Neurolaskentamenetelmien soveltamisesta puheen tuoton häiriöiden mallintamiseen

Antti Järvelin

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Kesäkuu 2003

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Antti Järvelin: Neurolaskentamenetelmien soveltamisesta puheen tuoton häiriöiden mallintamiseen
Pro gradu -tutkielma, 92 sivua, 9 liitesivua
Kesäkuu 2003

Tutkielmassa tarkastellaan afaattisten nimeämishäiriöiden mallintamista kolmen erilaisen neuroverkkomallin avulla. Esiteltävistä malleista kahden tarkastelu perustuu kirjallisuuteen ja kolmas toteutettiin tutkielman teon yhteydessä. Toteutetulla Learning SLIPNET -neuroverkkomallilla halutaan osoittaa, että puheentuoton häiriöitä voidaan mallintaa yleisesti käytettyjen monikerrosperceptron-neuroverkkojen avulla. Learning SLIPNETin kykyä simuloida afasiapotilaita testattiin sovittamalla verkko kymmeneen potilaaseen, jotka kärsivät erityyppisistä afasiaoireyhtymistä. Potilassovitukset onnistuivat tilastollisesti hyvin. Tulosten laadullinen analyysi jätettiin kuitenkin tekemättä, sillä sen suorittaminen vaatisi vankkaa neuropsykologian ja kielitieteen tieteenalojen asiantuntemusta.

Uuden simulaatiomallin kehittämisen yhteydessä jouduttiin pohtimaan sanojen semantiikan ja fonologian esittämistä neuroverkoille. Fonologian koodaamiseksi voitiin käyttää osaksi jo tunnettuja menetelmiä. Semantiikan koodaamiseksi tyydyttävää menetelmää ei kuitenkaan ollut, joten tutkielmassa kehitetään menetelmä semantiikan algoritmiseksi koodaamiseksi neuroverkoille. Menetelmä perustuu sanojen hierarkkiseen luokitukseen. Sen avulla on mahdollista muodostaa pienidimensioisia syötevektoreita, jotka soveltuvat suuridimensioisia paremmin monikerrosperceptron-neuroverkkojen syötteeksi. Tutkielmassa kehitetään lisäksi menetelmä, jolla simulaatiomallien sovittaminen potilasaineistoon voidaan automatisoida. Tämä vapauttaa tutkijan simulaatioparametrien hakemiselta tärkeämpien työtehtävien pariin.

CR-luokat: I.2.6 [**Artificial Inteligence**]: Learning - *connectionism and neural nets*; I.2.7 [**Artificial Inteligence**]: Natural Language Processing - *language models*

Avainsanat ja -sanonnat: afasia, neuroverkot, nimeämishäiriö, puheen tuoton mallintaminen

Sisällys

Kiitokset

Kiitokset opettajilleni sekä tutkielman tarkastajille. Erityisesti haluan kiittää Jorma Laurikkalaa, joka avusti viidennen luvun tilastollisten testien tekemisessä.

Termit, lyhenteet ja merkinnät

/a/	Suomenkielen grafeemia (kirjainta) 'a' vastaava foneemi.
α_L	Learning SLIPNET-neuroverkon leksikaalis-semanttisen aliverkon kohina-parametri.
α_F	Learning SLIPNET-neuroverkon fonologisen aliverkon kohina-parametri.
Afasia	Aivovaurion seurauksena syntynyt häiriö, jossa potilaan puheen tuottaminen tai ymmärtäminen on heikentynyt.
Anomia	Vaikeus nimetä esineitä tai olioita. Anomia on yleisin afasian oire.
d_E, $d_E(\mathbf{x},\mathbf{y})$	Euklidinen etäisyys vektoreiden **x** ja **y** välillä ($\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$).
DTS-malli	Discrete Two-Stage model of naming. Mallin mukaan leksikalisaation vaiheet ovat toisistaan täysin erillisiä.
IA-malli	Interactive activation model of naming. Mallin mukaan leksikalisaation vaiheet vaikuttavat toisiinsa.
Lekseemi	Sanan fonologinen esitys, joka sisältää artikulaatio-ohjeen.
Leksikalisaatio	Leksikalisaatiolla tarkoitetaan ajatusten muuttamista puhutuiksi sanoiksi. Tutkijat uskovat leksikalisaation tapahtuvan kahdessa vaiheessa, joista ensimmäisessä haetaan sanan lemma ja toisessa sen lekseemi.
Lemma	Sanan abstrakti syntaktis-semanttinen esitys, joka toimii välittäjänä sanan käsitteellisen ja fonologisen esityksen (lekseemin) välillä.
MLP	Multi-Layer Perceptron, monikerrosperceptron. Yleisesti käytetty eteenpäinsyöttävä neurolaskentamalli, joka koostuu yhdestä tai useammasta piilokerroksesta ja käyttää usein sigmoidiaktivaatio-funktiota.
Neologismi	Uudismuodoste. "Keksitty" sana, jota ei esiinny yleiskielessä. Afaatikot voivat tuottaa esim. nimeämistestien yhteydessä neologismeja.
Omissio	Omissiolla tarkoitetaan puheentuottotehtävän, kuten kuvan nimeämistestin yhteydessä tilannetta, jossa potilas ei kykene nimeämään hänelle esitettyä kuvaa.
SSE, E_{SSE}	Sum of Squared Errors, neliövirhesumma.
τ	Learning SLIPNET-neuroverkon kynnysarvoparametri.

1. Johdanto

Afasia on aivovaurion seurauksena syntynyt puheen tuottamis- tai ymmärtämishäiriö. Yleisimmät syyt afasiaan johtaneisiin aivovaurioihin ovat vasemman aivopuoliskon aivoinfarktit sekä aivoverenvuodot. Anomia, eli sanojen löytämisen vaikeus, on keskeinen afasian oire. Tätä vaikeutta on mahdollista tutkia kuvan nimeämistehtävän avulla. Tehtävässä potilaalle esitetään kuvia esineistä, jotka potilaan tulee nimetä, eli sanoa ääneen kuvaa vastaava sana.

Jos visuaalinen tunnistusvaihe sekä puheen artikulaatio suljetaan tarkastelusta pois, nimeämistä voidaan tarkastella *leksikalisaationa*. Leksikalisaatiolla tarkoitetaan prosessia, jonka aikana puhuja muuntaa sanan käsitteellisen esityksen sen fonologiseksi esitykseksi, eräänlaiseksi artikulaatio-ohjeeksi. Tällä hetkellä tutkijoiden keskuudessa hyväksytään laajalti ns. kaksivaiheinen teoria leksikalisaatiosta. Siinä leksikalisaatioprosessista erotetaan kaksi vaihetta: *lemman haku* (lemma access) ja *fonologinen haku* (phonological access). Ensimmäisessä vaiheessa, eli lemmanhakuvaiheessa, kuvataan nimettävän sanan käsitteellinen esitys sanan abstraktiksi syntaktis-semanttiseksi esitykseksi, *lemmaksi*. Toisessa vaiheessa eli fonologisen haun aikana lemma muutetaan sanan fonologiseksi esitykseksi, *lekseemiksi*. Tutkijat eivät kuitenkaan ole yksimielisiä siitä, osallistuuko leksikalisaation eri vaiheisiin koko puheentuoton järjestelmä vai vain osa siitä.

Nimeämistä ja nimeämishäiriöitä mallinnetaan neurolaskentamenetelmillä. Neuroverkot ovat luonnollinen tapa mallintaa nimeämishäiriöitä, sillä niiden rakenne vastaa pelkistetysti intuitiivisella tasolla aivojen rakennetta. Rakenteensa johdosta neuroverkot kestävät hyvin vaurioita ja vaurioiden tuottaminen neuroverkkoihin on yksinkertaista. Vaurionkestävyydellä tarkoitetaan sitä, että vaurioitettunakin neuroverkot antavat tuloksia, jotka muistuttavat jossain määrin oikeaa tulosta. Esimerkkinä nimeämishäiriöiden mallintamisesta neuroverkkojen avulla mainittakoon Tikkalan [1997] väitöskirjatyö, jossa mallinnetaan suomenkielisten afasiapotilaiden nimeämistä.

Tässä tutkielmassa tarkastellaan afaattisten nimeämishäiriöiden mallintamista kolmen erilaisen neuroverkkomallin avulla. Malleista kahden ensimmäisen tarkastelu perustuu kirjallisuuteen. Kumpikaan näistä malleista ei ole oppiva, vaan mallien tarvitsema tieto on asetettu niihin ennalta. Ensimmäisen, *erilliseen kaksivaiheteoriaan* (Discrete Two-Stage theory, DTS) perustuvan mallin mukaan nimeämisen vaiheet ovat toisistaan täysin erillisiä. Ensin valitaan lemma, jolle sitten haetaan foneemitason esitys erillään lemmatasosta. Tutkiel-

massa tarkastellaan suomenkielistä DTS-mallia pääasiassa Laineen *et al.* [1998] esittämän SLIPNET-neuroverkon pohjalta.

Toinen, *interaktiivisen aktivaation* (Interactive Activation, IA) teoriaan perustuva malli, olettaa, että lemmanhakuvaiheessa myös nimettävän sanan fonologia aktivoituu ja vahvistaa nimettävän sanan lemmaa takaisinkytkentöjen avulla. Interaktiiviseen aktivaatioon perustuvaa mallia tarkastellaan Dellin *et al.* [1996] ja [1997] esittämän mallin pohjalta.

Tutkielman pääosa koostuu kolmannen neuroverkkomallin, Learning SLIPNETin toteutuksesta sekä evaluoinnista. Käytännössä Learning SLIPNET on edellä mainitun SLIPNET-neuroverkon uudelleentoteutus käyttäen oppivia *monikerrosperceptron*-neuroverkkoja (Multi-Layer Perceptron, MLP). Learning SLIPNET toteutettiin MLP-neuroverkoilla, sillä ne ovat yleisimmin käytettyjä neuroverkkoja ja soveltuvat useiden erilaisten ongelmien mallintamiseen. Toteutettavasta neuroverkkomallista haluttiin oppiva, sillä oppivan neuroverkkomallin avulla on mahdollista mallintaa sellaisia ilmiöitä, joita staattisilla malleilla ei ole mahdollista mallintaa. Esimerkkinä tällaisesta ilmiöistä on potilaiden uudelleenopetuksen simulointi. Tutkielmassa keskitytään kuitenkin toteutetun mallin yleisten ominaisuuksien kartoittamiseen sekä mallin sovittamiseen potilasaineistoon. Mallin uusien ominaisuuksien hyödyntäminen jätetään tulevaisuuden työksi.

Mallin toteuttamisen yhteydessä sanojen semantiikan ja fonologian esittäminen sille vaati erityishuomiota. Fonologian koodaamiseksi kirjallisuudesta löytyi sopivia menetelmiä, joilla koodauksen saattoi suorittaa. Sen sijaan sanojen semantiikan koodaaminen muodostui ongelmalliseksi. Koska tyydyttävää ratkaisua ongelmaan ei kirjallisuudessa ollut, kehitettiin uusi menetelmä sanojen semanttisen koodauksen algoritmiseksi generoimiseksi. Menetelmä johtaa sanojen välisten semanttisten suhteiden yliyksinkertaistukseen, mutta sen avulla oli mahdollista ratkaista muita semanttiseen koodaukseen liittyviä ongelmia. Semantiikan koodaamista Learning SLIPNETille esitellään tarkemmin luvussa neljä.

Tutkielman toisessa luvussa esitellään neurolaskennan perusteita ja keskitytään myöhemmin esitettävien ongelmien kannalta tärkeisiin teemoihin. Näitä ovat mm. tiedon esittäminen neuroverkoille sekä neuroverkkojen oppimiskyky. Lisäksi luvussa käsitellään pinnallisesti MLP-neuroverkkoja, koska niillä on tärkeä osa luvussa neljä ja viisi esiteltävän nimeämistä simuloivan Learning SLIPNET-neuroverkon toteutuksessa. Luvun loppuosassa perustellaan neurolaskentamallien käyttöä kielen häiriöiden mallintamisessa.

Kolmannen luvun alussa suoritetaan suppea katsaus kielen psykologiaan sekä vallalla olevaan leksikalisaation kaksivaiheiseen teoriaan. Luvussa

tarkastellaan lisäksi lyhyesti klassisia afasiaoireyhtymiä, joita käytetään edelleen kliinisen afasialuokituksen perustana. Luvun loppuosa muodostuu kahden nimeämistä simuloivan neuroverkkomallin esittelystä. Mallien kykyä simuloida nimeämishäiriöistä kärsiviä potilaita sekä malleja vastaan esitettyä kritiikkiä tarkastellaan lyhyesti.

Luvussa neljä esitetään nimeämishäiriöitä simuloivan Learning SLIPNET-neuroverkon toteutus. Pääosa luvusta käsittelee sanojen semantiikan ja fonologian koodaamista muodostettavalle mallille. Luvussa esitetään mm. uusi menetelmä sanojen semanttisen koodauksen muodostamiseksi neuroverkoille. Lopuksi keskitytään toteutettavan verkon arkkitehtuurin tarkempaan esittelyyn.

Viidennen luvun alussa kartoitetaan toteutetun mallin yleisiä ominaisuuksia yrittämättä sovittaa verkkoa potilaisiin. Tämän jälkeen mallia sovitetaan kymmenen afasiapotilaan nimeämistuloksiin. Sovitus tapahtuu sitä varten kehitetyllä algoritmilla, joka yrittää minimoida mallin ja potilaan tuottamien jakaumien välisiä erotuksia. Tulosanalyysi suoritetaan vain tulosten tilastollisen jakautumisen mukaan. Virheluokkien sisäistä laadullista analyysia ei suoriteta, sillä se vaatisi vankkaa neuropsykologian ja kielitieteen tieteenalojen tuntemusta.

Tutkielman tarkoituksena oli selvittää, voidaanko afaattisia nimeämishäiriötä mallintaa yksinkertaisilla MLP-neuroverkoilla. Lisäksi tavoitteena oli luoda simulaatiomalli, jonka avulla pystyttäisiin simuloimaan potilaiden nimeämisen lisäksi mm. heidän toipumistaan ja uudelleenoppimistaan.

2. Neurolaskennan perusteita

Neuroverkkojen (artificial neural networks, neural networks, connectionist networks) tutkimusta on motivoinut alusta alkaen huomio, että ihmisaivot käsittelevät tietoa täysin eri tavalla kuin tavanomaiset tietokoneet [Haykin, 1994]. Ihmiset kykenevät sellaiseen massiiviseen rinnakkaisprosessointiin, josta tämänhetkisillä supertietokoneilla voidaan vain haaveilla. Lisäksi ihmiset oppivat kokemuksesta ja kykenevät yleistämään jo opittua tietoa uusissa asiayhteyksissä. Näiden huomioiden perusteella neurolaskentamallien innoittajina olivatkin ihmisaivot, tai -hermosto, mutta sittemmin mallit ovat alkaneet erkaantua yhä kauemmaksi esikuvistaan.

Ihmisaivojen perusyksikkö on *hermosolu* eli *neuroni* (neuron). Hermosolut vastaanottavat viestejä muilta hermosoluilta ja tuottavat niihin vastauksia. Hermosolun yleinen rakenne voidaan jakaa karkeasti neljään osaan: synapseihin, dendriitteihin, solukeskukseen ja aksoniin. Synapsit ovat kahden hermosolun välisiä liitospisteitä, dendriitit toimivat hermosoluun saapuvan informaation (sähköimpulssi) välittäjinä ja aksoni taas välittää solukeskuksesta lähtevän informaation eteenpäin muille hermosoluille. Aivoissa tietoa varastoituu hermosolujen välisiin yhteyspisteisiin [Rojas, 1996].

Myös neuroverkkojen peruslaskentayksikkönä on (keinotekoinen) neuroni. Tässä luvussa esitellään ensin keinotekoisen neuronin yleinen malli. Tämän jälkeen tarkastellaan niistä rakentuvien neuroverkkojen arkkitehtuureja. Arkkitehtuureista tarkastellaan lähemmin monikerrosperceptronia, joka on ehkä yleisin käytetty verkkoarkkitehtuuri. Lopuksi pohditaan vielä tiedon esittämistä neuroverkoille, sekä neuroverkkojen soveltumista kielenhäiriöiden mallintamiseen.

2.1. Neuroverkko

Vastaavasti kuin ihmisaivojen perusyksikkönä on hermosolu, neuroverkkojen perusyksikkönä on (keinotekoinen) neuroni. Neuroni koostuu *syötekanavista* (input channels) ja *tuloskanavasta* (output channel) sekä *aktivaatiofunktiosta* (activation function). Aktivaatiofunktio laskee neuronin tuloksen sen saamien syötteiden perusteella. Jokaiseen syötekanavaan liittyy *paino* (weight), jolla esitetään syötekanavasta saapuvan syötteen voimakkuus. Lisäksi neuroniin liittyy *siirtotermi* (threshold, bias), jolla säädellään neuronin aktivaativoitumistasoa, sekä *summain* (adder), jolla lasketaan neuroniin saapuvat syötteet.

Haykinin [1994] mukaan edellä mainituista käsitteistä yhden neuronin toiminnan kannalta tärkeimpiä ovat:

1. Neuronin syötekanavat, joista jokaisella on oma painoarvo.
2. Summain, joka laskee yhteen neuroniin saapuvat painollaan kerrotut syötteet.
3. Aktivaatiofunktio, joka laskee neuronin tuloksen. Yleensä käytetään aktivaatiofunktioita, jotka rajoittavat neuronin tuloksen suljetulle välille [0,1] tai [-1,1].

Matemaattisesti kuvattuna neuronin toimintaa voidaan mallintaa seuraavasti. Jokainen neuroniin saapuva reaalilukusyöte kerrotaan sitä vastaavan syötekanavan painolla. Näin saadut painotetut syötteet lasketaan yhteen, jolloin saadaan neuroniin saapuva kokonaissyöte. Kokonaissyöte u_i neuronille i voidaan siis laskea kaavalla

$$u_i = \sum_{j=1}^{n} w_{ij} x_j \,, \tag{2.1}$$

missä n on neuroniin kytkettyjen syötekanavien lukumäärä, w_{ij} syötekanavan j painoarvo ja x_j syötekanavaa j pitkin saapuva syöte. Kaava (2.1) vastaa siis edellä esitetyistä peruskäsitteistä summainta. Olkoon neuronin aktivaatiofunktiona (reaali)funktio φ. Neuronin i tulos y_i saadaan laskemalla aktivaatiofunktion arvo syötteellä $u_i - \theta$, missä θ on siirtotermi, toisin sanoen

$$y_i = \varphi(u_i - \theta)\,. \tag{2.2}$$

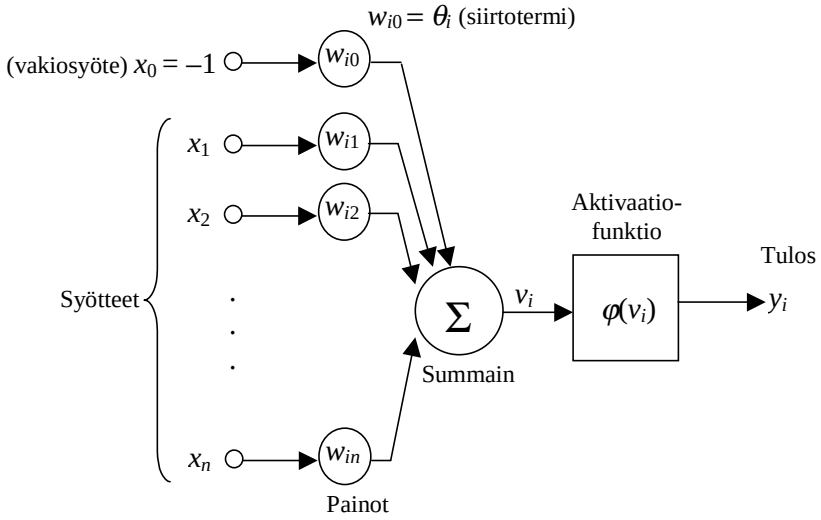
Periaatteessa aktivaatiofunktio φ voisi olla mikä tahansa reaalifunktio, mutta käytännössä vain muutamia aktivaatiofunktiotyyppejä käytetään [Rojas, 1996]. Näitä aktivaatiofunktiota esitellään tarkemmin luvussa 2.2.1. Siirtotermin θ laskeminen voidaan yhdistää neuroniin saapuvan kokonaissyötteen laskemiseen lisäämällä neuroniin vakiosyöte $x_0 = -1$ ja paino $w_{i0} = \theta_i$. Tällöin yhtälöt (2.1) ja (2.2) saadaan muotoon

$$v_i = \sum_{j=0}^{n} w_{ij} x_j \tag{2.3}$$

ja

$$y_i = \varphi(v_i)\,. \tag{2.4}$$

Kuva 2.1 havainnollistaa neuronin i toimintaa kaavojen (2.3) ja (2.4) pohjalta.



Kuva 2.1. Neuronin *i* toiminta.

Nimensä mukaisesti neuroverkko koostuu toisiinsa kytketyistä neuroneista. Neuroverkkoarkkitehtuureita on useita, mutta kaikkien perusrakennuspalikkoina on edelläkuvatun kaltaiset neuronit. Näistä rakennuspalikoista voidaan kuitenkin koota hyvinkin erilaisia ja erityyppisiin tehtäviin soveltuvia neuroverkkoja vaihtelemalla neuroverkon neuroneissa käytettävää aktivaatiofunktiota ja neuronien välisiä kytkentöjä.

2.2. Neuroverkkojen luokittelua

Neuroverkkoja voidaan luokitella ainakin niiden rakenteen sekä niillä suoritettavien tehtävien perusteella. Rakenteen perusteella tapahtuva luokittelu voidaan tehdä seuraavien kriteerien perusteella [Rojas, 1996]:

1. Neuronin rakenne,
2. Verkon arkkitehtuuri,
3. Oppimisalgoritmi, joka määrää verkon painot.

Bechtel ja Abrahamsen [1991] lisäävät listaan vielä kohdan "neuroverkon semanttinen tulkinta". Neuronin rakenteella tarkoitetaan pääasiassa sen aktivaatiofunktiota. Verkon arkkitehtuurilla tarkoitetaan neuroneiden lukumäärää ja niiden keskinäisiä kytköksiä. Oppimisalgoritmin tehtävänä on säätää neuronien välisten yhteyksien painoarvot siten, että neuroverkko tuottaa tietyillä syötteillä haluttuja vastauksia. Kaikissa neuroverkkoarkkitehtuureissa oppimisalgoritmilla ei ole merkittävää roolia (esim. luvussa 3.4 esiteltävät neuroverkot). Bechtel ja Abrahamsen tarkoittavat neuroverkkojen semanttisella tulkinnalla neuroverkon yksittäisten neuronien merkityksen tulkintaa. Tulkinnan vaikeus riippuu käytettävästä neuroverkkotyypistä.

Rakenteellisen luokittelun lisäksi neurolaskentamalleja voidaan luokitella myös niillä suoritettavien tehtävätyyppien mukaan. Tällä tavalla mallit voidaan jakaa neljään luokkaan [Fu, 1994]:

1. Luokittelumallit,
2. Assosiaatiomallit,
3. Optimointitehtäviä suorittavat mallit,
4. Itseorganisoituvat mallit.

Luokittelutehtäviä suorittavat mallit luokittelevat saamansa syötteen yhteen äärellisestä määrästä kategorioita. Esimerkkejä luokittelutehtävistä ovat mm. loogisia operaatiota suorittavat neuroverkot. Tällaiset neuroverkot luokittelevat saamansa syötteen joko tosiin tai epätosiin riippuen toteutettavasta loogisesta operaatiosta.

Assosiaatiomallit suorittavat joko *autoassosiaatiota* (autoassociation) tai *heteroassosiaatiota* (heteroassociation). Autoassosiaatiossa neuroverkko uudelleentuottaa sille esitetyn syötehahmon. Heteroassosiaatiossa verkko liittää syötehahmoon jonkin toisen hahmon tuloshahmojen joukosta. Tällöin tulos- ja syötejoukot ovat usein erillisiä.

Optimointitehtäviä suorittavat mallit yrittävät löytää tietylle ongelmalle parhaan mahdollisen ratkaisun. Tämä tehdään usein minimoimalla ongelmalle määriteltyä kustannusfunktiota [Fu, 1994]. Eräs esimerkki tunnetusta neuroverkoille soveltuvasta optimointiongelmasta on kauppamatkustajan ongelma.

Itseorganisoituvien mallien avulla on mahdollista järjestää tai luokitella mallille annettavia syötehahmoja, kun varsinainen luokitus on tuntematon. Itseorganisoituvia malleja voidaan myös käyttää sellaisten tehtävien suoritttamiseen, joiden algoritminen määrittäminen on erittäin hankalaa [Fu, 1994].

Tässä luvussa keskitytään kuitenkin luokittelemaan neuroverkkoja niiden rakenteiden perusteella. Tällöin edellä kuvattu Rojasin [1996] luokittelu soveltuu käytettäväksi myös seuraavissa aliluvuissa.

2.2.1. Neuronin rakenne: aktivaatiofunktio

Neuronin rakenteen kannalta tärkein osa neuronia on sen aktivaatiofunktio. Neuroverkkojen aktivaatiofunktiot $\varphi(\cdot)$ voidaan jakaa kolmeen luokkaan [Haykin, 1994]:

1. *Askelfunktiot* (threshold function),
2. *Paloittain lineaariset funktiot* (piecewise-linear function),
3. *Sigmoidifunktiot* (sigmoid function).

Askelfunktiota on käytetty ensimmäisistä neurolaskentamalleista lähtien. Se määritellään kaavalla

$$\varphi(v) = \begin{cases} 1, & \text{jos } v \geq 0 \\ 0, & \text{jos } v < 0 \end{cases}, \tag{2.5}$$

missä v on neuroniin saapuva kokonaissyöte. Askelfunktio antaa siis tuloksen 1, jos sen syöte on positiivinen. Muutoin sen tulos on = 0. Askelfunktio noudattaa siis "kaikki tai ei mitään" -periaatetta aktivoitumisen yhteydessä.

Paloittaan lineaarinen funktio määritellään vuorostaan kaavalla

$$\varphi(v) = \begin{cases} 1, & \text{jos } v \geq \frac{1}{2} \\ v, & \text{jos } -\frac{1}{2} < v < \frac{1}{2} \\ 0, & \text{jos } v \leq -\frac{1}{2} \end{cases}. \tag{2.6}$$

Paloittain lineaarinen funktio antaa siis tuloksen 1, jos sen syöte $v \geq ½$, tuloksen 0, jos $v \leq –½$, ja tuloksen v muutoin. Paloittain lineaarinen aktivaatiofunktio lähestyy askelfunktiota, jos väliä, jolla funktion arvot määräytyvät sen lineaarisen komponentin perusteella, pienennetään rajatta. Toisaalta paloittain lineaarinen askelfunktio lähestyy lineaarista aktivaatiofunktiota, jos väliä, jolla funktion arvot määräytyvät sen lineaarisen komponentin perusteella, kasvatetaan rajatta.

Sigmoidifunktiosta käytetään yleisesti kahdenlaisia versioita [Haykin, 1994]. Ensimmäinen on logistinen funktio, jonka arvot sijoittuvat välille [0,1]. Se määritellään kaavalla

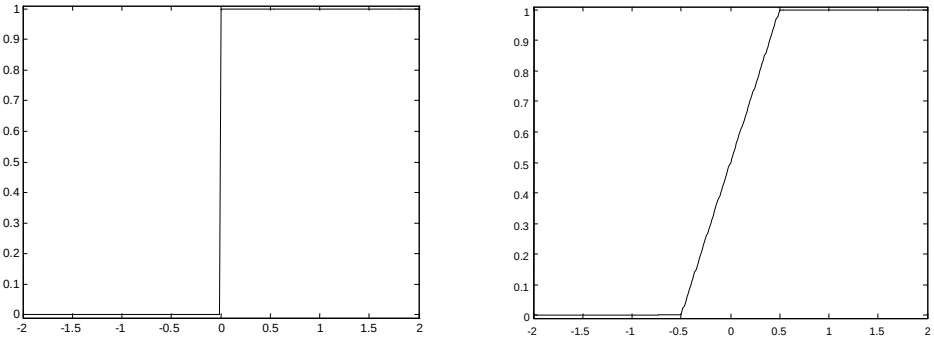
$$\varphi(v) = \frac{1}{1+e^{-av}}, \tag{2.7}$$

missä a on logistisen funktion leveyttä määräävä parametri. Kun vakion a annetaan kasvaa rajatta logistinen funktio lähestyy askelfunktiota. Jos neuronien tulosten halutaan sijoittuvan välille [–1,1], logistinen funktio on tähän tarkoitukseen riittämätön. Tällöin sigmoidifunktiona käytetään hyperbolista tangettifunktiota

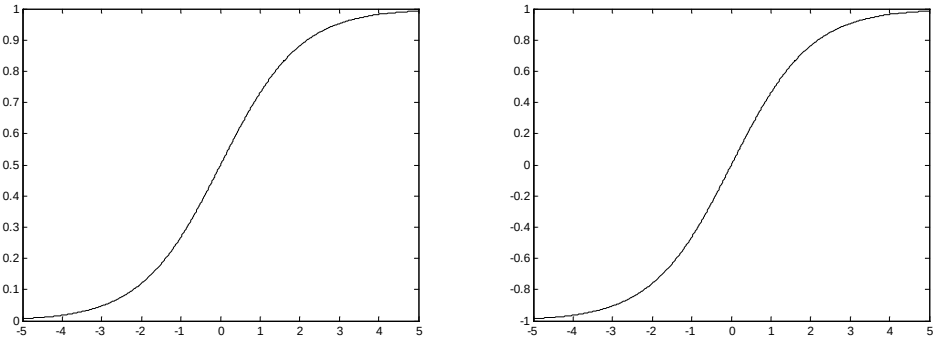
$$\varphi(v) = \tanh(v/2) = \frac{1-e^{-v}}{1+e^{-v}}. \tag{2.8}$$

Myös muut aktivaatiofunktiotyypit voidaan yleistää välille $v \in [–1,1]$.

Sigmoidifunktioista on tullut yleisesti käytettyjä aktivaatiofunktiota back-propagation -algoritmin kehittämisen myötä, joka vaatii jatkuvasti derivoituvan aktivaatiofunktion. Tätä ominaisuuttahan ei ole funktioilla (2.5) ja (2.6). Esimerkiksi eteenpäin syöttävät monikerrosperceptron-neuroverkot (ks. luku 2.3) käyttävät sigmoidiaktivaatiofunktiota. Funktioiden (2.5) - (2.8) kuvaajia esitellään kuvissa 2.2 ja 2.3.



Kuva 2.2. Askelfunktio sekä paloittain lineaarinen funktio.



Kuva 2.3. Sigmoidifunktioita: logistinen funktio ja hyperbolinen tangenttifunktio.

2.2.2. Verkon arkkitehtuuri

Verkon arkkitehtuurilla tarkoitetaan neuroverkon neuronien lukumäärää ja niiden keskinäisiä kytkentöjä. Arkkitehtuurin tarkasteleminen mahdollistaa neuronien jakamisen *kerroksiin* (layers) niiden tehtävien mukaan. Yksinkertaisimmissa kerroksellisissa neuroverkoissa on vain syöte- ja tuloskerros. Syötekerroksen neuronit välittävät saamansa syötteen eteenpäin sellaisenaan seuraavalle kerrokselle. Tuloskerroksen neuronit taas toimivat luvussa 2.1 esitetyllä tavalla. Monimutkaisemmissa arkkitehtuureissa voi olla lisäksi *piilokerroksia* (hidden layers), jotka välittävät aktivaatiota syöte- ja tuloskerroksen neuronien välillä. Myös nämä neuronit toimivat luvussa 2.1 esitetyllä tavalla. Piilokerrosten avulla neuroverkkojen laskentakapasiteettia voidaan parantaa.

Haykin [1994] jakaa neuroverkkoarkkitehtuurit kolmeen osaan: *eteenpäin syöttäviin neuroverkkoihin* (feedforward networks), *rekursiivisiin neuroverkkoihin* (recurrent networks) sekä *hilarakenteisiin neuroverkkoihin* (lattice structured networks). Eteenpäin syöttävillä neuroverkoilla tarkoitetaan verkkoja, joissa neu-

ronien lähettämät tulosimpulssit liikkuvat aina samaan suuntaan. Toisin sanoen neuronit lähettävät tuloksensa aina sellaisille seuraavan kerroksen neuroneille, joihin ne on yhdistetty. Eteenpäin syöttävät neuroverkot voivat olla joko *yksikerroksisia* (single-layered) tai *monikerroksisia* (multi-layered). Yksikerroksisissa eteenpäin syöttävissä neuroverkoissa on vain syötekerros ja tuloskerros. Monikerroksisissa eteenpäinsyöttävissä neuroverkoissa on syöte- ja tuloskerrosten välissä yksi tai useampi piilokerros. Eteenpäinsyöttävää neuroverkkoa, jossa on p syöteneuronia, h_1, h_2, …, h_l neuronia piilokerroksissa 1,2,…,l ja q neuronia tuloskerroksessa, kutsutaan p-h_1-h_2-…-h_l-q neuroverkoksi. Lisäksi sen sanotaan olevan *täysin kytketty* (fully connected), jos jokaisen kerroksen jokainen neuroni on yhdistetty viereisen kerroksen jokaiseen neuroniin.

Eteenpäinsyöttävät neuroverkot ovat erittäin suosittuja neuroverkkomalleja. Suosion takaa niiden yksinkertaisuus verrattuna moniin muihin neuroverkkoarkkitehtuureihin ja soveltuvuus monentyyppisten ongelmien ratkaisemiseen. On nimittäin todistettu, että mikä tahansa jatkuva funktio

$$f\colon [0,1]^n \rightarrow \mathbb{R}^m,\ f(\mathbf{x}) = \mathbf{y},$$

voidaan toteuttaa yhdellä piilokerroksella varustetulla eteenpäinsyöttävällä neuroverkolla. Verkossa tulee olla n syöteneuronia, $2n$+1 piiloneuronia sekä m tulosneuronia [Hecht-Nielsen, 1990]. Tämä nk. Kolmogorovin lause ei kuitenkaan anna tarkempaa tietoa kaikkien verkon neuronien aktivaatiofunktioiden valinnasta. Koska muutakaan menetelmää sopivan aktivaatiofunktion löytämiseksi ei tunneta, kyseessä on vain teoreettisessa mielessä merkittävä tulos [Hecht-Nielsen, 1990].

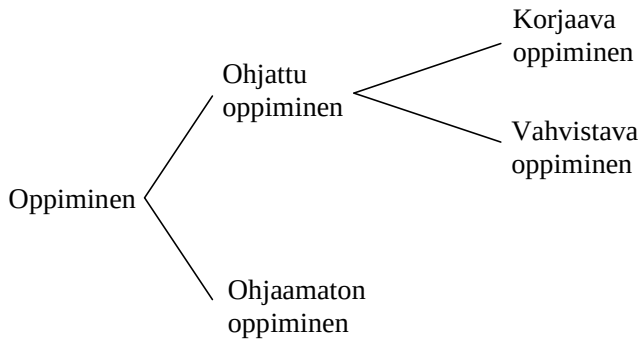
Rekursiivisissa neuroverkoissa on eteenpäinsyöttävien yhteyksien lisäksi ainakin yksi takaisinpäin syöttävä yhteys. Rekursiiviset yhteydet voivat olla neuronien välisiä tai itseissilmukoita, jolloin neuroni saa käyttöönsä edellisen tuloksensa. Takaisinpäin syöttävillä yhteyksillä on vaikutusta verkon oppimis- ja suorituskykyyn [Haykin, 1994]. Tunnetuimpia rekursiivisiin yhteyksiin perustuvia neuroverkkoja ovat mm. Boltzmannin kone ja Hopfieldin verkko.

Hilarakenteiset neuroverkot ovat eteenpäin syöttäviä neuroverkkoja, joiden tuloskerroksen neuronit on järjestetty riveihin ja sarakkeisiin [Haykin, 1994].

2.2.3. Oppiminen ja oppimisalgoritmit

Neuroverkkojen yhteydessä oppimista voidaan tarkastella kahdella eri tasolla. Ylempi taso on oppimisparadigmojen taso, jossa luonnehditaan oppimisen yleisiä ominaisuuksia. Paradigmataso voidaan nähdä myös oppimisalgoritmien karkeana luokituksena. Tarkasteltaessa oppimista alemmalla tasolla keskitytään tiettyyn oppimisparadigmaan kuuluvien oppimisalgoritmien toteutukseen ja niiden välisiin eroihin.

Rojas [1996] on luokitellut oppimisparadigmat kuvan 2.4 mukaisella tavalla. Ohjattu oppiminen tarkoittaa menetelmää, jossa neuroverkolle opetetaan kuvaus syöte- ja tulosvektoreiden välille opettajan avulla. Neuroverkon painoarvoja muutetaan opettajan antaman palautteen mukaan, joka riippuu neuroverkon tuottamasta tuloksesta. Opettajan tietomäärästä riippuu, luokitellaanko algoritmi *korjaavaksi* (corrective learning) vai *vahvistavaksi* (reinforcement learning) oppimiseksi [Rojas, 1996]. Korjaavaa oppimista voidaan käyttää, kun opettajalla on täsmällinen tieto neuroverkon tekemän virheen suuruudesta. Jos opettajalla on vain tieto neuroverkon tuottaman vastauksen oikeellisuudesta, tulee käyttää vahvistavaa oppimista. Ohjatun oppimisen tavoitteena on saada verkko jäljittelemään opettajaa [Haykin, 1994].



Kuva 2.4. Oppimisparadigmojen luokittelua [Rojas, 1996].

Ohjaamattomalla oppimisella tarkoitetaan tilannetta, jossa neuroverkolla ei ole käytettävissä ulkopuolista opettajaa. Tällöin verkolle esitettäville syötevektoreille ei ole tiedossa tulosvektoreita, vaan sen tulee itse muodostaa tuo kuvaus. Yleisesti ohjaamattomaan oppimiseen pohjautuvat algoritmit ovat epätarkempia kuin ohjattuun oppimiseen pohjautuvat algoritmit. Kuitenkin ne ovat usein laskennallisesti kevyempiä, joten niitä käytetään vähäisemmästä tarkkuudesta huolimatta esimerkiksi reaaliaikasovelluksissa ohjatun oppimisen sijasta [Fu, 1994]. Lisäksi ohjattua oppimista ei luonnollisestikaan voida käyttää tilanteissa, joissa ei ole tietoa, minkälaisia vastauksia neuroverkon pitäisi esitetyille syötteille tuottaa.

Haykin [1994] luokittelee oppimisalgoritmeja seuraavasti: virhettä korjaava oppiminen, Boltzmannin oppiminen, Thorndiken lain mukainen oppiminen, Hebbin oppiminen ja kilpaileva oppiminen. *Virhettä korjaavassa* oppimisessa algoritmi laskee erotuksen haluttujen vastausten ja neuroverkon tuottamien vastausten välillä ja säätää neuroverkon painoja suhteessa virheeseen. Tätä proseduuria toistamalla neuroverkon tekemä virhe pienenee, jolloin neuroverkko oppii kuvauksen syötteen ja tuloksen välillä. Esimerkiksi MLP-neuroverkot käyttävät virhettä korjaavaa back-propagation algoritmia.

Boltzmannin oppiminen on termodynamiikan menetelmiin pohjautuva tilastollinen oppimisalgoritmi. Boltzmannin oppimista käytetään tietyn neuroverkkotyypin, Boltzmannin koneiden, opettamiseen. Boltzmannin koneen neuronit ovat binaarisia neuroneja, jotka vaihtavat tilaansa tietyn todennäköisyyden mukaisesti. Boltzmannin kone soveltuu hyvin mm. autoassosiaatiotehtävien suorittamiseen [Haykin, 1994].

Thorndiken lain mukainen oppiminen on eräs vahvistavan oppimisen muoto. Oppimisalgoritmi perustuu ajatukseen, että jos oppivan järjestelmän toiminta on tuottanut halutun tuloksen, pyritään järjestelmän todennäköisyyttä toimia tällä tavalla kasvattamaan. Päinvastaisissa tapauksissa todennäköisyyttä pyritään pienentämään [Haykin, 1994]. Thorndiken lain mukainen oppiminen eroaa virhettä korjaavasta oppimisesta siinä, että Thorndiken lain mukaisen oppimisen yhteydessä ei tarvita tietoa tehdyn virheen suuruudesta [Rojas, 1996].

Hebbin oppimisessa vahvistetaan sellaisten neuronien välisiä yhteyksiä, jotka ovat aktivoituneena yhtäaikaisesti. Eri aikaan aktivoituneiden neuronien välisiä yhteyksiä heikennetään. Hebbin oppimissääntö on oppimissäännöistä vanhin ja kuuluisin [Haykin, 1994] ja se on esimerkki vahvistavasta oppimissäännöstä [Rojas, 1996].

Kilpaileva oppiminen perustuu neuronien väliseen kilpailuun aktivoitumisesta. Toisin kuin muissa oppimissäännöissä, kilpailevan oppimisen yhteydessä vain yksi neuroni kerrallaan voi olla aktivoitunut. Opetuksen alussa verkon painoarvot on alustettu pieniksi satunnaisluvuiksi, jolloin eri neuronit vastaavat eri tavalla samaan syötteeseen. Kilpailevan oppimisen tarkoituksena on luoda eri syötetyypeille erikoistuneita neuroneita, jotka vastaavat (eli voittavat kilpailun aktivoitumisesta) vain omiin syötetyyppeihinsä. Kilpailevaa oppimista käytetään ohjaamattoman oppimisen yhteydessä [Rojas, 1996].

Vaikka oppiminen onkin keskeinen alue neurolaskennassa, on kuitenkin neuroverkkoja, joita ei ole suunniteltu oppiviksi. Tällöin sopivat verkon painoarvot ovat tiedossa jo etukäteen, joten niitä ei tarvitse erikseen verkolle opettaa. Tällaisia verkkoja ovat mm. luvussa 3.4 esiteltävät nimeämistä simuloivat neuroverkkomallit. Usein oikeiden parametriasetusten päätteleminen on kuitenkin erittäin vaikeaa, ellei jopa mahdotonta, joten oppivien neuroverkkojen käyttäminen on välttämätöntä. Lisäksi monesti halutaan, että neuroverkko muodostaa itsenäisesti (opettajan avulla tai ilman opettajaa) kuvauksen syöteja tulosvektorien välillä. Näin on esimerkiksi silloin, kun neuroverkon avulla halutaan löytää syötteiden joukoista yhteneväisyyksiä ja eroavaisuuksia, joita aikaisemmin ei ole havaittu. Tällöin oppimisesta muodostuu tärkeä ja keskeinen osa neuroverkon toimintaa.

2.2.4. Neuroverkon semanttinen tulkinta

Neuroverkkomallit voidaan semanttisen tulkinnan suhteen jakaa kahteen osaan: *lokaalisiin* (localist) ja *hajautettuihin* (distributed) [Bechtel and Abrahamsen, 1991]. Lokaaleissa neuroverkoissa yksi neuroni vastaa yhtä simulaatiossa esiintyvää käsitettä. Hajautetuissa neuroverkoissa käsiteen esitys on hajautettu usean neuronin kesken. Neuroverkkojen semanttiseen tulkintaan vaikuttaa lisäksi verkon oppivuus.

Luvussa kolme käsiteltävät neuroverkot ovat kumpikin lokaaleja, eli yhtä simulaatiossa mukana olevaa käsitettä vastaa yksi verkon neuroni. Näissä neuroverkoissa esimerkiksi kunkin sanan lemmaa vastaa yksi neuroni. Toinen esimerkki näiden verkkojen alemmilta tasoilta on neuroni, joka vastaa tiettyä foneemia (esim. /*k*/). Tämä neuroni aktivoituu aina kun foneemi /*k*/ esiintyy sanassa.

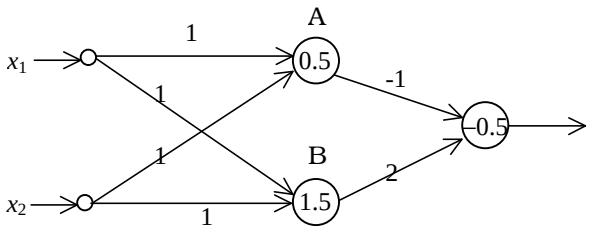
Hajautettua esitystä käyttävän verkon tapauksessa yhtä käsitettä vastaa usean neuronin aktivoituminen ja verkkoon tallennetut käsitteet voivat aiheuttaa osittain samojen neuronien aktivoitumisen. Tällöin yksi neuroni vastaa piirrettä, joka voi olla yhteinen usealle käsitteelle. Hajautettua esitystä käyttävässä nimeämistä simuloivassa neuroverkossa foneemi /*k*/ esitettäisiin piirteidensä avulla, jotka jakaantuisivat usean neuronin osalle. Luvuissa neljä ja viisi käsiteltävä Learning SLIPNET-neuroverkko on esimerkki hajautetusta neuroverkkoarkkitehtuurista.

Oppivan verkon käyttäminen lisää semanttisen tulkinnan vaikeutta. Tällöin osalle verkon neuroneista ei ole (välttämättä) annettu mitään ennaltamäärättyä merkitystä, vaan merkitys muodostuu oppimisen myötä. Vaikka oppivan neuroverkon myötä suunnittelija vapautuukin verkon painoarvojen asettamisesta, ongelmaksi muodostuu käsitteiden esitys neuroverkolle. Ongelma siirtyy siis vain toiselle tasolle, sillä sopiva koodauksen muodostaminen voi olla erittäin hankalaa. Koodauksessa tulisi ilmetä kohdealueen olioiden todelliset samankaltaisuudet ja erilaisuudet mahdollisimman tarkasti, jotta verkolla olisi mahdollisuus muodostaa tarkka kuvaus kohdealueesta. Koodaukseen liittyviä ongelmia sekä koodaustapoja esitellään luvussa 2.4.

2.2.5. Esimerkki

Esimerkkinä tarkastellaan kuvan 2.5 neuroverkkoa. Kuvassa ympyröillä merkitään neuroneja ja viivoilla niiden välisiä yhteyksiä. Verkon rakenne voidaan jakaa kerroksiin siten, että vasemmanpuoleisimpia neuroneita (pienet ympyrät) kutsutaan syötekerrokseksi, keskimmäisiä (kuvassa neuronit A ja B) piilokerrokseksi, ja oikeanpuoleisinta kutsutaan tuloskerrokseksi. Yhteyksissä esiintyvät luvut ovat niiden painoja ja neuroneissa esiintyvät luvut taas ovat

niihin liittyviä siirtotermejä. Syötekerroksen neuroneilla ei ole kynnysarvoja, sillä ne välittävät kokonaissyötteensä eteenpäin sellaisenaan. Kuvan 2.5 verkossa on siis kaksi syötekerroksen neuronia, kaksi piilokerroksen neuronia ja yksi tuloskerroksen neuroni. Neuronien väliset yhteydet ovat suunnattuja vasemmalta oikealle, joten verkko on eteenpäinsyöttävä 2-2-1 neuroverkko.



Kuva 2.5. Ekvivalenssiongelman ratkaiseva neuroverkko.

Esimerkkiverkko suorittaa loogisen ekvivalenssioperaation. Verkko saa syötteekseen kaksiulotteisia bittivektoreita (esim. (0,1)) ja antaa tuloksena joko bitin 1 tai 0, loogisen ekvivalenssin määritelmän mukaisesti. Aktivaatiofunktiona φ toimii kaavan (2.5) mukainen askelfunktio. Aktivaatiofunktion syöte v_i neuronille i lasketaan kaavalla (2.3) ja funktio antaa tuloksen 1, jos v_i on aidosti positiivinen, muutoin tuloksen 0. Koska verkon painoarvot on ennalta asetettu, verkko ei ole oppiva.

Tarkastellaan verkon toimintaa syötteellä $(x_1,x_2)=(0,0)$. Tällöin verkon laskeman tuloksen pitäisi loogisen ekvivalenssin määritelmän mukaan olla =1. Syötekerroksen neuronit (kuvassa 2.5 vasemmanpuoleisimmat neuronit) saavat siis kummatkin syötteekseen 0. Syötekerroksen neuroneina ne välittävät syötteensä suoraan seuraavalle kerrokselle. Nyt piilokerroksen neuroneille saadaan aktivaatiofunktiolla (2.5) tulokset

$$\varphi(1\cdot 0 + 1\cdot 0 - 0.5) = \varphi(-0.5) = 0 \text{ ja} \qquad \text{(neuronin A tulos)}$$

$$\varphi(1\cdot 0 + 1\cdot 0 - 1.5) = \varphi(-1.5) = 0. \qquad \text{(neuronin B tulos)}$$

Siis myös piilokerroksen neuronien tulokset ovat nollia. Nyt tuloskerroksen (oikeanpuoleisin neuroni) tulos saadaan piilokerroksen neuroneiden tuloksista laskemalla

$$\varphi(-1\cdot 0 + 2\cdot 0 - (-0.5)) = \varphi(0.5) = 1,$$

mikä on myös verkon tulos syötteellä (0,0). Kaikkia syötteitä vastaavat tulokset ovat taulukossa 2.1.

Syöte	Piilokerros		Tuloskerros
	A	B	
(0,0)	0	0	1
(0,1)	1	0	0
(1,0)	1	0	0
(1,1)	1	1	1

Taulukko 2.1. Kuvan 2.5 neuroverkon neuronien tulokset.

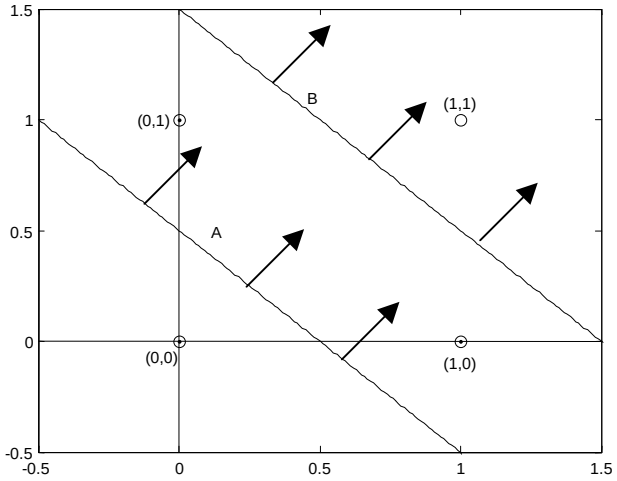
Kun taulukon 2.1 tuloksia ja verkon rakennetta vertaa toisiinsa huomaa, että piilokerroksen neuroni A tunnistaa tapaukset, joissa syötteenä on (1,0), (0,1) tai (1,1). Neuroni B taas tunnistaa tapaukset, joissa syötteessä kumpikin bitti on 1. Neuroni A tunnistaa siis loogisen tai-operaation ja neuroni B loogisen ja-operaation. Käytännössä neuroni A laskee siis epäyhtälön

$$x_1 + x_2 - 0.5 \geq 0$$

ja neuroni B epäpäyhtälön

$$x_1 + x_2 - 1.5 \geq 0.$$

Näiden neuronien tulokset yhdistämällä tuloskerroksen neuroni voi ratkaista ekvivalenssi-ongelman antamalla tuloksen 1, jos kumpikin piilokerroksen neuroni antaa saman tuloksen, ja tuloksen 0 muutoin. Verkon piilokerroksen neuronien päätösrajoja on havainnollistettu kuvassa 2.6.



Kuva 2.6. Neuroverkon piilokerroksen päätösrajat.

2.3. Monikerrosperceptron

Neurolaskenta-arkkitehtuureista todennäköisesti tunnetuin ja käytetyin on *monikerrosperceptron* (Multi-Layer Perceptron, MLP) [Fu, 1994]. Arkkitehtuuri on tärkeä myös tämän tutkielman kannalta, sillä luvussa neljä esiteltävä nimeämishäiriöitä simuloiva Learning SLIPNET-neuroverkko toteutetaan niiden avulla.

Monikerrosperceptronit pohjautuvat Frank Rosenblatin vuonna 1958 esittelemään perceptron-neuroniin, jonka toimintaa on myöhemmin yleistetty. Rosenblat ehdotti yleiseksi laskennan malliksi perceptron-neuronia, joka on kynnysfunktiota (kaava (2.5)) käyttävä neuroni [Rojas, 1996]. Perceptroneja tutkittiinkin aktiivisesti, ja niiden puutteeksi havaittiin, että ilman piilokerrosta toimiva eteenpäinsyöttävä perceptron-neuroverkko kykenee ratkaisemaan vain lineaarisesti erottuvia ongelmia. Kuuluisa esimerkki lineaarisesti erottumattomasta ongelmasta on XOR-ongelma. Toinen esimerkki on luvussa 2.2.5 esitetty ekvivalenssiongelma, joka on XOR-ongelman negaatio.

Edellä esitetyllä huomiolla ei sinänsä ollut mitään suurempaa merkitystä, sillä käyttämällä yhtä tai useampaa piilokerrosta lineaarisesti erottuvat ongelmat voidaan ratkaista. Perceptroneille ei kuitenkaan tunnettu algoritmia, jolla monikerroksisia perceptron-neuroverkkoja olisi voitu opettaa. Tämä pysäytti perceptrontutkimuksen miltei koko 1970 -luvuksi [Haykin, 1994].

Perceptroneja vaivannut oppimisongelma ratkaistiin usealla taholla 1980-luvulla. Perceptron-neuronin aktivaatiofunktiona alettiin käyttää sigmoidifunktiota (kaava (2.7)), jolloin uuden oppimisalgoritmin kehittäminen tuli mahdolliseksi. Kehitetty algoritmi tunnetaan back-propagation-algoritmina.

MLP-neuroverkot ovat siis eteenpäinsyöttäviä neuroverkkoja, jotka koostuvat syöte- ja tuloskerroksen lisäksi ainakin yhdestä piilokerroksesta [Haykin, 1994]. Piilo- ja syötekerrosten neuronien aktivaatio lasketaan sigmoidiaktivaatiofunktiolla (kaavat (2.7) ja (2.8)). Verkkoja opetetaan back-propagation-algoritmilla, josta on tullut yleisimmin sovellettu neuroverkkojen opetusalgoritmi.

MLP -neuroverkkojen suosio johtuu niiden soveltuvuudesta monenlaisiin tehtäviin. Lisäksi on voitu todistaa, että kolmen piilokerroksen MLP-neuroverkolla voidaan approksimoida kaikkia "mielenkiintoisia" funktioita

$$f\colon [0,1]^n \to \mathbb{R}^m,\ f(\mathbf{x}) = \mathbf{y}.$$

"Mielenkiintoisiin" funktioihin kuuluvat mm. kaikki jatkuvat funktiot ja sellaiset paloittain jatkuvat funktiot, jotka koostuvat äärellisestä määrästä paloja. Täten ulkopuolelle jäävillä funktioilla onkin usein vain matemaattista merkitystä [Hecht-Nielsen, 1990]. Tulos ei kerro mitään siitä, kuinka paljon MLP-neuroverkossa tulisi olla neuroneita, jotta jonkin tietyn funktion approksimointi onnistuisi. On syytä huomata, että luvussa 2.2.2 esitetty vastaavan kaltainen tulos koski yleisesti eteenpäinsyöttäviä neuroverkkoja, myös MLP-neuroverkkoja. Tässä esitetty tulos koskee sen sijaan vain logistista aktivaatiofunktiota käyttävää MLP-neuroverkkoa.

Esimerkkejä sovelluksista, joissa MLP-neuroverkoilla on ollut keskeinen rooli, on useita. Eräs tunnetuimmista sovelluksista on Sejnowskin ja Rosenbergin [1987] esittelemä NETtalk neuroverkkojärjestelmä, joka oppi lukemaan englanninkielistä tekstiä. Järjestelmän ytimenä on MLP-neuroverkko, joka suorittaa kuvauksen kirjainten ja foneemien välillä. Sejnowski ja Rosenberg havaitsivat NETtalk-järjestelmän käyttäytymisessä muutamia samankaltaisia piirteitä ihmisten kanssa. Näitä olivat mm. järjestelmän yleistyskyvyn paraneminen sen oppiessa uusia sanoja, vaurion kestävyys sekä nopea uudelleenoppiminen vaurion jälkeen.

NETtalk-järjestelmän lisäksi MLP-neuroverkkoja on sovellettu onnistuneesti mm. puheentunnistukseen, käsinkirjoitetun tekstin tunnistamiseen, tutka- ja kaikuluotainsignaalien tunnistamiseen ja luokitteluun sekä monille muille alueille [Haykin, 1994]. Yksistään jo erilaisten sovellusalueiden määrä kertoo MLP-neuroverkkojen soveltuvuudesta mitä erilaisimpien ilmiöiden mallintamiseen.

2.4. Syötteiden ja tulosten koodaaminen neuroverkoille

Syötteiden ja tulosten koodaaminen neuroverkolle on olennainen osa neuroverkkomallin rakentamista. Tässä yhteydessä koodauksella tarkoitetaan verkolle esitettävien syötteiden muuttamista numeeriseen muotoon. Koodaaminen ei rajoita ongelmatyyppejä, joita neuroverkoilla voidaan ratkaista, sillä ainakin periaatteessa on mahdollista koodata kaikentyyppisiä ongelmia. Swinglerin [1996] mukaan koodausmenetelmät voidaan luokitella kolmeen eri kategoriaan:

1. *Paikallinen koodaus* (local encoding),
2. *Hajautettu koodaus* (distributed encoding),
3. *Karkea koodaus* (coarse encoding).

Paikallisessa koodauksessa jokaista verkolle opetettavan hahmon arvoa varten varataan yksi syötevektorin komponentti. Syötevektorin komponentit x_i voivat olla joko päällä ($x_i = 1$) tai pois päältä ($x_i = 0$). Kun hahmo esitetään verkolle, vain hahmon arvoa vastaava komponentti on päällä ja muut komponentit ovat pois päältä. Jos halutaan esittää useita hahmoja samaan aikaan, asetetaan hahmojen arvoja vastaavat komponentit päälle muiden komponenttien ollessa pois päältä.

Paikallisen koodauksen ongelma on tarvittavien syöteneuronien lukumäärä. Jos koodattavia hahmoja on N kappaletta ja jokainen hahmo voi saada k erillistä arvoa, tarvitaan k^N-dimensioinen syötevektori jokaisen mahdollisen syötearvon koodaamiseksi. Paikallinen koodaus onkin usein käyt-

tökelvoton tapa koodata syötteet tai tulokset, sillä sen käyttäminen johtaa helposti suuriin neuroverkkoihin.

Hajautetulla koodauksella on mahdollista vähentää syötevektorien dimensiota. Hajautetussa koodauksessa syötevektorissa on yksi komponentti x_i jokaista koodattavaa muuttujaa i kohti. Komponenttien arvot ovat jatkuvia ja kunkin komponentin arvo x_i koodaa hahmon aseman dimensiossa i. Täten koodauksen avulla on mahdollista muodostaa pienidimensioisia syötevektoreita.

Hajautetun koodauksen avulla ei kuitenkaan voi kunnolla esittää neuroverkolle useita hahmoja kerrallaan, sillä yksi muuttuja voi koodata vain yhden arvon kerrallaan. Tätä ongelmaa kutsutaan *sitomisongelmaksi* (binding problem) [Hinton *et al.*, 1986]. Ainoa tapa esittää useita hahmoja kerrallaan on keskiarvoistaa esitettävien hahmojen komponenttien arvot ja esittää saatu keskiarvovektori verkolle. Saatu keskiarvovektori esittääkin nyt uutta pistettä syöteavaruudessa, eikä siinä suoranaisesti ole tietoa usean syötevektorin yhtäaikaisesta esittämisestä. Hajautettu koodaus sopiikin parhaiten neuroverkkomalleille, joille esitetään vain yksi hahmo kerrallaan.

Kolmas koodausmenetelmä, eli *karkea koodaus*, on kompromissi paikallisen ja hajautetun koodauksen välillä. Tässä koodaustavassa syöteavaruus jaetaan useisiin päällekkäisiin r-säteisiin hyperpalloihin, joista jokaiseen liitetään yksi syötevektorin komponentti x_i. Komponentti asetetaan x_i=1, jos koodattava syötehahmo sijaitsee komponentia x_i vastaavan pallon keskipisteessä. Jos syötehamo ei sijaitse komponenttia x_i vastaavan pallon alueella on x_i=0. Muutoin $x_i \in (0,1)$ riippuen syötehahmon etäisyydestä komponenttia x_i vastaavan pallon keskipisteestä. Karkeasta koodauksesta saadaan sitä tarkempi, mitä suurempi syötevektorin komponenteja vastaavien pallojen säde r on. Tällöin kukin hahmo koodataan useiden syötevektorin komponenttien avulla [Hinton *et al.*, 1986].

Karkean koodauksen avulla on mahdollista ratkaista sitomisongelma, jos yhtäaikaa koodattavat hahmot ovat suhteellisen harvassa toisiinsa nähden. Jos koodattavat hahmot sijaitsevat edes osittain samoilla syötekomponenttien x_i alueilla, karkean koodauksen menetelmä ei toimi. Tämä puoltaisi siis komponentteja vastaavien pallojen säteen r valitsemista mahdollisimman pieneksi. Täten koodattaessa useita hahmoja yhtäaikaa täytyy tehdä kompromisseja koodauksen tarkkuuden ja resoluution suhteen.

2.5. Miksi neuroverkot soveltuvat kielenhäiriöiden mallintamiseen?

Neuroverkkomalleja pidetään usein soveliaampina ihmisten kognitiivisten prosessien mallintamiseen kuin symbolista prosessointia suorittavia malleja. Tämä

ei sinänsä ole yllättävää, sillä neuroverkkomallien esikuvina toimivat alunperin juuri ihmisaivot tai hermosto. Tämän vuoksi neurolaskentamalleja voidaan pitää kielellisen prosessoinnin tarkempina kuvauksina kuin perinteisiä malleja [Harley, 2001]. Koska kaikki neuroverkkojen yhteydessä esiintyvät käsitteet eivät kuitenkaan ole yhteneviä neurobiologian käsitteiden kanssa, on syytä puhua neuroverkoista biologisten mallien sijasta aivojen kognitiivisina malleina [Bechtel and Abrahamsen, 1991].

Kielenhäiriötä mallintavia neuroverkkoja tulisikin täten ensisijaisesti pitää kielellisen kyvyn kognitiivisina malleina. Kognitiiviset mallit tarjoavat käsitteellisen viitekehyksen, joiden avulla on mahdollista tutkia kielen mentaalista organisoitumista. Malleilla tai niiden osilla ei ole kielellisen kyvyn kannalta suoranaisia neurofysiologisia tai neuroanatomisia vastineita, vaan ne on tarkoitettu vastaamaan kielellisen kyvyn psykologisia käsitteitä [Martin *et al.*, 2002].

Kognitiivisen soveltuvuuden lisäksi neurolaskenta-arkkitehtuurien suuri etu on niiden vaurionkestävyys [Bechtel and Abrahamsen, 1991]. Tällä tarkoitetaan sitä, että vaikka joitakin neuronien välisistä yhteyksistä hieman muutettaisiin tai poistettaisiin kokonaan, voi mallilla tämän jälkeen kuitenkin laskea. Sama koskee neuronien poistoa. Vaikka mallin tulokset eivät ole vaurion jälkeen enää välttämättä oikeita, ovat ne kuitenkin oikeansuuntaisia. Täten vaurioitunutkin neuroverkko antaa paremman tuloksen kuin neuroverkko, jonka painot on asetettu täysin satunnaisesti (olettaen tietenkin, että vaurio ei ole täysin satunnaistanut verkon toimintaa). Esimerkiksi loogisen ekvivalenssin ratkaisevasta neuroverkosta voitaisiin poistaa joko neuroni A tai B. Jos neuroni A poistettaisiin, tulosneuroniin tulisi syötteitä vain neuronilta B. Tällöin verkon tulos olisi 1 kaikilla syötteillä (ks. luku 2.2.5).

Edellä kuvattua ominaisuutta, jossa neuroverkkojen suoritus heikkenee, mutta ei kokonaan katoa, kutsutaan *sulavaksi taantumaksi* (graceful degradation) [Bechtel and Abrahamsen, 1991]. Koska sulava taantuma on verkon rakenteesta periytyvä ominaisuus, on jokaisella neuroverkolla tämä ominaisuus. Myös aivot taantuvat sulavasti. Esimerkiksi aivovaurion yhteydessä ihmisen puheentuottokyky saattaa kärsiä, mutta se ei useinkaan katoa kokonaan. Aivovaurio-potilaan artikulaatio voi olla häiriintynyttä, tai potilaalla voi olla vaikeuksia löytää haluamiansa sanoja, mutta potilas pystyy silti puhumaan jollakin tavalla.

Kognitiivisia puheentuoton malleja käytettäessä tulee huomioida, että ne ovat puheentuoton teorioiden pohjalta rakennettuja malleja. Teoreettisia malleja käytettäessä joudutaan usein tekemään yksinkertaistavia tai idealisoituja oletuksia mallinnettavan ilmiön olemuksesta tai käyttäytymisestä, mutta

silti niiden uskotaan kuvaavan ainakin jossakin määrin oikein mallinnettavaa ilmiötä [Niiniluoto, 1980].

Yksinkertaistavilla oletuksilla saattaa olla vaikutusta mallin käyttäytymiseen, jolloin mallin antamat tulokset eivät välttämättä ole yhteneviä niitä vastaavien teorioiden kanssa. Toisaalta mallien avulla voidaan testata teorioiden paikkansapitävyyttä, jos uskotaan että yksinkertaistavilla oletuksilla ei ole liiallista vaikutusta mallin käyttäytymiseen [Martin *et al.*, 2002]. Tällaiset testit eivät välttämättä muuten olisi mahdollisia, sillä esimerkiksi testien eettiset seuraukset voisivat olla arveluttavia.

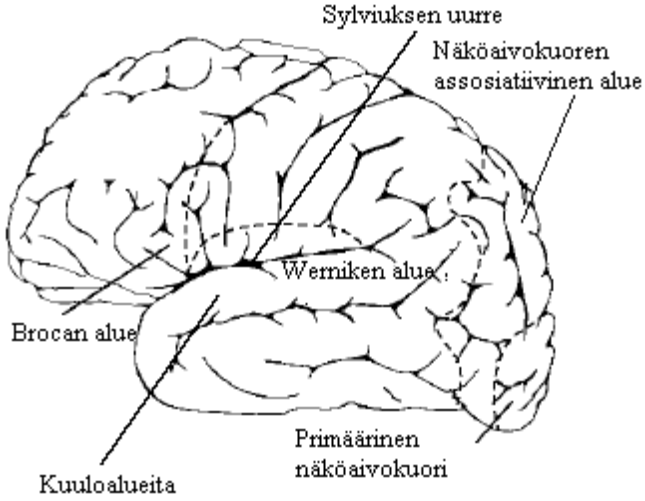
Kuten empiirisiä malleja ja teoriota yleensä, myös puheentuoton kognitiivisia malleja ja teorioita voidaan vain falsifioida. Siksi niiden testaaminen potilasaineiston avulla on keskeinen osa niiden kehitystä. Jos ei voida konstruoida sellaista testiä, jonka avulla tarkasteltava malli tai teoria olisi falsifioitavissa, ei sillä itseasiassa ole lainkaan selitysvoimaa. Tämä johtuu siitä, että falsifioivan testin puuttuessa mallilla tai teorialla voidaan selittää kaikki sen aluetta koskevat väitteet – erityisesti sellaiset väitteet, jotka ovat selvästi toisensa poissulkevia [Popper, 1995].

Uusien mallien ja teorioiden kehityksessä juuri jonkin mallin tai teorian falsifiointi näyttelee keskeistä roolia. Popperin [1995] mukaan teorioiden tarkoituskin on tulla joskus falsifioiduksi. Tällöin teoriassa havaittujen heikkouksien pohjalta on mahdollista kehittää uusi teoria, joka korjaa edellisen teorian heikkoudet. Kehitettyä uutta teoriaa testataan ja se tulee mahdollisesti myös falsifioitua. Jos kahdella teorialla on sama selitysvoima, usein pidetään parempana Occamin partaveitsen mukaisesti yksinkertaisempaa teoriaa.

3. Puheentuoton psykologiaa

Moderni psykolingvistiikka on kielen psykologiaa tutkiva kognitiivisen psykologian kaltainen kokeellinen tiede. Sen tutkimusaluesiiin kuuluvat kielen ymmärtäminen, tuottaminen ja muistaminen. Täten psykolingvististä tutkimusta tehdään tutkimalla ihmisten kuuntelemista, lukemista, puhumista, kirjoittamista sekä muistamista. Tutkimuksen tavoite on pilkkoa ihmisten kielijärjestelmä toiminnallisiin osiinsa ja osoittaa näiden osien väliset suhteet ja tehtävät [Harley, 2001].

Koska kielen ymmärtäminen, tuottaminen ja muistaminen ovat keskushermoston toiminnan kautta syntyviä kognitiivisia toimintoja, liittyy merkittävä osa psykolingvistisesta tutkimuksesta aivotutkimukseen. Aivotutkimus on osoittanut, että kielen kannalta keskeisimmät aivoalueet sijaitsevat useimmilla ihmisillä vasemmassa aivopuoliskossa Sylviuksen uurteen molemmilla puolilla [Laine ja Marttila, 1992] (ks. kuva 3.1). Tärkeimmät näistä alueista ovat Brocan alue ja Wernicken alue. Brocan alueelta puhetuottoon liittyvä informaatio kulkee motoriselle aivokuorelle, josta annetaan käskyt suun, huulien, kielen, kurkunpään ja pallean lihaksille. Wernicken alueella on tärkeä merkitys puheen ymmärtämisessä sekä lukemisessa ja kirjoittamisessa [Ilmoniemi, 2001].



Kuva 3.1. Vasen aivopuolisko [Ilmoniemi, 2001].

Eräs kielen tutkimuksen tärkeä osa on vaurioituneen kielijärjestelmän tutkiminen ja mallintaminen. Vaurioituneen kielijärjestelmän tutkimisella on mahdollista saada sellaista tietoa kielijärjestelmän osista ja niiden suhteista, jota muuten ei olisi saatavissa. Vaurioituneen kielijärjestelmän mallintamisella voidaan puolestaan testata teorioita kielijärjestelmän rakenteesta ja toiminnasta.

Lisäksi hyvien mallien avulla voidaan muodostaa hypoteeseja potilaiden parantumisesta ja kuntoutuksesta.

Tässä luvussa esitellään ensin aivoperäistä kielen häiriötä, afasiaa. Tämän jälkeen tarkastellaan lyhyesti nykyisiä puheen tuoton teorioita. Lopuksi esitellään kaksi eri puheen tuoton teoriaan perustuvaa nimeämistä simuloivaa neuroverkkomallia.

3.1. Afasiasta kärsivät potilaat: afasialuokituksia

Koska ihmisten kielellinen kyky on pitkälti kognitiivista toimintaa, johtuvat kielellisten toimintojen häiriöt usein aivovauriosta. Aivovaurion aiheuttamaa kielellisten toimintojen häiriötä kutsutaan *afasiaksi*. Lähes 90 % afasiaan johtaneista aivovaurioista on aivoinfarkteja tai aivoverenvuotoja. Keskeisiä afasiaoireita ovat erilaisten kielellisten toimintojen, kuten puheen tuoton ja ymmärtämisen, nimeämisen, toistamisen, lukemisen ja kirjoittamisen häiriintyminen. Häiriön laatu ja voimakkuus riippuu aivovaurion sijainnista ja laadusta [Laine ja Marttila, 1992].

Brocan alueen vaurio vaikuttaa haitallisesti puheen tuottamiseen, mutta ei juurikaan vaikuta saapuvan informaation käsittämiseen. Brocan ja lähialueiden vauriosta johtuvaan afasiaa kutsutaan *Brocan afasiaksi*. Wernicken alueen vauriosta johtuvaa afasiaa kutsutaan vastaavasti *Wernicken afasiaksi*. Wernicken afasiasiasta kärsivän potilaan puhe on sujuvaa, mutta sanat ovat usein vääriä. Myös puheen ymmärtäminen kärsii ratkaisevasti [Ilmoniemi, 2001].

Brocan ja Wernicken alueita yhdistävään *fasciculus arcuatus* -hermorataan kohdistunut vaurio aiheuttaa potilaan puheeseen äänteellisiä vääristymiä. Potilaan puhe voi muuten olla sujuvaa, hyvin artikuloitua sekä prosodiikan ja lauserakenteiden osalta normaalia [Laine ja Marttila, 1992]. Puhutun ja kirjoitetun ymmärtäminen on potilailla lähes normaalia, mutta toistaminen on vaivalloista sekä sanojen ja niiden semantiikan yhtenäisyys häiriytyy. Fasciculus arcuatus -hermoradan vaurion seuraksena puhjennutta afasiaa kutsutaan *konduktioafasiaksi* [Ilmoniemi, 2001].

Klassisia afasiaoireyhtymiä Brocan, Wernicken ja konduktioafasian lisäksi ovat *anominen afasia* sekä *transkortikaaliset afasiat*. Anomiselle afaatikoille on tyypillistä huomattava *anomia* (vaikeus nimetä esineitä tai olioita), mutta puheentuotto ja ymmärtäminen on yleensä sujuvaa. Transkortikaalisista afasioista kärsivien potilaiden puheen toistamiskyky on säilynyt hyvänä, mutta puheentuotto tai ymmärtäminen ovat heikentyneet [Laine ja Marttila, 1992].

Vaikka edellä painotettiinkin Brocan ja Wernicken alueen merkitystä ihmisen kielelliseen kyvyn kannalta, ei aivoissa ole selkeitä "puhekeskuksia", joiden tehtäviin mm. kielioppi, kielen ymmärrys tai ääntäminen kuuluisi. Pikem-

minkin nämä kyvyt emergoituvat aivojen hajautetun rakenteen pohjalta [Ilmoniemi, 2001].

Afasiatyypistä riippumatta afaattisiin oireisiin liittyy usein vaikeus löytää tai tuottaa oikeita sanoja. Tämän vuoksi yleinen tapa tutkia afasiaoireita on potilaan nimeämisen (yksittäisten sanojen tuottamisen) tutkiminen. Nimeämistutkimus suoritetaan kuvan nimeämistehtävänä, jossa koehenkilölle näytetään yksittäisiä sanoja esittäviä kuvia, jotka heidän pitää nimetä. Nimeämistesteissä potilaiden tuottamat virheelliset vastaukset luokitellaan virhetyyppien mukaan eri luokkiin. Liitteessä 1 esitellään Laineen [2003] nimeämisvirheiden luokittelumalli sekä esimerkkejä nimeämisvirheistä.

3.2. Kielijärjestelmän modulaarisuudesta

Kuten luvun alussa todettiin, psykolingvistien keskuudessa yleinen oletus on, että kielijärjestelmä koostuu moduleista, joista jokaisella on oma, muista moduleista riippumaton tehtävä. Täten kielen prosessointi voidaan jakaa moduleiden avulla prosessointitasoihin, jotka ovat jonkinlaisessa vuorovaikutuksessa keskenään. Yleensä tutkijat eivät kuitenkaan ole yksimielisiä modulien välisestä vuorovaikutuksen laadusta [Harley, 2001].

Ensimmäinen lähestymistapa vuorovaikutukseen on kysymys moduleissa tapahtuvan prosessoinnin erillisyydestä. *Erillisissä* malleissa (discrete models) modulin prosessointi voi alkaa vasta, kun edellisen modulin prosessointi on päättynyt. *Kaskadimalleissa* (cascade models) moduli voi sen sijaan aloittaa prosessoinnin jo ennen kuin edellinen moduli on päättänyt oman prosessointinsa. Tällöin ylemmiltä prosessointitasoilta "tihkuu" informaatiota alemmille tasoille ennen niiden prosessoinnin päättymistä.

Erillisyyden ja kaskadisuuden lisäksi tutkijat kiistelevät modulien välisen tiedonsiirron laadusta [Harley, 2001]. Jos modulien välillä on vuorovaikutusta, niin onko vuorovaikutus yksi- vai kaksisuuntaista? Yksisuuntaisissa malleissa tiedonsiirto tapahtuu vain ylemmältä tasolta alemmalle, joko erillisesti tai kaskadisti. Kaksisuuntaisissa malleissa tiedonsiirtoa tapahtuu modulien välillä kumpaakin suuntaan, jolloin alemman tason modulit voivat vaikuttaa ylemmän tason prosessointiin ja toisinpäin.

Yleensä oletetaan modulien erillisyys, ellei ole "hyvää syytä" uskoa toisin (vrt. Occamin partaveitsi). Tutkijat eivät yleensä ole yksimielisiä siitä minkälaiset syyt olisivat "hyviä" ja minkälaiset eivät [Harley, 2001].

3.3. Ajatuksista puheeksi: kaksivaiheinen teoria leksikalisaatiosta

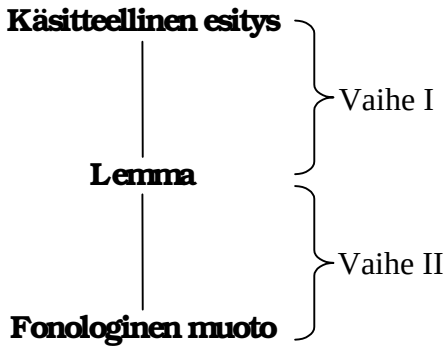
Puhetuotto koostuu kolmesta prosessista [Harley, 2001]: *käsitteellistämisestä* (conceptualization), *muotoilusta* (formulation) ja *artikulaatiosta* (articulation).

Käsitteellistämisen aikana puhuja valitsee muististaan tai ympäristöstä tarvitsemansa tiedot viestin tuottamiseksi. Käsitteellistämisen tuloksena on esikielellinen viesti, jonka puheentuotannon muut vaiheet tuottavat varsinaiseksi puheeksi.

Muotoiluvaiheessa valitaan puheeseen sisältyvät yksittäiset sanat ja järjestetään ne lauseiksi. Lisäksi sen aikana muodostetaan sanojen fonologinen esitys sekä suunnitellaan niiden artikulointi. Sanojen valintaa yhdessä fonologisten esitysten hakemisen kanssa kutsutaan *leksikalisaatioksi* (lexicalization). Sanojen järjestämistä lauseiksi kutsutaan vuorostaan *syntaktiseksi suunnitteluksi* (syntactic planning). Puheentuotannon viimeisen vaiheen (artikulaation) aikana tuotetaan edellisissä vaiheissa valmiiksi muotoiltu viesti äänteiksi.

Jos visuaalinen vaihe sekä puheen artikulaatio suljetaan tarkastelusta pois, yksittäisen kuvan nimeämistä voidaan tarkastella leksikalisaatioprosessina. Yleisesti uskotaan leksikalisaation tapahtuvan kahdessa vaiheessa. Ensimmäisessä vaiheessa muutetaan kohdesanan käsitteellinen esitys sanan abstraktiksi syntaktis-semanttiseksi esitykseksi, *lemmaksi*. Toisessa vaiheessa lemma muutetaan sanan fonologiseksi esitykseksi, *lekseemiksi* (lexeme) [Harley, 2001]. Leksikalisaation ensimmäistä vaihetta kutsutaan *lemman hauksi* (lemma access) ja toista vaihetta *fonologiseksi hauksi* (fonological access) [Dell *et al.*, 1997].

Kaksivaiheisen leksikalisaation malli esitetään yleisellä tasolla kuvassa 3.2. Kuvassa erotetut vaiheet I ja II vastaavat lemmanhakua ja fonologista hakua. Riippuen kaksivaiheista nimeämisteoriasta prosessointia voi kunkin vaiheen aikana tapahtua kaikilla tai vain osalla kuvaan merkityistä tasoista. Vaihe I merkitseekin vain sitä, että sen aikana nimettävän sanan käsitteellinen esitys muutetaan lemmaksi. Vastaavasti vaiheen II aikana lemma muutetaan sanan fonologiseksi esitykseksi. Kuva ei ota kuitenkaan kantaa siihen, *miten* muutos vaiheiden aikana tapahtuu.



Kuva 3.2. Kaksivaiheinen leksikalisaatio.

Vaikka tutkijoiden keskuudessa vallitseekin suurelta osalta yksimielisyys leksikalisaation kaksivaiheisuudesta, ei vaiheiden erillisyydestä ja niiden välisestä vuorovaikutuksesta olla yksimielisiä. Tiukin kaksivaiheisen nimeämisen malleista on leksikalisaation vaiheiden erillisyyttä korostava *DTS-malli* (Discrete Two-Stage model). Sen mukaan leksikalisoinnin vaiheet ovat sekä ajallisesti, että vuorovaikutuksellisesti erillisiä. Tällä tarkoitetaan sitä, että fonologinen haku ei voi alkaa ennen, kuin lemmanhaku on kokonaan päättynyt (ajallinen erillisyys). Vaikka lemmanhakuvaiheessa useita nimettävään käsitteeseen liittyviä lemmoja voi olla aktivoituneena, valitaan aktivoituneista lemmoista vain yksi *kohdelemma*, joka annetaan seuraavalla tasolle (vuorovaikutuksellinen erillisyys). Täten vain yksi lemma (kohdelemma) osallistuu fonologiseen hakuun [Levelt *et al.*, 1991].

Suurimpia DTS-mallien haastajia ovat olleet *vuorovaikutteiseen aktivaatioon* (Interactive Activation, IA) perustuvat mallit [Levelt *et al.*, 1991]. IA-mallit koostuvat yleensä kolmesta kerroksesta: (1) semanttisesta (tai käsite) kerroksesta, (2) lemmakerroksesta ja (3) fonologisesta kerroksesta. Kerrokset vastaavat siis kuvan 3.2 tasoja. Useimmissa IA -malleissa vierekkäiset kerrokset on liitetty toisiinsa kaksisuuntaisilla yhteyksillä, jotka mahdollistavat aktivaation leviämisen kerrosten välillä kumpaankin suuntaan [Levelt *et al.*, 1991]. Täten lemmanhaku ja fonologinen haku vaikuttavat toisiinsa ja tapahtuvat osittain päällekkäin.

IA-malleissa lemmanhakuvaiheessa valitaan lemmakerroksesta kohdelemma. Tämä tehdään levittämällä semanttisesta kerroksesta aktivaatiota lemmakerrokseen. Kohdelemman hakuun osallistuvat kuitenkin kaikki kerrokset: semanttisen kerroksen aktivaatio leviää ensin lemmakerrokselle, ja lemmakerrokselta edelleen fonologiselle kerrokselle. Koska yhteydet kerroksien välillä ovat kaksisuuntaisia, alkaa aktivaatio levitä takaisin alemmilta kerroksilta ylemmäksi, jolloin myös niiden aktivaatio vaikuttaa osaltaan lemmanhakuun.

Kohdelemman valinnan jälkeen fonologinen haku suoritetaan vastaavalla tavalla. Valittu kohdelemma aktivoidaan, ja aktivaatio leviää sekä semanttiselle, että fonologiselle kerrokselle kaksisuuntaisten yhteyksien ansiosta. Täten myös fonologisen muodon hakuun osallistuvat kaikki kerrokset. Joissakin IA-mallien toteutuksissa lemman haun jälkeen useita lemmoja voi olla aktivoituneina. Tällöin kaikkien aktivoituneiden lemmojen fonologista muotoa haetaan fonologisen haun yhteydessä.

Vaikka tutkijat ovatkin olleet melko yksimielisiä leksikalisoinnin kaksivaiheisuudesta, on sitä vastaan esitetty myös kritiikkiä. Kritiikin kärkenä on esitetty potilasaineistoon perustuvia todisteita, että lemmatasoa ei välttämättä tarvita ollenkaan leksikalisaatiossa [Caramazza, 1997]. Ilman lemmatasoa kak-

sivaiheinen leksikalisaatio muuttuu yksitasoiseksi. Kuitenkin ainakin toistaiseksi suurin osa tutkijoista kannattaa nimeämisen kaksivaiheista teoriaa [Harley, 2001].

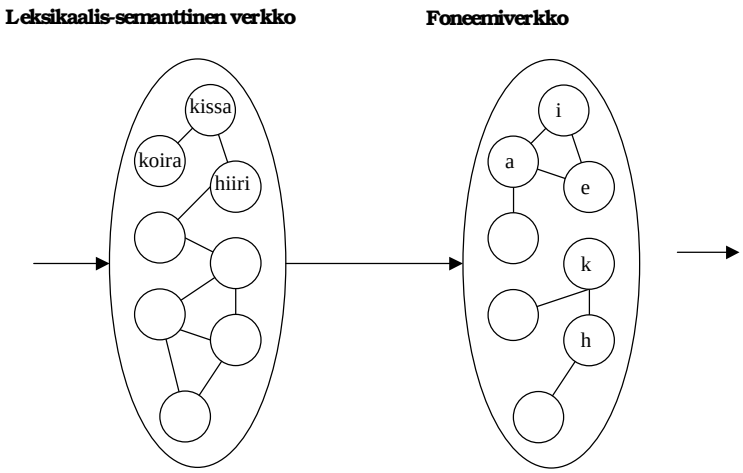
3.4. Kaksi nimeämishäiriöitä simuloivaa neuroverkkomallia

Tässä luvussa esitellään kaksi kaksivaiheisen nimeämisteorian pohjalle rakennettua nimeämistä simuloivaa neuroverkkomallia. Mallit olettavat, että visuaalinen vaihe ja puheen artikulointi eivät ole häiriintyneitä. Täten ne mallintavat pelkkää leksikalisaatioprosessia, eli lemmanhakua sekä fonologista hakua. Ensimmäinen malli on suomalaisten afasiapotilaiden simuloimista varten kehitetty DTS-malli. Toinen esittävistä malleista on englanninkielisten afasiapotilaiden mallintamista varten kehitetty IA-malli. Kumpikaan malli ei ole oppiva, vaan niiden painoarvot ovat ennalta asetettuja.

3.4.1. Suomenkielinen SLIPNET-malli

DTS-mallin oletusten mukaan nimeämisprosessin kaksi vaihetta ovat täysin erillisiä. Sanan lemmat aktivoidaan ilman fonologisen tason vaikutusta ja aktivoituneista lemmoista valitaan vain suurimman aktivaation omaava lemma. Valitun lemman fonologinen esitys haetaan fonologisella tasolla täysin erillään lemmatasosta. DTS-mallissa tasojen väliset yhteydet ovat täten vain eteenpäin syöttäviä. Sen sijaan tasojen sisäisillä yhteyksillä on DTS-mallissa keskeinen rooli, sillä aktivaation levitys tapahtuu DTS-malleissa niitä pitkin. Tässä tarkasteltava DTS-malli SLIPNET on toteutettu suomenkielisten afasiapotilaiden nimeämishäiriöiden mallintamista varten. Siitä on esitetty useita eri versiota (varhainen versio SLIPNETistä esitellään esim. [Juhola and Laine, 1992]). Eniten mallia on kehittänyt ja analysoinut Tikkala [1997] väitöskirjatyössään. Tässä luvussa tarkastellaan kuitenkin vain Laineen *et al.* [1998] esittelemää SLIPNETin viimeisintä versiota.

SLIPNET koostuu kahdesta aliverkosta: leksikaalis-semanttisesta aliverkosta ja foneemialiverkosta. Kuvassa 3.3 on esitetty SLIPNETin rakennetta. Leksikaalis-semanttinen aliverkko simuloi nimettävän sanan lemmanhakua. Verkkoon on koodattu 27 lemmaa, jotka on yhdistetty toisiinsa semanttisin perustein. Jos lemmat liittyvät semanttisesti toisiinsa, on niiden välillä voimakas yhteys. Jos lemmat ovat semanttisesti kaukaisia, ei niiden välillä ole yhteyttä ollenkaan.



Kuva 3.3. SLIPNET-neuroverkon rakennetta [Laine *et al.*, 1998].

Foneemialiverkko simuloi yksittäisten foneemien tuottamista. Se jakautuu kahteen aliverkkoon, vokaaliverkkoon sekä konsonanttiverkkoon. Jako on tehty, koska kokeellisesti on huomattu, että konsonantin muuttuminen vokaaliksi (tai toisinpäin) on erittäin harvinaista [Tikkala and Juhola, 1995]. Foneemiverkkoon on koodattu kaikki leksikaalis-semanttisen aliverkon sanoissa esiintyvät foneemit.

SLIPNETin neuronien välisten yhteyksien painot on rajoitettu viiteen eri tyyppiin: vahvoihin (paino w=0.40), kohtuullisiin (w=0.20) ja heikkoihin (w=0.15) yhteyksiin. Neljännellä painotyypillä merkitään puuttuvia yhteyksiä (w=0) ja viidennellä (w=1) merkitään jokaisen neuronin itseissilmukoita [Juhola and Laine, 1992]. Painoarvot määrättiin leksikaalis-semattisessa aliverkossa terveiden koehenkilöiden arvioiden perusteella. Foneemialiverkon painoarvot määrättiin foneemien lingvististen ominaisuuksien perusteella [Tikkala and Juhola, 1995].

Koska SLIPNETin kerrosten sisäiset yhteydet ovat kaksisuuntaisia on malli rekursiivinen. Siksi nimeämistä simuloidessa täytyy ottaa käyttöön diskreetti aika, jonka suhteen neuronien aktivaatiota tarkastellaan. Neuronin tulos kierroksella (ajanhetkellä) t lasketaan lisäämällä neuronin edellisen kierroksen tulokseen sen saamien painoarvollaan kerrottujen syötteiden summa. Täten siis neuronin i saama kokonaissyöte kierroksella t voidaan laskea kaavalla

$$v_i(t) = \sum_{j=1}^{n} w_{ij} x_j(t-1), \tag{3.1}$$

missä $x_j(t-1)$ on neuronin j aktivaatio ajanhetkellä $t-1$ ja w_{ij} on neuroneiden i ja j välisen yhteyden paino. Aktivaatiofunktiona käytetään lineaarista aktivaatiofunktiota

$$\varphi(v) = (1-q)v, \tag{3.2}$$

missä q on neuronin aktivoitumista säätelevä *vaimenemiskeroin* (decay rate). Neuronin tulos $y_i(t)$ kierroksella t saadaan siis laskemalla

$$y_i(t) = \varphi(v_i(t)). \tag{3.3}$$

Vaurioitunutta nimeämistä simuloidaan lisäämällä neuronien tuloksiin normaalijakautunutta satunnaiskohinaa. Vaurioituneen neuronin tulos $y_i'(t)$ kierroksella t lasketaan kaavalla

$$y_i'(t) = (1 + \alpha e_i) y_i(t), \tag{3.4}$$

missä αe_i on neuroniin i liittyvä satunnaiskohinaparametri kierroksella t. Satunnaiskohinaparametrissa e_i on saatu nollakeskiarvoisesta normaalijakaumasta, jonka keskihajontaa parametrisoi α. Tästä eteenpäin keskihajontaa α kutsutaan satunnaiskohinaksi tai kohinaksi.

Simulaation alussa aktivoidaan leksikaalis-semanttisen aliverkon neuroni, joka vastaa nimettävänä olevan sanan lemmaa. Tämän jälkeen aktivaatiota levitetään leksikaalis-semanttisessa aliverkossa m kierrosta kaavoilla (3.1) - (3.4). Sitten leksikaalis-semanttisesta aliverkosta valitaan se neuroni i, jonka tulos $y_i'(m)$ on suurin. Olkoon tällainen tulos y_{max}. Saatu arvo kynnystetään kaavalla

$$y = \begin{cases} y_{max}, & y_{max} \geq \tau \geq 0 \\ 0, & \text{muutoin} \end{cases}. \tag{3.5}$$

Siis jos valitun neuronin aktivaatio y_{max} ylittää kynnysarvon τ, annetaan leksikaalis-semanttisen aliverkon tulokseksi y_{max}, muutoin 0.

Foneemialiverkosta aktivoidaan kohdelemmaa vastaavat foneemit yksitellen. Aktivaation suuruuden määrää leksikaalis-semanttisen aliverkon kynnystetty maksimiarvo. Foneemialiverkko simuloi foneemin hakua kaavoilla (3.1) - (3.4). Jos syötteen arvo on 0, verkko ei tuota mitään vastausta. Muussa tapauksessa foneemialiverkon tuottamat foneemit konkatenoidaan, jolloin tuloksena on joko suomenkielinen sana tai suomen kieleen kuulumaton foneemijono. Foneemialiverkon tuottamia tuloksia ei kynnystetä.

Nimeämisvirheitä SLIPNETillä tuotettiin lisäämällä satunnaiskohinaa leksikaalis-semanttiseen ja fonologiseen aliverkkoon sekä muuntelemalla kerrosten välistä kynnysarvoa. Satunnaiskohinan lisääminen leksikaalis-semanttiseen aliverkkoon lisää todennäköisyyttä, että väärän sanan lemma tulee valituksi aiheuttaen semanttisen virheen (esim. koira → kissa). Vastaavasti satunnaiskohinan lisääminen foneemialiverkkoon aiheuttaa joko formaaleja parafasioita (kirahvi → karahvi) tai neologismeja (esim. helikopteri → rapuli). Fonologiset parafasiat (esim. lusikka → tusikka) luokiteltiin myös neologismeiksi. Kynnysarvoon liittyvä virhetyyppi on omissio (yleisesti virhetyyppejä on esitelty

liitteessä 1). Omissioita sattuu, kun leksikaalis-semanttisen aliverkon tulos ei ylitä kynnysarvoa. Mainittujen virheiden lisäksi SLIPNET tuottaa muita virheitä, jotka eivät ole luokiteltavissa oikeiksi vastauksiksi tai joksikin edellä mainituista virhetyypeistä [Laine *et al.*, 1998].

Laine *et al.* [1998] testasivat SLIPNETin kykyä simuloida potilaiden nimeämisvirheitä vertaamalla sen suoritusta kymmeneen suomalaiseen afasia-potilaaseen. Potilaat kärsivät neljästä erityyppisestä afasiaoireyhtymästä: Brocan afasiasta (potilaat B1 ja B2), Wernicken afasiasta (potilaat W1, W2 ja W3), konduktioafasiasta (potilaat C1 ja C2) sekä anomisesta afasiasta (potilaat A1, A2 ja A3). Potilaiden nimeämistestien tulokset vaihtelivat. Joillakin potilailla neologistiset virheet muodostivat suurimman osan kaikista virheistä, kun taas toisilla suurin osa virheistä muodostui omissioista. Potilaiden nimeämistestien tulokset oli luokiteltu edellä kuvattujen SLIPNETin virheluokituksen mukaisesti oikeiksi vastauksiksi, semanttisiksi virheiksi, neologismeiksi, omissioiksi sekä muiksi virheiksi.

Laine *et al.* [1998] yrittivät parametrisoida SLIPNETin simuloimaan jokaista potilasta mahdollisimman tarkasti vaihtelemalla satunnaiskohinan määrää kummassakin aliverkossa, sekä kynnysarvoa τ aliverkkojen välillä. Potilaiden ja heidän nimeämisjakaumiinsa sovitetun verkon jakaumat on esitetty taulukoissa 3.1A ja 3.1B.

Potilas	Oikein %	Semanttinen %	Neolog. %	Omissio %	Muu %
B1	77.1	3.0	0.0	19.3	0.6
B2	87.3	3.0	0.0	8.4	1.2
A1	51.7	4.2	0.0	44.1	0.0
A2	77.1	8.4	0.0	13.9	0.6
A3	39.2	0.6	0.0	59.6	0.6
C1	65.7	3.6	22.3	7.2	1.2
C2	84.3	0.6	9.6	4.8	0.6
W1	54.8	4.8	15.1	20.5	4.8
W2	43.8	13.9	10.8	21.1	11.4
W3	36.7	0.6	19.3	37.3	6.0

Taulukko 3.1A. SLIPNETin evaluointiin käytettyjen potilaiden vastausjakaumat nimeämistesteissä.

Simul.	Oikein %	Semanttinen %	Neolog. %	Omissio %	Muu %
B1	78.3	2.8	0.0	18.5	0.0
B2	87.2	4.1	0.0	8.1	0.6
A1	51.1	4.3	0.4	44.1	0.2
A2	76.9	9.4	0.7	13.0	0.0
A3	38.7	0.7	0.4	60.2	0.0
C1	65.4	3.9	18.5	7.8	4.4
C2	84.1	1.1	9.3	4.6	0.9
W1	54.6	6.3	14.6	20.7	3.7
W2	43.1	13.9	16.7	20.4	5.9
W3	36.7	1.1	22.6	35.9	3.7

Taulukko 3.1B. SLIPNETin sovitus potilaiden vastausjakaumiin nimeämistesteissä.

Kuten taulukoista 3.1A ja 3.1B ilmenee, SLIPNETin parametrisointi eri potilaiden suoritustusta vastaavaksi onnistui hyvin. Erot potilaan ja verkon tuottamien nimeämistulosten välillä eivät olleet tilastollisesti merkitseviä (Laine *et al.* arvioivat potilaan ja verkon jakaumien välisiä eroja χ^2-testeillä). Potilasovitukset esitetään taulukkoina, jotta vertailu luvuissa neljä ja viisi käsiteltävään Learning SLIPNETiin olisi helpompaa.

3.4.2. Interaktiiviseen aktivaatioon perustuva neuroverkko

Interaktiivisen aktivaation teorian mukaan nimeämisen kumpaankin vaiheeseen osallistuu koko puheentuottojärjestelmä. Haettaessa lemmaa siihen liittyvät foneemit alkavat myös aktivoitua ja vaikuttavat lemman valintaan takaisinsyöttävien yhteyksien avulla. Fonologisen haun aikana taas foneemit lähettävät aktivaatiota lemmalle, josta on edelleen yhteyksiä takaisin foneemeihin. Täten lemmanhakuvaiheessa haettavat lemmat saavat lisäaktivaatiota niihin liittyviltä foneemeilta, ja fonologisen haun yhteydessä foneemit saavat lisäaktivaatiota lemmoilta. Esiteltävä malli pohjautuu Dellin [1986] alunperin esittelemään puheentuoton malliin. Nimeämishäiriöiden mallintamiseen sitä ovat soveltaneet ainakin Dell *et al.* [1996, 1997] sekä Foygel ja Dell [2000].

Dellin *et al.* [1996, 1997] IA-malli koostuu kolmesta kerroksesta: semanttisesta kerroksesta, lemmakerroksesta sekä foneemikerroksesta. Vierekkäisten kerrosten välillä on kaksisuuntaisia yhteyksiä. Kerrosten sisäisiä yhteyksiä ei sen sijaan ole. Mallin syötekerroksena toimii semanttinen kerros, joka koostuu nimettävän kohteen semanttisista piirteistä, semeemeistä. Semeemejä liittyy 10 kpl jokaiseen verkkoon tallenttuun lemmaan. Semanttisesta kerroksesta on kaksisuuntaisia yhteyksiä lemmakerrokseen. Tuloskerroksena toimii foneemikerros. Myös foneemi- ja lemmakerroksen välillä on kaksisuuntaisia yhteyksiä. Foneemikerros on jaettu kolmeen osaan: alkukonsonantiin, vokaaliin ja loppukonsonanttiin. Täten tarkasteltavaan IA-malliin voidaan koodata vain yhden tavun mittaisia sanoja. IA-mallin rakennetta on esitetty kuvassa 3.4.

Koska IA-mallin kerrosten väliset yhteydet ovat kaksisuuntaisia on malli SLIPNETin tapaan rekursiivinen. Siksi nimeämistä simuloidessa täytyy ottaa käyttöön diskreetti aika, jonka suhteen neuronien aktivaatiota tarkastellaan. Neuroniin i saapuva kokonaissyöte $v_i(t)$ kierroksella (ajanhetkellä) t lasketaan kaavalla (3.1). Neuronien i ja j väliset painot ovat $w_{ij}=c$ (c on jokin vakio), kun $i \neq j$. Paino on sen sijaan $w_{ii}=1-q$, missä q on vaimenemiskerroin. Vaimenemiskertoimella siis säädellään kuinka voimakkaana neuroni saa edellisen kierroksen aktivaationsa käyttöönsä. Neuronien aktivaatiofunktiona toimii lineaarinen funktio

$$\varphi(v) = v, \tag{3.6}$$

jolloin neuronin i tulos $y_i(t)$ kierroksella t saadaan kaavojen (3.1) ja (3.6) perusteella laskemalla

$$y_i(t) = \varphi\big(v_i(t)\big). \tag{3.7}$$

Nimeämisvirheiden tuottamiseksi verkkoon lisättiin kahta erillistä satunnaiskohinaa. Ensimmäinen parametrisoi verkossa esiintyvää yleistä kohinaa $noise_1$. Se lasketaan kaavalla

$$noise_1 = \alpha_I e, \tag{3.8}$$

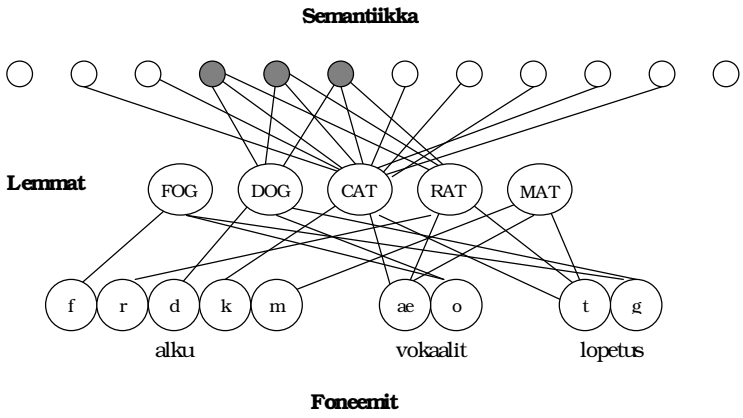
missä e on saatu nollakeskiarvoisesta normaalijakaumasta, jonka keskihajonta on α_I. Toinen satunnaiskohina $noise_2$ parametrisoi neuronin i aktivaatioon $y_i(t)$ liittyvää satunnaiskohinaa. Se lasketaan kaavalla

$$noise_2 = \alpha_A e_i y_i(t), \tag{3.9}$$

missä e_i on saatu nollakeskiarvoisesta normaalijakaumasta, jonka keskihajonta on α_A. Täten kohinaisen verkon neuronin i tulos $y_i'(t)$ kierroksella t saadaan kaavalla

$$y_i'(t) = y_i(t) + noise_1 + noise_2. \tag{3.10}$$

Dell *et al.* [1997] toteuttivat IA-mallin kahtena kuuden sanan naapurustona. Kummassakin naapurustossa on sama kohdesana "cat". Ensimmäisessä naapurustossa on kohdesanan lisäksi siihen semanttisesti liittyvä sana, kaksi fonologisesti liittyvää sanaa, sekä kaksi kohdesanaan liittymätöntä sanaa. Toisessa naapurustossa on kohdesanan lisäksi siihen semanttisesti liittyvä sana, yksi fonologisesti liittyvä sana, yksi sekä semanttisesti että fonologisesti liittyvä sana, sekä kaksi kohdesanaan liittymätöntä sanaa. Sanojen pituus oli rajoitettu yhteen tavuun, jotta ne voitaisiin koodata kolmella foneemilla (alkukonsonantti, vokaali, loppukonsonantti).



Kuva 3.4. IA-neuroverkon rakennetta [Dell *et al.*, 1997].

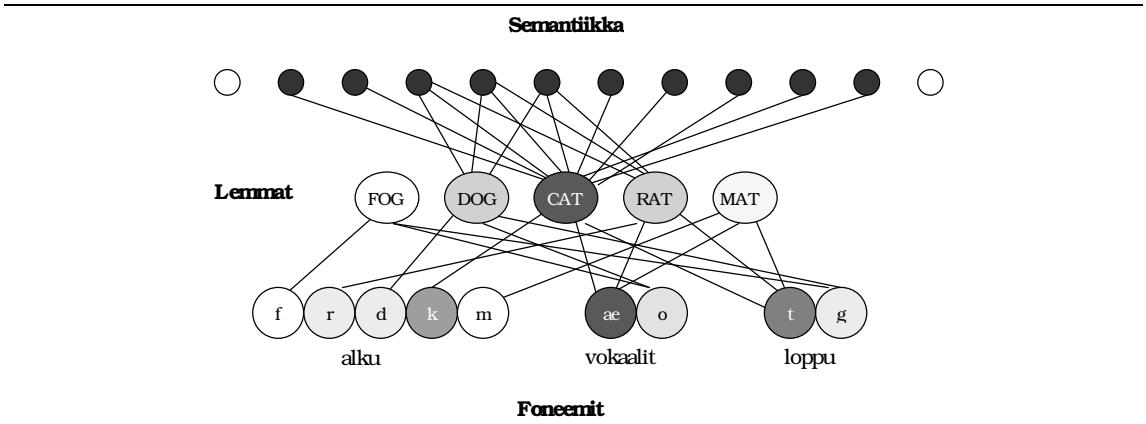
Ensimmäisessä naapurustossa kohdesanalla ja tähän semanttisesti liittyvällä sanalla on kolme yhteistä semanttista piirrettä. Toisessa naapurustossa kohdesanalla on edellisen lisäksi kolme yhteistä semanttista piirrettä siihen semanttisesti ja fonologisesti liittyvän sanan kanssa. Kuvassa 3.4 on esitetty sanan kohdesanan "cat" semanttiset piirteet. Harmaalla merkityt semanttiset piirteet kuuluvat kohdesanan lisäksi sanalle "dog", joka liittyy sanaan kohdesanaan semanttisesti, ja sanalle "rat", joka liittyy kohdesanaan semanttisesti ja fonologisesti naapuri. Muut kohdesanaan liittyvät semanttiset piirteet taas erottavat sen sanoista "dog" ja "rat".

Foneemikerroksen naapurustoihin on sisällytetty kaikki mahdolliset foneemit, joita verkkoon tallennetuilla sanoilla voi olla. Jokaisesta lemmakerroksen neuronista on yhteys sen alkukonsonanttiin, vokaaliin sekä loppukonsonanttiin. Täten esimerkiksi sanalla "cat" on yhteys foneemeihin /*k*/ (alkukonsonantti), /*ae*/ (vokaali) sekä /*t*/ (loppukonsonantti). Kohdesanalla ja siihen fonologisesti liittyvällä sanalla on yhteisiä foneemeja. Esimerkiksi sanat "mat" ja "cat" eroavat vain yhdellä foneemilla (alkukonsonantilla /*m*/ - /*k*/). Niiden semantiikalla ei kuitenkaan ole mitään tekemistä keskenään. Tämä mahdollistaa formaaleiden parafasioiden syntymisen vaurioitetussa verkossa [Dell *et al.*, 1996].

Simulaatio käynnistyy, kun nimettävän sanan (kohdesanan) semanttiset piirteet aktivoidaan semanttisesta kerroksesta. Aktivaatiota levitetään kaavojen (3.1) ja (3.6) - (3.10) mukaisesti n kierrosta, jonka jälkeen valitaan lemmakerroksesta suurimman aktivaation saavuttanut neuroni. Valittu neuroni ei välttämättä ole neuroni, jonka semanttiset piirteet on aktivoitu simulaation alussa, sillä semanttisen kerroksen neuronit ovat yhteydessä myös muihin kuin kohdesanaan liittyvään lemmatason neuroniin. Tällöin myös muut lemmat aktivoituvat lemmakerroksessa. Lisäksi kaikki aktivoituneet lemmakerroksen neuronit levittävät aktivaatiota takaisin niitä vastaaviin semanttisen kerroksen neu-

roneille sekä eteenpäin fonologisen kerroksen neuroneihin. Aktivoituneet fonologisen kerroksen neuronit vahvistavat vuorostaan niihin liittyviä lemmakerroksen neuroneja. Kun lemmakerroksen voimakkaimmin aktivoitunut neuroni on valittu, nollataan kaikkien neuronien aktivaatiot. Tämän jälkeen valittu neuroni aktivoidaan uudelleen, ja simulaatio jatkuu vastaavalla tavalla aktivaatiota levittäen *m* kierrosta. Lopuksi valitaan foneemikerroksen jokaisesta foneemipaikasta suurimman aktivaation saavuttanut neuroni (tavun alku- ja loppukonsonantti sekä vokaali). Näin saatu foneemijono tulkitaan verkon tulokseksi.

Hypoteettista aktivaation leviämistä on havainnollistettu kuvassa 3.5. Alussa on aktivoitu kohdesanan "cat" semanttiset piirteet. Mitä tummempi neuronin väri on, sitä suurempi on sen aktivaatiotaso. Kuvassa ei esitetä semanttisia piirteitä, jotka lemmojen "dog" ja "rat" aktivoituminen on aiheuttanut, eikä näiden ei-kuvattujen semanttisten piirteiden edelleen aiheuttamia aktivoitumisia.



Kuva 3.5. Aktivaation leviäminen.

Kun aktivaatiota on levitetty muutamia kierroksia, on lemmakerroksen neuroneista lemman "cat" lisäksi aktivoituneet lemmat "dog" ja "rat". Tämä johtuu siitä, että niillä on yhteisiä semanttisia piirteitä sanan "cat" kanssa. Niiden aktivaatiotaso ei ole kuitenkaan yhtä suuri kuin sanan "cat", sillä tämä saa lisäaktivaatiota myös seitsemältä muulta siihen liittyvältä semanttiselta piirteeltä. Aktivoituneet lemmat ovat vuorostaan alkaneet levittää aktivaatiota niitä vastaaviin foneemikerroksen neuroneihin. Tavun alkuosaa vastaavista neuroneista foneemia */k/* vastaava neuroni on aktivoitunut voimakkaimmin, koska se saa aktivaatiota lemmakerroksen voimakkaimmin aktivoituneelta lemmalta "cat". Foneemeja */r/* ja */d/* vastaavat neuronit ovat alkaneet myös aktivoitua niiden saadessa aktivaatiota vähemmän aktivoituneilta lemmakerroksen neuroneilta "dog" ja "rat". Samalla tavalla voidaan tulkita myös tavun vokaali- ja lopetusosaa vastaavien neuronien aktivaatiota. Vokaali */ae/* on

kuitenkin aktivoitunut muita voimakkaammin, koska se saa syötteitä sekä lemmalta "cat" että "rat". Mielenkiintoista on kuitenkin huomata, että alkuperäiseen syötteeseen semanttisesti täysin liittymätön lemma "mat" on alkanut aktivoitua. Tämä johtuu siitä, että osa sitä vastaavista foneemikerroksen neuroneista on aktivoitunut.

Esimerkistä ilmenee, kuinka IA-malli tuottaa nimeämisvirheitä. Jos lemman valinnan yhteydessä jokin muu kuin kohdesanan lemma tulee valituksi, aiheutuu joko semanttinen virhe, formaali parafasia, sekoittunut virhe (semanttinen ja fonologinen) tai kohdesanaan liittymätön virhe. Virheen tyyppi riippuu siitä, mikä on valitun lemman suhde kohteeseen. Fonologisen haun aikana voi syntyä neologismeja sekä formaaleja että fonologisia parafasioita samalla periaatteella kuin lemman haun yhteydessä. Esimerkkejä näistä virheistä on annettu liittessä 1. Kun verkko on parametrisoitu terveiden koehenkilöiden nimeämisen mukaan, virheiden todennäköisyys on tietenkin pieni. Mallin parametreja (painot, vaimenemiskerroin, kohinat α_I ja α_A) muuttamalla virheiden todennäköisyys kasvaa ja verkko alkaa tuottaa oikeiden vastausten ohella myös virheitä.

Mallin evaluointia varten Dell *et al.* [1997] parametrisoivat mallin ensin mahdollisimman lähelle terveiden englanninkielisten koehenkilöiden nimeämistä. Tällöin mallin parametrit olivat w_{ij}=0.1, kun $i \neq j$, q=0.5, α_I=0.01 ja α_A=0.16. Leksikalisaation kumpaankin vaiheeseen käytettiin kahdeksan kierrosta. Tämän jälkeen Dell *et al.* parametrisoivat mallin mahdollisimman lähelle 21 afasiapotilaan nimeämistä. Parametrisointi suoritettiin manuaalisesti (ilman formaalia sovitusprosessia) vaihtelemalla vaimenemiskerrointa q ja neuronien välisten yhteyksien painoarvoja w_{ij}. Jotta simulaatiosta ei olisi tullut liian monimutkaista, oletettiin afasiapotilaiden aivovaurioiden olevan kokonaisvaltaisia, jolloin vaimenemiskertoimen ja painoarvojen vaihtelut kohdistettiin koko verkkoon samalla tavalla.

Dellin *et al.* mukaan mallin sovitus potilasaineistoon onnistui hyvin. Lisäksi potilaat oli mahdollista luokitella mallin painoarvojen perusteella suuri- ja pienipainoisiksi. Suuripainokertoimisten potilaiden ennustettiin tekevän enemmän sekoittuneita virheitä, koska aktivaation leviäminen on näillä potilailla voimakkaampaa. Tämä ennustus vahvistettiin kokeellisesti.

3.4.3. Tuloksista

Vaikka edellä kuvatut nimeämistä simuloivat neuroverkkomallit perustuivatkin eri teorioihin kaksivaiheisesta nimeämisestä, kummankin verkkoarkkitehtuurin avulla oli mahdollista simuloida erityyppisten afasiapotilaiden nimeämisvirheitä. DTS-malli SLIPNETiä vaurioitettiin lisäämällä yhteyksiin normaalijakautunutta satunnaiskohinaa, kun taas IA-malliin vauriot aiheutet-

tiin muuntelemalla vaimenemiskerrointa q ja yhteyksien välisiä painoarvoja w_{ij} kohinaparametrien pysyessä vakiona.

SLIPNETin rakenteesta johtuen sillä ei voitu tuottaa sekoittuneita virheitä, joita terveillä koehenkilöillä on havaittu. Kyvyttömyys sekoittuneiden virheiden tuottamiseen SLIPNET-verkolla johtuu siitä, että niiden tuottaminen vaatisi fonologisen aliverkon vuorovaikutusta leksikaalis-semanttisen aliverkon kanssa foneemien haun aikana. Koska DTS-malleissa vuorovaikutus ei ole sallittu, ei SLIPNETissä ole minkäänlaista keinoa sekoittuneiden virheiden tuottamiseksi. IA-malli voi tuottaa tällaisia virheitä, sillä vuorovaikutus kerrosten välillä mahdollistaa semanttisten virheiden synnyn vielä fonologisen haun aikana. Sekoittuneiden virheiden puuttuminen SLIPNETin virhejakaumasta heikentää DTS-mallien uskottavuutta suhteessa IA-malleihin [Laine *et al.*, 1998].

IA-mallilla ei vuorostaan voitu tuottaa omissiota, koska mallissa ei ollut minkäänlaista kynnysarvoa tai muuta menetelmää omissioiden tuottamiseksi. Dell *et al.* [1996, 1997] eivät kuitenkaan yrittäneet mallintaa omissoita tuottavia potilaita. Tällaisia potilaita olisi todennäköisesti mahdollista mallintaa käyttämällä vastaavan kaltaista kynnysarvoa kuin SLIPNETissä käytettiin.

Kumpikin esitellyistä malleista kärsii simulaatiossa mukana olevien sanojen vähyydestä. On kyseenalaista, voidaanko muutamilla sanoilla suoritetuista onnistuneista simulaatiosta tehdä yleisiä päätelmiä nimeämisprosessin kulusta. IA-mallin heikkoutena voidaan myös pitää sitä, että simulaatiossa mukana olevat sanat olivat yksitavuisia. Tämä yksipuolistaa entisestään simulaatiossa käytettävää sanajoukkoa. SLIPNETissä sanojen pituutta ei tällä tavalla rajattu ja simulaatiossa esiintyi myös pitkiä yhdyssanoja, kuten "pyykkipoika".

Caramazza ja Miozzo [1997] sekä Ruml ja Caramazza [2000] ovat voimakkaasti kritisoineet IA-mallia. Miozzon ja Caramazzan kritiikki perustuu yleisemmin kaksivaiheiseen nimeämisen teoriaan, joten heidän esittämä kritiikki soveltuu myös SLIPNETiin. Tutkiessaan italiankielisiä koehenkilöitä he huomasivat, että koehenkilöt pystyivät tuottamaan osan nimettävän sanan syntaktisesta informaatiosta (esim. substantiivin suvun), vaikka he eivät kyenneet nimeämään itse sanaa. Tämä on ristiriidassa kaksivaiheisen nimeämisteorian kanssa, koska siinä oletetaan, että sanan lemmaan on tallennettu sen syntaktiset ja semanttiset tiedot. Toisin sanoen, jos puhuja pystyy hakemaan sanan lemman, tulisi hänen kyetä tuottamaan tällöin koko sana. Jos taas puhuja ei kykene hakemaan sanan lemmaa, ei hänellä pitäisi olla minkäänlaista tietoa myöskään sanan syntaktisista ominaisuuksista. Tämän havainnon perusteella Caramazza ja Miozzo halusivat hylätä kaksivaiheisen teorian nimeämisestä ja ehdottavat uutta nimeämisteoriaa, jossa lemmoilla ei ole minkäänlaista roolia.

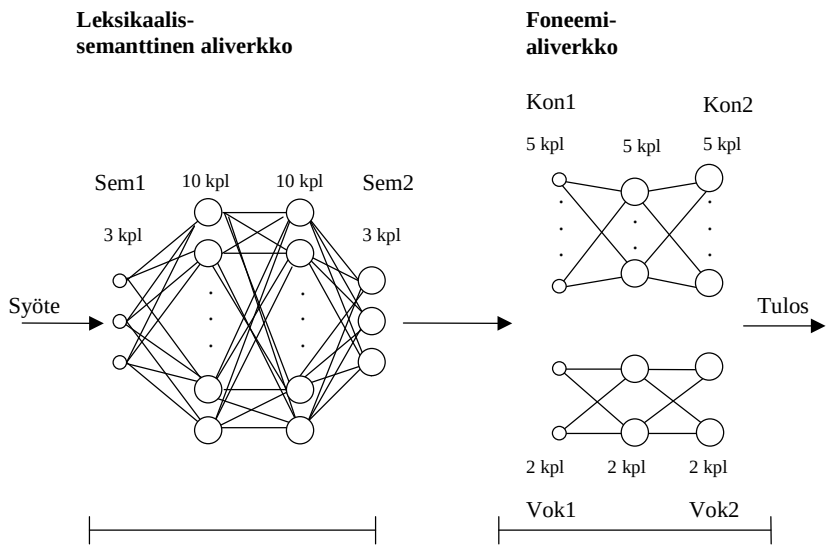
Ruml ja Caramazza vuorostaan kritisoivat Dellin *et al.* [1997] suorittamaa IA-mallin analyysia. Suoritettuaan tarkemman analyysin he väittivät, että mallin sovitus potilaisiin oli kaikkea muuta kuin hyvä ja että sen epäonnistuminen potilassovituksissa johtuu IA -mallin rakenteesta.

4. Uuden simulaatiomallin toteutus: Learning SLIPNET

Tässä luvussa esitellään SLIPNET-neuroverkon oppiva versio Learning SLIPNET. Kyseessä on alkuperäisen SLIPNETin uudelleentoteutus MLP-neuroverkoilla. Tavoitteena oli luoda verkkoarkkitehtuuri, jonka avulla voitaisiin simuloida aivovaurion jälkeistä toipumista ja erityisesti kuntoutuksen vaikutusta afaatikkojen nimeämiseen. Ensimmäinen tehtävä olisi ollut mahdollinen myös alkuperäisellä SLIPNETillä, sillä toipumista voi simuloida vähentämällä verkkoon lisättävän satunnaiskohinan määrää ja madaltamalla kynnysarvoa. Kuntoutuksen vaikutuksen simulointi sen sijaan ei onnistu SLIPNETillä, sillä kuntouttamisen ja kuntouttamisen vaikutuksien simulointi vaatii oppivaa neuroverkkoa. Tämän vuoksi toteutettavan neuroverkon tulee olla oppiva.

Toisaalta verkkoarkkitehtuuri haluttiin pitää mahdollisimman yksinkertaisena. Koska alkuperäisen SLIPNETin aliverkkojako sekä arkkitehtuuri on yksinkertainen ja selkeä, päädyttiin SLIPNETin uudelleentoteuttamiseen yleisesti käytetyillä MLP-neuroverkoilla. Tässä luvussa käsitellään tarkemmin Learning SLIPNETin toteutusta. Esiinnousevista ongelmista keskeisin on syötteiden ja tulosten koodaus leksikaalis-semanttiselle ja fonologiselle aliverkolle. Seuraavassa luvussa (luku 5) tutkitaan kehitetyn verkon yleisiä ominaisuuksia ja sovitetaan se potilasaineistoon. Tavoitteena on osoittaa, että Learning SLIPNETillä voidaan määrällisesti simuloida afaattisten potilaiden nimeämistä. Varsinainen kuntoutuksen vaikutusten simulointi jätettään tulevaisuuden työksi.

Learning SLIPNET koostuu SLIPNETin tapaan leksikaalis-semanttisesta aliverkosta ja foneemialiverkosta. Leksikaalis-semanttinen aliverkko on kahdella piilokerroksella varustettu MLP-neuroverkko. Foneemialiverkko koostuu kahdesta erillisestä MLP-neuroverkosta, joista ensimmäiseen on tallennettu vokaalit ja toiseen konsonantit. Vokaalit ja konsonantit on erotettu toisistaan, koska todennäköisyys niiden keskinäiselle sekoittumiselle on erittäin pieni [Tikkala ja Juhola, 1995]. Ilman erillisiä vokaali- ja konsonanttiverkkoja Learning SLIPNET ei käytännössä voisi toteuttaa vokaali-konsonantti rajoitusta. Samanlainen jako on tehty myös alkuperäisessä SLIPNET-verkossa [esim. Laine *et al.*, 1998]. Learning SLIPNETin kaikki aliverkot ovat autoassosiatiivisia, eli ne yrittävät uudelleentuottaa saamansa syötehahmon. Neuroverkon rakennetta on esitetty kuvassa 4.1.



Kuva 4.1. Learning SLIPNET -neuroverkon arkkitehtuuri.

Koko järjestelmän syötekerroksena on kerros Sem1. Kerros Sem2 on leksikaalis-semanttisen aliverkon tuloskerros. Kerrokset Kon1 ja Vok1 ovat foneemialiverkon konsonantti- ja vokaaliverkon syötekerroksia. Vastaavasti Kon2 ja Vok2 ovat konsonantti- ja vokaaliverkon tuloskerroksia. Kerrosten yhteyteen merkityt neuronien lukumäärät riippuvat tulos- ja syötekerrosten kohdalla syötteiden ja tulosten koodaustavasta. Piilokerrosten koot on määrätty kokeellisesti (ks. luvut 4.2 ja 4.3).

Syötteeksi Learning SLIPNETille annetaan nimettävän sanan semanttinen esitys, jonka leksikaalis-semanttinen aliverkko yrittää tuottaa uudelleen. Aliverkon tuottama tulos, sanan lemma, syötetään eteenpäin foneemialiverkolle, joka yrittää tuottaa lemmaa vastaavat foneemit perätysten. Foneemialiverkon tuottama foneemijono tulkitaan verkon vastaukseksi eli kohdesanan nimeksi. Seuraavissa aliluvuissa esitellään tarkemmin Learning SLIPNETin aliverkkoja, sekä niiden toteutusta. Ensin täytyy kuitenkin tarkastella syötteiden ja tulosten koodausta leksikaalis-semanttiselle ja fonologisille aliverkoille.

4.1. Syötteiden ja tulosten koodaus

Jotta MLP-neuroverkolle voitaisiin opettaa jokin tehtävä, täytyy tehtävä esittää verkolle joukkona syötevektoreita ja niihin liittyviä tulosvektoreita. Kuten jo luvussa 2.4 todettiin, tämä ei periaatteessa rajoita ongelmatyyppejä, joita MLP-neuroverkoilla voidaan ratkaista. Käytännössä sopivan koodaustavan kehittäminen voi olla kuitenkin vaikeaa.

Edellisessä luvussa esitetyillä neuroverkkomalleilla syötteiden ja tulosten koodaaminen tapahtui paikallisesti, eli verkkojen neuronit vastasivat yhtä

sanaa tai foneemia. MLP-neuroverkoilla olisi myös mahdollista koodata syötteet ja tulokset vastaavasti. Käytännössä tästä seuraisi kuitenkin se, että syöte- ja tulosvektoreista tulisi suuridimensiosia (ks. luku 2.4). Tavoitteena onkin muodostaa sellainen tiivis semanttinen koodaus, joka huomioi mahdollisimman hyvin sanojen semantiikan samankaltaisuudet ja erilaisuudet. Samoin foneemien koodamisen yhteydessä sopiva koodaustapa säilyttää foneemien keskinäiset suhteet.

Learning SLIPNETille opetettiin kaikkiaan 279 sanaa. Sanat saatiin potilaan "YK" nimeämistestistä (190 sanaa) [Laine, 2002], jota täydennettiin Snodgrassin ja Vanderwartin [1980] käyttämillä sanoilla (89 sanaa). Kaikki simulaatiossa käytetyt sanat on esitetty liitteessä 2. Seuraavissa aliluvuissa käsitellään sanojen semantiikan ja fonologian koodaamista tarkemmin sekä perustellaan niille valitut koodaustavat.

4.1.1. Semantiikan koodaaminen

Suoraviivainen tapa sanojen semantiikan koodaamiseksi olisi paikallisen koodauksen käyttäminen, jossa jokainen simulaation sana koodattaisiin omaan dimensioonsa. Paikallisessa koodauksessa sanaa vastaava komponentti asetettaisiin ykköseksi muiden komponenttien ollessa nollia. Menetelmällä on yksinkertaisuudestaan huolimatta paikallisen koodauksen heikkoudet. Ensimmäinen heikkous on koodauksen epätaloudellisuus: syöte- ja tulosvektoreissa täytyisi olla yksi dimensio kullekin simulaation sanalle. Learning SLIPNETillä tehtävien simulaatioiden yhteydessä sanoja on 279 kappaletta, joten syöte- ja tulosvektoreiden dimensio olisi 279. Tämä riittää jo yksin paikallisen koodauksen hylkäämiseen. Lisäksi paikalliseen esitykseen ei voi koodata sanojen semanttista samankaltaisuutta, sillä sanan etäisyys mihinkä tahansa muuhun sanaan on aina vakio (vakion arvo riippuu käytettävästä etäisyysmitasta). Näin ollen kehittyneempien semantiikan koodaustapojen käyttäminen on välttämätöntä.

Toinen vaihtoehto semantiikan koodaamiseksi on ns. *semanttisten piirteiden* (semantic features) käyttäminen. Piirrepohjaisessa semantiikan esittämisessä valitaan joukko ominaisuuksia, joiden avulla sanojen semantiikkaa voidaan kuvailla. Näitä ominaisuuksia kutsutaan semanttisiksi piirteiksi. Sanojen semantiikka muodostetaan asettamalla siihen liittyvät piirteet ykkösiksi muiden piirteiden ollessa nollia. Täten semanttisten piirteiden avulla tapahtuvaa semantiikan koodaamista voidaan pitää hajautettuna koodauksena. Esimerkiksi sanan "susi" semanttinen esitys voisi koostua mm. piirteistä "nisäkäs" ja "peto". Vaikka yksittäisten sanojen kuvaaminen onnistuukin suhteellisen helposti, on

pienen mutta kattavan piirrejoukon löytäminen pienellekin sanajoukolle vaikea tehtävä.

Esimerkiksi Hinton ja Sejnowski [1986] käyttivät kaikkiaan 30 semanttista piirrettä, joiden avulla koodattiin simulaatiossa käytetyt 40 keinotekoista "sanaa". Hinton ja Shallice [1991] käyttivät 68 semanttista piirtettä mallintaessaan dysleksiapotilaiden nimeämistä (dysleksia on aivovaurion seurauksena syntynyt lukihäiriö). Simulaatiossa oli yhteensä 40 englanninkielen kolme- tai neljäkirjaimista sanaa, joista kuhunkin liittyi keskimäärin 15 semanttista piirrettä. Täten keskimäärin 53 semanttista piirrettä oli käyttämättä yhtä sanaa kohti, joten myös semanttisten piirteiden avulla saatava koodaus on melko epätaloudellinen. Esimerkkejä Hintonin ja Shallicen [1991] käyttämistä semanttista piirteistä on annettu taulukossa 4.1.

id	Piirre
1	koko < jalka
2	jalka < koko < 2 jaardia
3	koko < 2 jaardia
4	päämuoto 1D
5	päämuoto 2D
6	poikkileikkaus suorakaide
7	poikkileikkaus ympyrä
8	jalallinen
9	valkoinen
10	ruskea
…	…
68	komponentti

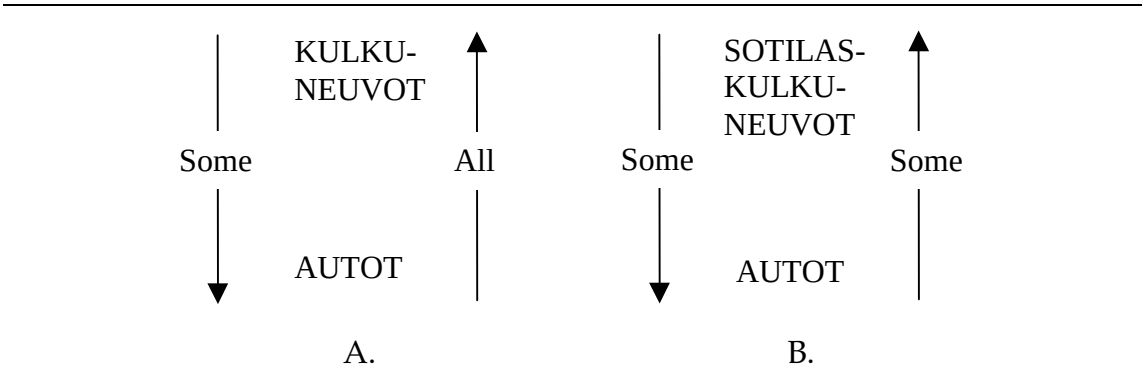
Taulukko 4.1. Hintonin ja Shallicen [1991] käyttämiä semanttisia piirteitä.

Toinen syvällisempi semanttisiin piirteisiin liittyvä ongelma on niiden valinta. Hintonin ja Shallicen [1991] käyttämät sanat oli valittu viidestä konkreettisesta luokasta: kodin esineistä (indoor objects), eläimistä, ruumiinosista, elintarvikkeista sekä ulkona olevista kohteista (outdoor objects). Samaan kategoriaan sijoittuvilla sanoilla oli tyypillisesti useampia yhteisiä semanttisia piirteitä kuin eri kategoriaan kuuluvilla sanoilla. Hinton ja Shallice eivät kuitenkaan osoittaneet, että heidän valitsemansa semanttisten piirteiden joukko olisi kattava, tai esittäisi sanojen todelliset semanttiset samankaltaisuudet.

Koska myös semanttisten piirteiden käyttäminen semantiikan koodamiseksi johtaa suuridimensiosiin syötevektoreihin, soveltuu koodaustapa huonosti MLP-neuroverkoille. Tämän vuoksi semantiikan koodaus tehtiin kolmannella tavalla. Tämäkään koodaustapa ei takaa, että sanojen väliset todelliset semanttiset erot olisivat edustettuina koodauksessa. Tämä on itse asiassa hyvin epätodennäköistä. Menetelmä takaa kuitenkin sen, että koodauksesta saadaan hyvin tiivis. Ideana on muodostaa hajautettu semanttinen koodaus algoritmisesti satunnaisvektoreista *semanttisen puun* avulla.

Semanttinen puu on *ylä-* ja *alakäsitteiden* avulla toteutettu sanojen puumainen luokitus. Puun solmut määrittelevät kategoriota, joihin simulaatiossa käytettävät sanat sijoittuvat. Puun juuri on yläkäsite kaikille puun solmuille. Jokaisen solmun v lapsi c on solmun v alakäsite ja solmu v on vuorostaan solmun c yläkäsite. Mitä syvemmälle semanttisessa puussa mennään sitä kapeampialaiseksi solmun määräämä käsite muuttuu.

Puun ylä- ja alakäsitehierarkia noudattaa ISO:n [1986] antamia suosituksia ylä- ja alakäsitteiden tunnistamisesta. Keskeisenä ominaisuutena kaikilla puun solmuilla on, että ne toteuttavat "all-and-some" -testin [ISO, 1986]. Testiä on havainnollistettu kuvassa 4.2.



Kuva 4.2. "all-and-some" -testi.

Kuvassa 4.2A yläkäsite-ehdokkaana on "kulkuneuvot" ja alakäsite-ehdokkaana "autot". Koska aito sisältymisrelaatio

$$\text{"autot"} \subset \text{"kulkuneuvot"},$$

toteuttaa pari "kulkuneuvot" - "autot" "all-and-some"-testin. Täten käsitepari on hyväksyttävä ylä - alakäsitepari ja se voitaisiin hyväksyä semanttiseen puuhun.

Kuvassa 4.2B yläkäsite-ehdokkaana on "sotilaskulkuneuvot" ja alakäsite-ehdokkaana "autot". Nyt "all-and-some"-testi ei toteudu, sillä

$$\text{"autot"} \not\subset \text{"sotilaskulkuneuvot"}.$$

Täten kuvan 4.2B käsitepari ei ole hyväksyttävä ylä -alakäsitepari. Siksi sitä ei voida myöskään hyväksyä semanttiseen puuhun.

Jokainen simulaation sana on sijoitettu sellaiseen solmuun v, joka määrittää "täydellisimmin" sanan paikan semanttisessa puussa. Sana sijoitettiin solmuun v, jos

1. solmun v vanhempi ei määrittele "riittävän tarkasti" sanan asemaa semanttisessa puussa ja
2. solmun v lapset määrittelisivät sanan aseman semanttisessa puussa "liian tarkasti".

Termejä "riittävän tarkasti" ja "liian tarkasti" ei voi määritellä kunnolla. Kyseessä on siis intuitioon perustuva luokittelu, aivan kuten Hintonilla ja Shallicella [1991]. Seuraavia periaatteita on kuitenkin noudatettu termien "liian tarkasti" ja "riittävän tarkasti" soveltamisen yhteydessä: solmu v määrittelee "liian tarkasti" sanan, jos käsitteen ala koskee vain kyseistä sanaa. Täten esimerkiksi sanaa "leijona", ei sijoitettaisi solmuun, joka määrittelee vain leijonat. Sanaa "leijona" ei sijoitettaisi myöskään solmuun "eläimet", sillä tällöin liian erilaiset eläimet joutuisivat samaan solmuun, eikä sana "leijona" tulisi määriteltyä "riittävän tarkasti". Täten sana "leijona" sijoitetaan johonkin solmusta "eläimet" lähtevän alipuun solmuun, kuten solmuun "eläimet" → "nisäkkäät" → "(nisäkäs)pedot". Tärkeä ehto puun solmujen tunnistamisen yhteydessä oli "all-and-some" -testin toteutuminen. Edellä mainituin periaattein muodostettu semanttinen puu Learning SLIPNETille opettavista sanoista on kokonaisuudessaan esitetty liittessä 3.

Kun semanttinen puu on muodostettu, voidaan algoritmin 4.1 avulla muodostaa sitä vastaava sanojen semantiikan vektoriesitys. Algoritmi jakaa syötteeksi saamansa satunnaisvektorit klustereihin semanttisen puun määräämällä tavalla ja valitsee puun solmuissa oleville sanoille semanttisen esityksen solmua vastaavien satunnaisvektorien joukosta. Syötteekseen algoritmi saa neljä parametria:

1. puun *tree*, jossa määritellään sanojen semanttinen hierarkia. Puun jokaiseen solmuun voi olla tallennettu sanoja. Lisäksi jokaisella solmulla voi olla n kpl lapsia.
2. joukon C satunnaisesti valittuja vektoreita $\mathbf{v} \in \mathbb{R}^m$, $m \in \mathbb{Z}_+$,
3. reaaliluvun d_1, joka parametrisoi vektoreiden etäisyyttä oman klusterinsa keskipisteeseen,
4. reaaliluvun d_2, joka parametrisoi vektoreiden etäisyyttä kaikkien paitsi oman klusterinsa keskipisteeseen.

Algoritmin toimintaperiaate on seuraava: Olkoon (semanttisen) puun *tree* juuri r ja n juuren r lasten lukumäärä. Jos puun juuressa r on sanoja, valitaan jokaiselle sanalle vastine $\mathbf{v} \in C$. Olkoon sanoja m kappaletta. Sanojen vektoriesityksen valinta tehdään klusteroimalla joukon C vektorit m klusteriin. Tämän jälkeen kunkin sanan semanttinen koodaus on sitä vastaavan klusterin keskipistevektori $\mathbf{v}$. Klusterointiin käytetään tunnettua K-means-algoritmia (K:n keskiarvon-algoritmi) [esim. Hand *et al.*, 2001].

Sanojen valinnan jälkeen syötevektoreiden joukko C jaetaan joukkoihin C_i, missä $i \in \{0,1,\ldots,n-1\}$. Jako tehdään jälleen K-means-algoritmilla. Muodostetuista joukoista C_i muodostetaan joukot C'_i poistamalla etäisyysparametrien avulla

sellaiset vektorit $\mathbf{u} \in C_i$, jotka ovat liian kaukana joukon C_i keskipistevektorista tai ovat liian lähellä jonkun toisen joukon C_j, $i \neq j$, keskipistevektoria. Etäisyysmittana käytetään euklidista etäisyyttä d_E, joka lasketaan vektoreille $\mathbf{x} = (x_1, x_2, \ldots, x_l)$ ja $\mathbf{y} = (y_1, y_2, \ldots, y_l)$ kaavalla

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{l} (x_i - y_i)^2} \ . \tag{4.1}$$

Nyt joukot C'_i voidaan esittää joukko-opillisesti muodossa

$$C'_i = \left\{ \mathbf{u} \in C_i \mid d_E(\mathbf{u}, \hat{\mathbf{u}}_i) \leq d_1 \delta_i \wedge d_E(\mathbf{u}, \hat{\mathbf{u}}_j) > d_2 \delta_j,\ i \neq j \right\},$$

missä $\hat{\mathbf{u}}_i$ on joukon C_i keskipistevektori ja δ_i on joukon C_i vektoreiden keskimääräinen euklidinen etäisyys keskipistevektorista $\hat{\mathbf{u}}_i$. Etäisyysparametrien avulla joukot C'_i saadaan erottumaan toisistaan.

Lopuksi algoritmi kutsuu juuren r jokaisen lapsen $child_i$ kohdalla rekursiivisesti itseään. Uutena syötteenä algoritmille on lapsesta $child_i$ lähtevä alipuu ja uutena syötejoukkona on joukko C'_i, maksimietäisyyden parametrin d_1 ja minimietäisyyden parametrin d_2 pysyessä ennallaan.

Input: Syötevektoreiden joukko C,
semanttinen puu *root*,
etäisyysparametrit d_1 ja d_2.

Output: Sanojen vektoriesitys

1. **begin**
2. **for** $i \leftarrow 0$ **to** *root.numberOfWords()* $- 1$ **do**
3. Laske vektori $\mathbf{v}_i$ vastaamaan sanaa i.
4. **od**
5. $n \leftarrow$ *root.numberOfChildren()*.
6. **if** $n > 0$ **then**
7. Muodosta n lapsijoukkoa C_i klusteroimalla syöteklusterin C sisältämät vektorit *K-means*-algoritmilla.
8. Muokkaa muodostettuja lapsiklustereita etäisyysparametreilla d_1 ja d_2.
9. **fi**
10. **for** $i \leftarrow 0$ **to** $n - 1$ **do**
11. Kutsu algoritmia 4.1 arvoilla *root.childAt(i)*, C_i sekä etäisyysparametreilla d_1 ja d_2.
12. **od**
13. **end**

Algoritmi 4.1. Sanojen semanttisen esityksen muodostaminen semanttisen puun avulla.

Algoritmin tuloksena on puussa *tree* määriteltyjen sanojen vektoriesitys, jossa kaksi erillistä vektoria ovat lähempänä toisiaan silloin, kun ne ovat lähellä toisiaan semanttisessa puussa. Tämä johtuu siitä, että mitä syvemmälle semanttisessa puussa mennään, sitä pienempi etäisyys siinä esiintyvien sanojen vektoriesityksillä toisistaan on. Semanttisesta koodauksesta saadaan haluttaessa hyvin tiivis, sillä satunnaisvektorien dimensio voidaan valita vapaasti.

Algoritmin 4.1 asymptoottinen aikakompleksisuus saadaan johdettua *K*-means-algoritmin aikakompleksisuuden perusteella. Analyysissa oletetaan yksinkertaisuuden vuoksi, että klustereita ei muotoilla algorimin suorituksen aikana.

On tunnettua, että *K*-means-algoritmin aikakompleksisuus on $O(KnI)$, missä K on klustereiden lukumäärä, n on syötteen koko ja I on iteraatioiden lukumäärä [Hand *et al.*, 2001]. Oletetaan nyt, että jokaisella puun sisäsolmulla on K lasta ja että *K*-means-algoritmi tekee aina I iteraatiota (tämä ei rajoita yleisyyttä, sillä K ja I voidaan valita "riittävän" suureksi). Edelleen yleisyyttä rajoittamatta voidaan olettaa, että *K*-means-algoritmin tuottamat klusterit ovat saman kokoisia. Oletetaan vielä, että sanoja on talletettu ainoastaan puun lehtiin ja että niitä on jokaisessa lehdessä l kpl. Olkoon s puun lehtien lukumäärä. Tällöin puun korkeus on $\log_K s$ (puu on täydellinen).

Algoritmin suorituksen edetessä syötevektorien joukko jaetaan aina K osaan siirryttäessä puussa alaspäin. Tällöin puun tasolla i olevassa solmussa suoritettavaan laskentaan kuluva aika on *K*-means-algoritmin aikakompleksisuuden perusteella

$$O\left(\frac{KnI}{K^i}\right) = O\left(\frac{nI}{K^{i-1}}\right). \tag{4.2}$$

Koska tasolla i on K^i solmua, on tasolla i tehtävä kokonaislaskenta kaavan (4.2) perusteella

$$K^i O\left(\frac{nI}{K^{i-1}}\right) = O(KnI). \tag{4.3}$$

Koska puun korkeus on $\log_K s$, saadaan klusterointeihin käytetty laskenta-aika puun korkeuden ja kaavan (4.3) perusteella kaavalla

$$(\log_K s) O(KnI) = O(KnI \log s). \tag{4.4}$$

Kun oletetaan, että lehdissä tapahtuva sanojen valinta vie lineaarisesti aikaa suhteessa sanojen määrään l, saadaan algoritmin kokonaislaskenta-aika kaavan (4.4) ja lehdissä tehtävän kokonaislaskennan avulla kaavalla

$$O(KnI \log s) + O(sl). \tag{4.5}$$

Kaavan (4.5) perusteella algoritmin 4.1 aikakompleksisuus on $O(KnI \log s)$, jos $(\log_K s)KnI >> sl$. Päinvastaisessa tapauksessa sen aikakompleksisuus on $O(sl)$.

4.1.2. Semanttisen koodauksen analysointia

Semantiikan koodaus muodostettiin algoritmin 4.1 avulla käyttämällä syötevektoreina 10000 joukon $\mathbb{R}^3$ satunnaisvektoria, joiden komponentit kuuluvat välille [0,1]. Etäisyysparametrien olivat d_1 = 1.5 ja d_2 = 1.5. Pienemmällä syötevektoreiden määrällä etäisyysparametrit karsivat pois niin paljon vektoreita, että joihinkin klustereihin jäi vähemmän vektoreita kuin koodattavia sanoja. Toisaalta suuremman syötevektorijoukon käyttäminen ei paranna tulosta, sillä syötevektorit ovat jakautuneet tasaisesti koko syötejoukon alueelle. Suuremman syötejoukon käyttäminen johtaa vain syötejoukon tiheyden kasvamiseen, mutta ei muuta syötevektoreiden jakaumaa, joka vaikuttaa algoritmin lopputulokseen. Lisäksi ylisuuren syötejoukon käyttäminen kasvattaa turhaan algoritmin laskenta-aikaa. Etäisyysparametreista d_1 valittiin mahdollisimman pieneksi ja d_2 mahdollisimman suureksi. Kokeellisesti huomattiin, että karsintavaiheessa (algoritmi 4.1, askel 8) ei karsittu liikaa vektoreita pois suhteessa valittavien sanojen määrään, kun oli d_1=d_2=1.5. Täten semantiikan koodaus muodostettiin käyttämällä näitä etäisyysparametrien arvoja.

Semantiikan koodaus onnistuisi algoritmin 4.1 avulla myös joukon $\mathbb{R}^2$ tai $\mathbb{R}$ satunnaisvektoreilla. Tällöin neuroverkon opettaminen koodatulla aineistolla todettiin kuitenkin erittäin hankalaksi tai mahdottomaksi. Tämä johtuu siitä, että siirryttäessä pienempidimensioiseen avaruuteen algoritmin syötteenä olevien satunnaisvektorien keskimääräinen (euklidinen) etäisyys pienenee. Siksi koodatut sanat ovat keskimäärin lähempänä toisiaan kuin suurempidimensioisissa avaruuksissa.

Syötevektoreiden joukoksi valittiinkin pienidimensioisin reaalilukuvektorien joukko, jonka avulla muodostettu koodaus oli opetettavissa Learning SLIP-NETille. Koodaus tulkittiin opittavaksi, jos oli olemassa korkeintaan kahdesta 20-neuronisesta piilokerroksesta koostuva MLP-neuroverkko, joka ei tehnyt yhtään virhettä koko sanajoukon nimeämisen yhteydessä. Neuroverkon tulos tulkittiin virheeksi, jos sen tuottamaa tulosvektori **o** oli lähempänä jotain muuta kohdesanojen joukon vektoria **t**' kuin haluttua kohdesanaa **t**. Opetuksen kustannusfunktiona käytettiin *neliövirhesummaa* (sum of squared errors, SSE). Se määritellään jokaiselle tulosneuronille kaavalla

$$E_{SSE} = \sum_{i=1}^{n} \left(d_i - y_i\right)^2, \tag{4.6}$$

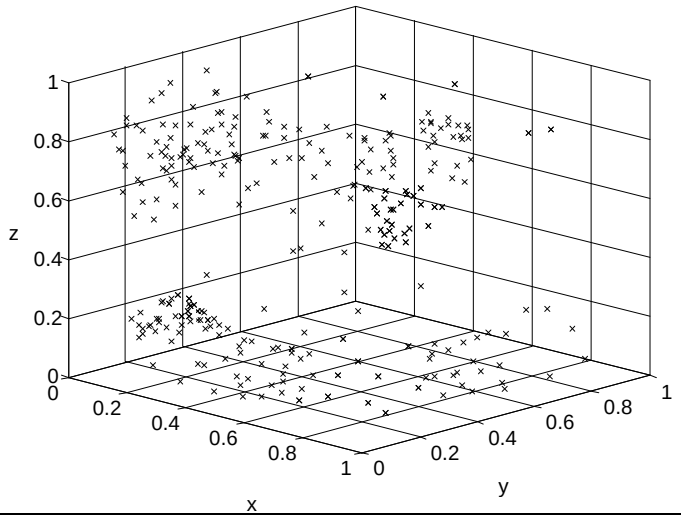
missä d_i on syötehahmon i haluttu tulos ja y_i on neuronin tuottama tulos. Opetuksen yhteydessä verkkojen SSE yritettiin saada pienemmäksi kuin E_{SSE}<0.005 (pienemmillä neliövirhesummien arvoilla opetukseen tarvittava aika moninkertaistui). Opetuskierroksia käytettiin maksimissaan 10000. Testien pe-

rusteella havaittiin, että joukon $\mathbb{R}$ ja $\mathbb{R}^2$ vektoreilla semantiikan koodaus ei ollut opittava. Sen sijaan joukon $\mathbb{R}^3$ vektoreilla opetustavoitteet saavutettiin, joten algoritmin 4.1 syötevektoreiksi valittiin joukon $\mathbb{R}^3$ vektoreita.

Semantiikan koodausta tutkittiin suorittamalla sille hierarkkinen klusterointi [esim. Hand *et al.*, 2001]. Klusterointi suoritettiin SPSS-ohjelmistolla ja klustereiden välisenä etäisyysmittana käytettiin *naapurikeskiarvoa* (average-linkage). Klusteroinnin tuloksena saatu dendogrammi on esitetty liitteessä 4 A. SPSS-ohjelmiston tuottamaa dendogrammia on muutettu siten, että lehtitason klustereita on yhdistetty keskenään dendogrammin selkeyttämiseksi. Täten liitteen 4 A dendogrammissa ei esiinny yksittäisiä sanoja, vaan lehtitasonkin solmut ovat useiden sanojen muodostamia klustereita. Mielenkiintoinen huomio on kuitenkin se, että suurin osa semanttisessa puussa määritellyistä kategorioista erottuu myös dendogrammissa. Tämä kertoo siitä, että semantiikan koodaus on onnistunut hyvin suhteessa muodostettuun semanttiseen puuhun (vrt. liite 3). Poikkeuksia ovat vain kategoriat "leikkikalut" ja "kirjallisuus", jotka ovat sekoittuneet keskenään. Lisäksi "hyönteiset" liittyvät dendogrammin mukaan läheisemmin kasveihin, kuin eläimiin. Toinen huomio on "ruumiinosien" jakautuminen kahteen eri kategoriaan (ruumiinosat I ja II). Ruumiinosat II -kategoriaan kuuluu kaksi sanaa, sanat "nenä" ja "peukalo".

Nisäkkäät on merkitty dendogrammiin yhdeksi käsitteeksi dendogrammin selkeyden säilyttämiseksi. Nisäkäs-kategoriaan kuuluvien sanojen koodausta tutkittiin siksi erikseen klusteroimalla vain ne uudelleen. Tämän klusteroinnin tulokset on esitetty liitteen 4 B dendogrammissa. Klusterointimenetelmät ja -ohjelmisto olivat samat kuin koko aineiston klusteroinnin yhteydessä. Klusterointi osoittaa, että myös nisäkkäiden semantiikan koodaus on onnistunut hyvin suhteessa semanttiseen puuhun. Esimerkiksi ihmiset -kategoriaan kuuluvat sanat erottuvat muista nisäkässanoista täydellisesti. Semanttisesta puusta poikkeaa selvästi vain sana "leopardi", joka klusteroinnin mukaan liittyy läheisemmin kotieläimiin kuin petoihin.

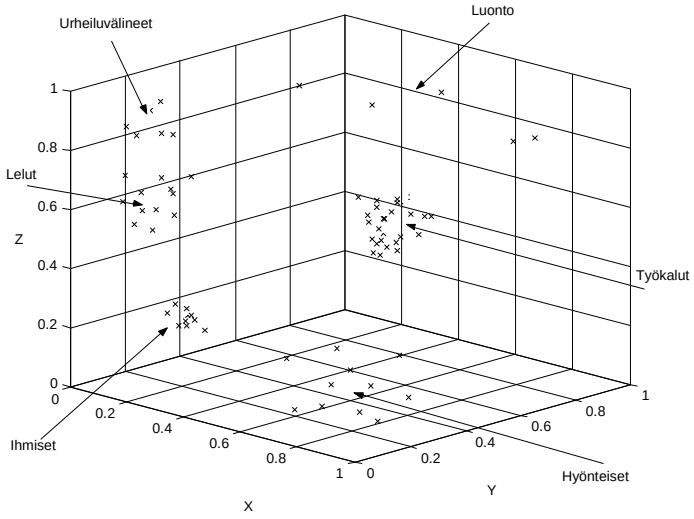
Semanttisessa puussa (ks. liite 3) ensimmäisenä oleva käsitteiden jako "elottomiin" ja "elollisiin" käy ilmi kuvan 4.3 keskellä olevan tyhjän alueen kautta. Alue on syntynyt kun algoritmi 4.1 on jakanut semanttisen puun perusteella käsitteen "oliot" käsitteisiin "elolliset" ja "elottomat". Jako on noudatellut suurin piirtein tasoa z=0.5. Jaon jälkeen suoritettu klustereiden muokkaaminen on karsinut pois keskialueen vektorit. Tämän jälkeen "elollisille" on jäänyt tila $x \in [0,1]$, $y \in [0,1]$, $z \in [0,0.35]$ ja "elottomille" tila $x \in [0,1]$, $y \in [0,1]$, $z \in [0.55,1.0]$. Muuttujille x, y ja z ei voida antaa mitään tarkkaa merkitystä, sillä algoritmi 4.1 on muodostanut koodauksen satunnaisluvuista.



Kuva 4.3. Algoritmin 4.1 tuottama semantiikan koodaus.

Kuvaan 4.4 on merkitty muutamien semanttisten luokkien paikkoja kuvan 4.3 koodauksessa. Käsitteet "ihmiset", "lelut", "urheiluvälineet" ja "työkalut" ovat melko tiiviitä, eli käsitteiden alaan kuuluvat sanat muodostavat tiiviin alueen. Tämä kertoo siitä, että klusterit, joista sanojen semanttiset koodaukset on valittu, ovat olleet melko pieniä. Klustereiden pienuus johtuu siitä, että käsitteet ovat semanttisessa puussa melko syvällä, tai että niillä on useita sisarsolmuja (sisarille jäävä tila jakautuu melko tasaisesti niiden kesken).

Käsitteet "luonto" ja "hyönteiset" ovat sen sijaan melko hajanaisia. Tämä johtuu siitä, että ne sijaitsevat melko lähellä semanttisen puun juurta ja niillä on vain muutama sisarsolmu. Tästä seuraa, että käsitteille jäävän klusterin koko kasvaa ja klusterista valittavat sanat joutuvat hajalleen. Toisin sanoen, mitä yleisemmin käsite on määritelty semanttisessa puussa, sitä hajanaisempi käsiteen piiriin kuuluvien sanojen koodauksesta tulee. Tämä on perusteltua siinä mielessä, että laaja-alaisen käsitteen piiriin kuuluvat sanat liittyvät semanttisessa mielessä toisiinsa harvemmin kuin pienialaisen käsitteen piiriin kuuluvat sanat.



Kuva 4.4. Eräitä algoritmin 4.1 tuottamia semanttisia luokkia.

Vaikka semanttisen puun avulla muodostettu semantiikan koodaus on erittäin tiivis, on sillä myös heikkouksia. Selvin heikkous on semanttisten suhteiden yksinkertaistaminen ylä- alakäsitepareiksi. Esimerkiksi semanttisessa puus-sa on mahdotonta ilmaista sanojen "hiiri" ja "juusto" välinen assosiaatiosuhde. Kuvasta 4.4 voi saada sellaisen käsityksen, että algoritmi olisi tarkoituksella asettanut käsitteet "ihmiset", "lelut" sekä "urheiluvälineet" lähekkäin. Näin ei kuitenkaan ole, sillä algoritmilla ei ole tietoa assosiaatiosuhteista. Täten esimerkiksi "ihmisten" ja "lelujen" kohtalainen läheisyys johtuu vain sattumasta. Assosiaatio-ongelma voitaisiin kyllä itse puussa kiertää sallimalla eri alipuiden väliset suhteet, jolloin semanttisesta puusta muodostuisi verkko. Semanttisen verkon käsitteleminen algoritmin 4.1 avulla ei kuitenkaan onnistuisi, joten semanttisen verkon käyttäminen vaatisi uuden algoritmin kehittämisen. Tähän kehitystyöhön ei kuitenkaan haluttu ryhtyä, sillä tutkielman ensisijainen tavoite oli SLIPNETin oppivan version toteuttaminen.

Toinen esitetyn koodaustavan heikkouksista tulee esille liian yksityiskohtaisen semanttisen puun yhteydessä. Jos esimerkiksi kuvan 4.4 käsitettä "ihmiset" edelleen jaettaisiin alakäsiteisiin, tulisi alakäsitteitä vastaavista klustereista niin tiiviitä, ettei neuroverkko erottaisi niiden piiriin kuuluvia sanoja toisistaan. Tämä johtuu siitä, että jaettaessa klusteri osiin, kunkin osaklusterin sisällä olevien vektorien keskimääräinen etäisyys toisistaan pienenee. Kun klusterin koko tulee riittävän pieneksi, neuroverkko tulkitsee klusterin pisteet yhdeksi pisteeksi. Ongelman esiintuloa voidaan pitkittää opettamalla verkkoa pidempään. Käytännössä tämä johtaa kuitenkin tarvittavan opetusajan moninkertaistumiseen, joten menetelmä käy lopulta mahdottomaksi. Ratkaisuksi ongelmaan voi kokeilla koodauksen muodostamista jossakin suurempi-

dimensioisessa avaruudessa, jolloin vektorien keskimääräinen etäisyys toisistaan kasvaa (mikäli algoritmin syötevektoreiden määrä pidetään vakiona).

Kolmantena heikkoutena voidaan pitää sitä, että muodostettu koodaus on hajautettu. Siksi leksikaalis-semanttiselle aliverkolle voidaan esittää vain yksi sana kerrallaan (ks. luku 2.4). Jos verkolle halutaan esittää useampia sanoja kerrallaan, täytyy semantiikan koodaustapaa muuttaa. Tämän tutkimuksen aikana useampia sanoja ei kuitenkaan tarvinnut esittää yhtäaikaa verkolle, joten luvussa 2.4 esiteltyä sitomisongelmaa ei esiinny.

Semanttisen puun avulla suoritetun semanttisen koodauksen hyvät puolet ovat kuitenkin ilmeiset. Erityisesti tuloksina saatavien vektorien pienidimensioisuus tekee menetelmästä erittäin sopivan MLP-neuroverkkoja käytettäessä. Lisäksi menetelmä on nopea suurenkin sanajoukon semantiikan koodaamiseksi verattaessa esimerkiksi käsin tehtävään piirrepohjaiseen koodaukseen. Tästä ominaisuudesta voi olla hyötyä, jos simulaatioissa käytettävää sanajoukkoa halutaan kasvattaa. Lisäksi semanttisen koodauksen muuttaminen on melko yksinkertaista, sillä muutoksia tarvitsee tehdä vain semanttiseen puuhun. Tämän jälkeen uuden semanttisen koodauksen saa ajamalla algoritmin 4.1 päivitetyllä semanttisella puulla.

4.1.3. Foneemien koodaaminen

Toinen Learning SLIPNETin aliverkoista on foneemialiverkko. Se tuottaa leksikaalis-semanttisen aliverkon tuottaman lemman fonologisen muodon hakemalla lemmaan liittyvät foneemit peräkkäin. Foneemialiverkko simuloi foneemien hakua tuottamalla uudelleen sille esitetyt syötefoneemit. Täten syötteiden koodaus foneemialiverkolle on samalla sen tulosten koodaamista. Vokaalien ja konsonanttien sekoittumisen estämiseksi foneemialiverkko on jaettu konsonantti- ja vokaaliverkkoon. Siksi vokaaleille ja konsonanteille on myös mahdollista käyttää erilaisia koodaustapoja.

Fonologian koodaaminen oli huomattavasti yksinkertaisempaa kuin semantiikan koodaaminen. Tämä johtuu siitä, että sanojen äänneasu esitetään foneemien avulla, joita suomenkielessä on 19 - 25 kappaletta laskentatavasta riippuen [Karlsson, 1983]. Foneemien erottavat piirteet on myös tunnistettavissa, jolloin foneemit voidaan luokitella tunnistettujen piirteiden perusteella. Näitä piirteitä ovat mm. foneemin soinnillisuus / soinnittomuus, sekä niiden ääntöpaikka ja -tapa [Yule, 1999]. Lisäksi erityisesti suomen yleispuhekielen foneemit ja oikeinkirjoitusjärjestelmän grafeemit vastaavat läheisesti toisiaan muutamaa poikkeusta lukuunottamatta [Karlsson, 1983]. Tämän vuoksi suomenkielen foneemien koodaaminen on melko yksinkertaista, sillä käytännössä riittää koodata taulukossa 4.2 esitetyt grafeemit. Koska simulaatiossa

käytetyissä sanoissa ei esiintynyt vierasperäisiä foneemeja /*c,q,w,x,z,å*/, ei niitä myöskään ole mukana esitetävissä foneemien koodauksissa. Täten suurenkin sanajoukon fonologian koodaamiseen tarvitaan vain 23 grafeemia. Tästä eteenpäin puhuttaessa foneemeista tarkoitetaan todellisuudessa grafeemeja. Tällä ei kuitenkaan ole merkitystä suomen kielessä grafeemien ja foneemien vastaavuuden johdosta.

Koodattavat grafeemit (foneemit)			
/*a*/	/*h*/	/*n*/	/*u*/
/*b*/	/*i*/	/*o*/	/*v*/
/*d*/	/*j*/	/*p*/	/*y*/
/*e*/	/*k*/	/*r*/	/*ä*/
/*f*/	/*l*/	/*s*/	/*ö*/
/*g*/	/*m*/	/*t*/	
Yhteensä 23 kpl.			

Taulukko 4.2. Koodattavat suomen kielen grafeemit.

Koodaustehtävää helpottaa myös se, että valmiita foneemien esitystapoja on olemassa useita. Esimerkiksi englannin kielen foneemien koodaamiseksi on esitetty erilaisia menetelmiä [esim. Rumelhart and McClelland, 1986, Miikkulainen, 1997]. Näistä ainakin Miikkulaisen esittämä menetelmä soveltuisi sellaisenaan myös suomen kielen foneemien koodaamiseen. Thyme [1993] on vuorostaan esittänyt menetelmän pelkästään suomen kielen foneemien koodaamiseksi. Konsonanttien osalta päädyttiinkin käyttämään juuri Thymen esittämää menetelmää, tosin hieman muutettuna. Miikkulaisen [1997] esittämää koodaustapaa ei käytetty, sillä koodaus oli ehditty muodostaa ennen artikkelin lukemista. Vokaaleille kehitettiin sen sijaan uusi koodaustapa, jotta vokaalisointua koskevat rajoitukset voitaisiin huomioida simulaation aikana.

Koodatut konsonantit on esitetty taulukossa 4.3. Konsonanttien koodaus eroaa Thymen käyttämästä koodauksesta vain siinä, että siitä on jätetty muuttuja "konsonantti / vokaali" turhana pois (koodauksessa on vain konsonantteja). Muuttujalla "soinnillinen" kuvattaan konsonantin soinnillisuutta (1) tai soinnittomuutta (0). *Soinnillinen* foneemi syntyy, kun äänihuulia puristetaan yhteen. Tällöin keuhkoista virtaava ilma saa äänihuulet värähtelemään, joka havaitaan foneemin soinnillisuutena. *Soinniton* foneemi tuotetaan vuorostaan pitämällä äänihuulet erillään, jolloin ilma pääsee vapaasti virtaamaan keuhkoista ulos. Soinnillisia foneemeja ovat mm. /*b,m,l*/ [Yule, 1999].

Foneemi	Soinnillinen	Ääntötapa		Ääntöpaikka	
/b/	1	1	1	1	1
/d/	1	1	1	1	0
/f/	0	1	0	1	1
/g/	1	1	1	0	0
/h/	0	0	1	0	0
/j/	1	0	1	0	0
/k/	0	1	1	0	0
/l/	1	0	1	0	1
/m/	1	0	0	1	1
/n/	1	0	0	1	0
/p/	0	1	1	1	1
/r/	1	0	1	1	0
/s/	0	1	0	1	0
/t/	0	1	1	1	0
/v/	1	1	0	1	1

Taulukko 4.3. Koodatut konsonantit [Thyme, 1993].

Muuttuja "ääntötapa" kuvaa nimensä mukaisesti foneemin ääntötapaa. Se saa neljä erillistä arvoa, joista arvo (0,0) kuvaa nasaaleita, arvo (0,1) puolivokaaleita, arvo (1,0) frikatiiveja ja arvo (1,1) klusiileja. *Nasaalit* tuotetaan laskemalla kitapurje alas, jolloin ilma pääsee virtaamaan nenäonteloon. Nasaaleita ovat mm. /*m*,*n*/. *Puolivokaalien* artikulaatioon vaikuttaan voimakkaasti niitä seuraavat vokaalit. Tyypillisesti puolivokaaleita tuotettaessa kieli liukuu joko kohti seuraavan vokaalin ääntöpaikkaa tai poispäin edellisen vokaalin ääntöpaikasta. Puolivokaaleja ovat esim. /*j*,*l*,*r*/. *Frikatiiveja* äännettäessä ilmavirta pysäytetään lähes kokonaan ja jäljelle jäänyt ilmavirta "työnnetään" ulos hampaiden välistä. Frikatiiveja ovat /*f*,*s*,*v*/. *Klusiilit* tuotetaan pysäyttämällä ilmavirta kokonaan ja antamalla sen tämän jälkeen purkautua äkkinäisesti. Esimerkkejä klusiileista ovat /*k*,*p*,*t*/ [Yule, 1999].

"Ääntöpaikka"-muuttujalla kuvataan foneemin ääntöpaikkaa. Myös tämä muuttuja saa neljä erilistä arvoa, joista arvolla (0,0) merkitään palatovelaareja sekä foneemia /*h*/, arvolla (0,1) foneemia /*l*/, arvolla (1,0) alveolaareja ja arvolla (1,1) labiaaleja. *Palatovelaarit* äännetään suulaen pehmeän osan alueella (lat. velum palatinum). Palatovelaareja ovat mm. foneemit /*j*,*k*/. *Alveolaarit* äännetään kielen kärki kiinni hammaskuoppaharjanteessa (lat. jugum alveolare), joka sijaitsee ylimmäisten etuhampaiden takana. Alveolaarisia foneemeja ovat mm. /*d*,*t*,*s*,*r*,*l*/. Foneemia /*l*/ ei kuitenkaan voida koodata alveoraaliseksi, koska muuten se ei erottuisi foneemista /*r*/. Ilmeisesti tämän vuoksi Thyme [1993] on koodannut foneemin /*l*/ arvolla (0,1). Hän ei kuitenkaan perustele tai selitä asiaa millään tavalla. Karlsson [1983] luokittelee alveolaarit dentaaleiksi. Näin voidaan tehdä, sillä suomen kielessä ei ole varsinaisia dentaalisia äänteitä [*ð*,*θ*] (esimerkiksi englannin kielen sanojen *three* (*ð*) ja *there* (*θ*) alussa). *Labiaalit*

äännetään käyttämällä joko ylä- ja alahuulta (lat. labium) tai pelkästään alahuulta ja ylähampaita. Labiaalisia foneemeja ovat mm. */p,b,m,f/* [Yule, 1999].

Vokaalien koodaamiseen Thymen menetelmä ei sovi, koska vokaalisoinnun alaiset vokaaliparit eivät erotu toisistaan systemaattisesti tietyn piirteen (esim. place1) osalta. Vokaalisoinnulla tarkoitetaan koko sananmuodon vokaaliston jakaumarajoitusta, missä yksinkertaisessa sananmuodossa ei voi esiintyä sekä etuvokaaleita */y,ö,ä/* että takavokaaleita */a,o,u/* [Karlsson, 1983]. Yhdessä etu- ja takavokaaleita kutsutaan *harmoniavokaaleiksi*. Koska vokaalisointu säilyy myös aivovaurion yhteydessä täytyy vokaalit koodata tavalla, joka estää etu- ja takavokaalien keskinäisen vaihtumisen. Tämän vuoksi harmoniavokaaleita ei koodattu sellaisenaan ollenkaan. Niiden sijasta koodattiin vastaavat morfofoneemit */A,O,U/*, sekä neutraalivokaalit */i,e/*. Morfofoneemeista morfofoneemi */A/* vastaa foneemeja */a,ä/*, morfofoneemi */O/* foneemeja */o,ö/* ja morfofoneemi */U/* foneemeja */u,y/*. Tarkasti ottaen myös neutraalivokaalit koodattiin morfofoneemimuodossa */I,E/*, mutta koska niillä ei ole vokaalipareja, kuten harmoniavokaaleilla, vastaa morfofoneemi */I/* foneemia */i/* ja morfofoneemi */E/* foneemia */e/*.

Vokaaliverkolle opetetaan siis vain morfofoneemit. Kun tiedetään onko tuotettava sana etu- vai takavokaalisana, valitaan simulaation aikana oikea foneemi vokaaliverkon tuottaman morfofoneemin pohjalta seuraavin periaattein:

1. Jos sanassa on etuvokaaleja, tulkitaan morfofoneemit */A,O,U/* foneemeiksi */ä,ö,y/*. Morfofoneemit */I,E/* tulkitaan neutraalivokaaleiksi */i,e/*.
2. Jos sanassa on takavokaaleja, tulkitaan morfofoneemit */A,O,U/* foneemeiksi */a,o,u/*. Morfofoneemit */I,E/* tulkitaan neutraalivokaaleiksi */i,e/*.
3. Jos sanassa on pelkkiä neutraalivokaaleja, tulkitaan verkon tuottamat morfofoneemit */A,O,U/* etuvokaaleiksi */ä,ö,y/*, ja morfofoneemit */I,E/* neutraalivokaaleiksi */i,e/*.

Valintaperiaatteista 1 - 3 seuraa, että etu- ja takavokaalit eivät voi korvautua toisillaan. Lisäksi vain neutraalivokaaleita sisältävän sanan vokaalit voivat korvautua joko toisillaan tai sitten foneemeilla */y,ä,ö/*. Etuvokaalisanan neutraalivokaalit voivat korvautua vain toisillaan tai etuvokaaleilla. Vastaavasti takavokaalisanan neutraalivokaalit eivät voi korvautua etuvokaaleilla. Täten valintaperiaatteet 1 - 3 estävät täysin etu- ja takavokaalien sekoittumisen ja vauriotunutkin verkko noudattaa tiukasti vokaaliharmoniaa.

Jos tuotettavana sanana on esimerkiksi sana "kissa", tuottasi foneemialiverkko tuloksen

/k I s s A/,

missä morfononeemi /I/ tulkittaan foneemiksi /i/ ja morfofoneemi /A/ foneemiksi /a/, koska kissa on takavokaalisana. Vastaavasti, jos tuotettavana sanana on sana "pöytä", tuottaisi foneemiverkko tuloksen

/p O U t A/.

Koska sana "pöytä" on etuvokaalisana tulkittaan morfofoneemi /O/ foneemiksi /ö/, morofofoneemi /U/ foneemiksi /y/ ja morfofoneemi /A/ foneemiksi /ä/.

Vokaalien koodaus muodostettiin Karlssonin [1983] esittämän vokaalien luokittelun pohjalta (ks. taulukko taulukko 4.4).

	Pyöreä		Lavea	
	Takainen	Etinen	Takainen	Etinen
Suppea	u	y		i
Puolisuppea	o	ö		e
Väljä			a	ä

Taulukko 4.4. Vokaalien luokitus [Karlsson, 1983].

Kuten taulukosta 4.4 voi todeta, harmoniavokaaliparit muodostavat pareja pyöreyden suhteen (eli harmoniavokaaliparilla on sama pyöreysarvo). Neutraalivokaaleilla ei ole takaisia vokaalipareja, joten vokaalisoinnun kannalta ne jäävät yksin. Taulukon 4.4 perusteella voidaan muodostaa vokaalien morfofoneeminen esitys, kun muuttuja "takainen / etinen" poistetaan. Tällöin vokaalit voidaan kuvata morfofoneemeina muuttujilla "suppea" ja "pyöreä". Muuttuja "suppea" saa arvot joukosta {0, 0.5, 1}, missä arvo 0 vastaa ominaisuutta suppea ja 1 ominaisuutta väljä. Muuttuja "pyöreä" saa arvonsa joukosta {0, 1}, missä 0 vastaa ominaisuutta lavea ja 1 ominaisuutta pyöreä. Koodatut vokaalit on esitetty taulukossa 4.5.

Morfofoneemi	Pyöreä	Suppea
/A/	0	0
/E/	0	0.5
/I/	0	1
/O/	1	0.5
/U/	1	1

Taulukko 4.5. Koodatut vokaalit.

4.2. Leksikaalis-semanttisen aliverkon arkkitehtuuri

Learning SLIPNETin leksikaalis-semanttinen aliverkko simuloi siis lemmanhakua. Verkkoon tarvitaan kolme syöte- ja tulosneuronia, sillä sanan semantiikan esittämiseen tarvitaan kolme muuttujaa ja verkko on autoassosiatiivinen (ks. luvut 4.1.1 ja 4.1.2). Tarvittavien piilokerrosten määrää ja kokoa arvioitiin testaamalla muutamia arkkitehtuurivaihtoehtoja. Tavoitteena oli minimoida verkon koko suhteesssa opetuksen onnistumisen todennäköisyyteen.

Opetuksen katsottiin onnistuneen, kun verkon tekemä neliövirhesumma oli E_{SSE}<0.005 alle 3000 opetuskierroksella. Neliövirhesumma haluttiin E_{SSE}<0.005, sillä tällöin opetettut verkot eivät tehneet virheitä simuloidessaan lemmanhakua. Suuremmilla neliövirhesummien arvoilla virheitä esiintyi, eli verkon tuottaman tulosvektorin **o** euklidinen etäisyys d_E oikeaan kohdevektoriin **t** oli suurempi kuin etäisyys johonkin toiseen kohdevektorijoukon vektoriin **t**'. Yhden opetuskierroksen aikana verkolle esitettiin sanajoukko kokonaisuudessaan. Testattuja arkkitehtuurivaihtoehtoja on esitelty taulukossa 4.6.

Taulukon ensimmäisessä sarakkeessa esitetään testatut arkkitehtuurit. Toisessa sarakkeessa on arkkitehtuurityypin opetuksen onnistumistodennäköisyys. Kolmannessa sarakkeessa on onnistuneeseen opetukseen keskimäärin vaadittavat opetuskierrokset. Jos onnistuneita opetuksia oli vähemmän kuin 20 (20 % kaikista opetetuista verkoista), keskilukuna käytetään mediaania. Muutoin keskilukuna käytetään keskiarvoa. Viimeisessä sarakkeessa esitetään onnistuneiden opetuskierrosten keskihajonta.

Arkkitehtuuri	Onnistumis %	Opetuskierrokset	Keskihajonta
3-10-3	7.0	841.00	382.74
3-15-3	20.0	758.90	461.91
3-10-5-3	12.0	1239.50	658.62
3-10-10-3	35.0	1023.11	567.88
3-15-10-3	36.0	696.89	375.86
		N = 100	

Taulukko 4.6. Leksikaalis-semanttisen aliverkon arkkitehtuurivaihtoehtoja.

Taulukosta 4.6 havaitaan, että niillä kerroilla, joilla leksikaalis-semanttisen aliverkon opetus onnistui, tarvittiin keskimäärin vain noin kolmannes suurimmasta sallitusta opetuskierrosten määrästä. Täten on epätodennäköistä, että opetuskierrosten määrän kasvattamisella olisi vaikutusta eri arkkitehtuurien käyttäytymiseen opetuksen aikana, vaikka keskihajonta olikin melko suurta. Vain arkkitehtuurissa 3-10-10-3 suurin tarvittu opetuskierrosten määrä (2983) oli lähellä suurinta sallittua opetuskierrosten määrää. Keskimääräinen maksimiarvo oli 2141. Testin pohjalta päädyttiin arkkitehtuuriin 3-10-10-3.

4.3. Foneemialiverkon arkkitehtuuri

Learning SLIPNETin foneemialiverkko simuloi nimettävän sanan fonologian hakua. Verkko saa syötteekseen leksikaalis-semanttisen aliverkon tuottaman lemman ja tuottaa sitä vastaavat foneemit peräkkäin. Kuten leksikaalis-semanttinen aliverkko, myös foneemialiverkko on autoassosiatiivinen. Jos syötefoneemina on konsonantti, kuvauksen suorittaa konsonanttiverkko, muutoin vokaaliverkko. Konsonantti- ja vokaaliverkon syötteiden ja tulosten

koodaukset on esitetty luvussa 4.1.3. Koodaustapojen erilaisuudesta johtuen vokaali- ja konsonanttiverkkojen arkkitehtuurit ovat erilaisia.

Koska konsonanttien koodaus vaatii viisi bittiä, täytyy konsonanttiverkon syöte- ja tuloskerroksissa olla viisi neuronia. Piilokerroksen neuronien lukumäärä määrättiin kokeellisesti. Testattuja arkkitehtuureita oli kolme kappaletta, joista ensimmäisessä oli neljä piiloneuronia, toisessa viisi ja kolmannessa kuusi. Onnistuneiden opetusten prosenttiosuudet, keskimäärin onnistuneeseen opetukseen tarvitut opetuskierrokset sekä onnistuneiden opetuskierrosten keskihajonnat on esitetty taulukossa 4.7A. Opetus tulkittiin onnistuneeksi, kun verkon tekemä neliövirhesumma oli E_{SSE}<0.02. Suurin sallittu opetuskierrosten määrä oli 3000.

Taulukosta havaitaan, että niillä kerroilla, joilla konsonanttiverkon opetus onnistui, tarvittiin keskimäärin vain murto-osa suurimmasta sallitusta opetuskierrosten määrästä. Koska keskihajontakin oli pientä, opetuskierrosten määrän kasvattamisella ei ole vaikutusta eri arkkitehtuurien käyttäytymiseen opetuksen aikana. Suurin tarvittu opetuskierrosten lukumäärä oli 201 arkkitehtuurilla 5-6-5.

Konsonanttiverkon arkkitehtuurivaihtoehdoista ylivoimaisesti paras oli viiden piiloneuronin verkko. Vaikka neljän piiloneuronin arkkitehtuuri olisi ollut pienin mahdollinen vaihtoehto, valittiin viiden piiloneuronin arkkitehtuuri parhaana konsonanttialiverkon arkkitehtuuriksi.

Arkkitehtuuri	Onnistuminen %	Opetuskierrokset	Keskihajonta
5-4-5	22.0	62.09	25.50
5-5-5	80.0	25.34	18.21
5-6-5	59.0	46.92	37.49
		N = 100	

Taulukko 4.7A. Konsonanttialiverkon arkkitehtuurivaihtoehtoja.

Arkkitehtuuri	Onnistuminen %	Opetuskierrokset	Keskihajonta
2-1-2	0.0	0.00	0.00
2-2-2	33.0	45.33	38.52
2-3-2	68.0	36.24	35.38
		N = 100	

Taulukko 4.7B. Vokaalialiverkon arkkitehtuurivaihtoehtoja.

Vokaaliverkon syöte- ja tuloskerroksissa on kaksi neuronia, sillä vokaalien koodaukseen tarvittiin kaksi muuttujaa. Piilokerroksen neuronien lukumäärä määrättiin kokeellisesti. Testattuja arkkitehtuureita oli kolme kappaletta, joista ensimmäisessä oli yksi piiloneuroni, toisessa kaksi ja kolmannessa kolme. Onnistuneiden opetusten prosenttiosuudet, keskimäärin onnistuneeseen opetukseen tarvitut opetuskierrokset sekä onnistuneiden opetuskierrosten

keskihajonnat on esitetty taulukossa 4.7B. Opetus katsottiin onnistuneeksi, kun verkon tekemä neliövirhesumma oli E_{SSE}<0.02. Suurin sallittu opetuskierrosten määrä oli jälleen 3000.

Taulukosta 4.7B nähdään, että kerroilla, joilla vokaaliverkon opetus onnistui, tarvittiin keskimäärin vain murto-osa suurimmasta sallitusta opetuskierrosten määrästä. Koska keskihajontakin oli pientä suhteessa maksimikierrosten määrään, opetuskierrosten määrän kasvattaminen ei vaikuta eri arkkitehtuurien käyttäytymiseen opetuksen aikana. Tämän vahvistaa myös suurin tarvittu opetuskierrosten lukumäärä, joka oli 198 kierrosta arkkitehtuurilla 2-3-2. Koska piilokerroksen koko haluttiin määrätä mahdollisimman pieneksi, kahden piiloneuronin arkkitehtuuri valittiin vokaaliverkon arkkitehtuuriksi.

4.4. Verkolla laskeminen

Koska kaikki Learning SLIPNETin aliverkot ovat MLP -neuroverkkoja, voidaan, syötekerroksen neuroneja lukuunottamatta, jokaisen neuronin j tulos y_j laskea kaavoilla (2.3) ja (2.7), (kuva 2.3). Aivovaurion simuloimiseksi kaavaa (2.3) muutettiin lisäämällä satunnaiskohinaparametri αe_{ij} jokaiseen painoarvoon w_{ij}, missä e_{ij} on saatu nollakeskiarvoisesta normaalijakaumasta, jonka keskihajontaa (tästä eteenpäin satunnaiskohina, kohina) parametrisoi α. Nyt, kaavan (2.3) perusteella, jokaiseen ei-syötekerroksen neuroniin i saapuva painotettu kohinainen syöte v_i' saadaan kaavalla

$$v_i' = \sum_{i=0}^{n}(w_{ij} + \alpha e_{ij})x_j = \sum_{i=0}^{n} w_{ij}x_j + \alpha\sum_{i=0}^{n} e_{ij}x_j \overset{(2.3)}{=} v_i + \alpha\sum_{i=0}^{n} e_{ij}x_j . \qquad (4.7)$$

Kaavan (4.7) oikeanpuoleisimmasta muodosta nähdään, että painoarvojen muuttaminen voidaan tulkita myös neuronin i tuloksen x_i muuttamiseksi. Nyt kohinaisella verkolla voidaan laskea jokaiselle neuronille i tulos y_i kaavalla (2.7) antamalla sigmoidifunktiolle syötteeksi kaavan (4.7) mukaan laskettu painotettu syöte v_i'.

Satunnaiskohinaa α kasvattamalla verkon tulos muuttuu satunnaisemmaksi. Täten vakavia afaattisia oireita voidaan simuloida käyttämällä suurta satunnaiskohinan arvoa ja lieviä afaattisia oireita käyttämällä pientä satunnaiskohinan arvoa. Satunnaiskohina α on kummallekin aliverkolle erillinen, mutta aliverkon sisällä yhteinen. Siis esimerkiksi leksikaalis-semanttisen aliverkon kohina α_L on simulaation aikana sama verkon jokaiselle painolle. Vastaavasti foneemialiverkon kohina α_F on sama sekä konsonantti- että vokaaliverkon painolle. Täytyy kuitenkin huomata, että aliverkon painoja muutetaan silti (todennäköisesti) eri arvolla, sillä lopullisen painojen muutoksen määrää satunnaiskohinaparametri. Satunnaiskohinaparametrihan laskettaan kertomalla kohina α satunnaisluvulla e_{ij}, joka on (todennäköisesti) erillinen jokaiselle painolle w_{ij}.

Koko aliverkon tulos lasketaan sen tuottamasta tulosvektorista **o** etsimällä aliverkolle opetettujen kohdevektorien joukosta vektoria **o** lähinnä oleva vektori **n**. Vektori **n** tulkitaan aliverkon tulokseksi. Etäisyysmittana käytetään euklidista etäisyyttä d_E (kaava 4.1). Saatuun tulokseen voidaan soveltaa vielä kynnysarvoa τ, joka karsii pois sellaiset vastaukset, jotka ovat liian kaukana mistään verkolle opetusta sanasta. Kynnysarvoa τ käytettäessä verkon tulos **r** lasketaan kaavalla

$$\mathbf{r} = \begin{cases} \mathbf{n}, & \text{jos } \tau \le \dfrac{1}{c \cdot d_E(\mathbf{o},\mathbf{n})}, \quad d_E(\mathbf{o},\mathbf{n}) > 0, \\ \mathbf{n}, & \text{jos } d_E(\mathbf{o},\mathbf{n}) = 0, \\ \text{ei määritelty}, & \text{muuten} \end{cases}, \tag{4.8}$$

missä $c > 0$ on skaalausvakio. Skaalausvakiolla voidaan skaalata etäisyys $d_E(\mathbf{o},\mathbf{n})$ halutun suuruiseksi. Tavoitteena oli skaalata etäisyyttä siten, että simulaatioiden aikana voitaisiin käyttää kynnysarvoja $\tau \in [0,1]$. Simulaatioden aikana oli c = 100.

Käyttämällä kynnysarvoa τ kohteena oleva aliverkko saadaan tuottamaan omissioita, kun omissioksi tulkitaan sellaiset etäisyyden $d_E(\mathbf{o},\mathbf{n})$ arvot, joilla kynnystetyn verkon tulos **r** ei ole määritelty. Simulaatioiden aikana kynnysarvoa käytettiin ainoastaan leksikaalis-semanttisessa aliverkossa (ts. foneemialiverkossa τ_F=0). Jos leksikaalis-semanttinen aliverkko tuottaa omission, tulkitaan se samalla koko Learning SLIPNETin tulokseksi. Muussa tapauksessa leksikaalis-semanttisen aliverkon tulos syötetään eteenpäin foneemiverkolle.

Edellä esitettyjen kaavojen pohjalta muodostettu algoritmi 4.4 laskee Learning SLIPNETin aliverkkojen tulokset.

1. Määritä aliverkossa käytettävä satunnaiskohina α ja kynnysarvo τ.
2. Esitä aliverkolle syöte **p**.
3. Laske jokaiselle aliverkon neuronille tulos kaavoilla (4.7) ja (2.7).
4. Etsi aliverkon tuottamaa tulosta **o** vastaava lähin vektori **n**.
5. Laske aliverkon tulos **r** kaavalla (4.8) käyttämällä kynnysarvoa τ, tulosta **o** sekä vektoria **n**.

Algoritmi 4.2. Learning SLIPNETin aliverkkojen tulosten laskeminen.

Algoritmia 4.2 sovelletaan jokaisen nimettävän sanan kohdalla joko yhden kerran tai m+1 kertaa, missä m on nimettävän sanan pituus. Ensin algoritmia sovelletaan leksikaalis-semanttiseen aliverkkoon sanan lemman hakemiseksi. Jos leksikaalis-semanttisen aliverkon tuottama tulos on riittävän hyvä, eli verkon tulos **r** on määritelty (ks. kaava 4.8), annetaan se foneemialiverkolle.

Foneemialiverkko tuottaa leksikaalis-semanttisen aliverkon tulosta **r** vastaavat foneemit, joita on siis *m* kappaletta, yksi kerrallaan algoritmin 4.2 avulla. Jos leksikaalis-semanttisen aliverkon tulosta **r** ei ole määritelty, ei foneemiverkolle anneta mitään syötettä. Tällöin nimeämisen aikana sovelletaan algoritmia 4.2 vain kerran leksikaalis-semanttiseen aliverkkoon.

5. Learning SLIPNET -neuroverkon evaluointi

Learning SLIPNETin kykyä simuloida afasiapotilaita kartoitettiin kahdessa eri vaiheessa. Ensimmäisessä vaiheessa tutkittiin simulaatioparametrien vaikutusta Learning SLIPNETin tuottamaan virhejakaumaan. Testeissä keskityttiin tutkimaan Learning SLIPNETin aliverkkojen käyttäytymistä erikseen sekä yhdessä suhteessa simulaatioparametrien vaihteluun.

Testien toinen vaihe koostui simulaatiomallin sovittamisesta potilasaineistoon. Potilassovitusten tavoitteena on osoittaa, että Learning SLIPNETillä voidaan ainakin määrällisesti simuloida erityyppisistä afaattisista oireista kärsiviä potilaita. Potilassovitusten yhteydessä käytettävät potilaat olivat samoja potilaita, joita Laine *et al.* [1998] käyttivät SLIPNETin tutkimiseen.

Kaikki tämän luvun testeissä käytetyt Learning SLIPNETin ilmentymät on opetettu back-propagation-algoritmin perusversiosta kehitetyllä BFGS-algoritmilla. Se on Newtonin menetelmään perustuva opetusalgorimi, joka mahdollistaa nopeamman oppimisen kuin perinteinen back-propagation algoritmi [Demuth and Beale, 2000]. Opetusalgoritmin kustannusfunktiona käytettiin neliövirhesummafunktiota (kaava (4.6)). Tavoiteneliövirhesumman arvona leksikaalis-semanttiselle aliverkolle oli E_{SSE}=0.005 ja fonologiselle aliverkolle E_{SSE}=0.02. Tällöin kohinattomat ilmentymät eivät tehneet yhtään virhettä sanajoukon nimeämisen yhteydessä. Opetuskierroksia käytettiin enintään 3000. Yhden opetuskierroksen aikana leksikaalis-semanttiselle aliverkolle esitettiin kerran jokainen sana. Fonologiselle aliverkolle esitettiin yhden opetuskierroksen aikana kerran jokainen sille opetettava foneemi. Opetus ja simulaatiot suoritettiin Matlab 6.0 -ohjelmiston neuroverkkofunktiota sisältävän Neural Network Toolbox 4.0 -kirjaston avulla. Kirjasto sisältää valmiiksi toteutettuja funktioita mm. MLP-neuroverkkojen simuloimiseen [Demuth and Beale, 2000].

5.1. Verkon yleiset ominaisuudet

Kokonaiskäsityksen saamiseksi simulaatioparametrien vaikutuksesta Learning SLIPNET-neuroverkkoon suoritettiin kolme testiä. Ensimmäisessä testissä tutkittiin semanttisen aliverkon käyttäytymistä suhteessa semanttiseen kohinaan α_L ja kynnysarvoon τ. Toisessa testissä tutkittiin fonologisen aliverkon käyttäytymistä fonologisen kohinan α_F muuttuessa. Viimeisessä testissä tutkittiin koko mallin käyttäytymistä suhteessa kohinaparametrien α_L ja α_F muutoksiin.

5.1.1. Leksikaalis-semanttisen aliverkon ominaisuudet

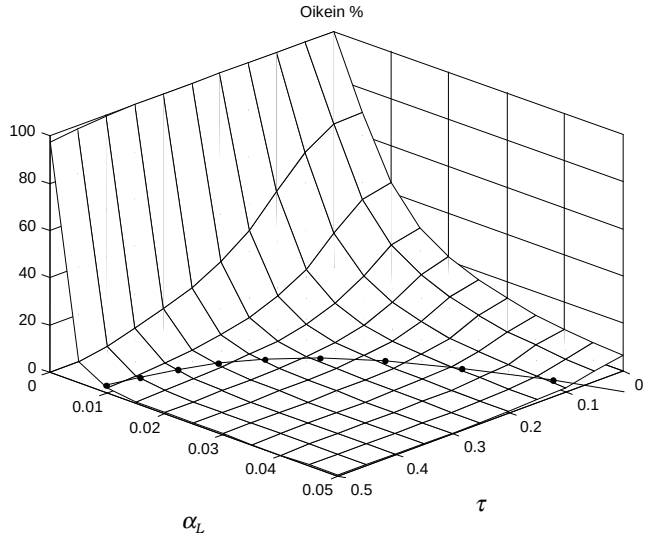
Leksikaalis-semanttisen aliverkon ominaisuuksia tutkittiin säätelemällä kohinaa α_L ja kynnysarvoa τ. Aliverkosta luotiin 10 ilmentymää, joista jokaista tes-

tattiin erillisillä (α_L,τ) -parin yhdistelmillä 10 kertaa. Täten kaikkiaan suoritettiin 100 testikierrosta kullekin (α_L,τ) -parin yhdistelmälle. Koska yhden testikierroksen jokainen sana oli täsmälleen kerran kohdesanana, oli testin aikana jokainen verkolle opetettu sana tietyllä parametrien asetuksella kohdesanana 100 kertaa.

Parametrin α_L arvoja kasvatettiin aina 0.005:llä α_L lähtien arvosta 0 päätyen arvoon 0.1. Parametrin τ arvoja kasvatettiin aina 0.05:llä lähtien arvosta 0 päätyen arvoon 1. Kumpikin parametri sai siis kaikkiaan 21 erillistä arvoa. Täten yhden testikierroksen suorittaminen vaati $21 \cdot 21 = 441$ semanttisen aliverkon täydellistä ajoa (täydellisessä ajossa jokainen sana on täsmälleen kerran kohdesanana). Koska yhden testikierroksen laskenta-aika oli hieman alle 2.5 h, vei yhden ilmentymän testaaminen (10 kierrosta) noin 24 h. Täten koko testin (10 ilmentymää) laskenta-aika oli noin 10 vuorokautta. Laskenta suoritettiin Matlab 6.0 -ohjelmistolla, jonka alustana oli 800 Mhz Windows NT 4.0-kone. Keskusmuistia koneessa oli 128 Mb.

Seuraavaksi esitetään analyysin tulokset graafisessa muodossa kynnysarvon τ ja satunnaiskohinan α_L funktiona. Kunkin vastaustyypin suhteellisen esiintymisen lisäksi esitetään myös vastaustyypin keskihajonta parametrien τ ja α_L funktiona. Keskihajonnan avulla saadaan tietoa analyysin luotettavuudesta tietyillä parametrien arvoilla. Tulosten esittelyssä rajoitutaan selkeyden vuoksi parametrien arvoihin $\alpha_L \in [0,0.05]$ ja $\tau \in [0,0.5]$.

Kuvassa 5.1 esitetään oikeiden vastausten suhteellinen lukumäärä kohinan ja kynnysarvon funktiona. Kun kohinaa ei ole, kynnysarvon kasvattaminen ei juurikaan vaikuta oikeiden vastausten määrään, eli omissiota ei esiinny pelkän kynnysarvon vaikutuksesta. Tämä johtuu kynnysarvon laskentatavasta. Jos kynnysarvon annettaisiin saada suurempia arvoja kuin 1.0 (parametrin maksimiarvo testin aikana), saataisiin kohinattomissakin oloissa oikeiden vastausten osuus nollille. Jos taas τ=0 ja säädellään vain kohinaa, saadaan oikeiden vastausten osuus putoamaan aina hieman alle 10 %:iin. Tällöin lähes kaikki verkon tuottamat vastaukset ovat semanttisia virheitä. Jos kynnysarvoa ja kohinaa kasvatettaan yhtäaikaa oikeiden vastausten osuus putoaa erittäin nopeasti nollille. Tämä johtuu siitä, että kohinan kasvattaminen saa verkon vastaukset poikkeamaan yhä enemmän kohteista, jolloin kasvava kynnysarvo karsii nämä vastaukset omissioiksi.



Kuva 5.1. Oikeat vastaukset.

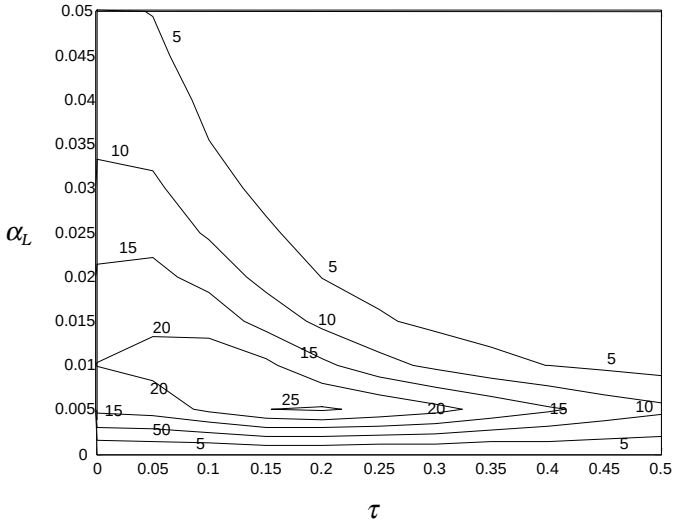
Kuvaan 5.1 on merkitty käyrä, jonka kohdalla oikeiden vastausten osuus on noin 2 %. Parametrien α_L ja τ välinen suhde voidaan tällöin esittää yhtälönä

$$\alpha_L = -0.46263\tau^3 + 0.69445\tau^2 - 0.3685\tau + 0.078186.$$

Yhtälö on laskettu sovittamalla kolmannen asteen polynomi sellaisten parametriparien (α_L,τ) muodostamaan joukkoon, jossa oikeiden vastausten osuus oli 1 % - 3 %. Käyrän sovituksessa käytettiin Matlab 6.0-ohjelmiston käyränsovitusfunktiota.

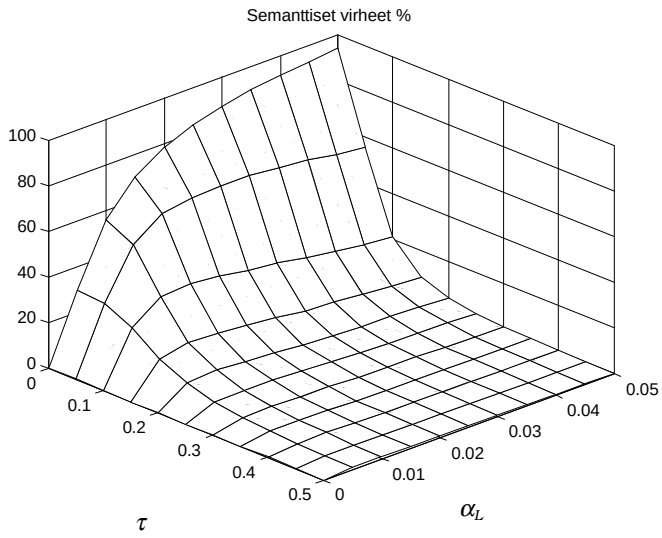
Kuvassa 5.2 esitetään oikeissa vastauksissa esiintynyt keskihajonta korkeuskäyrien avulla. Keskihajonta on ollut suurinta, kun kohina $\alpha_L \in [0.005,0.015]$. Tämä johtuu siitä, että testissä käytetyt leksikaalis-semanttisen aliverkon ilmentymät reagoivat kohinaan eri tavoilla. Toiset ilmentymistä sietävät paremmin kohinaa kuin toiset. Erot johtuvat ilmentymien painoarvojen eroista. Kun kohina $\alpha_L \geq 0.015$ alkavat ilmentymien väliset erot tasoittua painoarvoihin vaikuttavan satunnaisuuden lisääntyessä.

Kun α_L=0, ei kynnysarvon kasvattaminen vaikuta keskihajontaan, sillä verkko tuottaa omissioita vain kohinaisena. Kun kynnysarvoa ja kohinaa säädellään yhdessä, keskihajonta kasvaa ensin huippuunsa ($\alpha_L \in [0.005,0.01]$, $\tau \in [0,0.3]$). Tämä johtuu kohinan α_L erilaisesta vaikutuksesta eri ilmentymiin. Kun kynnysarvoa ja kohinaa edelleen kasvatetaan, keskihajonta vähenee omissioiden lisääntyessä. Suurilla parametrien arvoilla keskihajontaa ei esiinny, koska verkko ei tuota oikeita vastauksia. Keskihajonnan suurin arvo on ≈26 prosenttiyksikköä. Tällöin α_L=0.005 ja τ=0.20.



Kuva 5.2. Oikeiden vastausten keskihajonta (%).

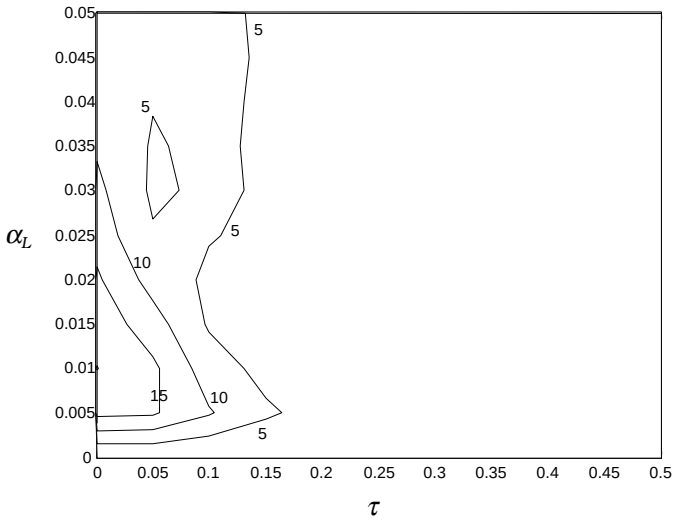
Kuvassa 5.3 esitetään semanttisten virheiden suhteellinen osuus kohinan ja kynnysarvon funktiona. Kun α_L=0, semanttisia virheitä ei voi esiintyä, sillä kynnysarvo säätelee omissioiden osuutta. Kun kynnysarvo τ=0 ja kohinaa lisätään, semanttisten virheiden osuus kasvaa lähelle 98 %. Verkko on yleensä hyvin herkkä kohinalle; jo pienikin kohinan määrä saa verkon tuottamaan lähes 10 % semanttisia virheitä ja kohinan arvolla α_L=0.01 verkko tuottaa niitä hieman yli 55 %. Kun kynnysarvoa kasvatetaan, semanttisten virheiden osuus pienenee omissioiden yleistyessä. Kun kynnysarvo $\tau \geq 0.25$, ei verkossa esiinny juurikaan semanttisia virheitä. Tämä johtuu siitä, että vastaukset, joihin kohina vaikuttaa vääristävästi, joutuvat kynnysarvon karsimiksi, jolloin syntyy omissio.



Kuva 5.3. Semanttiset virheet.

Kuvassa 5.4 esitetään semanttisissa virheissä esiintynyt keskihajonta korkeuskäyrien avulla. Keskihajontaa ei esiinny, kun kohinaa α_L=0, koska tällöin semanttisia virheitä ei voi syntyä. Kun kynnysarvo τ=0, niin keskihajonta lisääntyy kohinaa kasvatettaessa, kunnes se saavuttaa arvon α_L=0.020. Tämän jälkeen keskihajonta pienenee, kohinan α_L vaikutuksen kasvasssa ilmentymien painoarvoissa.

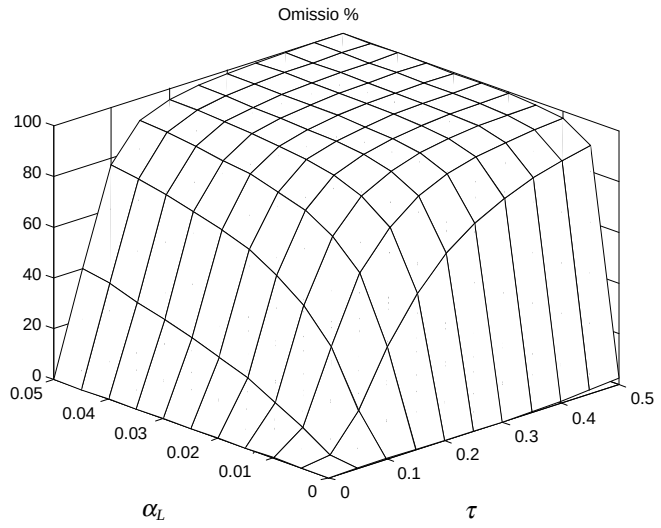
Kun kynnysarvoa τ kasvatetaan, keskihajonta vähenee hyvin nopeasti omissioiden lisääntyessä. Kuitenkin, kun $\alpha_L \leq 0.015$, keskihajonta vähenee hitaammin kuin suuremmilla kohinan α_L arvoilla. Tämä johtuu siitä, että pienet kohinan α_L arvot polarisoivat kynnysarvon τ vaikutusta eri ilmentymiin. Toisin sanoen ilmentymät, jotka sietävät paremmin kohinaa sietävät myös paremmin kynnysarvon kasvattamista. Kun kohina α_L on kiinnitetty, hyvin kohinaa sietävien ilmentymien tuottamat tulosvektorit **o** ovat keskimäärin lähempänä lähintä lemmavektoria **n** kuin huonosti kohinaa sietävien verkkojen tulosvektorit. Tällöin huonosti kohinaa sietävät ilmentymät tuottavat enemmän omissioita kuin hyvin kohinaa sietävät ilmentymät. Ilmiö häviää suurilla kohinan α_L arvoilla, koska kohinan vaikutus ilmentymien painoarvoihin on niin suurta, että niiden painoarvojen erot tasoittuvat satunnaisuuden vaikutuksesta. Kun kynnysarvo $\tau \geq 0.15$ ei keskihajontaa enää esiinny. Suurimmillaan keskihajonta on ≈20 prosenttiyksikköä. Tällöin α_L=0.01 ja τ=0.



Kuva 5.4. Semanttisten virheiden keskihajonta (%).

Kuvassa 5.5 esitetään omissioiden suhteellinen lukumäärä kohinan ja kynnysarvon funktiona. Kun kynnysarvo τ=0, ei omissiota luonnollisestikaan esiinny. Mielenkiintoinen havainto on myös se, että pieni kohinan määrä α_L on välttämätöntä, jotta omissioita ylipäänsä esiintyisi. Kuten jo oikeiden vastausten

analysoinnin yhteydessä todettiin, saataisiin omissioiden osuus nousemaan 100 %:iin ilman kohinaakin, jos kynnysarvoa kasvatettaisiin suuremmaksi kuin 1 tai sen laskentatapaa muutettaisiin (skaalausvakiota *c* muutettaisiin). Kun sekä kynnysarvoa että kohinaa kasvatetaan, nousee omissioden osuus nopeasti lähelle 100 %. Nousu on vain hieman loivempaa pienillä kohinan arvoilla.

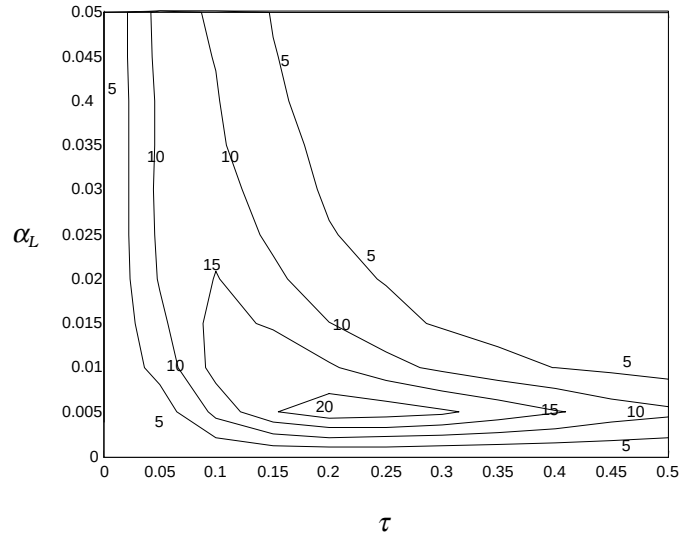


Kuva 5.5. Omissiot.

Kuvassa 5.6 esitetään omissiossa esiintynyt keskihajonta korkeuskäyrien avulla. Koska kynnysarvon τ käyttö on välttämätöntä omissioiden tuottamiseksi, on omissioiden keskihajonta nolla, kun τ=0. Kun kohinaa ei ole, pysyy keskihajonta lähes nollana, vaikka kynnysarvoa kasvatetaan.

Keskihajonta kasvaa voimakkaasti, kun $\tau \geq 0.1$ ja $\alpha_L > 0$. Erityisen voimakasta keskihajonta on, kun $\tau \in [0.1,0.4]$ ja $\alpha_L \in [0.005,0.015]$. Tällöin ilmentymien tulosten vaihtelun syynä on kohina α_L, joka polarisoi kynnysarvon τ vaikutusta ilmentymiin. Polarisaation tuloksena kohinalle herkät ilmentymät tuottavat paljon omissioita, kun taas kohinaa paremmin sietävät ilmentymät eivät tuota näitä virheitä juuri lainkaan.

Kun $\tau \geq 0.2$ ja kohinan α_L annetaan kasvaa, semanttiset virheet alkavat karsiutua pois ja jäljelle jää vain omissiota. Keskihajonta pienenee, kun suuret kohinan α_L arvot tekevät ilmentymien väliset erot merkityksettömiksi. Kun kumpaakin parametria tästä edelleen kasvatetaan, tuottavat ilmentymät vain omissioita. Tällöin keskihajonta luonnollisesti häviää. Suurimmillaan keskihajonta on ≈23 prosenttiyksikköä. Tällöin kynnysarvo on τ=0.2 ja kohina on α_L =0.005.



Kuva 5.6. Omissioiden keskihajonta (%).

Leksikaalis-semanttisen aliverkon dynamiikan analysoinnin perusteella saatiin hyödyllistä tietoa sen keskimääräisestä käyttäytymisestä simulaatioparametrien suhteen. Tulosten pohjalta on mahdollista arvioida potilassovituksissa käytettäviä parametrien asetuksia. Potilassovitusten kannalta tärkeä havainto on merkittävien simulaatioparametrien arvoalueiden kapeus. Käytännössä potilaihin sovitettaessa voidaan käyttää vain parametrien arvoja $\alpha_L \in [0,0.01]$ ja $\tau \in [0,0.25]$. Lisäksi parametrien ollessa välillä $\alpha_L \in [0,0.01]$ ja $\tau \in [0,0.25]$, niiden aikaansaama virhejakauman muutos on suurimmillaan (ks. kuvat 5.1, 5.3 ja 5.5).

Keskihajonnan analysoinnin perusteella voidaan tehdä päätelmiä yksittäisten ilmentymien käyttäytymisestä suhteessa verkon keskimääräiseen käyttäytymiseen. Erityisen tärkeä havainto on keskihajonnan suuruus kohtalaisilla kohinan α_L ja kynnysarvon τ arvoilla. Tämä viittaa siihen, että ilmentymät on mahdollista jakaa hyvin ja huonosti kohinaa kestäviin verkkoihin. Vaikka keskihajonta suurilla simulaatioparametrien arvoilla häviääkin, tulee keskihajonta vaikeuttamaan Learning SLIPNETin sovittamista potilasaineistoon. Keskihajontahan on suurinta juuri väleillä $\alpha_L \in [0,0.01]$ ja $\tau \in [0,0.25]$, jolloin semanttisen aliverkon tuottama virhejakauma on lähellä potilaiden virhejakaumaa. Tästä seuraa se, että oikeiden simulaatioparametrien arvioiminen verkon eri ilmentymille on hankalaa, sillä yhdelle ilmentymälle löydetyt parametrit eivät välttämättä sovi jollekin toiselle ilmentymälle.

Suoritetun leksikaalis-semanttisen aliverkon analyysin perusteella voidaan kuitenkin olettaa, että Learning SLIPNETillä on mahdollista mallintaa afasiapotilaiden semanttisia virheitä ja omissiota.

5.1.2. Fonologisen aliverkon ominaisuudet

Fonologisen aliverkon ominaisuuksia tutkittiin säätelemällä kohinaa α_F. Testin tarkoituksena oli selvittää yksittäisten foneemien käyttäytymistä kohinaisessa foneemialiverkossa. Varsinaisten fonologisten virheiden määrää suhteessa kohinaan α_F on tutkittu luvussa 5.1.3.

Fonologisesta aliverkosta luotiin 10 ilmentymää, joilla jokaista aliverkon foneemia testattiin samalla kohinan arvolla 100 kertaa. Täten jokaista foneemia on testattu jokaisella kohinan α_F arvolla 1000 kertaa. Kohinaparametri sai testin aikana arvonsa joukosta $\alpha_F \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Testin laskenta-aika oli noin 1 h 50 min. Laskenta suoritettiin Matlab 6.0-ohjelmistolla, jonka alustana oli 800 Mhz Windows NT 4.0-kone. Keskusmuistia koneessa oli 128 Mb.

Testin tulokset on esitetty liitteissä 5 (konsonantit) ja 6 (vokaalit). Konsonanttiverkko käyttäytyi kohinaisena melko säännöllisesti. Vain konsonantti */g/* on suurilla kohinan arvoilla selvästi virhealttiimpi kuin muut konsonantit. Vokaalialiverkon käyttäytyminen ei sen sijaan ole yhtä säännöllistä kuin konsonanttialiverkon. Erityisesti morfofoneemi */E/* on virhealttiimpi kuin muut verkon morfofoneemit. Lisäksi morfofoneemi */A/* kestää huomattavasti paremmin kohinaa kuin muut morfofoneemit.

Vaikka yksittäisten foneemien virhetuotokset ovatkin melko epätodennäköisiä, tuottaa foneemialiverkko silti suuremman prosentuaalisen osuuden fonologisia virheitä kuin mitä yksittäisten foneemien perusteella voisi päätellä. Tämä johtuu siitä, että tuottaessaan sanan fonologista muotoa foneemialiverkko tuottaa peräkkäin sanan foneemit. Tällöin sanan virhetodennäköisyydeksi tietyllä kohinan α_F arvolla muodostuu siihen liittyvien foneemien virhetodennäköisyyksien yhdistetty todennäköisyys. Jos nimettävänä sanana olisi sana "kettu" ja foneemialiverkon kohina olisi α_F=0.20, olisi

$$\begin{aligned} P(/\text{kettu}/) &= P(/k/) \cdot P(/e/) \cdot P(/t/) \cdot P(/t/) \cdot P(/u/) \\ &= 1 \cdot 0.972 \cdot 0.997 \cdot 0.997 \cdot 1 \\ &\approx 0.966. \end{aligned}$$

Tällöin sana "kettu" tulee hyvin todennäköisesti nimettyä oikein foneemialiverkossa (todennäköisyydet saatu liitteistä 5 ja 6). Jos kohina kasvatetaan arvoon α_F=0.40 on $P(/\text{kettu}/)$=0.645. Sanalle "hampurilainen" vastaavat todennäköisyydet ovat 0.949 ja 0.593. Yleisestikin pidempien sanojen virhetodennäköisyys on suurempi kuin lyhyiden foneemien peräkkäisen tuotannon vuoksi.

5.1.3. Koko verkon ominaisuudet

Learning SLIPNETin käyttäytymistä simulaatioparametrien suhteen tutkittiin säätelemällä leksikaalis-semanttisen aliverkon kohinaa α_L ja fonologisen

aliverkon kohinaa α_F. Kynnysarvon käyttö τ jätettiin testeistä pois laskenta-ajan säästämiseksi. Kynnysarvon vaikutusta tuloksiin voidaan kuitenkin arvioida jäljempänä esitettävällä tavalla.

Learning SLIPNETistä luotiin 10 ilmentymää, joista jokaista testattiin kaikilla kohinaparin (α_L,α_F) yhdistelmillä 10 kierrosta. Täten kaikkiaan suoritettiin 100 testikierrosta kullekin kohinaparin yhdistelmälle, jolloin jokainen verkon sana oli kohteena 100 kertaa samoilla parametrien asetuksilla.

Leksikaalis-semanttisen aliverkon kohina α_L sai testin aikana arvonsa joukosta $\alpha_L \in \{0.000635, 0.00125, 0.0025, 0.005, 0.01, 0.03, 0.05\}$ ja fonologisen aliverkon kohina α_L joukosta $\alpha_F \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Täten kohina α_L sai kaikkiaan seitsemän erillistä ja kohina α_L kuusi erillistä arvoa. Yhden testikierroksen suorittamisen aikana Learning SLIPNETin täytyi siis nimetä jokainen sana 42 kertaa (kerran jokaisella kohinaparin arvolla). Koska yhden testikierroksen laskenta-aika oli hieman alle 2.5 tuntia, vei yhden ilmentymän testaaminen (10 kierrosta) noin 24 tuntia. Koko testin laskenta-ajaksi muodostui tällöin noin 10 vuorokautta. Laskenta suoritettiin Matlab 6.0-ohjelmistolla, jonka alustana oli 233 Mhz Windows NT 4.0-kone. Keskusmuistia koneessa oli 384 Mb.

Testin tulokset on esitetty liitteessä 7. Koska fonologinen aliverkko suorittaa sanan fonologisen muodon hakemisen vasta leksikaalis-semanttisen aliverkon haettua sanan lemman, riippuu fonologisten virheiden jakauma kohinan α_F lisäksi vain kynnysarvosta τ. Kun kynnysarvoa ei ole testin aikana käytetty, ainoa fonologisiin virheisiin vaikuttava tekijä on kohina α_F. Siksi Learning SLIPNETin fonologisten virheiden jakauma pysyy lähes samanlaisena riippumatta leksikaalis-semanttisessa aliverkossa käytettävän kohinan α_L arvoista.

Fonologisten virheiden laadulliseen jakaumaan kohinan α_L käyttö sen sijaan vaikuttaa. Pienillä kohinan α_L arvoilla pääosa fonologisista virheistä on neologismeja tai muita foneemien vaihtumisesta syntyneitä virheitä. Suurilla kohinan α_L arvoilla suurin osa fonologisista virheitä syntyy semanttisen virheen jälkeen, jolloin hallitseva virhetyyppi on semanttis-fonologinen virhe. Käytännössä potilassovitusten yhteydessä ei kuitenkaan tarvitse käyttää sellaisia parametrien arvoja, että semanttis-fonologiset virheet nousisivat hallitsevaan osaan kaikista fonologisista virheitä.

Kynnysarvon vaikutusta verkon tuloksiin voidaan arvioida liitteen 7 perusteella, koska kynnysarvo muuntaa pääasiassa muita vastaustyypejä omissioiksi. Esimerkiksi kun α_L=0.005 ja α_F=0.20 verkon vastauksista keskimäärin 72.3 % on oikein, 20.2 % on semanttisia virheitä ja 7.5 % fonologisia virheitä. Jos verkossa käytettäisiin kynnysarvoa, jolla verkko tuottaisi noin 20 % omissioita, arviolta reilut kaksi kolmasosaa näistä virheitä syntyisi oikeista vastauksista, viidennes semanttisista virheistä ja loput fonologisista virheistä.

5.2. Sovitus potilasaineistoon

Learning SLIPNETin kykyä simuloida afasiapotilaiden nimeämistä testattiin sovittamalla verkko kymmenen potilaan dataan, jota Laine *et al.* [1998] käyttivät SLIPNETin evaluoimiseen (ks. luku 3.4.1). Potilaat oli luokiteltu kliinisten afasialuokitusten perusteella neljään ryhmään: anomisiin afaatikkoihin (kolme potilasta), Brocan afaatikkoihin (kaksi potilasta), konduktioafaatikkoihin (kaksi potilasta) ja Wernicken afaatikkoihin (kolme potilasta).

Sovituksen yhteydessä yhdistettiin Laineen *et al.* [1998] käyttämistä virhetyypeistä neologistiset ja muut virheet uudeksi virhetyypiksi fonologiset virheet. Tämä tehtiin, koska muut -tyyppisten virheiden tunnistamista ei voitu automatisoida, vaan ne olisi jouduttu tunnistamaan verkon tuottamista nimeämistuloksista simulaation jälkeen. Koska jälkeenpäin suoritetut tarkistamiset vaikuttavat saatuun tulosjakaumaan, vaikeuttaa ja hidastaa tämä verkon sovittamista. Lisäksi tulosten käsintarkastuksesta seuraa, että verkon sovittamista ei voida automatisoida. Automatisointi oli tarpeellista, jotta samalle potilaalle voitiin tehdä useita sovituksia luotettavien tulosten saamiseksi. Yhdistämällä neologistiset ja muut virheet fonologisiksi virheiksi, tuli jokaisesta virhetyyppistä automaattisesti tunnistettava.

Simulaation tuottamasta syötesanasta poikkeava tulos luokiteltiin joko semanttiseksi, fonologiseksi tai omissiovirheeksi seuraavien periaatteiden mukaisesti:

1. Tulos tulkittiin *omissioksi*, jos leksikaalis-semanttinen aliverkko ei tuottanut mitään vastausta annetulle syötteelle.
2. Tulos tulkittiin *semanttiseksi virheeksi*, jos leksikaalis-semanttisen aliverkon tuottama tulossana erosi sen saamasta syötesanasta.
3. Tulos tulkittiin *fonologiseksi virheeksi*, jos foneemialiverkon tuottama tulos erosi leksikaalis-semanttisen aliverkon tuloksesta.

Jos verkko tuotti ensin semanttisen ja sitten fonologisen virheen, luokiteltiin tällainen virhe vain fonologiseksi virheeksi. Laineen *et al.* [1998] luokittelun mukaan yhtäaikainen semanttinen ja fonologinen virhe olisi kuulunut kategoriaan muut virheet. Kategoriaan muut virheet he luokittelivat lisäksi osan pelkästään foneemialiverkossa sattuneista virheistä.

5.2.1. Algoritmi potilassovitusten automatisoimiseksi

Koska verkon sovittaminen potilasaineistoon oli työlästä, kehitettiin algoritmi, joka hakee verkolle oikeita parametreja. Algoritmi ei ole sidottu tiettyyn verkkoarkkitehtuuriin, vaan se on yleiskäyttöinen. Algoritmi pitää käytettävää verkkoa mustana laatikkona, joka tuottaa tietyillä simulaatioparametrien

asetuksella jonkin virhejakauman. Tämän jälkeen se laskee potilaan ja verkon tuottamien virhejakaumien väliset erotukset ja päivittää simulaatioparametreja saatujen erotusten mukaisesti. Algoritmia suoritetaan, kunnes potilaan ja simulaation tuottamat jakaumat ovat riittävän samankaltaisia.

Muodostettavaa sovitusalgoritmia voidaan tarkastella siis ohjattuna oppimistehtävänä. Algoritmin suorituksen aikana se yrittää oppia oikeat parametriasetukset ulkopuolisen opettajan avulla, jolla on tiedossa haluttu tulosjakauma (potilaan jakauma). Oppimissääntönä käytetään virhettä korjaavaa oppimissääntöä (ks. luku 2.2.3). Potilaan ja simulaation tuottamien jakaumien samankaltaisuutta mitataan χ^2 -yhteensopivuustestillä, joka toimii tällöin oppimisalgoritmin kustannusfunktiona. χ^2-testisuure lasketaan kaavalla

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}, \tag{5.1}$$

missä O_i on luokan i havaitut frekvenssit, E_i luokan i odotetut frekvenssit ja k on luokkien lukumäärä.

Olkoon $\alpha_L(n)$ verkon semanttinen kohina, $\tau(n)$ verkon kynnysarvo ja $\alpha_F(n)$ verkon fonologinen kohina algoritmin suorituskierroksella n. Olkoon lisäksi d_s potilaan ja $y_s(n)$ simulaation semanttisten virheiden määrä. Olkoon vastaavasti d_f, $y_f(n)$, d_o ja $y_o(n)$, potilaan ja simulaation fonologisten virheiden ja omissioiden määrä. Tällöin verkon ja potilaan jakaumien välille voidaan määritellä virheet

$$e_s = d_s - y_s(n), \tag{5.2}$$

$$e_f = d_f - y_f(n) \text{ ja} \tag{5.3}$$

$$e_o = d_o - y_o(n). \tag{5.4}$$

Koska $\alpha_L(n)$ vaikuttaa pääasiassa semanttisten, $\alpha_F(n)$ fonologisten ja $\tau(n)$ lähinnä omissiovirheiden syntyyn, voidaan simulaatioparametreja päivittää suhteessa algoritmin tekemiin virheisiin kaavoilla

$$\alpha_L(n+1) = \alpha_L(n) + \eta e_s, \tag{5.5}$$

$$\alpha_F(n+1) = \alpha_F(n) + \eta e_f \text{ ja} \tag{5.6}$$

$$\tau(n+1) = \tau(n) + \eta e_o, \tag{5.7}$$

missä η on oppimisnopeutta säätelevä *oppimiskerroin* (learning rate).

Oppimiskertoimen valinta vaikuttaa oppimisproseduurin suppenemiseen kohti oikeita parametrien arvoja. Pienellä oppimiskertoimella oppiminen on tasaista, koska parametreja muutetaan vain vähän kerrallaan. Siksi proseduuri suppenee tasaisesti, mutta hitaasti kohti ratkaisua. Suuri oppimiskerroin aiheuttaa vastaavasti suuria muutoksia päivitettäviin parametreihin. Liian suuri oppimiskerroin voi johtaa oppimisproseduurin oskilloimiseen oikean ratkaisun

ympärillä tai jopa oppimisproseduurin hajaantumiseen [Haykin, 1994]. Siksi oppimiskertoimen valintaan täytyy kiinnittää huomiota.

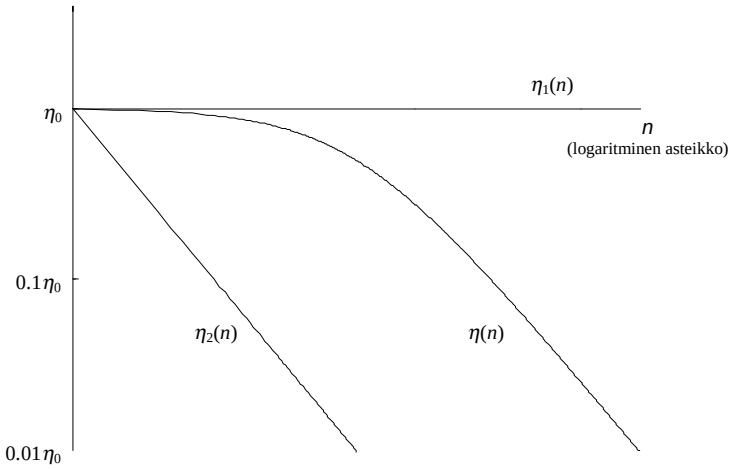
Haykinin [1994] mukaan hyvä ratkaisu oppimiskertoimen valintaan on käyttää muuttuvaa oppimiskerrointa. Oppimisen alkuvaiheessa käytetään suurta kerrointa, jota pienennetään etsinnän edetessä. Tällainen vaihtuva oppimiskerroin voidaan määritellä esimerkiksi funktiolla

$$\eta(n) = \frac{\eta_0}{1 + (n/t)}, \tag{5.8}$$

missä η_0 on oppimiskertoimen alkuarvo ja $t>0$ vakio. Kun n on pieni, $\eta(n) \approx \eta_0$. Suurilla muuttujan n arvoilla

$$\eta(n) = \frac{\eta_0}{1 + (n/t)} \approx \frac{\eta_0}{n/t} = \frac{\eta_0 t}{n} \overset{c=\eta_0 t}{=} \frac{c}{n}. \tag{5.9}$$

Kuvaan 5.7 on merkitty funktioiden $\eta_1(n) = \eta_0$, $\eta_2(n) = c/n$ ja $\eta(n)$ kuvaajat. Kuten kuvastakin havaitaan, pienillä muuttujan n arvoilla funktio $\eta(n)$ approksimoi funktiota $\eta_1(n)$ ja suurilla arvoilla funktiota $\eta_2(n)$. Täten funktio $\eta(n)$ jakaa parametrien päivityksen *etsintä-* ja *suppenemisvaiheeseen*. Kaavassa (5.8) esiintyvä vakio t parametrisoikin juuri etsintävaiheeseen käytettyä aikaa. Etsintävaiheessa suuret funktion $\eta(n)$ arvot mahdollistavat suurien muutosten tekemisen simulaatioparametreihin. Tällöin algoritmin on mahdollista löytää lähes oikeat parametrien arvot suhteellisen nopeasti. Etsintävaiheessa algoritmi voi jäädä kuitenkin oskilloimaan ratkaisun ympärille. Suppenemisvaiheessa funktion $\eta(n)$ arvot kuitenkin pienenevät, jolloin algoritmi suppenee kohti ratkaisua.



Kuva 5.7. Oppimiskertoimen laskentatapoja [Haykin, 1994].

Sovitusalgoritmin (algoritmi 5.1) toteutus edellä esitettyjen ideoiden pohjalta on suoraviivainen. Pääsilmukassa evaluoidaan ensin sovitettava verkko

simulaatioparametreilla $\alpha_L(n)$, $\tau(n)$ ja $\alpha_F(n)$. Tämän jälkeen lasketaan virheet e_s, e_o ja e_f sekä χ^2-testisuure kaavoilla (5.1) - (5.4). χ^2-testisuuretta verrataan tavoitetestisuureeseen χ^2_{target}. Jos on testisuure $\chi^2 > \chi^2_{target}$, päivitetään simulaatioparametreja suhteessa virheisiin e_s, e_f ja e_o kaavoilla (5.5) - (5.7) Jos on $\chi^2 \leq \chi^2_{target}$, voidaan algoritmin suoritus lopettaa, sillä riittävän hyvät simulaatioparametrit on löydetty.

Input: Oppimiskerroin η_0,
etsintävaiheeseen käytettävää aikaa parametrisoiva t,
simulaatioparametrit $\alpha_L(0)$, $\tau(0)$ ja $\alpha_F(0)$,
tavoitetestisuureen arvo χ^2_{target}.

Output: Verkon tuottama virhejakauma.

1. **begin**
2. $n \leftarrow 0$, $\chi^2 \leftarrow \infty$.
3. **while** $\chi^2 > \chi^2_{target}$ **do**
4. Evaluoi verkko parametreilla $\alpha_L(n)$, $\tau(n)$ ja $\alpha_F(n)$.
5. Olkoon χ^2 potilaan ja verkon jakaumien välisen testisuureen arvo.
6. **if** $\chi^2 > \chi^2_{target}$ **then**
7. $\eta \leftarrow \eta(n)$ (kaava 5.8).
8. $\alpha_L(n+1) \leftarrow \alpha_L(n) + \eta \cdot e_s$.
9. $\tau(n+1) \leftarrow \tau(n) + \eta \cdot e_o$.
10. $\alpha_F(n+1) \leftarrow \alpha_F(n) + \eta \cdot e_f$.
11. **fi**
12. $n \leftarrow n + 1$.
13. **od**
14. **end**

Algoritmi 5.1. Potilasdatan sovittaminen.

Sovitusalgoritmin käyttäminen edellyttää kohinan alaisena melko säännöllisesti käyttäytyvää verkkoa, jotta algoritmi voi päivittää simulaatioparametreja oikeaan suuntaan. Lisäksi on huomattava, että pienennettäessä tavoitetestisuuretta χ^2_{target}, algoritmin laskenta-aika pidentyy, koska riittävän hyviä parametrien asetuksia on vaikeampi löytää. Tämä johtuu siitä, että pienillä tavoitetestisuureen arvoilla verkon tuottaman jakauman epäsäännöllisyydet vaikeuttavat parametrien päivitystä.

Ilmiötä voidaan havainnollistaa seuraavasti. Oletetaan että on olemassa sellainen ideaali simulaatiomalli, joka tuottaa tietyillä parametrien p_1, p_2,…, p_n asetuksilla aina samanlaisen jakauman. Olkoon lisäksi

$$e_i(p_i) = d_i - y_i(p_i) \tag{5.10}$$

potilaan ja mallin jakauman komponentin i välinen virhe parametrilla p_i. Nyt tavoitetestisuuretta χ^2_{target} minimoidaan minimoimalla virheitä $e_i(p_i)$. Ideaalisen mallin tapauksessa minimointi suoritetaan siis päivittämällä simulaatioparametreja edellä kuvatulla tavalla.

Oletetaan sitten, että tutkittava malli poikkeaa ideaalisesta mallista siten, että se antaa samoilla simulaatioparametrien $p_1, p_2,\ldots, p_n$ asetuksilla hieman erilaisia tuloksia. Oletetaan että nämä tulokset saadaan ideaalimallin tuloksista kaavalla

$$y_i'(p_i) = y_i(p_i) + z_i, \tag{5.11}$$

missä z_i on normaalijakaumaa noudattava (diskreetti) satunnaismuuttuja. Muuttuja z_i siis kuvaa tarkasteltavan mallin satunnaista poikkeamaa ideaalimallista. Tällöin tarkasteltavan mallin tekemä virhe parametrien asetuksilla p_1, $p_2,\ldots, p_n$ voidaan laskea kaavalla

$$e_i'(p_i) = d_i - y_i'(p_i) \overset{(5.11)}{=} [d_i - y_i(p_i)] - z_i. \tag{5.12}$$

Kun erotus $d_i - y_i(p_i) \approx 0$, eli kun ideaalimallin ja potilaan jakaumat ovat hyvin lähellä toisiaan, kaavan (5.12) perusteella on tarkasteltavalle mallille

$$e_i'(p_i) = [d_i - y_i(p_i)] - z_i \approx -z_i. \tag{5.13}$$

Kaavasta (5.13) seuraa, että pienillä erotuksen $d_i - y_i(p_i)$ arvoilla virheeseen $e_i'(p_i)$ vaikuttaa pääasiassa satunnaismuuttuja z_i, koska $e_i'(p_i) \approx -z_i$. Tällöin sovitusalgoritmi päivittää parametria p_i lähinnä satunnaisesti.

5.2.2. Yleistä potilasdatasovituksista

Potilasdatasovituksia varten Learning SLIPNETistä luotiin 11 ilmentymää (ilmentymien opettaminen tapahtui luvun alussa esitellyllä tavalla). Jokainen ilmentymä sovitettiin jokaisen potilaan dataan kerran. Yhden sovituskierroksen aikana verkko nimesi jokaisen sille opetetun sanan viisi kertaa, joten yhden sovituskierroksen sanojen kokonaismääräksi tuli N=279·5=1395 sanaa. Suuren sanajoukon tarkoituksena oli tasoittaa kohinaisen verkon suoritusta ja helpottaa näin potilasdatasovituksia. Potilasta edustavaksi simulaatiotulokseksi valittiin sovitusaineiston ja potilasaineiston perusteella laskettujen χ^2-testisuureiden mediaani. Potilasdatasovitusten yhteydessä raportoitava verkon jakauma vastaa juuri tätä χ^2-testisuureen arvoa. Lisäksi tulosten yhteydessä annetaan sekä paras että huonoin χ^2-testisuuren arvo sekä testisuureiden p-arvot.

Kaikki potilassovitukset suoritettiin Matlab 6.0-ohjelmistolla, jonka alustana oli 800 Mhz Windows NT 4.0-kone. Keskusmuistia koneessa oli 128 Mb. Sovitusalgoritmin tarvitsema tavoitetestisuure oli χ^2_{target}=3.0. Tällä varmistettiin, että algoritmi hylkää potilaan tuottamasta jakaumasta liian paljon poikkeavat

simulaatiojakaumat. Itseasiassa tavoitetestisuureen valinnasta seuraa välittömästi se, että potilaiden ja hyväksyttyjen potilassovitusten välille ei tule tilastollisesti merkitsevää eroa. Pienemmän tavoitetestisuureen valinta olisi kasvattanut sovitusten laskenta-aikaa huomattavasti, koska verkon tuottaman jakauman epäsäännöllisyydet olisivat vaikeuttaneet parametrien päivitystä yhä enemmän. Toisaalta suuremman tavoitetestisuureen valinta olisi ollut ristiriidassa verkon sovituksen optimoinnin kanssa. Sovituksen alussa sovitusalgoritmin oppimiskerroin oli $\eta_0 = 0.5$ ja etsintävaiheeseen käytettävän ajan parametri oli $t = 5$.

5.2.3. Brocan afaatikot

Brocan afasiasta kärsivillä potilailla nimeämisvirheitä hallitsevat pääasiassa omissiot [Laine ja Marttila, 1992]. Sovituksen yhteydessä mallinnetut Brocan afaatikot tekivät lisäksi jonkin verran fonologisia virheitä. Potilasaineistossa oli käytettävissä kaksi Brocan afaatikkoa B1 ja B2.

Potilaan B1 ja verkon tuottamat jakaumat on esitetty taulukossa 5.1A. Taulukon 5.1B χ^2-yhteensopivuustestit osoittavat, ettei potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa. Sovitusalgoritmin tarvitsemat simulaatioparametrien alkuarvot olivat: semanttisen aliverkon kohina α_L=0.005, kynnysarvo τ=0.20 ja fonologisen aliverkon kohina α_F=0.20. Yhden ilmentymän sovittaminen vei keskimäärin 54 min.

	B1				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	77.1	77.5	1075.5	1081.0	-5.5
Semanttinen	3.0	2.5	41.9	35.0	6.9
Fonologinen	0.6	0.6	8.4	8.0	0.4
Omissio	19.3	19.4	269.2	271.0	-1.8
N = 1395			1395.0	1395.0	

Taulukko 5.1A. Potilaan B1 ja simulaation nimeämisjakaumat.

	B1				
	χ^2	p	α_L	τ	α_F
Paras	0.875	0.834	0.00584	0.2271	0.1686
Mediaani	1.177	0.758	0.00066	0.2059	0.1282
Huonoin	2.921	0.402	0.00736	0.2170	0.0949
Vapausaste = 3					

Taulukko 5.1B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle B1.

Potilaan B2 ja verkon tuottama jakauma on esitetty taulukossa 5.2A. Kuten jakaumista näkee, myös potilaan B2 datan sovitus on onnistunut hyvin. Tätä tukevat myös χ^2-yhteensopivuustestit, joiden tulokset on esitetty taulukossa

5.2B. Sovitusalgoritmin alkuarvot olivat: α_L=0.005, τ=0.20 ja α_F=0.10. Yhden ilmentymän sovittaminen vei keskimäärin 1 h 40 min.

	B2				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	87.3	87.7	1217.8	1224.0	-6.2
Semanttinen	3.0	2.7	41.9	38.0	3.9
Fonologinen	1.2	0.9	16.7	12.0	4.7
Omissio	8.4	8.7	117.2	121.0	-3.8
N = 1395			1393.6	1395.0	

Taulukko 5.2A. Potilaan B2 ja simulaation nimeämisjakaumat.

	B2				
	χ^2	p	α_L	τ	α_F
Paras	0.378	0.946	0.00055	0.1753	0.0424
Mediaani	1.849	0.603	0.00269	0.1947	0.1361
Huonoin	2.481	0.478	0.00404	0.2002	0.1452
Vapausaste = 3					

Taulukko 5.2B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle B2.

5.2.4. Anomiset afaatikot

Anomisesta afasiasta kärsiville potilaille on tyypillistä huomattava anomia [Laine ja Marttila, 1992]. Täten nimeämisvirheitä hallitsivat pääasiassa omissiot. Lisäksi mallinnetut anomiset afaatikot tekivät jonkin verran fonologisia virheitä. Potilasaineistossa oli käytettävissä kolme anomista afaatikkoa A1, A2 ja A3.

Potilaan A1 ja verkon tuottamat jakaumat on esitetty taulukossa 5.3A. Taulukon 5.3B χ^2-yhteensopivuustestit osoittavat, ettei potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa. Sovitusalgoritmin alkuarvot olivat: α_L=0.008, τ=0.25, α_F=0. Yhden ilmentymän sovittaminen vei keskimäärin 51 min. χ^2-yhteensopivuustestien yhteydessä ei huomioitu fonologisia virheitä, sillä potilas ei tuottanut niitä.

	A1				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	51.7	53.3	721.2	743.0	-21.8
Semanttinen	4.2	4.4	58.6	61.0	-2.4
Fonologinen	0.0	0.0	0.0	0.0	0.0
Omissio	44.1	42.4	615.2	591.0	24.2
N = 1395			1395.0	1395.0	

Taulukko 5.3A. Potilaan A1 ja simulaation nimeämisjakaumat.

	A1				
	χ^2	p	α_L	τ	α_F
Paras	0.351	0.841	0.00447	0.2440	0.0000
Mediaani	1.709	0.423	0.00991	0.2405	0.0000
Huonoin	2.915	0.231	0.00373	0.2334	0.0000
	Vapausaste = 2				

Taulukko 5.3B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle A1.

Potilaan A2 ja verkon tuottamat jakauma on esitetty taulukossa 5.4A. Taulukon 5.4B. Myöskään tämän potilaan ja simulaation jakaumien välillä ei havaittu tilastollisesti merkitsevää eroa (ks. taulukko 5.4B). Sovitusalgoritmin alkuarvot olivat: α_L=0.010, τ=0.15, α_F=0.20. Yhden ilmentymän sovittaminen vei keskimäärin 58 min.

	A2				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	77.1	76.5	1075.5	1067.0	8.5
Semanttinen	8.4	8.5	117.2	119.0	-1.8
Fonologinen	0.6	0.9	8.4	12.0	-3.6
Omissio	13.9	14.1	193.9	197.0	-3.1
	N = 1395		1395.0	1395.0	

Taulukko 5.4A. Potilaan A2 ja simulaation nimeämisjakaumat.

	A2				
	χ^2	p	α_L	τ	α_F
Paras	0.329	0.956	0.00512	0.1506	0.1852
Mediaani	1.720	0.633	0.00374	0.1521	0.1255
Huonoin	2.800	0.422	0.00085	0.1525	0.0382
	Vapausaste = 3				

Taulukko 5.4B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle A2.

Potilaan A3 ja verkon tuottama jakauma on esitetty taulukossa 5.5A. Taulukon 5.5B χ^2-yhteensopivuustestit osoittavat, ettei tämänkään potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa. Sovitusalgoritmin alkuarvot olivat: α_L=0.010, τ=0.50, α_F=0.20. Yhden ilmentymän sovittaminen vei keskimäärin 1 h 13 min.

	A3				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	39.2	39.2	546.8	547.0	-0.2
Semanttinen	0.6	0.4	8.4	5.0	3.4
Fonologinen	0.6	0.6	8.4	9.0	-0.6
Omissio	59.6	59.8	831.4	834.0	-2.6
	N = 1395		1395.0	1395.0	

Taulukko 5.5A. Potilaan A3 ja simulaation nimeämisjakaumat.

	A3				
	χ^2	p	α_L	τ	α_F
Paras	0.480	0.921	0.0046	0.4613	0.1666
Mediaani	1.412	0.708	0.0043	0.4878	0.1681
Huonoin	2.887	0.406	0.0034	0.4225	0.2048
Vapausaste = 3					

Taulukko 5.5B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle A3.

5.2.5. Konduktioafaatikot

Konduktioafasiasta kärsivillä potilailla nimeämisvirheitä hallitsevat neologismit. Mallinnetut konduktioafaatikot tekivät lisäksi jonkin verran fonologisia virheitä. Potilasaineistossa oli käytettävissä kaksi konduktioafaatikkoa C1 ja C2.

Potilaan C1 ja verkon tuottama jakauma on esitetty taulukossa 5.6A. Potilaan ja simulaation jakaumien välillä ei havaittu tilastollisesti merkitsevää eroa (taulukko 5.6B). Sovitusalgoritmin alkuarvot olivat: α_L=0.005, τ=0.20 ja α_F=0.30. Yhden ilmentymän sovittaminen vei keskimäärin 47 min.

	C1				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	65.7	66.4	916.5	926.0	-9.5
Semanttinen	3.6	3.2	50.2	45.0	5.2
Fonologinen	23.5	22.8	327.8	318.0	9.8
Omissio	7.2	7.6	100.4	106.0	-5.6
N = 1395			1395.0	1395.0	

Taulukko 5.6A. Potilaan C1 ja simulaation nimeämisjakaumat.

	C1				
	χ^2	p	α_L	τ	α_F
Paras	0.634	0.889	0.00487	0.1698	0.3381
Mediaani	1.243	0.743	0.00296	0.1683	0.3191
Huonoin	2.848	0.416	0.00394	0.1581	0.4181
Vapausaste = 3					

Taulukko 5.6B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle C1.

Potilaan C2 ja verkon tuottama jakauma on esitetty taulukossa 5.7A. Taulukon 5.7B χ^2-yhteensopivuustestit osoittavat, ettei potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa. Sovitusalgoritmin alkuarvot olivat: α_L=0.005, τ=0.25 ja α_F=0.30. Yhden ilmentymän sovittaminen vei keskimäärin 2 h 45 min.

	C2				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	84.3	85.4	1176.0	1192.0	-16.0
Semanttinen	0.6	0.6	8.4	9.0	-0.6
Fonologinen	10.2	9.9	142.3	138.0	4.3
Omissio	4.8	4.0	67.0	56.0	11.0
N = 1395			1393.6	1395.0	

Taulukko 5.7A. Potilaan C2 ja simulaation nimeämisjakaumat.

	C2				
	χ^2	p	α_L	τ	α_F
Paras	0.600	0.897	0.00198	0.2504	0.3095
Mediaani	1.871	0.599	0.00141	0.2478	0.2844
Huonoin	2.957	0.396	0.00303	0.2572	0.3024
Vapausaste = 3					

Taulukko 5.7B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle C2.

5.2.6. Wernicken afaatikot

Wernicken afasiasta kärsivien potilaiden nimeämisvirheet eroavat toisistaan, eikä heitä voi erottaa muista potilasryhmistä pelkästään jakaumien perusteella. Potilasaineistossa oli käytettävissä kolme Wernicken afaatikkoa W1, W2 ja W3. Potilaan W1 jakaumaa hallitsivat omissiot sekä fonologiset virheet (ks. taulukko 5.8A). Potilaalla W2 virhejakauma oli tasaisempi ja jokainen virhetyyppi on selkeästi esillä (ks. taulukko 5.9A). Potilaalla W3 oli vuorostaan huomattava anomia, joka ilmenee omissioiden yleisyytenä. Lisäksi potilas tekee runsaasti neologistisia virheitä, mikä ilmenee fonologisten virheiden suurena osuutena. Yhdessä fonologiset virheet muodostavat neljänneksen koko vastausjakaumasta. Potilaan W3 virhejaukauma on esillä taulukossa 5.10A.

Potilaan W1 ja verkon tuottama jakauma on esitetty taulukossa 5.8A. Jälleen suoritetut χ^2-yhteensopivuustestit osoittavat, ettei potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa (taulukon 5.8B). Sovitusalgoritmin alkuarvot olivat: α_L=0.010, τ=0.20 ja α_F=0.30. Yhden ilmentymän sovittaminen vei keskimäärin 50 min.

	W1				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	54.8	53.8	764.5	750.0	14.5
Semanttinen	4.8	5.1	67.0	71.0	-4.0
Fonologinen	19.9	20.6	277.6	287.0	-9.4
Omissio	20.5	20.6	286.0	287.0	-1.0
N = 1395			1395.0	1395.0	

Taulukko 5.8A. Potilaan W1 ja simulaation nimeämisjakaumat.

	W1				
	χ^2	p	α_L	τ	α_F
Paras	0.231	0.972	0.00304	0.1874	0.4023
Mediaani	0.839	0.840	0.00088	0.1728	0.1679
Huonoin	2.617	0.455	0.00883	0.1829	0.3683
	Vapausaste = 3				

Taulukko 5.8B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle W1.

Potilaan W2 ja verkon tuottama jakauma on esitetty taulukossa 5.9A. Potilaan ja simulaation jakaumien välillä ei havaittu tilastollisesti merkitsevää eroa (taulukko 5.9B). Sovitusalgoritmin alkuarvot olivat: α_L=0.001, τ=0.15 ja α_F=0.30. Yhden ilmentymän sovittaminen vei keskimäärin 50 min.

	W2				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	42.9	44.2	598.5	616.0	-17.5
Semanttinen	13.9	13.9	193.9	194.0	-0.1
Fonologinen	22.2	20.9	309.7	292.0	17.7
Omissio	21.1	21.0	294.3	293.0	1.3
	N = 1395		1396.4	1395.0	

Taulukko 5.9A. Potilaan W2 ja simulaation nimeämisjakaumat.

	W2				
	χ^2	p	α_L	τ	α_F
Paras	0.475	0.925	0.00152	0.1216	0.3154
Mediaani	1.617	0.656	0.00536	0.1230	0.4279
Huonoin	2.989	0.393	0.01281	0.1280	0.4264
	Vapausaste = 3				

Taulukko 5.9B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle W2.

Potilaan W3 ja verkon tuottama jakauma on esitetty taulukossa 5.10A. Tehdyt χ^2-yhteensopivuustestit osoittavat, ettei potilaan ja simulaation jakaumien välillä ole tilastollisesti merkitsevää eroa (taulukko 5.10B). Sovitusalgoritmin alkuarvot olivat: α_L=0.005, τ=0.30 ja α_F=0.40. Yhden ilmentymän sovittaminen vei keskimäärin 54 min.

	W3				
	%		Frekvenssit		
Vastaus	Potilas	Simulaatio	Potilas	Simulaatio	Erotus
Oikein	36.7	36.7	512.0	512.0	0.0
Semanttinen	0.6	0.6	8.4	9.0	-0.6
Fonologinen	25.3	23.7	352.9	330.0	22.9
Omissio	37.3	39.0	520.3	544.0	-23.7
	N = 1395		1393.6	1395.0	

Taulukko 5.10A. Potilaan W3 ja simulaation nimeämisjakaumat.

	W3				
	χ^2	p	α_L	τ	α_F
Paras	0.629	0.891	0.00525	0.3653	0.5086
Mediaani	2.610	0.456	0.00067	0.3100	0.3896
Huonoin	2.822	0.418	0.00507	0.4039	0.3424
	Vapausaste = 3				

Taulukko 5.10B. χ^2-testisuureiden ja niitä vastaavien parametrien arvoja potilaalle W3.

5.3. Yhteenveto simulaatioista

Suoritetut potilastestit osoittavat, että Learning SLIPNETillä voidaan määrällisesti simuloida erityyppisistä afaattisista oireista kärsiviä potilaita. Tulosten luotettavuutta lisää useiden sovitusten suorittaminen samalle potilaalle. On kuitenkin huomattava, että verkkoon lisättävän satunnaiskohinan seurauksena samoilla parametrien asetuksilla voidaan saada hieman erilaisia tulosjakaumia eri simulaatiokerroilla. Tyypillisesti vaihtelu simulaatiotulosten välillä oli muutaman prosenttiyksikön luokkaa. Vaihtelulla ei ole simulaatiomallin luotettavuuden kannalta merkitystä, sillä myös afasiapotilaiden nimeämistulokset vaihtelevat eri testikerroilla.

Kuten jo luvussa 5.1.1 ennakoitiin Learning SLIPNETin ilmentymät voidaan jakaa hyvin ja huonosti kohinaa sietäviin verkkoihin. Jakaminen voidaan tehdä kummankin aliverkon osalta. Potilasdatasovitusten perusteella verkon kohinaherkkyys ei kuitenkaan vaikuta sen kykyyn simuloida potilaita.

Potilaiden sijoittumista simulaatioparametrien suhteen tutkittiin sovittamalla potilaiden datat 11 kertaa samaan ilmentymään. Näistä sovituksista valittiin potilaita edustamaan χ^2-testisuureiden mediaania vastaava sovitus. Sovitusalgoritmin alkuarvot olivat samat kuin edellisissä potilasdatasovituksissa. Sovituksessa käytettiin ilmentymää, joka oli potilasdatasovitusten perusteella havaittu keskimääräisesti kohinaa sietäväksi.

Valittujen potilasdatasovitusten simulaatioparametrien yli laskettiin niiden mediaanit sekä vaihteluvälit. Semanttisen kohinan mediaani oli α'_L=0.00376 ja vaihteluväli oli $\alpha_L \in [0.00201,0.00769]$, kynnysarvon mediaani on τ'=0.2003 ja vaihteluväli oli $\tau' \in [0.1248,0.4175]$ ja fonologisen kohinan mediaani on α'_F=0.2572 ja vaihteluväli oli $\alpha'_F \in [0,0.5117]$. Koska kynnysarvon ja fonologisen kohinan vaihteluvälit ovat melko suuria, erottuvat potilaat näiden parametrien suhteen melko hyvin. Semanttisen kohinan vaihteluväli on sen sijaan pieni, joten hyvinkin erilaiset potilaat saavat lähes samoja kohinan arvoja.

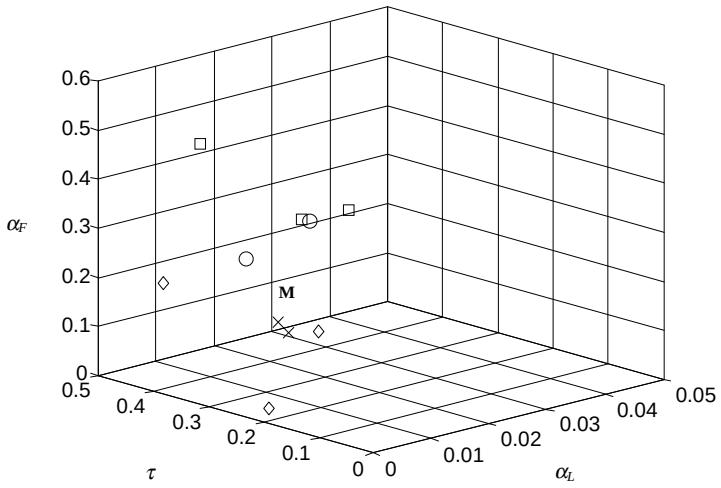
Taulukkossa 5.11 on yhteenveto edellisen testin potilaiden simulaatioparametreista. Sarakkeeseen d_E on laskettu potilastapauksien euklidinen etäisyys mediaanipotilaasta. Mediaanipotilaan parametrit ovat α'_L, τ' ja α'_F. Poti-

laista lähimpänä mediaanipotilasta ovat potilaat B1, C2, B2 sekä A2 (tässä järjestyksessä).

Potilas	α_L	τ	α_F	d_E
A1	0.00487	0.2405	0.0000	0.2604
A2	0.00507	0.1518	0.1862	0.0860
A3	0.00351	0.4175	0.2044	0.2236
B1	0.00373	0.2112	0.1891	0.0690
B2	0.00334	0.1893	0.1760	0.0820
C1	0.00380	0.1562	0.4129	0.1618
C2	0.00201	0.2518	0.3101	0.0738
W1	0.00468	0.1791	0.4066	0.1509
W2	0.00769	0.1248	0.4331	0.1914
W3	0.00299	0.3458	0.5117	0.2932

Taulukko 5.11. Yhteenveto potilaiden simulaatioparametreista.

Potilaiden sijoittumista simulaatioparametreihin nähden on havainnollistettu kuvassa 5.8. Anomisia afaatikkoja merkitään timanteilla [◇], Brocan afaatikkoja risteillä [X], konduktioafaatikkoja ympyröillä [O] ja Wernicken afaatikkoja neliöillä [□]. Mediaanipotilas sijoittuu kuvaan 5.8 merkityn **M**-kirjaimen kohdalle.



Kuva 5.8. Potilaiden sijoittuminen simulaatioparametreihin nähden.

Leksikaalis-semanttisen aliverkon herkkyys kohinan α_L ja kynnysarvon τ vaihteluille tulee konkreettisesti esille tarkasteltaessa potilaita C1 ja W1. Vaikka nämä potilaat (kuvassa 5.8 keskellä) ovat lähes päällekäin, on potilas W1 on tehnyt lähes kolme kertaa enemmän omissiota kuin potilas C1. Kuitenkin on vain

$$\alpha_L(\text{W1}) - \alpha_L(\text{C1}) = 0.00088 \text{ ja}$$

$$\tau(\text{W1}) - \tau(\text{C1}) = 0.0228,$$

joten pienikin kynnysarvon korotus yhdessä semanttisen kohinan lisäämisen kanssa saa aikaan suuria muutoksia verkon nimeämisjakaumassa. Tämä johtuu siitä, että kohinaa α_L kasvatettaessa kynnysarvon ollessa vakio verkon tuottamat tulosvektorit **o** ovat keskimäärin kauempana lähimmästä lemmavektorista **n**, jolloin useampi vastausvektori joutuu kynnysarvon karsimaksi. Jos myös kynnysarvoa kasvatetaan, omissiot edelleen lisääntyvät. Tämän vuoksi pienet muutokset semanttiseen kohinaan α_L ja kynnysarvoon τ saavat aikaan suuria muutoksia virhejakaumaan. Tarkemmin ilmiötä on havainnollistettu luvussa 5.1.1.

Afasialuokituksittain tarkasteltaessa vain Brocan afaatikot muodostavat selkeän ryhmän myös simulaatioparametrien suhteen. Muut potilasryhmät ovat enemmän tai vähemmän hajallaan tai päällekäin toisten potilasryhmien kanssa. Hajanaisuus selittyy toisaalta sillä, että kliinisissä luokituksessa potilaat luokitellaan muiden seikkojen kuin nimeämisvastausten jakauman perusteella [Laine ja Marttila, 1992]. Siksi huomattavan erilaisetkin nimeämisjakaumat tuottaneet potilaat voivat kliinisesti katsoen kuulua samaan ryhmään. Koska verkko sovitetaan potilaan jakaumaan, riippuvat simulaatioparametrit vain jakaumasta, jolloin samankaltaisen jakauman tuottaneet potilaat saavat myös samankaltaiset simulaatioparametrit. Toisaalta on myös huomattava, että yleistysten tekeminen vain kymmenen potilaan aineiston pohjalta ei ole kovin luotettavaa.

Vaikka Learning SLIPNETin avulla onnistuttiinkin hyvin simuloimaan potilaiden virhejakaumia tilastollisesti, ei tämä vielä takaa mallin hyvyyttä laadullisessa mielessä. Ensinnäkin neologisten ja fonologisten virheiden yhdistäminen yhdeksi ryhmäksi aiheuttaa välttämättä laadullista epätarkkuutta simulaatiotuloksiin. Tämän lisäksi Learning SLIPNET kärsii muistakin DTS-malleihin liittyvistä heikkouksista (ks. luku 3.4.3). Tämän tutkielman tarkoituksena oli kuitenkin selvittää Learning SLIPNETin yleisiä ominaisuuksia sekä kykyä simuloida tilastollisesti afasiapotilaiden nimeämistä. Tässä mielessä malli on onnistunut. Täytyy myös huomata, että laadullisten ominaisuuksien tarkempaan analysointiin tarvittaisiin vankkaa neuropsykologian ja kielitieteen tieteenalojen asiantuntemusta.

6. Yhteenveto

Tutkielmassa tarkasteltiin afaattisten nimeämishäiriöiden mallintamista neurolaskentamenetelmillä. Esitellyistä malleista kahden tarkastelu perustui kirjallisuuteen ja kolmas toteutettiin tutkielman teon yhteydessä. Toteutetulla Learning SLIPNET-neuroverkkomallilla haluttiin osoittaa, että puheentuoton häiriöitä voidaan mallintaa yleisesti käytettyjen MLP-neuroverkkojen avulla. Learning SLIPNETin kykyä simuloida afasiapotiaita testattiin sovittamalla verkko kymmenen potilaan datoihin, jotka käsittivät erityyppisisiä afasiaoireyhtymiä. Potilasdatasovitukset onnistuivat tilastollisesti hyvin. Tulosten laadullinen analyysi jätettiin kuitenkin tekemättä, sillä sen suorittaminen vaatisi vankkaa neuropsykologian ja kielitieteen tieteenalojen asiantuntemusta. Laadullinen analyysi saattaisi paljastaa Learning SLIPNETin ja potilaiden jakaumien olevan laadullisesti hyvinkin erilaisia, vaikka ne määrällisesti ovatkin hyvin läheisiä.

Learning SLIPNETiä tarkasteltaessa täytyy kuitenkin muistaa, että DTS-mallina sitä vaivaavat samat rajoitukset, kuin muitakin DTS-malleja. Eräs tärkeä tällainen rajoitus on luvussa 3.4.1 mainittut fonologisen haun yhteydessä sattuvat semanttiset virheet. DTS-mallina Learning SLIPNET ei kykene tuottamaan näitä virheitä, sillä niiden tuottaminen vaatisi lemmatason osallistumista fonologiseen hakuun. Tässä mielessä IA-mallin toteutus olisi ollut parempi vaihtoehto. DTS-mallin toteutukseen päädyttiin kuitenkin sen yksinkertaisuuden vuoksi.

Learning SLIPNETin kehittämisen yhteydessä jouduttiin pohtimaan semantiikan ja fonologian koodaamista neuroverkoille. Fonologian koodaamiseksi voitiin käyttää osaksi jo tunnettuja menetelmiä. Semantiikan koodaamiseksi tyydyttävää menetelmää ei kuitenkaan ollut saatavilla, joten tutkielmassa kehitettiin menetelmä semantiikan algoritmiseksi koodaamiseksi neuroverkoille (algoritmi 4.1). Menetelmä perustuu sanojen hierarkkiseen luokitukseen semanttisen puun avulla. Sen avulla on mahdollista muodostaa pienidimensioisia syötevektoreita, jotka soveltuvat suuridimensioisia paremmin MLP-neuroverkoille. Lisäksi tutkielmassa kehitettiin menetelmä, jonka avulla simulaatiomallien sovittaminen potilaisdataan voidaan automatisoida (algoritmi 5.1). Tämä vapauttaa tutkijan simulaatioparametrien hakemiselta tärkeämpien työtehtävien pariin.

Simulaatiomallin toteutuksen yhteydessä täytyi tehdä lukuisia valintoja, joiden vaikutusta lopputulokseen oli mahdotonta tai vaikeaa arvioida. Esimerkkejä tällaisista valinnoista ovat mm. semanttisen koodauksen suoritta-

van algoritmin parametrisointi ja sen avulla saatavan semanttisen koodauksen valinta. Ideaalitilanteessa algoritmin parametrisoinnin vaikutusta malliin olisi testattu toteuttamalla useita simulaatiota erilaisilla semantiikan koodauksilla. Tämä oli kuitenkin mahdotonta, mallin testaamiseen kuluvan ajan vuoksi. Tehtyjen valintojen lisäksi suurimpia tuloksissa mahdollisesti esiintyvien virheiden lähteitä ovat tutkimuksen lähes jokaisessa vaiheessa käytetyt tietokoneohjelmat. Näistä ohjelmista useat ovat itse kirjoitettuja ja ne voivat aina sisältää käyttäjälleen näkymättömiä virheitä.

Tutkimuksen aikana syntyi lukuisia mielenkiintoisia jatkotutkimusaiheita. Tällaisia ovat mm. semanttisen koodauksen suorittavan algoritmin parantaminen, suuremman sanamäärän opettaminen mallille, suuremman potilasjoukon sovittaminen malliin, mallintamisen laajentaminen uusiin kieliin, potilaiden toipumisen ja uudelleenopettamisen mallintaminen sekä kehittyneemmän neuroverkkomallin kehittäminen. Semanttisen koodauksen suorittavan algoritmin parantamisessa keskeistä olisi sanojen semanttisten suhteiden esityksen parantaminen. Tähän liittyy oleellisesti semanttisen puun kehittäminen semanttisen verkon suuntaan. Tällöin olisi mahdollista esittää mm. assosiaatiosuhteita sanojen välillä. Nykyinen algoritmin totetus rajoittaa semanttisten suhteiden tarkastelun vain ylä - alakäsitetasolle.

Suuremman sanamäärän opettaminen neuroverkolle mahdollistaisi mm. sanojen frekvenssiominaisuuksien vaikutusten tutkimisen. Kokeellisesti on huomattu, että potilaat kykenevät nimeämään paremmin yleisesti esiintyviä sanoja kuin harvoin esiintyviä sanoja. Lisäksi usein esiintyvät sanat nimetään nopeammin kuin harvemmin esiintyvät. Toinen suuremman sanamäärän mahdollistama tutkimuskohde olisi Learning SLIPNETin yleistyskyvyn tutkiminen. Tällöin mallille opettaisiin vain osa sanoista opetusjoukkona ja testattaisiin sen kykyä yleistää opittua koodausta opetusjoukon ulkopuolelle. Koska Learning SLIPNETin aliverkoille käytännössä opetetaan identiteettikuvaus, yleistämisen pitäisi ainakin periaatteessa olla tämän hetkisellä Learning SLIPNETillä mahdollista.

Suuremman potilasjoukon sovittaminen malliin saattaisi mahdollistaa potilaiden luokittelun simulaatioparametrien suhteen. Nyt sovitettujen kymmenen potilaan datojen perusteella tällaista luokittelua ei ole mahdollista tehdä, sillä vain Brocan afaatikot muodostivat sovitetuista potilaista selkeän ryhmän simulaatioparametrien suhteen (ks. luku 5.3). Kahden Brocan afaatikon perusteella ei kuitenkaan voida tehdä minkäänlaisia päätelmiä Brocan afaatikkojen luokittelusta yleisesti.

Nimeämishäiriöiden mallintamisen laajentaminen uusiin kieliin (erityisesti englannin kieleen) mahdollistaisi suuren potilasmäärän sovittamisen malliin.

Käytännössä kielen vaihtamisen yhteydessä vain foneemialiverkko täytyisi uudelleentoteuttaa, sillä kuvan nimeämistä mallinnettaessa leksikaalis-semanttisen aliverkon voidaan ajatella olevan samankaltainen kielestä riippumatta. Englannin lisäksi mielenkiintoinen simulaatiokieli olisi toinen kotimainen kielemme ruotsi.

Potilaiden toipumisen ja erityisesti uudelleenopettamisen mallintamisen avulla voitaisiin tutkia kuntoutuksen tehokkuutta. Tähän liittyviä tutkimuskohteita ovat mm. kuntoutustavan valinta yksittäisille potilaille, sekä kuntoutuksessa käytettävän sanajoukon ominaisuudet. Kuntoutustavan valintaan liityviä avoimia kysymyksiä ovat mm., kannattaako (esimerkiksi) semanttisia virheitä tekevää potilasta kuntouttaa semanttisilla harjoitteilla vai tulisiko potilaan puheentuotonjärjestelmää harjoittaa kokonaisvaltaisesti. Tällaista mallintamista on tehty dysleksiasta (aivovaurion seurauksena syntynyt lukihäiriö) kärsivien potilaiden simulaatiota varten toteutetulla neuroverkkomallilla [Plaut, 1996]. Tällainen mallintaminen puheen tuoton *kognitiivisilla malleilla* on kuitenkin hieman kyseenalaista ja siksi saataviin tuloksiin täytyy suhtautua varauksella. Kognitiivisen prosessin taustalla olevat fyysiologiset ominaisuudet ovat ihmisillä paljon monimutkaisempia kuin kognitiivista prosessointia simuloivalla matemaattisella mallilla.

Kehittyneempien verkkoarkkitehtuurien käyttäminen mahdollistaisi Learning SLIPNETin rakenteellisten heikkouksien korjaamisen. Eräs mahdollisuus olisikin toteuttaa oppiva IA-malli, sillä sen rakenne ei rajoita mallia niin paljon kuin nyt toteutetun DTS-mallin rakenne. Toinen syy kehittyneempien verkkoarkkitehtuurien käyttämiselle on neuropsykologiassa usein havaitun *virittymisen* (priming) mallintaminen. Tällä tarkoitetaan ilmiötä, jossa samalla (kielen) prosessointitasolla toisiinsa liittyvät käsitteet vaikuttavat toisiinsa joko prosessointia nopeuttaen tai hidastaen [Harley, 2001]. Leksikalisaation liittyvät viritystutkimukset ovat keskittyneet *semanttiseen virittämiseen* ja *fonologiseen virittämiseen* (semantic / phonological priming). Esimerkiksi on helpompi tunnistaa sana (esim. "leipä"), jos on juuri aikaisemmin kuullut tai nähnyt sen merkitykseen liittyvän sanan (esim. "voi"). Käytännössä virittämisen mallintaminen vaatisi joko kahden sanan yhtäaikaista aktivointia verkossa tai erikoistyyppisten painoarvojen käyttämistä neuroverkossa.

Mielenkiintoista olisi myös kehittää sellainen malli, joka mahdollistaisi nimeämisen käsitetason prosessoinnin mallintamisen. Tällaisessa mallissa käsitetaso koostuisi useasta aliverkosta, jotka mallintaisivat karkealla tasolla esimerkiksi visuaalista ja taktiilista (tuntoaistia) ja auditiivista prosessointia. Näiden aliverkkojen yhteinen aktivaatio muodostaisi nimettävän sanan käsitteen ja toimisi syöteenä leksikaalis-semanttiselle aliverkolle. Verkot voitaisiin

toteuttaa esimerksi som-verkkoilla (self-organizing map). Mallin kehittämistä lähiaikoina voidaan kuitenkin pitää epätodennäköisenä.

Suoritettu tutkimus saavutti päätavoitteensa. Se osoittaa, että afaattisia nimeämishäiriöitä on mahdollista mallintaa yksinkertaisella oppivalla neuroverkoarkkitehtuurilla. Tulevaisuudessa kehitetään tarkempia ja monipuolisempia nimeämistä simuloivia neuroverkkomalleja, jotka sovittuvat yhä paremmin empiirisiin havaintoihin.

Viitteluettelo

[Bechtel and Abrahamsen, 1991] William Bechtel and Adele Abrahamsen, *Connectionism and Mind. An Introduction to Parallel Processing in Networks*. Basil Blackwell, 1991.

[Caramazza, 1997] Alfonso Caramazza, How many levels of processing are there in lexical access? *Cognitive Neuropsychology* **14** (1997), 177-208.

[Caramazza and Miozzo, 1997] Alfonso Caramazza and Michele Miozzo, The relation between syntactic and phonological knowledge in lexical access: evidense from the 'tip-of-tongue' phenomenon. *Cognition* **64** (1997), 309-343.

[Dell, 1986] Gary S. Dell, A spreading-activation theory of retrieval in sentence production. *Psychol. Rev.* **93** (1986), 283-321.

[Dell *et al.*, 1996] Gary S. Dell, Myrna F. Schwartz, Nadine Martin, Eleanor M. Saffran and Deborah A. Gagnon, A connectionist model of naming errors in aphasia. In: James A. Reggia, Eytan Ruppin and Rita Sloan Berndt (eds.), *Neural Modelling of Brain and Cognitive Disorders*. World Scientific Publishing, 1996, 135-156.

[Dell *et al.*, 1997] Gary S. Dell, Myrna F. Schwartz, Nadine Martin, Eleanor M. Saffran and Deborah A. Gagnon, Lexical access in aphasic and nonaphasic speakers. *Psychol. Rev.* **104** (1997), 801-838.

[Demuth and Beale, 2000] Howard Demuth and Mark Beale, *Neural Network Toolbox for Use with Matlab*. MathWorks, Inc., 2000.

[Foygel and Dell, 2000] Dan Foygel and Gary S. Dell, Models of impaired access in speech production. *Journal of Memory and Language* **43** (2000),182-216.

[Fu, 1994] LiMin Fu, *Neural Networks in Computer Inteligence*. McGraw-Hill, 1994.

[Hand *et al.*, 2001] David Hand, Heikki Mannilla and Padhraic Smyth, *Principles of Data Mining*. MIT Press, 2001.

[Harley, 2001] Trevor Harley, *The Psychology of Language. From Data to Theory*. Psychology Press, 2001.

[Haykin, 1994] Simon Haykin, *Neural Netwoks. A Comprehensive Foundation*. Prentice Hall, 1994.

[Hecht-Nielsen, 1990] Robert Hecht-Nielsen, *Neurocomputing*. Addison-Wesley, 1990.

[Hinton *et al.*, 1986] Geofrey E. Hinton, James L. McClelland and David E. Rumelhart, Distributed representations. In: James L. McClelland and David E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, 77-109. MIT Press, Cambridge, MA, 1986.

[Hinton and Sejnowski, 1986] Geofrey E. Hinton and Terrence T. Sejnowski, Learning in Bolztmann machines. In: James L. McClelland and David E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, 282-317. MIT Press, Cambridge, MA, 1986.

[Hinton and Shallice, 1991] Geofrey E. Hinton and Tim Shallice, Lesioning an attractor network: Investigations of acquired dyslexia. *Psychol. Rev.* **98** (1991), 74-95.

[Ilmoniemi, 2001] Risto Ilmoniemi, Aivojen rakenne. *http://www.biomag.helsinki.fi/braincourse/luentomoniste2001.html* (viitattu 9.4.2003), 2001.

[ISO, 1986] ISO 2788-1986 (E), *Documentation - Guidelines for the Establishment and Development of Monolingual Thesauri*. International Organization for Standardization, 1986.

[Juhola and Laine, 1992] Martti Juhola and Matti Laine, Preliminary simulation of aphasic naming errors with a network model. In Eero Hyvönen, Jouko Seppänen and Markku Syrjänen (eds.), *New Directions of Artificial Inteligence* **2**, 224-230. *Proc. of the Finnish Artificial Inteligence Conference*.

[Karlsson, 1983] Fred Karlsson, *Suomen Kielen Äänne- ja Muotorakenne*. WSOY, Juva, 1983.

[Laine, 2002] Matti Laine, Erään potilaan (potilas "YK") nimeämistestissä käytetyt sanat. *Henkilökohtainen tiedonanto*, 2002.

[Laine, 2003] Matti Laine, Virhetuotosten luokittelusta kielellisissä tehtävissä. *Henkilökohtainen tiedonanto*, 2003.

[Laine *et al.*, 1998] Matti Laine, Anneli Tikkala and Martti Juhola, Modeling anomia by the discrete two-stage word production model. *J. Neurolinguistics* **11** (1998), 275-294.

[Laine ja Marttila, 1992] Matti Laine ja Reijo Marttila, Aikuisen afasia. *Duodecim* **108** (1992), 1039-1047.

[Levelt *et al.*, 1991] Willem J. Levelt, Herbert Schriefers, Dirk Vorberg, Antje S. Mayer, Thomas Pechman and Jaap Havinga, The time course of lexical access in speech production: a study of picture naming. *Psychol. Rev.* **98** (1991), 112-142.

[Martin *et al.*, 2002] Nadine Martin, Matti Laine and Trevor Harley, How can cognitive models of language inform models of language rehabilitation? In: Hillis A. (ed.), *Handbook on Adult Language Disorders*, Psychology Press, 2002.

[Miikkulainen, 1997] Risto Miikkulainen, Dyslexic and category-specific aphasic impairments in a self-organizing feature map model of the lexicon. *Brain and Language* **59** (1997), 334-366.

[Niiniluoto, 1980] Ilkka Niiniluoto, *Johdatus Tieteenfilosofiaan*. *Käsitteen - ja teorianmuodostus*. Otava, Keuruu, 1980.

[Plaut, 1996] David C. Plaut, Relearning after damage in connectionist networks: toward a theory of rehabilitation. *Brain and Language* **52** (1996), 25-82.

[Popper, 1995] Karl R. Popper, Tiede: Arvauksia ja Kumoamisia. Teoksessa: Karl R. Popper, *Arvauksia ja Kumoamisia. Tieteellisen tiedon kasvu*, 33-65. Gaudeamus, Helsinki, 1995.

[Rojas, 1996] Raúl Rojas, *Neural Networks. A Systematic Introduction*. Springer-Verlag, 1996.

[Ruml and Caramazza, 2000] Wheeler Ruml and Alfonso Caramazza, An evaluation of a computational model of lexical access: comments on Dell *et al.* (1997), *Psychol. Rev.* **107** (2000), 609-634.

[Rumelhart and McClelland, 1986] David E. Rumelhart and James L. McClelland, On learning the past tenses of English verbs. In: James L. McClelland and David E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*, 216-271. MIT Press, Cambridge, MA, 1986.

[Sejnowski and Rosenberg, 1987] Terrence J. Sejnowski and Charles R. Rosenberg, Parallel networks that learn to pronounce English text. *Complex Systems* **1** (1987), 145-168.

[Snodgrass and Vanderwart, 1980] Joan Gay Snodgrass and Mary Vanderwart, A standardized set of 260 pictures: norms for name agreement, image agreement, familiarity and visual complexity. *Journal of Experimental Psychology: Human Learning and Memory* **6** (1980), 174-215.

[Swingler, 1996] Kevin Swingler, 1996, *Applying Neural Networks. A Practical Guide*. Academic Press, San Diego, CA, 1996.

[Thyme, 1993] Ann E. Thyme, *A Connectionist Approach to Nominal Inflection: Paradigm Patterning and Analogy in Finnish*. PhD Dissertation. University of California, San Diego, 1993.

[Tikkala, 1997] Anneli Tikkala, *Neural Networks and Lexical Processing: Three Models for Finnish.* PhD Dissertation. Kuopio University Printing Office, Kuopio, 1997.

[Tikkala and Juhola, 1995] Anneli Tikkala and Martti Juhola, A neural network simulation method of aphasic naming errors: properties and behavior. *Neural Computing & Applications* **3** (1995), 191-201.

[Yule, 1999] George Yule, *The Study of Language*. Cambridge University Press, 1999.

Liite 1. Virhetuotosten luokittelusta [Laine, 2003]

Yleisiä afasiapotilaiden virhevastausten luokittelumahdollisuuksia: ensimmäinen sanatason vastaus, viimeinen sanatason vastaus, koko vastaus (jolloin voi olla useampia virheitä).

Peruskysymys: onko potilaan tuotos suomen kielen sana?

1. Virhetuotos on oikea suomen kielen sana
   - 1.1 Vastaus liittyy semanttisesti kohdesanaan: *semanttinen parafasia* (semantic paraphasia)
     - 1.1.1 Semanttis-visuaalinen (esim. viulu → kitara)
     - 1.1.2 Kategoriajäsen (esim. omena → banaani)
     - 1.1.3 Kategorianimi (esim. maissi → vihannes)
     - 1.1.4 Piirrekuvaus eli ns. *kiertelevä puhe* (circumlocution) (esim. huuliharppu → siihen puhalletaan)
     - 1.1.5 Assosiatiivinen (esim. hiiri → juusto)
   - 1.2 Vastaus liittyy fonologisesti kohdesanaan: *formaali parafasia* (formal paraphasia). Kriteerit vaihtelevat eri tutkimuksissa (esim. kirahvi → karahvi)
   - 1.3 Vastaus ei kytkeydy semanttisesti eikä fonologisesti kohdesanaan: *liittymätön sana* (unrelated word response) (esim. takki → koppi)
   - 1.4 Vastaus kytkeytyy sekä semanttisesti, että fonologisesti kohdesanaan: ns. sekoittunut virhe (*mixed error*) (esim. korva → karva)
   - 1.5 Vastaus on kohdesanan morfologinen variantti: morfologinen virhe
   - 1.6 Vastaus liittyy visuaalisesti kohteeseen
   - 1.7 Muut vastaukset
2. Virhetuotos ei ole suomenkielen sana
   - 2.1 Virhetuotos kattaa fonologiseti yli puolet kohdesanasta: *fonologinen parafasia*
     - 2.1.1 Äänteellinen lisäys (esim. kattila → krattila)
     - 2.1.2 Äänteellinen poisto (esim. ranta → rana)
     - 2.1.3 Äänteellinen muutos (esim. lusikka → tusikka)
     - 2.1.4 Paikan vaihtuminen (esim. tapuli → patuli)
   - 2.2 Virheellinen tuotos kattaa alle puolet kohdesanasta: *neologismi* (esim. helikopteri → rapuli)
   - 2.3 Monisanainen, neologismeja ja mahdollisesti muitakin sanoja vilisevä vastaus: jatkettu neologistinen jargon
   - 2.4 Olemassa olevien morfeemien väärä kombinaatio: *morfofonologinen parafasia* (esim. pyykkinaru → pyykkinaku)
3. Omissio

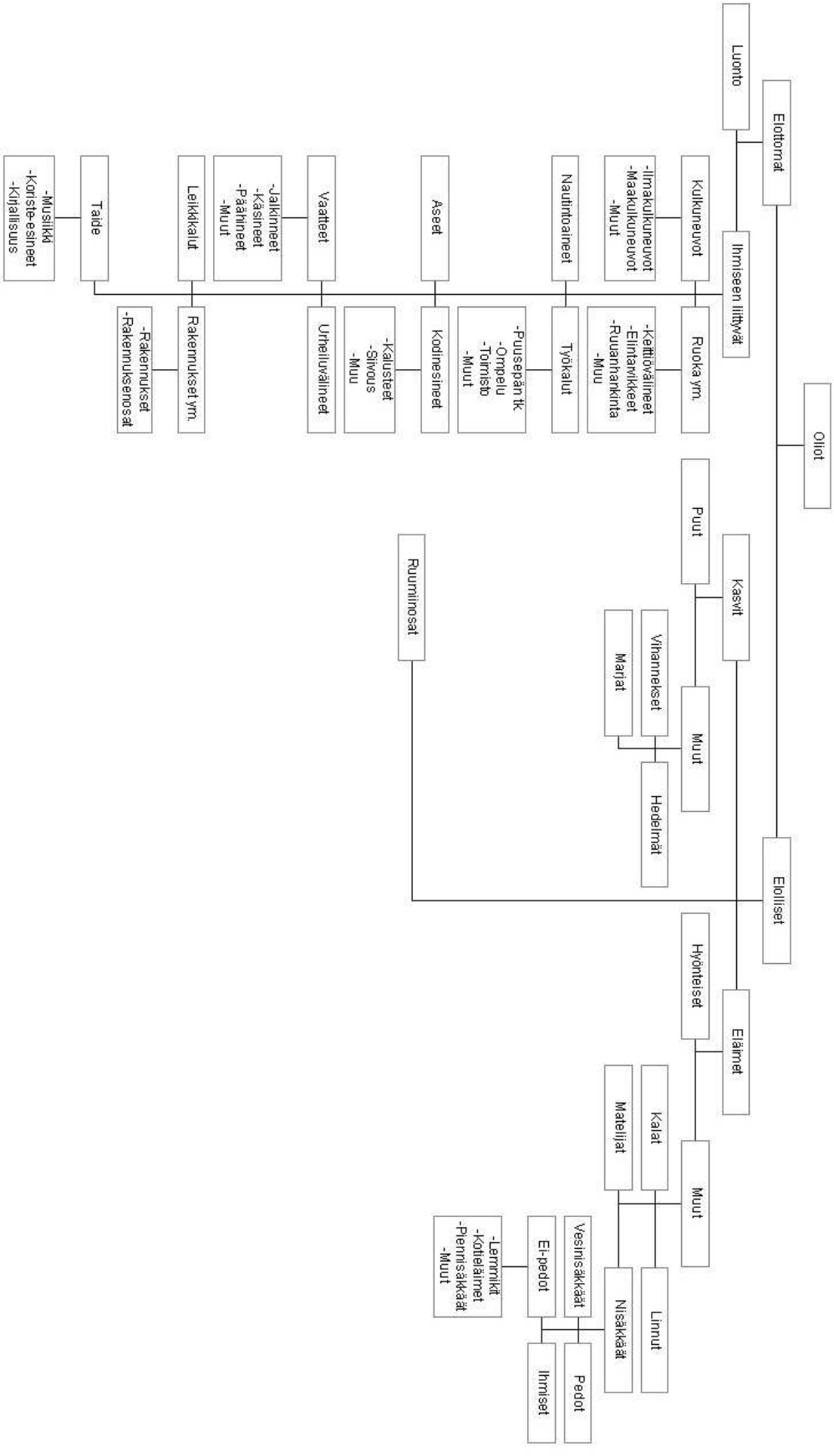
   Potilas on hiljaa tai sanoo "en tiedä" tms.

Liite 2. Simulaatiossa käytetyt sanat

1 aasi
2 ahven
3 alligaattori
4 ambulanssi
5 ananas
6 ankerias
7 ankka
8 appelsiini
9 artisokka
10 aurinko
11 auto
12 banaani
13 biisoni
14 bodaaja
15 elefantti
16 eskimo
17 etana
18 fasaani
19 gorilla
20 haarukka
21 haavi
22 haikara
23 haitari
24 hämähäkki
25 hame
26 hampurilainen
27 hamsteri
28 hanska
29 harppu
30 hattu
31 hauki
32 hävittäjä
33 heinäsirkka
34 helikopteri
35 helistin
36 hella
37 helmet
38 henkari
39 henkselit
40 hevonen
41 hiihtäjä
42 hiiri
43 hiukset
44 housut
45 huilu
46 hylje
47 hyrrä
48 hyttynen
49 ikkuna
50 ilmapallo
51 intiaani
52 jääkaappi
53 jakoavain
54 jalka
55 jalkaterä
56 jänis
57 jojo
58 joutsen
59 juna
60 kaappi
61 käärme
62 käärö
63 kaivo
64 kakku
65 kalastaja
66 kalossi
67 kameli
68 kampela
69 kana
70 kannu
71 karhu
72 kärpänen
73 kasetti
74 käsi
75 kattila
76 katto
77 kaulin
78 keidas
79 keinu
80 keinutuoli
81 kela
82 kelkka
83 kello
84 kenguru
85 kenkä
86 kettu
87 kilpikonna
88 kirahvi
89 kirja
90 kirkko
91 kirsikka
92 kirves
93 kissa
94 kitara
95 kivääri
96 koira
97 koppakuoriainen
98 kori
99 kotka
100 kottarainen
101 kravatti
102 kruunu
103 kukko
104 kulho
105 kurki
106 kurpitsa
107 kuu
108 kynä
109 lahna
110 lakki
111 lammas
112 lamppu
113 lankarulla
114 lapanen
115 lasi
116 lehmä
117 leija
118 leijona
119 leimasin
120 leipä
121 leivänpaahdin
122 leka
123 lentokone
124 leopardi
125 letku
126 levysoitin
127 liivi
128 linja-auto
129 lipasto
130 lohi
131 lokki
132 luistelija
133 luistimet
134 lumiukko
135 luola
136 lusikka
137 luuta
138 made
139 mansikka
140 mehiläinen
141 meloni
142 metso
143 miekka
144 moottoripyörä
145 mortteli
146 mutteri
147 muurahainen
148 nalle
149 nappi
150 nauhuri

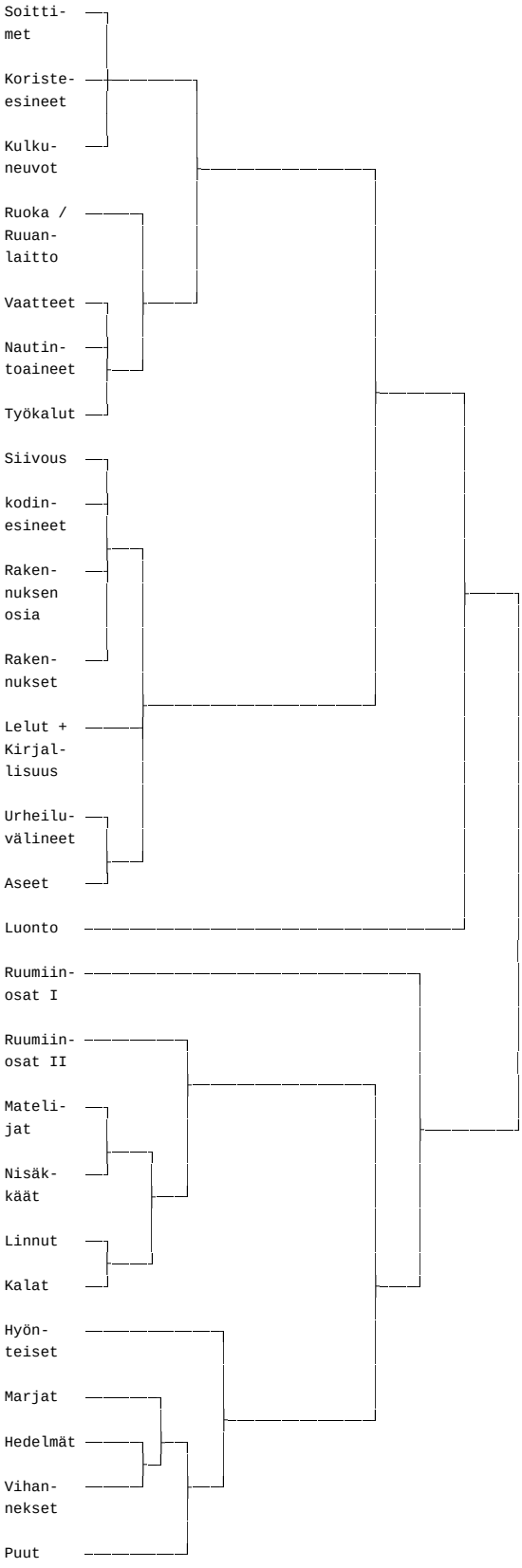
151 naula
152 navetta
153 nenä
154 neula
155 nitoja
156 noppa
157 omena
158 orava
159 ovenkahva
160 ovi
161 päärynä
162 pääskynen
163 pähkinä
164 paimen
165 painija
166 paistinpannu
167 paita
168 pakettiauto
169 palli
170 pallo
171 paprika
172 parsa
173 partiolainen
174 patsas
175 patteri
176 peili
177 penseli
178 perhonen
179 persikka
180 peruna
181 pesäpallomaila
182 peukalo
183 peura
184 piano
185 pihdit
186 pihlaja
187 piippu
188 piirakka
189 piisami
190 pilli
191 pingviini
192 pipari
193 pipo
194 pistooli
195 polkupyörä
196 pöllö
197 porkkana
198 portti
199 pöytä
200 puku
201 pullo
202 punkki
203 purjevene
204 pusero
205 pyykkipoika
206 raamattu
207 rausku
208 rekka
209 retiisi
210 riikinkukko
211 ritsa
212 rukki
213 rullaluistin
214 ruuvimeisseli
215 saapas
216 saha
217 sakset
218 salaatti
219 sänky
220 sarvikuono
221 seepra
222 selleri
223 sika
224 sikari
225 silityslauta
226 silitysrauta
227 sipuli
228 sitruuna
229 sohva
230 sormi
231 sormustin
232 sorsa
233 sotilas
234 strutsi
235 suihku
236 sukka
237 sukset
238 suu
239 tähti
240 takka
241 takki
242 taltta
243 tankki
244 teippi
245 tiikeri
246 tikka
247 tomaatti
248 torni
249 toukka
250 trumpetti
251 tuhkakuppi
252 tulppa
253 tuoppi
254 tupakka
255 tussi
256 tuuba
257 tuuletin
258 tuulimylly
259 tykki
260 työpöytä
261 vadelma
262 valas
263 varvas
264 vasara
265 vatkain
266 veitsi
267 verhot
268 viehe
269 viila
270 viinilasi
271 viinimarja
272 viivotin
273 virsu
274 virtahepo
275 virveli
276 viulu
277 voileipä
278 vuohi
279 vyö

Liite 3. Semanttinen puu

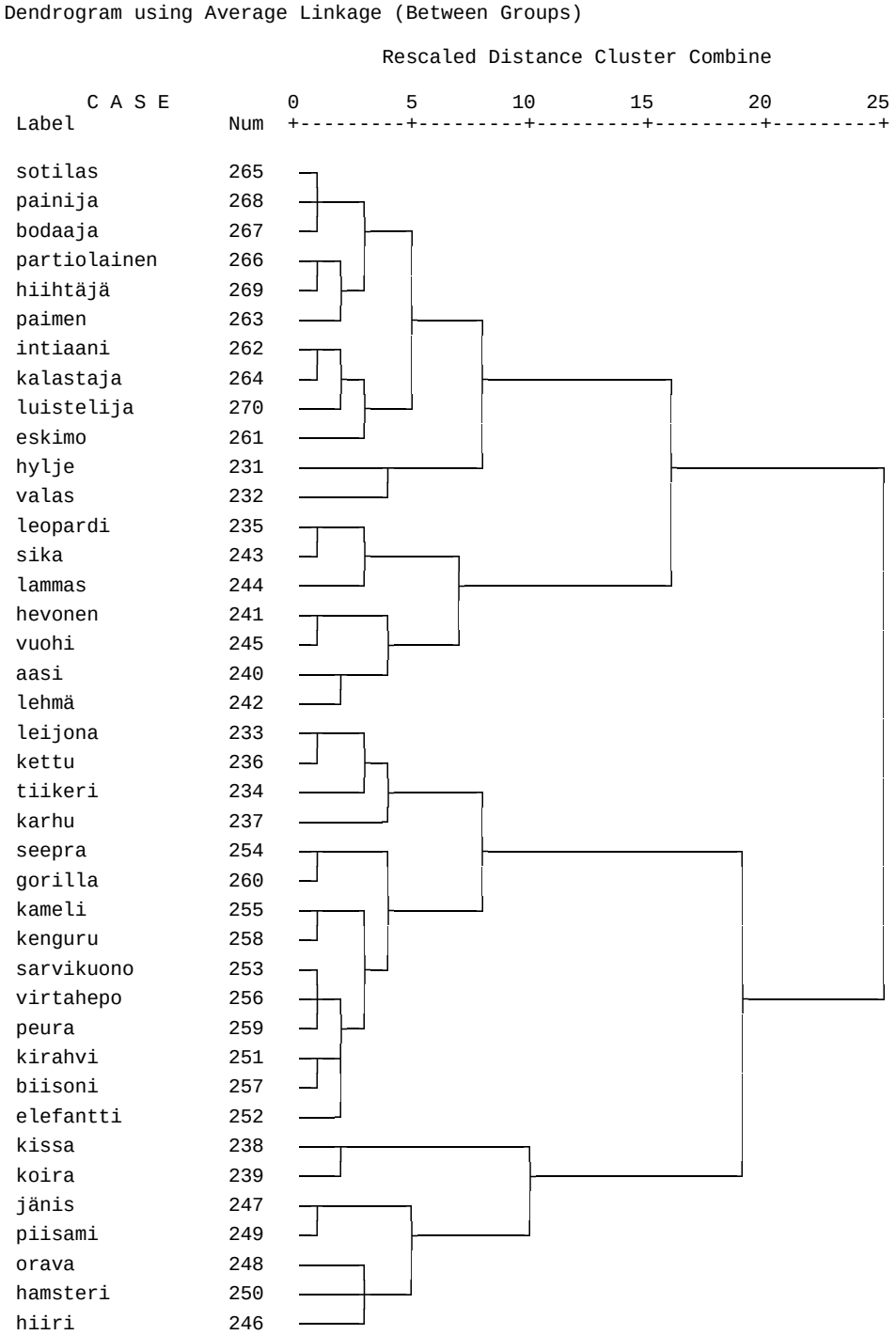


Liite 4. Semanttisen koodauksen klusterianalyysi

4 A. Semanttisen koodauksen klusterianalyysi



4 B. Nisäkkäiden semanttisen koodauksen klusterianalyysi



Liite 5. Konsonanttialiverkon vastausjakauma

Foneemi		α_F 0.00	0.20	0.40	0.60	0.80	1.00
/*b*/	Oikein %	100.00	98.80	92.20	80.70	72.30	63.50
	Väärin %	0.00	1.20	7.80	19.30	27.70	36.50
/*d*/	Oikein %	100.00	100.00	97.70	92.10	82.20	73.60
	Väärin %	0.00	0.00	2.30	7.90	17.80	26.40
/*f*/	Oikein %	100.00	99.30	91.90	81.20	72.80	69.60
	Väärin %	0.00	0.70	8.10	18.80	27.20	30.40
/*g*/	Oikein %	100.00	97.70	90.40	77.50	63.00	59.90
	Väärin %	0.00	2.30	9.60	22.50	37.00	40.10
/*h*/	Oikein %	100.00	100.00	99.70	97.60	95.40	89.50
	Väärin %	0.00	0.00	0.30	2.40	4.60	10.50
/*j*/	Oikein %	100.00	100.00	99.70	97.50	93.80	89.10
	Väärin %	0.00	0.00	0.30	2.50	6.20	10.90
/*k*/	Oikein %	100.00	100.00	99.40	97.80	95.10	89.40
	Väärin %	0.00	0.00	0.60	2.20	4.90	10.60
/*l*/	Oikein %	100.00	99.50	95.30	89.30	82.10	74.20
	Väärin %	0.00	0.50	4.70	10.70	17.90	25.80
/*m*/	Oikein %	100.00	99.90	99.00	96.20	90.20	88.60
	Väärin %	0.00	0.10	1.00	3.80	9.80	11.40
/*n*/	Oikein %	100.00	99.80	98.50	94.90	89.40	82.10
	Väärin %	0.00	0.20	1.50	5.10	10.60	17.90
/*p*/	Oikein %	100.00	98.90	94.30	88.40	83.70	74.40
	Väärin %	0.00	1.10	5.70	11.60	16.30	25.60
/*r*/	Oikein %	100.00	99.70	94.00	84.80	75.40	67.70
	Väärin %	0.00	0.30	6.00	15.20	24.60	32.30
/*s*/	Oikein %	100.00	100.00	99.40	95.00	89.70	82.70
	Väärin %	0.00	0.00	0.60	5.00	10.30	17.30
/*t*/	Oikein %	100.00	99.70	92.30	85.60	76.70	70.80
	Väärin %	0.00	0.30	7.70	14.40	23.30	29.20
/*v*/	Oikein %	100.00	99.40	93.90	82.80	76.00	68.70
	Väärin %	0.00	0.60	6.10	17.20	24.00	31.30

N = 1000

Liite 6. Vokaalialiverkon vastausjakauma

Foneemi		α_F 0.00	0.20	0.40	0.60	0.80	1.00
/A/	Oikein %	100.00	100.00	99.50	98.40	93.60	89.60
	Väärin %	0.00	0.00	0.50	1.60	6.40	10.40
/E/	Oikein %	100.00	97.20	79.30	62.20	49.60	42.40
	Väärin %	0.00	2.80	20.70	37.80	50.40	57.60
/I/	Oikein %	100.00	100.00	98.60	93.10	85.40	77.80
	Väärin %	0.00	0.00	1.40	6.90	14.60	22.20
/O/	Oikein %	100.00	98.90	88.30	75.70	73.80	65.70
	Väärin %	0.00	1.10	11.70	24.30	26.20	34.30
/U/	Oikein %	100.00	100.00	96.10	87.00	76.90	69.40
	Väärin %	0.00	0.00	3.90	13.00	23.10	30.60

N = 1000

Liite 7. Learning SLIPNETin vastausjakauma

	Vastaustyyppi %			
α_F	Oikeat	Semanttiset	Fonologiset	Yhteensä
		α_L = 0.000625, τ = 0		
0.00	98.94	1.06	0.00	100.00
0.20	91.39	0.77	7.83	100.00
0.40	66.54	0.54	32.92	100.00
0.60	43.84	0.38	55.78	100.00
0.80	28.53	0.33	71.14	100.00
1.00	18.86	0.20	80.94	100.00
		α_L = 0.00125, τ = 0		
0.00	95.75	4.25	0.00	100.00
0.20	88.62	3.70	7.68	100.00
0.40	64.82	2.26	32.92	100.00
0.60	42.63	1.55	55.82	100.00
0.80	27.92	1.04	71.04	100.00
1.00	18.05	0.74	81.21	100.00
		α_L = 0.0025, τ = 0		
0.00	88.96	11.04	0.00	100.00
0.20	83.16	9.26	7.58	100.00
0.40	61.30	6.16	32.54	100.00
0.60	39.95	4.11	55.94	100.00
0.80	25.42	2.84	71.75	100.00
1.00	16.25	2.34	81.41	100.00
		α_L = 0.005, τ = 0		
0.00	76.83	23.17	0.00	100.00
0.20	72.33	20.19	7.48	100.00
0.40	53.60	14.14	32.26	100.00
0.60	34.99	8.98	56.03	100.00
0.80	22.30	6.14	71.56	100.00
1.00	14.25	4.43	81.31	100.00
		α_L = 0.01, τ = 0		
0.00	57.60	42.40	0.00	100.00
0.20	54.51	38.06	7.43	100.00
0.40	40.53	26.92	32.55	100.00
0.60	26.49	17.35	56.16	100.00
0.80	16.95	11.92	71.13	100.00
1.00	10.67	8.18	81.15	100.00
		α_L = 0.03, τ = 0		
0.00	23.32	76.68	0.00	100.00
0.20	22.11	70.86	7.03	100.00
0.40	16.27	51.12	32.61	100.00
0.60	10.76	33.81	55.43	100.00
0.80	6.77	21.52	71.71	100.00
1.00	4.42	13.78	81.80	100.00
		α_L = 0.05, τ = 0		
0.00	12.40	87.60	0.00	100.00
0.20	11.75	81.02	7.24	100.00
0.40	8.71	58.23	33.06	100.00
0.60	5.61	37.62	56.77	100.00
0.80	3.85	24.29	71.86	100.00
1.00	2.30	16.16	81.54	100.00

N = 27900