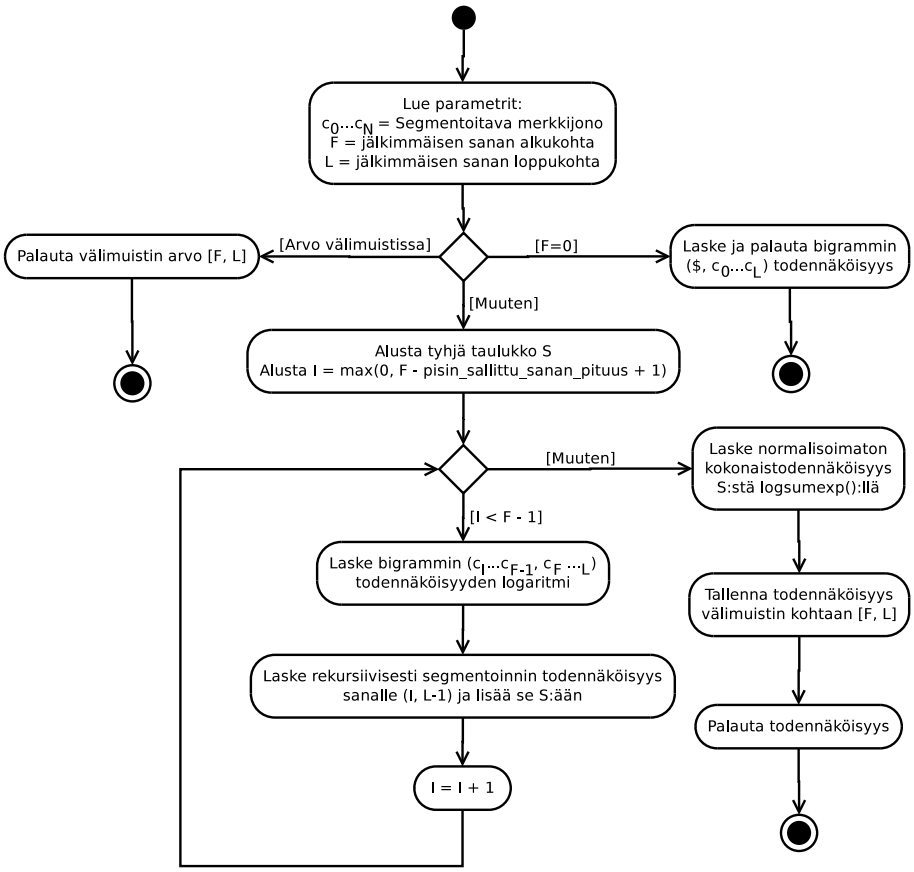
$w_1 =$ Edellinen sana.

$w_2 =$ Jälkimmäinen sana.

$\Theta_W =$ Sanojen HPYP:n tila.

6.3.2.3. Sanan todennäköisyys taaksepäin

Sanan todennäköisyys taaksepäin laskettiin Mochihashin [52] dynaamisella ohjelmointimenetelmällä. Algoritmin aktiviteettidiagrammi on esitetty kuvassa 6.5.



Kuva 6.5: Sanan logaritmisen todennäköisyyden taaksepäin laskemisen aktiviteettidiagrammi.

Algoritmin toiminta perustuu siihen, että kaikkien hypotetisoitua sanaa edeltävien segmentointien todennäköisyydet lasketaan rekursiivisesti ja summataan logaritmisessa avaruudessa. Tämä summa vastaa edeltävien segmentointien normalisoimatonta logaritmista kokonaistodennäköisyyttä, joten sitä voidaan käyttää suoraan toisen sanojen erottelijan ylimmän tason silmukan multinomiaalikokeen painokertoimena.

Koska edes 64-bittisten liukulukujen tarkkuus ei riittänyt todennäköisyyksien esittämiseen pitkiä kappaleita segmentoitaessa, laskettiin todennäköisyydet logaritmisessa avaruudessa logsumexp-tempulla, kuten Mochihashi suositteli.

6.3.2.4. Sanan todennäköisyys eteenpäin

Sanan todennäköisyys eteenpäin laskettiin hypotetisoidun sanan ja sitä seuraavan, jo segmentoidun, sanan bigrammin todennäköisyydestä HPYP:ssä. Tämä menetelmä vastaa siis suoraan Mochihashin tapaa laskea sanan todennäköisyys eteenpäin.

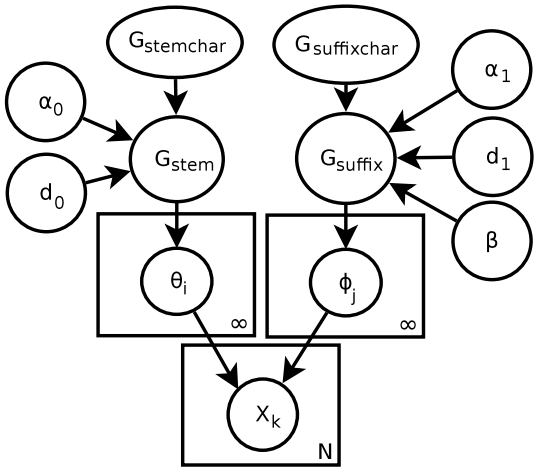
Mochihashin menetelmästä eroten, jos kyseessä oli kahden ei-terminaalin sanan segmentointiongelma, sanatason todennäköisyyttä muokattiin vielä lisäksi morfologiaprosessin tietojen perusteella. Tällöin todennäköisyyttä pienennettiin lausekkeen (6.3) mukaisesti silloin, kun ensimmäinen sana oli morfologiasegmentoinnin todennäköisyysmallin mukaan todennäköisesti vartalo ja toinen pääte.

$$
\begin{aligned}
P_{fwd}(w_1, w_2) &= \begin{cases} P_{bigram}(w_1, w_2) P_s(w_2) P_m(w_1 w_2) & \text{jos } w_1 \neq \$ \text{ ja } w_2 \neq \$ \\ P_{bigram}(w_1, w_2) & \text{muuten} \end{cases} \\
P_s(w) &= 1 - P_{suffix}(w) \\
P_m(c_0 ... c_k) &= 1 - \max_{i=1...k} \left[P_{morphseg}(c_0 ... c_i, c_{i+1} ... c_k) \right] \\
P_{suffix}(s) &= \text{Morfologiaprosessin vartalon todennäköisyys.} \\
P_{morphseg}(s, t) &= \text{Morfologiasegmentoinnin todennäköisyys.} \\
w_1 &= \text{Edellinen sana.} \\
w_2 &= \text{Jälkimmäinen sana.} \\
\$ &= \text{Kappaleen alkua tai loppua vastaava terminaalisana.}
\end{aligned}
\tag{6.3}
$$

Tämä todennäköisyys kertoo periaatteessa sen, kuinka usein kyseiset sanat ovat esiintyneet peräkkäin, lisäksi pienentäen hieman yksittäisten morfeemien todennäköisyyttä segmentoitua erillisiksi sanoiksi. Tätä todennäköisyyttä käytettiin sanojen erottelijan multinomiaalikokeen toisena painokertoimena.

6.3.3. Morfologian induktio

MI:ssä tyydyttiin käyttämään kuvassa 6.6 esitettyä yksinkertaista generatiivista mallia, jossa sana koostuu yksinkertaisesta vartalosta ja mahdollisesti tyhjästä päätteestä. Vartalo ja pääte peräkkäin kirjoitettuna muodostavat suoraan taivutetun sanan tässä mallissa.

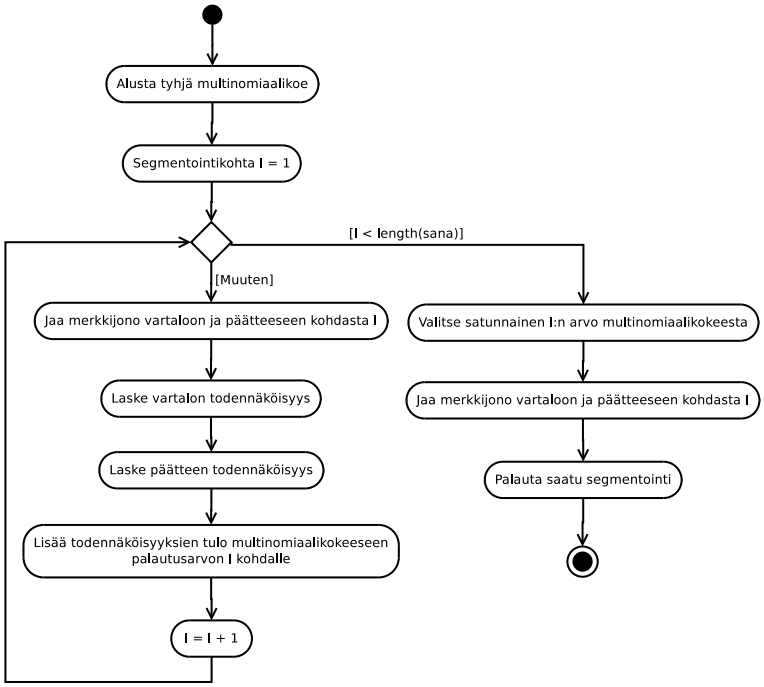


Kuva 6.6: Morfologiasegmentoinnin generatiivinen malli. Jokaisella havaitulle sanalle x_k näytteistetään satunnaisesti prosessista G_{stem} yksi äärettömän monesta vartalosta θ_i. Vartalon merkit puolestaan näytteistetään satunnaisesti prosessista $G_{stemchar}$. Vastaavasti, päätteet ϕ_j ja niiden merkit ovat peräisin prosesseista G_{suffix} ja $G_{suffixchar}$. Hyperparametri β määrää tyhjän päätteen todennäköisyyden.

Generatiivisen mallin ratkaiseva induktio toteutettiin kahdella PYP:llä, joista ensimmäiseen tallennettiin sanojen vartaloita ja toiseen päätteitä. Lisäksi vartaloille ja päätteille oli merkkitason HPYP:t, joita käytettiin PYP:ien pohjatodennäköisyyksinä sa-

maan tapaan kuin sanojen erottelussakin. Näistä prosesseista saatuja todennäköisyyksiä käytettiin sellaisenaan eri morfologiasegmentointien todennäköisyyksinä.

Jokaisen sanan morfologinen segmentointi ratkaistiin yksitellen lohkotetulla Gibbsin näytteistyksellä. Sanan kaikkien eri segmentointien todennäköisyydet laskettiin liu'uttamalla segmentointikohtaa sanan yli ja laskemalla todennäköisyydet PYP:istä ja HPYP:istä, ja näistä segmentoinneista valittiin yksi multinomiaalikokeella kuvan 6.7 aktiviteettidiagrammin mukaisesti.



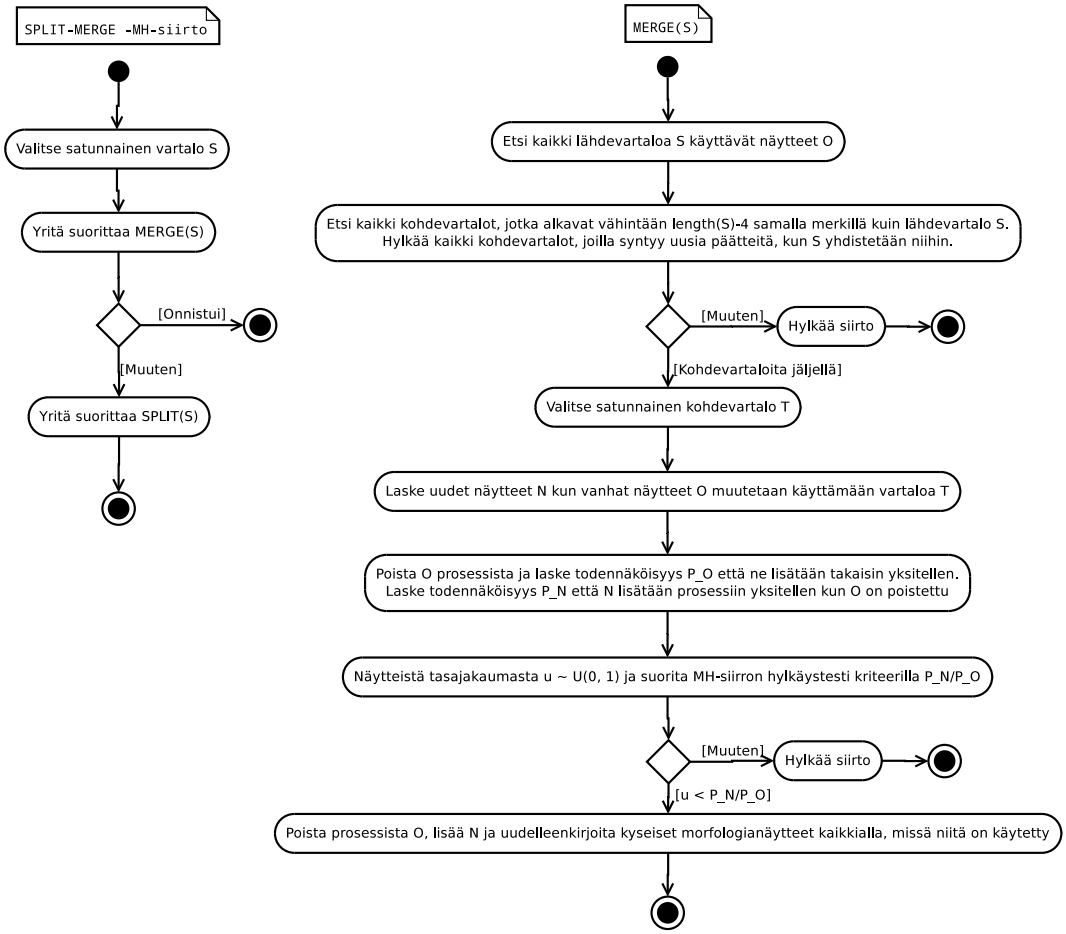
Kuva 6.7: Morfologiasegmentoinnin näytteistämisen aktiviteettidiagrammi.

Koska Gibbsin näytteistys vaikutti löytävän huonosti pidempiä päätteitä, sen rinnalla suoritettiin sopivin väliajoin kaksi MH-siirtoa, jotka yhdistivät ja hajottivat Gibbsin näytteistyksen löytämiä vartaloita. MH-siirtojen toiminta on esitetty kuvien 6.8 ja 6.9 aktiviteettidiagrammeissa.

Käytännössä MH-siirtojen suorittamat operaatiot joko pidensivät tai lyhensivät vartaloita, siirtäen merkkejä joko sanojen päätteistä niiden vartaloihin tai niiden vartaloista päätteisiin. Koska nämä MH-siirrot kohdistuivat morfologiaprosessin yhden vartalon kaikkiin näytteisiin, odotettiin niiden tehokkaasti löytävän samojen sanojen eri taivutusmuodoille samat vartalot ja hajottavan vartaloita, joissa on useita eri sanoja.

Aina kun tekstikappaleesta erotellulle sanalle näytteistettiin segmentointi, tämän segmentoinnin vartalo ja pääte sekä niiden merkit lisättiin induktioon kuuluviin prosesseihin. Vastaavasti, kun tekstikappale segmentoitiin uudelleen, sen kaikki morfologiatieto poistettiin prosesseista. Nämä kaksi operaatioita on esitetty kuvassa 6.10.

Numeromerkeillä alkaville sanoille ei johdettu tässä ratkaisussa ollenkaan morfologiaa, sillä lukusanoja ei haluttu poimia osaksi sanastoa. Lisäksi tyhjää päätettä ei lisätty päätteiden PYP:hen, sillä se heikensi todennäköisyysmallin toimintaa. Sen sijaan, tyhjälle päätteelle käytettiin erillistä käsin asetettua todennäköisyyttä.



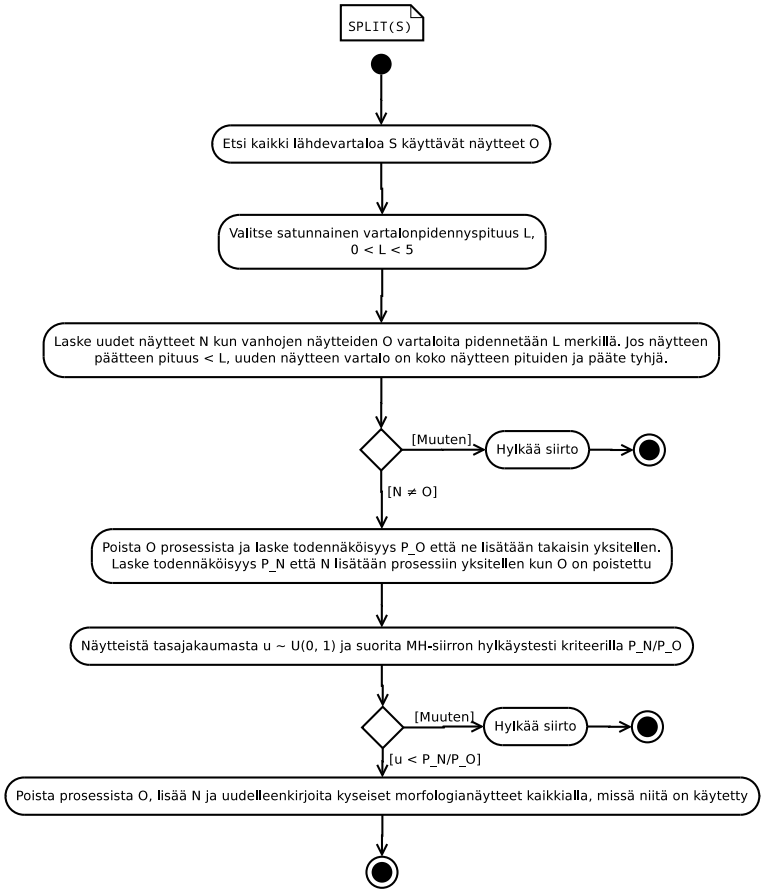
Kuva 6.8: MI:n MH-siirtojen kutsuminen sekä MH-siirto MERGE, joka yhdistää pidemmän vartalon lyhyempään, pidentäen näin sen päätteiden pituuksia.

6.3.3.1. Segmentoinnin todennäköisyys

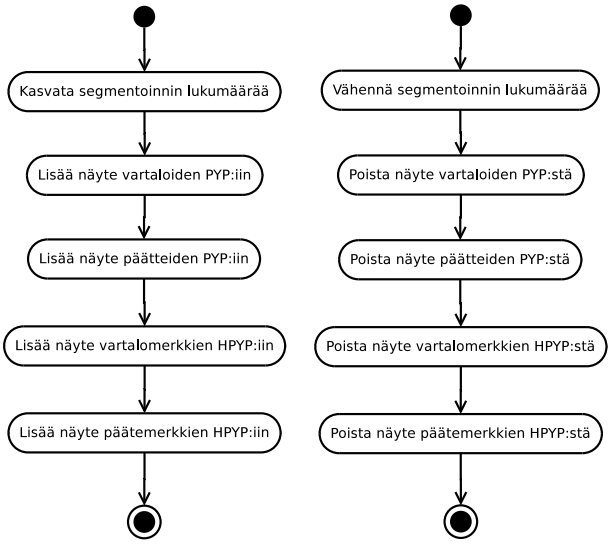
Hypotetisoidun morfologisen segmentoinnin todennäköisyyttä estimoitiin laskemalla erikseen sanan vartalon ja päätteen todennäköisyydet. Koko segmentoinnin todennäköisyys oli näiden tulo, kuten on esitetty lausekkeessa (6.4).

$$P_{morphseg}(s,t) = P_{stem}(t)P_{suffix}(s,t) \tag{6.4}$$
$$s = \text{Hypotetisoitu vartalo.}$$
$$t = \text{Hypotetisoitu pääte.}$$

Vartalon todennäköisyys laskettiin aikaisemmin johdettujen vartaloiden PYP:n ja näiden vartaloiden merkeistä kerätyn HPYP:n avulla lausekkeen (6.5) mukaisesti. Vartaloiden PYP sisälsi yhden klusterin kullekin uniikille vartalon merkkijonolle. Vartaloiden merkkien HPYP taaskin oli samanlainen merkkien prosessi kuin mitä käytettiin sanojen erottelussa.



Kuva 6.9: MI:n käyttämä MH-siirto SPLIT, joka hajottaa yhden vartalon useaksi pidemmäksi vartaloksi, lyhentäen näin sen päätteiden pituuksia.



Kuva 6.10: Morfologianäytteiden lisääminen MI-prosessiin ja poistaminen siitä.

$$
\begin{aligned}
P_{stem}(s) &= P_{PYP}(s|s_{stem}, P_{stemchar}(s)) \qquad (6.5)\\
P_{stemchar}(s) &= P_{string}(s, c_{stem})\\
s &= \text{Hypotetisoitu pääte.}\\
s_{stem} &= \text{Vartaloprosessissa olevat vartalot.}\\
c_{stem} &= \text{Vartalomerkkiprosessissa olevat merkit.}
\end{aligned}
$$

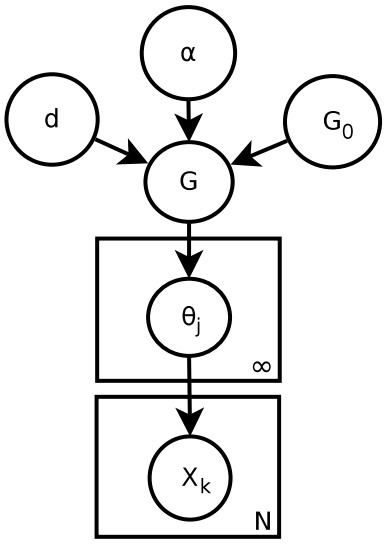
Päätteen todennäköisyys laskettiin myös PYP:n ja HPYP:n avulla. Suurin ero vartalon todennäköisyyden laskemiseen oli kuitenkin, että tyhjälle päätteelle käytettiin omaa käsin asetettua todennäköisyyttä PYP:n klusterin sijaan. Tämä tehtiin sen vuoksi, että tyhjä pääte on tyypillisesti niin yleinen verrattuna muihin päätteisiin, että PYP alkaa helposti suosimaan sitä liikaa, jos sillä on oma klusterinsa.

Päätteen todennäköisyyden laskukaava on annettu lausekkeessa (6.6).

$$
\begin{aligned}
P_{suffix}(t) &= \begin{cases} \gamma P_{PYP}(t|t_{stem}, P_{suffixchar}(t)) & \text{jos } \mathrm{length}(t) > 0 \\ (1-\gamma) P_{suffixchar}(t) & \text{jos } \mathrm{length}(t) = 0 \end{cases} \\
P_{suffixchar}(t) &= P_{string}(t|c_{suffix}) \\
t &= \text{Hypotetisoitu pääte.} \\
t_{suffix} &= \text{Pääteprosessissa olevat päätteet.} \\
c_{suffix} &= \text{Päätemerkkiprosessissa olevat merkit.} \\
\gamma &= \text{Käsin asetettu päätteen todennäköisyys.}
\end{aligned}
\tag{6.6}
$$

6.3.4. Sanaluokan induktio

Sanaluokat johdettiin MI:n tuloksista. Jokaisen uniikin vartalon oletettiin vastaavan yhtä uniikkia sanaa, ja jokaisen sanan oletettiin kuuluvan täsmälleen yhteen äärettömän monesta sanaluokasta. Tämän generatiivinen malli on esitetty kuvassa . Se on PYP:n vastine taustassa esitetylle DP:n mikstuurimallille, johon erona ovat mallin käyttämät hyperparametrit.

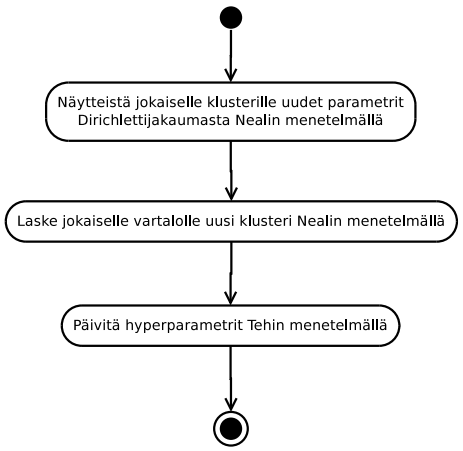


Kuva 6.11: SLI:n generatiivinen malli. Prosessissa on N kappaletta vartaloita X_k, joista jokainen kuuluu yhteen mahdollisesti äärettömän monesta sanaluokasta j. Jokaisella sanaluokalla j on tyypillisten päätteiden mikstuuri θ_j, joka on generoitu prosessista G, jolla on hyperparametrit d ja α. G:n perusjakaumana toimii tasajakauma G_0.

SLI esitettiin siis yksinkertaisena klusterointiongelmana, jossa morfologiaprosessista saadut vartalot klusteroitiin niihin liittyvien päätteiden perusteella. Vartaloiden piirteinä käytettiin pelkästään niihin yhdistettyjen päätteiden normalisoituja lukumääriä,

eli klusterointi pohjautui pelkästään ortografiaan, eikä distributionaalista tietoa hyödynnetty.

Klusteroiti toteutettiin kuvan 6.12 aktiviteettidiagrammin mukaisesti käyttäen perinteisiä mikstuurimallin induktiomenetelmiä.



Kuva 6.12: SLI toteutettiin suoraan PYP:n mikstuurimallin induktiomenetelmillä.

Mikstuurimallin induktiomenetelmänä käytettiin PYP:lle muunnettua Nealin kahdeksatta algoritmiä [46, s. 262]. Tämä algoritmi päivittää aluksi yksitellen jokaisen klusterin mikstuuriparametrit näytteistämällä kyseisen klusterin näytteiden ominaisuuksien määrillä painotettua Dirichlettijakaumaa. Tämän jälkeen algoritmi iteroi kaikkien näytteiden lävitse yksitellen. Näytteet poistetaan klusterista yksitellen, niille valitaan uudet klusterit kaikkien muiden klusterien ja näytteiden tilan perusteella, ja ne lisätään valittuihin klustereihin.

PYP:n hyperparametrit johdettiin automaattisesti Tehin [50] menetelmällä, joka on Escobarin ja Westin [47, s. 585] menetelmän muunnelma PYP:lle. Hyperparametrien induktion ansiosta SLI toimi täysin ilman esitietoa tai asiantuntijan ohjausta.

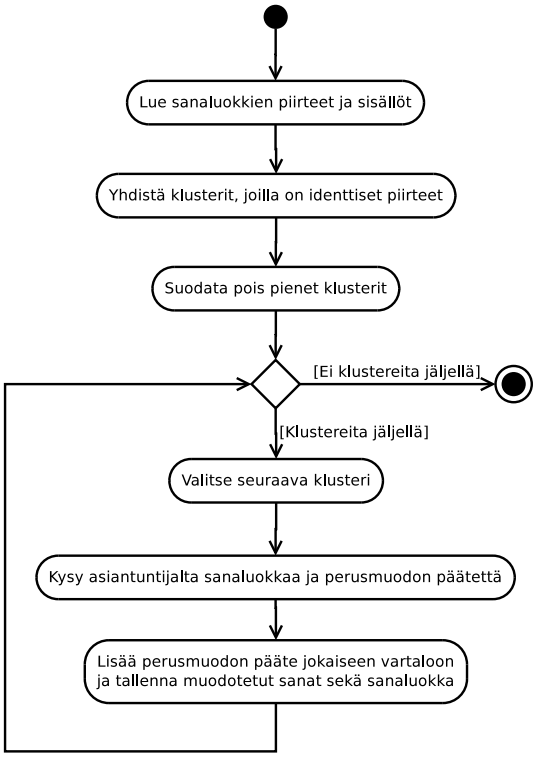
Klusteroinnin helpottamiseksi vartaloiden klusterointipiirteinä käytettiin 99:ää koko morfologiaprosessissa yleisimmin esiintynyttä päätettä. Lisäksi käytettiin yhtä piirrettä sanoille, joilla oli pelkästään tyhjä pääte. Käytettyjen ominaisuuksien listaa muokattiin dynaamisesti määräajoin induktion yhteydessä, sillä korpuksen yleisimpiä päätteitä ei luonnollisesti tunnettu induktion alkuvaiheessa.

6.3.5. Sanaston koostaminen

Sanaston koostaminen toteutettiin ohjatusti erillisellä työkalulla, jonka toimintaa on havainnollistettu kuvan 6.13 aktiviteettidiagrammissa.

Ohjelmiston ohjaamattoman osan tuottamat sanat sekä niiden morfologia ja sanaluokkien klusterointi luettiin aluksi tiedostosta ja suodatettiin automaattisesti. Suodatuksen ensimmäisessä vaiheessa klusterit, joiden piirteet olivat täysin samat, yhdistettiin keskenään. Tämän jälkeen jäljelle jäävistä klustereista poistettiin ne, joissa oli hyvin vähän sanoja, jotta asiantuntijan työmäärää voitaisiin vähentää.

Seuraavassa vaiheessa työkalu valitsi jokaisesta klusterista yleisimmän päätteen ja yleisimmän sen omaavan sanan. Asiantuntijalta pyydettiin kullekin klusterille tekstimuotoinen sanaluokan merkkaus sekä perusmuodon pääte. Näiden yksinkertaisten sääntöjen perusteella kaikille klusterin sanoille johdettiin perusmuodot ja sanaluokan merkkaus.



Kuva 6.13: Sanaston koostamisen aktiviteettidiagrammi.

7. RATKAISUN TESTAAMINEN JA MITTAUSTULOKSET

Tässä osiossa arviodaan, miten hyvin ratkaisu saavutti sille asetetut vaatimukset.

Aluksi tarkastellaan ratkaisun toiminnallisia ominaisuuksia, käytännöllisyyttä ja laajennettavuutta.

Tämän jälkeen esitellään testaussuunnitelma, jonka avulla järjestelmän suorituskykyä voidaan arvioida.

Testaussuunnitelman pohjalta tehtyjen testien tulokset esitellään tämän jälkeen.

Lopuksi pohditaan, mitä tulokset tarkoittavat käytännössä, ja niitä verrataan uusimpien vastaavien menetelmien tuloksiin.

7.1. Toiminnallisuus

Yksi tämän työn keskeisimmistä tavoitteista oli, että sanaston poiminta vaatisi mahdollisimman vähän asiantuntijan avustusta. Koska toteutettu järjestelmä oli lähes kokonaan automaattinen sanaston koostamista lukuun ottamatta, ja sanaston koostamissovelluksen käyttäminen vaati vain joitakin minuutteja aikaa asiantuntijalta, voidaan ratkaisun toiminnallisuuteen olla periaatteessa tyytyväisiä.

Järjestelmän toiminnallisuudelle asetettiin myös vaatimus, jonka mukaan sen piti pystyä poimimaan merkkaamattomasta korpuksesta sanojen perusmuodot sekä sanaluokat. Työssä toteutettu ratkaisu tuottikin näitä tietoja, sekä niiden lisäksi myös antoi mahdollisuuden saada yksinkertaista tietoa sanojen morfologiasta. Vaatimukset siis periaatteessa täytettiin tässäkin mielessä.

Järjestelmän haluttiin olevan monikielinen tai ainakin laajennettavissa sellaiseksi. Toteutettua ratkaisua pystyttiin soveltamaan kolmeen toisistaan huomattavan paljon poikkeavaan kieleen, joten ainakin perusta monen kielen tukemiseen pitäisi olla olemassa. Lisäksi ratkaisu tarjoaa useita selkeitä edellytyksiä monikielisyyden parantamiseen tulevassa kehityksessä.

Tarkkuutensa puolesta ratkaisun toiminta oli kuitenkin hyvin heikkoa. Tällaisenaan se ei vielä yksistään ole toimiva ratkaisu täysin ohjaamattomaan sanaston poimintaan, mutta sitä voidaan silti käyttää yhtenä työkaluna sanaston poiminnassa. Koska ratkaisu pystyy hyödyntämään asiantuntijan merkkaamaa tietoa, voidaan sitä käyttää esimerkiksi niin, että asiantuntija poimii oikeita tietoja sen edellisestä ajosta ja syöttää ne seuraavalle. Tällöin tuloksia voitaisiin parantaa ja uusia sanoja löytää inkrementaalisesti.

Kehyksenä käytetty Bayesin menetelmä tarjoaa paljon haluttuja laajennusmahdollisuuksia, joilla esimerkiksi tuloksia voitaisiin parantaa ja monikielisyyden tukea lisätä. Koska ratkaisun koko ohjaamaton osa käyttää samaa kehystä, se on rakenteeltaan varsin elegantti, ja sen aliosia voidaan helposti sulauttaa yhteen tai laajentaa tulevassa kehityksessä.

7.2. Testaussuunnitelma

Sanaston poimijaa kokonaisuudessaan testattiin aluksi käyttämällä aineistona suomalaisen Turku Dependency Treebankin tekstikatkelmia. Katkelmien merkkauksista kerättiin kaikkien niissä esiintyvien sanojen perusmuodot sekä sanaluokat. Tämän jälkeen katkelmat syötettiin ilman merkkauksia sanaston poimijalle. Johdetut tiedot

koostettiin sanaston koostajalla, ja asiantuntijan koostamiseen käyttämä aika mitattiin. Koostamisesta saatuja perusmuotoja ja sanaluokkia verrattiin korpuksen merkkauksiin F-mitalla. Lisäksi sanaston poimijan monikielisyyden perustoiminnallisuutta arvioitiin syöttämällä sille 5000 japanin- ja englanninkielisen Wikipedian tekstikappaletta ja tarkastelemalla tuloksia silmämääräisesti.

Sanojen erottelua testattiin käyttämällä aineistona japanilaista Tanaka-korpusta. Korpuksesta otettiin 50 000 asiantuntijan merkkaamaa tekstikatkelmaa, joita käytettiin referenssisegmentointina. Testiaineistona käytettiin samoja katkelmia, mutta sanavälitiedot poistettuna. Nämä katkelmat syötettiin sanojen erottelijalle yksi kerrallaan, ja tuotettua segmentointia verrattiin referenssisegmentointiin sanavälien F-mitan avulla. Lisäksi sanojen erottelijaa testattiin vastaavalla tavalla käyttämällä aineistona Turku Dependency Treebankiä.

MI:tä testattiin käyttämällä aineistona Turku Dependency Treebankin 50 000 ensimmäistä sanaa. Sanojen merkkauksista hyödynnettiin sanoja sekä niiden perusmuotoja. Sanoille luotiin vartalorelaatiot sen mukaan, olivatko niiden perusmuodot samat. Useammin kuin kerran esiintyneistä sanoista huomioitiin vain viimeinen, jotta hyvin yleiset sanat eivät kasvattaisi helppojen relaatioiden määrää. Korpuksen sanat syötettiin MI:lle, ja johdettujen vartaloiden perusteella luotiin toinen sanojen vartalorelaatioiden kokoelma. Sitä verrattiin Treebankin vartalorelaatioihin F-mitan avulla. Lisäksi MI:n sanaston poiminnan yhteydessä johtamia yleisimpiä päätteitä tarkasteltiin silmämääräisesti.

SLI:tä testattiin niin ikään samalla otoksella Turku Dependency Treebankin katkelmia. Katkelmien merkkauksista hyödynnettiin taivutettuja sanoja sekä niiden sanaluokkia, jotka Turku Dependency Treebankin merkkausten mukaisesti olivat seuraavat: adjektiivi, adposition, adverbi, konjunktio, interjunktio, substantiivi, numeraali, pronomini, välimerkki, symboli, verbi ja muunkielinen sana. Katkelmien sanat syötettiin ilman merkkauksia MI:lle ja SLI:lle. Jokaiselle sanalle johdettua sanaluokan klusterimerkkausta verrattiin korpuksen merkkauksiin V-mitan avulla.

7.3. Testien tulokset

Testit suoritettiin testaussuunnitelman mukaisesti sekä sanaston poimijalle kokonaisuudessaan että sen eri osille erikseen. Sanaston poiminnan, sanojen erottelun, MI:n ja SLI:n tulokset on esitetty seuraavaksi.

7.3.1. Sanaston poiminta

Suomenkielisten sanojen perusmuotojen ja sanaluokkien poiminnan tulokset on esitetty taulukossa 7.1. Tulokset poimittiin prosessista 13:n täyden iteraatiokierroksen jälkeen.

Tehtävä	Tarkkuus	Saanti	F-mitta
Perusmuoto	16.2%	0.5%	1.0%
Perusmuoto ja sanaluokka	3.2%	0.1%	0.2%

Taulukko 7.1: Sanaston poiminnan tulokset suomen kielellä.

Asiantuntijalta vaadittiin noin 7 minuuttia aikaa klusterien merkkaukseen, kun syötteenä oli kohtuullisen pieni, 13 000:n lauseen kokoinen, korpus. Tämän kokoisella

syötteellä tuotettiin 3.8 oikeellista sanan ja sanaluokan yhdistelmää työminuuttia kohti. Kun sanaluokkien oikeellisuutta ei huomioida, saatiin 19.1 oikeellista sanaa työminuuttia kohti.

Silmämääräisesti arvioituna englanninkieliset tulokset vaikuttivat olevan samaa luokkaa suomen kanssa. Japaninkieliset tulokset olivat myös pääasiallisesti heikkoja, lukuunottamatta kokonaan kanjilla tai katakanalla kirjoitettuja sanoja.

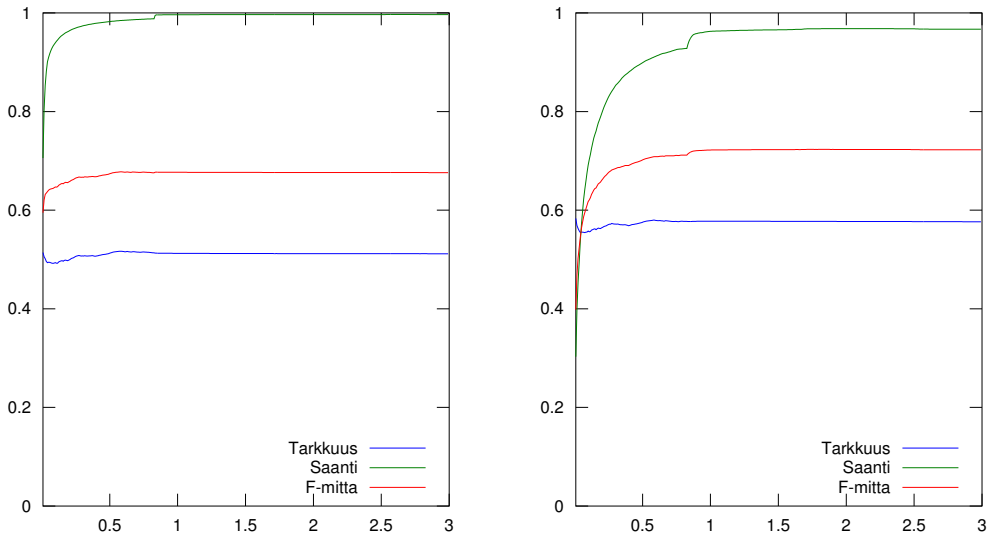
7.3.2. Sanojen erottelu

Sanojen erottelun tulokset on esitetty taulukossa 7.2. Japanin kielen erottelussa käytettiin 15 merkin maksimisananpituutta, ja testi suoritettiin sekä ilman morfologiatiedon hyödyntämistä sekä sen kanssa. Suomen kielen erottelussa testattiin 20 ja 25 merkin maksimisananpituuksia morfologiatietoa hyödynnettäessä.

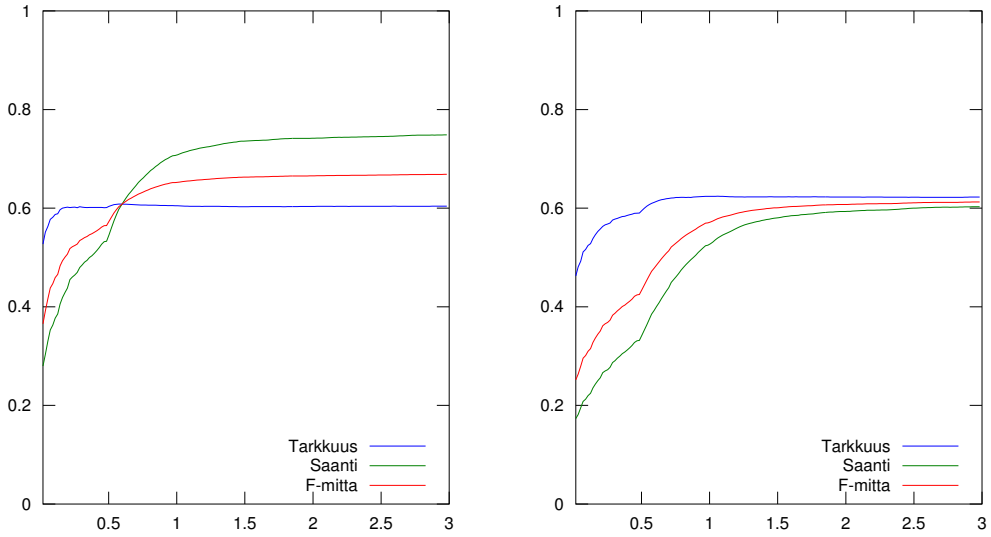
Kieli	Morfologiatieto	Maksimisananpituus	Tarkkuus	Saanti	F-mitta
Japani	Ei	15	51.1%	99.6%	67.6%
Japani	Kyllä	15	57.6%	96.7%	72.2%
Suomi	Kyllä	20	60.3%	74.8%	66.8%
Suomi	Kyllä	25	62.2%	60.2%	61.2%

Taulukko 7.2: Sanojen erottelun tulokset japanin- ja suomenkielisissä erottelutehtävissä eri asetuksilla.

Kuvassa 7.1 on esitetty mitattujen suureiden muuttuminen iteraation edetessä japanin kielellä. Tulokset pysyvät lähes vakiona sen jälkeen, kun jokainen tekstikappale on segmentoitu ensimmäisen kerran. Kuvassa 7.2 on esitetty induktion eteneminen suomenkielellä.



Kuva 7.1: Sanojen erottelun F-mitan arvo japanin kielellä iteraatiokierroksen funktiona. Vasemmalla ovat tulokset ilman morfologiatietoa. Oikealla ovat tulokset morfologiatietoa hyödynnettäessä.



Kuva 7.2: Sanojen erottelun F-mitan arvo suomen kielellä iteraatiokierroksen funktiona. Vasemmalla ovat tulokset 20 merkin sananpituudella. Oikealla ovat tulokset 25 merkin sananpituudella.

7.3.3. Morfologian induktio

Taulukossa 7.3 on esitetty MI:n tulokset suomen kielellä sekä ilman MH-siirtoja että niitä hyödynnettäessä. Nämä arvot saatiin laskemalla induktion viimeisen kymmenen kierroksen tulosten keskiarvot.

MH-siirrot	Tarkkuus	Saanti	F-mitta
Ei	24.3%	29.6%	26.6%
Kyllä	38.7%	30.3%	33.9%

Taulukko 7.3: MI:n tulokset suomenkielisessä tehtävässä. Arvoinnissa käytettiin vartalorelaatioiden F-mittaa.

Kuvassa 7.3 on esitetty vartalorelaatioiden F-mitan muutos iteraation edetessä. Taulukossa 7.4 on esitetty sanaston poiminnan yhteydessä johdetut yleisimmät päätteet suomelle, englannille ja japanille.

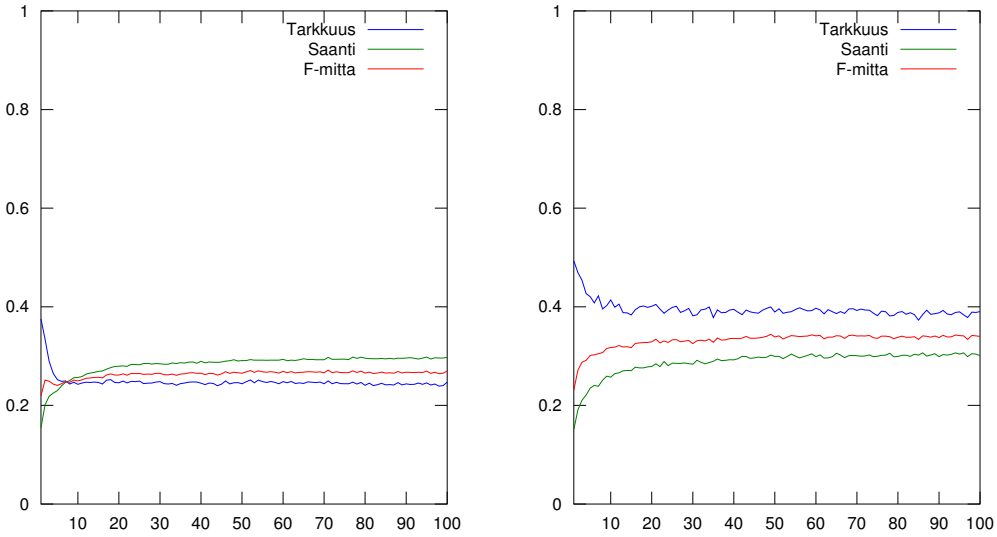
7.3.4. Sanaluokan induktio

SLI:n tulokset suomenkielissä testissä on esitetty taulukossa 7.5. Nämä arvot saatiin laskemalla induktion viimeisen kymmenen kierroksen tulosten keskiarvot. SLI:tä testattiin pelkästään suomen kielellä johtuen ongelmista sopivan korpuksen hankkimisessa muille kielille.

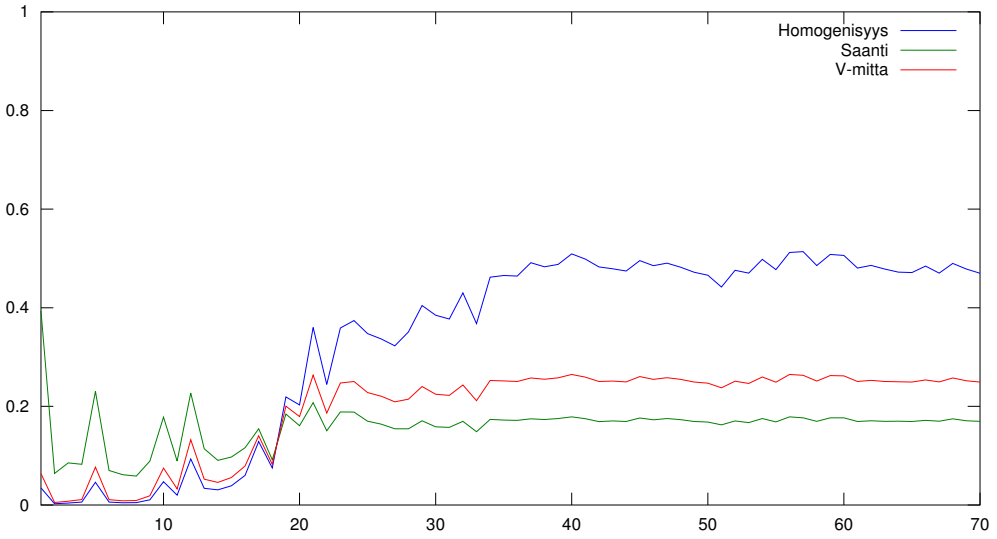
Kuvassa 7.4 on esitetty V-mitan muutos iteraation edetessä.

7.4. Pohdinta

Sanojen poiminta kokonaisuudessaan toimi huonosti suomen kielellä. Sanojen erottelija segmentoi kappaleita varsin usein väärin, ja tämän lisäksi MI segmentoi sanojen



Kuva 7.3: MI:n vartalosuhteiden F-mitan arvo iteraatiokierroksen funktiona. Vasemmalla on esitetty pelkällä Gibbsin näytteistyksellä saadut tulokset. Oikealla ovat tulokset, kun on lisäksi käytetty MH-siirtoja.



Kuva 7.4: Sanaluokan induktion V-mitan muutos iteraatiokierroksen funktiona.

morfologian väärin suurimmalle osalle sanoja. Näin SLI, joka hyödynsi yksinomaan sanojen morfologisia piirteitä, sai syötteekseen paljon joko kokonaan virheellisiä sanoja tai sanoja, joiden morfologia oli väärin. SLI:n tarkkuus oli myös erittäin heikko, joten myös se lisäsi tuloksiin virheitä. Koska nämä kolme menetelmää oli kytketty sarjaan, eikä niiden välillä ollut muuta taaksepäin kytkentää kuin sanojen erottelijan yksinkertainen morfologiasegmentoinnin todennäköisyyksien hyödyntäminen, voidaan olettaa, että eri vaiheiden virheet ovat siirtyneet eteenpäin seuraaville vaiheille ja vahvistuneet entisestään niiden aikana. Tämä selittäisi osaltaan lopullisten tulosten huonoutta.

Muita varsin selkeitä syitä tulosten huonouteen ovat SLI:n erityisen huono suorituskyky sekä sanaston koostamismenetelmän rajoittuneisuus. SLI:n ongelmien vuoksi

Pääte	Lukumäärä
a	11390
n	11025
ä	4068
i	3721
en	2180
t	2139
e	1720
ttä	1474
ta	1271
u	1242
s	1200
in	994
iin	796
sta	778
ssa	752
o	728
lla	691
si	666
an	651
ti	595

Pääte	Lukumäärä
e	11834
n	7218
nd	5537
s	5335
r	2837
f	2830
ed	2228
g	1847
re	1785
y	1577
ith	1163
ing	1119
]	1064
t	995
al	880
d	839
it	822
to	725
tion	680
es	662

Pääte	Lukumäärä
た	967
語	684
的	573
学	488
る。	355
国	336
合	334
化	323
者	298
って	293
て	254
ー	251
音	246
ス	233
ル	219
。	213
言語	203
ン	197
代	182
詞	177

Taulukko 7.4: Sanaston poiminnan yhteydessä kerätyt yleisimmät päätteet. Vasemmalla ovat päätteet suomelle, keskellä englannille ja oikealla japanille.

Homogenisyys	Saanti	V-mitta
48.0%	17.1%	25.2%

Taulukko 7.5: SLI:n tulokset suomenkielisessä tehtävässä.

sanaluokkien klusterit olivat hyvin epähomogeenisiä, joten monessa isommista klusterissa oli sekaisin sekä verbejä että nomiineja. Numeraaleja, adjektiiveja ja pronomineja ei myöskään pystytty pystytty erottamaan omiin klustereihinsa, mikä ei sinänsä ole yllätys, sillä tämä on jo teoreettisestikin hyvin haastava tehtävä. Näiden ongelmien vuoksi, kun SLI:n tuloksille merkattiin sanaluokkia ja johdettiin perusmuotoja koostamisohjelmassa, paljon klusterissa olleita sanoja sai väärän sanaluokan ja tyypillisesti klusterille valittu perusmuodon johtamistapakin oli väärä. Lisäksi, vaikka sanaluokka olisikin ollut oikein, ei yksinkertainen koostamismenetelmä pystynyt muokkaamaan vartalonmuunnoksia sisältäviä sanoja oikeaan muotoon. Näin saatuja tuloksia arvioitiin erittäin tiukalla arviointimenetelmällä, joka hylkäsi sanan kokonaan lievempienkin virheiden, kuten normalisoimattoman ison alkukirjaimen tai sanan jäljessä olevan ylimääräisen välilyönnin vuoksi.

Pienellä korpuksella asiantuntijan työtä vaadittiin vielä selkeästi liikaa suhteessa löydettyjen oikeiden sanojen määrään. Suhteellista työmäärää voidaan kuitenkin pienentää kasvattamalla korpuksen kokoa. Merkatun opetustiedon hyödyntäminen voisi myös olla suhteellisen kannattavaa, jos merkkaamattoman korpuksen koko kasvatetaan tarpeeksi suureksi.

Tiettävästi tämä on ensimmäinen tutkimus, jossa ilmoitetaan objektiivisesti mitattu tarkkuus ja saanti sekä sanojen perusmuodoille että sanaluokille. Ilman suoraa verrokkiakin voidaan kuitenkin sanoa, että tulokset ovat erittäin heikkoja kaikilla kokeilluista kielistä.

Sanojen erottelun tuloksissa mielenkiintoista oli, että erottelijan oppimisnopeus oli varsin korkea. Tämän osoittaa se, että F-mitan arvo vakautui ennen kuin kaikkia kappaleita oli edes käsitelty ensimmäistä kertaa. Toinen mielenkiintoinen seikka sanojen erottelussa oli, että heikkolaatuisenkin morfologiatiedon kevyehkö hyödyntäminen paransi erottelijan tarkkuutta. Tämä johtuu oletettavasti siitä, että ilman morfologiaa algoritmilla oli taipumus erotella yleisiä päätteitä omiksi sanoikseen [52]. Sanavälien todennäköisyyksien pienentäminen morfologiasegmentoinnin perusteella vaikuttaa tulosten perusteella tähän. Morfologiatiedon hyödyntäminen sanojen erottelussa vaikuttaisi siis järkevältä ajatukselta.

Sanojen erottelijan tuottamien segmentointien laatu näytti heikentyvän, kun pisintä sallittua sanan pituutta kasvatettiin. Suomen kielellä algoritmin tarkkuus kyllä kasvoi hieman suurinta sallittua sanan pituutta kasvatettaessa, mutta saanti huononi selvästi. Tämä viittaa oletettavasti siihen, että menetelmä ei pystynyt enää löytämään lyhyiden sanojen sanavälejä yhtä tehokkaasti, kun sallittua sanan pituutta kasvatettiin. Tämän vuoksi menetelmä ei ole sellaisenaan ihanteellinen suomen kaltaisille kielille, jotka käyttävät sekä hyvin lyhyitä että hyvin pitkiä sanoja.

Sanojen erottelija toimi japanilla hieman huonommin kuin Mochihashin [52] menetelmän bigrammivariantti, ainakin kun niitä verrataan pelkästään tarkkuuden perusteella. Mochihashin algoritmin trigrammivariantille ratkaisu häviää selvästi, mikä vastaa odotuksia hänen omien tulostensa perusteella. Koska testiaineisto ja arviointimenetelmä ovat erilaisia, tulokset eivät kuitenkaan ole suoraan verrannollisia, mutta joka tapauksessa trigrammimallia voidaan pitää selkeästi parempana.

Morfologiamalli oli selvästi riittämätön suomelle, sillä se ei esimerkiksi huomioinut vartaloiden muutoksia ollenkaan eikä normalisoinut isoja ja pieniä alkukirjaimia. Tästäkin huolimatta se pystyi ilmeisesti johtamaan hyödyllistä tietoa sen perusteella, että tietoja pystyttiin kohtuullisen tuloksekkaasti hyödyntämään sanojen erottelussa ja sanaluokan induktiossa. MI:hin lisätyillä MH-siirroilla pystyttiin myös parantamaan tulosten tarkkuutta selvästi ilman mitään muutoksia generoivaan malliin.

MI ei toiminut hyvin englannillakaan, sillä yleisimpien löydettyjen päätteiden perusteella se erotteli päätteitä liian ahneesti ja löysi oikeiden päätteiden lisäksi paljon tavallisten perusmuotoisten sanojen loppuosia. Japanissa morfeemien tunnistaminen ei toiminut pääosin sen vuoksi, että sanojen erottelijalla oli vahva taipumus segmentoida morfeemit erillisiksi sanoiksi. Tämän vuoksi MI ei saanut suurta osaa sanojen taivutusmuodoista käyttöönsä, ja se löysi enemmän sanaliittojen loppuosia oikeiden morfeemien sijaan.

MI:n tulokset jäivät parhaiden vastaavien menetelmien tuloksista, sillä esimerkiksi Linguistica pystyy saavuttamaan yli 40%:n tarkkuuden suomessa ja muut MDL-menetelmät yli 50%:n tarkkuuden [84]. Tässä työssä käytetty arviointimenetelmä ei ole suoraan verrattavissa näihin tuloksiin, ja oletettavasti se on anteeksiantavampi kuin verrokkitutkimusten asiantuntijoiden voimin tehty arviointi. Morfologiamallin ja induktiomenetelmän rajoittuneisuudet näyttäisivät olevan suurimmat syyt, joiden vuoksi ratkaisu hävisi parhaimmille MI-menetelmille.

SLI:n tulokset ovat selkeästi pienempiä kuin uusimmilla vastaavilla menetelmillä, joilla on päästy yli 70%:n [80] V-mittaan. Tästä voidaan päätellä, ettei pelkkä heikkolaatuinen morfologiatieto ole riittävää tehtävän ratkaisuun. Uusimmat SLI-menetelmät ovat kauttaltaan tukeutuneet vahvasti distributionaaliseen tietoon, jota ei tässä ratkaisussa käytetty ollenkaan. Distributionaalisen tiedon puute ja käytettyjen morfologiapiirteiden heikko laatu ovat todennäköisesti suurimmat syyt, joiden vuoksi ratkaisu ei pääse lähellekään parhaimpien SLI-menetelmien tuloksia.

8. TYÖN JATKOKEHITYS

Tässä työssä toteutettua ratkaisua voitaisiin kehittää monin tavoin, ja tuloksia sekä monikielisyyden tukea näin parantaa. Selkeimmät kehityssuunnat ovat järjestelmän ohjaamattomien komponenttien kehittäminen ja sulauttaminen toisiinsa.

Sanojen erottelussa voitaisiin hyödyntää trigrammimallia bigrammimallin sijaan, sillä tämän on osoitettu parantavan tuloksia muissa vastaavissa menetelmissä. Lisäksi MI:n tuloksia voitaisiin hyödyntää paremmin sanojen erottelussa, sillä morfologiatieto näytti vaikuttavan positiivisesti tuloksiin.

Sanojen erottelussa voitaisiin myös käyttää samanlaisia MH-siirtoja, kuin mitä tämän työn MI:ssä käytettiin. Tämä voisi parantaa tuloksia, sillä induktio toipuisi paremmin alussa tehdyistä vääristä segmentoinneista, joiden vuoksi tietyt sanat oli virheellisesti jaettu hyvin lyhyisiin osiin. Jos tietty sanatason n-grammi voitaisiin yhdistää yhdeksi sanaksi tai yksi sana hajottaa useaksi sanaksi kaikkialla, jossa se esiintyi, induktio voisi tehokkaammin päästä pois globaalista minimistä ja konvergoitua kohti globaalia optimaalista ratkaisua.

MI:tä on mahdollista kehittää huomattavasti. Siihen voitaisiin lisätä tuki esimerkiksi vartalonmuunnosten induktiolle. Vartalonmuunnoksia voitaisiin esimerkiksi generoida Pitman-Yor prosessista tietyille vartalon alun tai lopun kirjainyhdistelmille, ja vartalonmuunnossäännöt voitaisiin valita sanoille uudella vartaloita yhdistävällä MH-siirrolla.

MI voisi myös tukea useampaa kuin yhtä päätettä sekä myös etuliitteitä ja yhdyssanojen osia. Tällöin sanat voitaisiin segmentoida morfologisiin osiin sanojen erottelijan toimintaperiaatteita hyödyntäen. Morfologiasegmentointien todennäköisyyksien arvioinnissa voitaisiin käyttää myös hyväksi sanaluokan induktion tuloksia, etenkin jos sanaluokan induktioon lisättäisiin tuki distributionaaliselle luokittelulle. Tällöin vartaloiden todennäköisyyksiä voitaisiin varioida niiden sanaluokkien todennäköisyyksien mukaan. Toisaalta, sanojen erottelu ja MI voitaisiin integroida pidemmälle, jolloin molemmat näistä ongelmista voitaisiin ratkaista esimerkiksi adaptorikielioppien kaltaisilla menetelmillä.

SLI:hin pitäisi selvästikin lisätä tuki distributionaalisen tiedon hyödyntämiselle, sillä pelkkä morfologiatieto ei riitä yksistään. Distributionaalista tietoa olisi saatavilla esimerkiksi sanojen erottelijan sanojen n-grammimallista. Sen integrointi osaksi sanaluokan induktiota olisikin kiinnostavaa. Vaikka morfologian hyödyntäminen yksistään ei ollut riittävää, sen hyödyntäminen distributionaalisen tiedon rinnalla vaikuttaa kuitenkin lupaavalta. Tällainen malli, joka hyödyntää molemman tyyppistä tietoa, olisi myös kiintoisa tutkimuskohde.

Sanaston koostamiseen olisi kiinnostavaa löytää ohjaamattomia menetelmiä. Esimerkiksi ohjaamaton perusmuotojen tunnistaminen vähentäisi jäljellä olevaa asiantuntijoiden tekemää työtä huomattavasti. Tämän tehtävän automatisointi kieliriippumattomalla tavalla voi kuitenkin olla hyvin haastavaa, ja se vaatisi huomattavaa lisätutkimusta.

9. YHTEENVETO

Tässä diplomityössä tehtiin katsanto uusimpiin sanaston poiminnan menetelmiin ja teoriaan sekä esiteltiin niiden pohjalta suunniteltu ja toteutettu järjestelmä sanaston poimintaan. Työssä keskityttiin ennen kaikkea ohjaamattomiin ja pääosin ohjaamattomiin menetelmiin.

Sanaston todettiin olevan merkittävässä roolissa monien LKK-sovellusten toiminnan kannalta, sillä laajempi sanasto tyypillisesti lisää niiden toimintavarmuutta. Tästä huolimatta laajaa sanastoa ei kuitenkaan ole saatavilla monilla kielillä tai moniin aloihin liittyen, ja sanaston tuottaminen manuaalisesti on hidasta ja kallista. Tämän vuoksi ohjaamattoman tai pääosin ohjaamattoman sanaston poimijan todettiin olevan hyödyllinen käytännön apuväline, jos sellainen voitaisiin toteuttaa.

Työssä esiteltiin sanaston poiminnan kannalta olennaisin teoria sekä tärkeimmät olemassa olevat menetelmät ja arviointimenetelmät. Sanaston poiminnan todettiin koostuvan neljästä perusosasta: sanojen erottelu, MI, SLI sekä sovelluskohtainen sanaston koostaminen. Näistä kolmen ensimmäisen ratkaisemiseen esiteltiin useita menetelmiä, joiden joukossa keskityttiin erityisesti uusimpiin ei-parametrisiin Bayesin menetelmiin. Niillä todettiin olevan monia ohjaamattoman oppimisen kannalta hyödyllisiä ominaisuuksia sekä laajennusmahdollisuuksia tulevaa kehitystä ajatellen.

Taustatutkimuksen pohjalta kehitettiin ei-parametriseen Bayesin menetelmään perustuva sanaston poimija. Se koostui neljästä osasta, jotka olivat teorian mukaisesti sanojen erottelu, MI, SLI ja sanaston koostaminen. Sanaston koostamista lukuun ottamatta ratkaisu oli lähes täysin ohjaamaton.

Järjestelmän tuottamat tulokset olivat odotetusti heikkoja. Suomen kielellä sanoista noin 16% oli oikein, kun sanaluokkia ei huomioitu, ja sanojen ja sanaluokkien yhdistelmistä vain 3% oli oikein. Saanti oli molemmissa tapauksissa alle 1%:n. Myös järjestelmän jokainen yksittäinen osa suoriutui selkeästi huonommin kuin parhaimmat vastaavat ohjaamattomattomat menetelmät. Yksi merkittävä syy tähän oli se, että ratkaisussa käytetyt menetelmät eivät kyenneet mallintamaan ongelmia yhtä monipuolisesti kuin parhaimmat menetelmät.

Lopuksi todettiin, että järjestelmää voitaisiin kehittää edelleen runsaasti. Etenkin MI:n ja SLI:n todettiin kaipaavaan runsaasti parannuksia. Lisäksi todettiin, että menetelmän eri osien sulauttaminen tiiviimmin toisiinsa olisi kannattava tulevan tutkimuksen kohde. Sulautus voisi tapahtua käyttämällä järjestelmän eri osien keräämiä tietojen voimakkaammin hyödyksi sen toisissa osissa.

10. LÄHTEET

[1] Zernik U (1991) Lexical acquisition: exploiting on-line resources to build a lexicon. Associates 9: 429.

[2] Mikheev A (1997) Automatic rule induction for unknown-word guessing. Computational Linguistics 23(3): 405–423.

[3] Arranz MV (1997) Lexical bottleneck in machine translation and natural language processing: A case study. Proceedings Translating and the Computer 19.

[4] Abney S & Bird S (2010) The human language project: Building a universal corpus of the worlds languages. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 88–97. Association for Computational Linguistics.

[5] Maxwell M & Hughes B (2006) Frontiers in linguistic annotation for lower-density languages. In: Proceedings of the workshop on frontiers in linguistically annotated corpora 2006, pp. 29–37. Association for Computational Linguistics.

[6] Haspelmath M (2007) Pre-established categories don't exist: consequences for language description and typology. Linguistic Typology 11(1): 119–132.

[7] Song JJ (2001) Linguistic Topology: Morphology and Syntax. Pearson.

[8] Sproat R, Gale W, Shih C & Chang N (1996) A stochastic finite-state word-segmentation algorithm for chinese. Computational linguistics 22(3): 377–404.

[9] Shopen T (1985) Language typology and syntactic description, volume 3. Cambridge University Press.

[10] Bybee JL (1985) Morphology: a study of the relation between meaning and form. Benjamin.

[11] Radford A (1997) Syntax: a minimalist introduction. Cambridge University Press.

[12] Lyons J (1981) Language and linguistics. Cambridge University Press.

[13] Hakulinen A, Korhonen R, Vilkuna M & Koivisto V (2004) Iso suomen kielioppi. Suomalaisen kirjallisuuden seura.

[14] Miller RA (1967) The Japanese Language. Charles E. Tuttle.

[15] Hamming RW (1950) Error detecting and error correcting codes. Bell System technical journal 29(2): 147–160.

[16] Damerau FJ (1964) A technique for computer detection and correction of spelling errors. Communications of the ACM 7(3): 171–176.

[17] Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. In: Soviet physics doklady, volume 10, p. 707.

[18] Ukkonen E (1992) Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science 92(1): 191–211.

[19] Gravano L, Ipeirotis PG, Jagadish HV, Koudas N, Muthukrishnan S, Srivastava D *et al.* (2001) Approximate string joins in a database (almost) for free. In: Proceedings of the international conference on very large data bases, pp. 491–500.

[20] Deerwester SC, Dumais ST, Landauer TK, Furnas GW & Harshman RA (1990) Indexing by latent semantic analysis. JASIS 41(6): 391–407.

[21] Landauer TK, Foltz PW & Laham D (1998) An introduction to latent semantic analysis. Discourse processes 25(2-3): 259–284.

[22] Hofmann T (1999) Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pp. 289–296. Morgan Kaufmann Publishers Inc.

[23] Hofmann T (2001) Unsupervised learning by probabilistic latent semantic analysis. Machine learning 42(1-2): 177–196.

[24] Rissanen J (1983) A universal prior for integers and estimation by minimum description length. The Annals of statistics pp. 416–431.

[25] Clark A, Fox C & Lappin S (2010) The handbook of computational linguistics and natural language processing, volume 57. Wiley-Blackwell.

[26] Goldsmith J (2006) An algorithm for the unsupervised learning of morphology. Natural Language Engineering 12(04): 353–371.

[27] Goldwater SJ (2007) Nonparametric bayesian models of lexical acquisition. Ph.D. thesis, Citeseer.

[28] Clark A (2003) Combining distributional and morphological information for part of speech induction. In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1, pp. 59–66. Association for Computational Linguistics.

[29] Griffiths TL, Kemp C & Tenenbaum JB (2008) Bayesian models of cognition. Cambridge handbook of computational cognitive modeling pp. 59–100.

[30] Bayes T (1763/1958) An essay toward problem solving in the doctrine of chaos. reprinted in. Biometrika 45: 293–315.

[31] Johnson M (2008) Unsupervised word segmentation for sesotho using adaptor grammars. In: Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology, pp. 20–27. Association for Computational Linguistics.

[32] Chib S & Greenberg E (1995) Understanding the metropolis-hastings algorithm. The American Statistician 49(4): 327–335.

[33] Resnik P & Hardisty E (2010) Gibbs sampling for the uninitiated. Technical report, DTIC Document.

[34] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH & Teller E (1953) Equation of state calculations by fast computing machines. The journal of chemical physics 21: 1087.

[35] Hastings WK (1970) Monte carlo sampling methods using markov chains and their applications. Biometrika 57(1): 97–109.

[36] Geman S & Geman D (1984) Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. Pattern Analysis and Machine Intelligence, IEEE Transactions on (6): 721–741.

[37] Ferguson TS (1973) A bayesian analysis of some nonparametric problems. The annals of statistics pp. 209–230.

[38] Jordan MI (2005) Dirichlet processes, chinese restaurant processes and all that. In: Tutorial presentation at the NIPS Conference.

[39] Teh YW (2007) Dirichlet processes: Tutorial and practical course. Machine Learning Summer School .

[40] Teh YW (2010). Dirichlet process.

[41] Gershman SJ & Blei DM (2012) A tutorial on bayesian nonparametric models. Journal of Mathematical Psychology 56(1): 1–12.

[42] Aldous D (1985) Exchangeability and related topics. École d'Été de Probabilités de Saint-Flour XIII1983 pp. 1–198.

[43] Blackwell D & MacQueen JB (1973) Ferguson distributions via pólya urn schemes. The annals of statistics pp. 353–355.

[44] Sethuraman J (1991) A constructive definition of dirichlet priors. Technical report, DTIC Document.

[45] Pitman J (2002) Combinatorial stochastic processes. Technical report, Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course.

[46] Neal RM (2000) Markov chain sampling methods for dirichlet process mixture models. Journal of computational and graphical statistics 9(2): 249–265.

[47] Escobar MD & West M (1995) Bayesian density estimation and inference using mixtures. Journal of the american statistical association 90(430): 577–588.

[48] Pitman J & Yor M (1997) The two-parameter poisson-dirichlet distribution derived from a stable subordinator. The Annals of Probability 25(2): 855–900.

[49] Teh YW (2006) A hierarchical bayesian language model based on pitman-yor processes. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 985–992. Association for Computational Linguistics.

[50] Teh YW (2006) A bayesian interpretation of interpolated kneser-ney .

[51] Teh YW, Jordan MI, Beal MJ & Blei DM (2006) Hierarchical dirichlet processes. Journal of the American Statistical Association 101(476).

[52] Mochihashi D, Yamada T & Ueda N (2009) Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pp. 100–108. Association for Computational Linguistics.

[53] Johnson M, Griffiths TL & Goldwater S (2007) Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. Advances in neural information processing systems 19: 641.

[54] Johnson M (2008) Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In: 46th Annual Meeting of the ACL, pp. 398–406.

[55] Schone PJ (2001) Toward knowledge-free induction of machine-readable dictionaries. Ph.D. thesis, University of Colorado.

[56] Haruechaiyasak C, Kongyoung S & Dailey M (2008) A comparative study on thai word segmentation approaches. In: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on, volume 1, pp. 125–128. IEEE.

[57] Vapnik VN (1998) Statistical learning theory .

[58] Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery 2(2): 121–167.

[59] Lafferty J, McCallum A & Pereira FC (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

[60] Kruengkrai C, Sornlertlamvanich V & Isahara H (2006) A conditional random field framework for thai morphological analysis. In: Proceedings of LREC, pp. 2419–2424.

[61] Kudo T, Yamamoto K & Matsumoto Y (2004) Applying conditional random fields to japanese morphological analysis. In: EMNLP, volume 4, pp. 230–237.

[62] Argamon S, Akiva N, Amir A & Kapah O (2004) Efficient unsupervised recursive word segmentation using minimum description length. In: Proceedings of the 20th international conference on Computational Linguistics, p. 1058. Association for Computational Linguistics.

[63] Saffran JR, Aslin RN & Newport EL (1996) Statistical learning by 8-month-old infants. Science 274(5294): 1926–1928.

[64] Goldwater S, Griffiths TL & Johnson M (2006) Contextual dependencies in unsupervised word segmentation. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 673–680. Association for Computational Linguistics.

[65] Ando RK & Lee L (2000) Mostly-unsupervised statistical segmentation of japanese: Applications to kanji. In: Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, pp. 241–248. Association for Computational Linguistics.

[66] Goldwater S, Griffiths TL & Johnson M (2009) A bayesian framework for word segmentation: Exploring the effects of context. Cognition 112(1): 21–54.

[67] Pearl L, Goldwater S & Steyvers M (2010) How ideal are we? incorporating human limitations into bayesian models of word segmentation. In: Proceedings of the 34th Annual Boston University Conference on Child Language Development. Cascadilla Press, Somerville. Citeseer.

[68] Brent MR (1999) An efficient, probabilistically sound algorithm for segmentation and word discovery. Machine Learning 34(1-3): 71–105.

[69] Gaussier É (1999) Unsupervised learning of derivational morphology from inflectional lexicons. In: Proceedings of ACL99 Workshop: Unsupervised Learning in Natural Language Processing. Citeseer.

[70] Neuvel S & Fulop SA (2002) Unsupervised learning of morphology without morphemes. In: Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6, pp. 31–40. Association for Computational Linguistics.

[71] Wilson C (2004) Combining part of speech induction and morphological induction .

[72] Baroni M, Matiasek J & Trost H (2002) Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In: Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6, pp. 48–57. Association for Computational Linguistics.

[73] Schone P & Jurafsky D (2001) Knowledge-free induction of inflectional morphologies. In: Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, pp. 1–9. Association for Computational Linguistics.

[74] Snover MG (2002) An unsupervised knowledge free algorithm for the learning of morphology in natural languages .

[75] Goldsmith J (2001) Unsupervised learning of the morphology of a natural language. Computational linguistics 27(2): 153–198.

[76] Naradowsky J & Goldwater S (2009) Improving morphology induction by learning spelling rules. In: IJCAI, pp. 1531–1536.

[77] Christodoulopoulos C, Goldwater S & Steedman M (2010) Two decades of unsupervised pos induction: How far have we come? In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 575–584. Association for Computational Linguistics.

[78] Brown PF, Desouza PV, Mercer RL, Pietra VJD & Lai JC (1992) Class-based n-gram models of natural language. Computational linguistics 18(4): 467–479.

[79] Goldwater S & Griffiths T (2007) A fully bayesian approach to unsupervised part-of-speech tagging. In: Annual meeting-association for computational linguistics, volume 45, p. 744.

[80] Blunsom P & Cohn T (2011) A hierarchical pitman-yor process hmm for unsupervised part of speech induction .

[81] Makhoul J, Kubala F, Schwartz R, Weischedel R *et al.* (1999) Performance measures for information extraction. In: Proceedings of DARPA Broadcast News Workshop, pp. 249–252.

[82] Snover MG, Jarosz GE & Brent MR (2002) Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In: Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6, pp. 11–20. Association for Computational Linguistics.

[83] Rosenberg A & Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: EMNLP-CoNLL, volume 7, pp. 410–420.

[84] Creutz M & Lagus K (2002) Unsupervised discovery of morphemes. In: Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6, pp. 21–30. Association for Computational Linguistics.