Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Institut für Akustik und Sprachkommunikation

Studienarbeit

Optimierung einer trainingsbasierten Prosodiegenerierung

Michael Hofmann
mh21@gmx.net

31. Juli 2004

Verantwortlicher Hochschullehrer: Prof. Dr. R. Hoffmann
Betreuer: Dipl.-Ing. Oliver Jokisch

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik

Aufgabenstellung für die Studienarbeit

Name des Studenten: Herr Michael Hofmann

Thema: Optimierung einer trainingsbasierten Prosodiegenerierung

Zielsetzung:

Die Prosodiegenerierung (Akzentuierung, Intonation, Rhythmus) stellt eine Schwachstelle in Text-to-Speech (TTS)-Systemen dar. Um die Natürlichkeit von Sprachsignalen des Dresdner Sprachsynthesesystems (DRESS) zu erhöhen und den Forschungssaufwand für neue Sprachen wie Italienisch und Spanisch oder für verschiedene Sprechstile zu begrenzen, werden zur Prosodiegenerierung in DRESS u.a. trainingsbasierte Verfahren (z.B. Neuronale Netze) eingesetzt.

Folgende Teilaufgaben sollen im Rahmen der Studienarbeit bearbeitet werden:

1. Literaturrecherche zu aktuellen TTS-Systemen, Prosodiemodellen und Lernverfahren
2. Statistische Evaluation I (Fehlerkriterien, Korrelationsmaße) eines vorhandenen trainingsbasierten Verfahrens (NN-IGM)
3. Vergleich und Neukonzeption verschiedener Optimierungsansätze:
   a) Evolutionäre Optimierung der Netzwerkstruktur nach Kossebau/ Jokisch, 2003,
   b) Erweiterung der Netzeingangsinformation (Merkmalsvektoren) um linguistische Merkmale,
   c) Berücksichtigung der Intensitätsstudie Kühne/ Jokisch, 2003,
   d) weitere Alternativen aus der Literatur.
4. Implementation eines ausgewählten Optimierungsansatzes (für Deutsch oder US-Englisch)
5. Objektive und subjektive Evaluation II nach Optimierung (Fehlerkriterien, Korrelationsmaße, Hörtests)
6. Dokumentation

Es werden Vorarbeiten für den Stuttgarter Neuronalen Netz Simulator (SNNS), vorhandene Beispieldatensätze und ein Evolutions-Toolkit aus der Diplomarbeit Kossebau genutzt.

Betreuer: Dipl.-Ing. O. Jokisch

Ausgehändigt am: 01.05.2004

Einzureichen am: 31.07.2004

Prof. Dr:-Ing.habil. R. Hoffmann
Verantwortlicher Hochschullehrer

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage eingereichte Studienarbeit zum Thema

Optimierung einer trainingsbasierten Prosodiegenerierung

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 31. Juli 2004

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungen

IGM	Integrated Model of German Prosody
KNN	Künstliche Neuronale Netze
MFGI	Mixdorff-Fujisaki Model of German Intonation
MFN	Multi-layer Feedforward Neural Network
MSE	Mean Square Error – Mittlere Summe der quadratischen Fehler über alle Ausgänge pro Muster
NN-MFGI	Neuronales Netz mit MFGI-Parametern zur Schätzung von Fujisaki-Parametern
POS	Part of Speech
RMSE	Root Mean Square Error – Wurzel aus MSE
SAMPA	Speech Assessment Methods Phonetic Alphabet
SNNS	Stuttgarter Neuronale-Netze-Simulator
SSE	Sum Square Error – Summe der quadratischen Fehler über alle Ausgänge und Muster
TTS	Text To Speech

1. Einleitung

„Es war einmal eine kleine süße Dirne, die hatte jedermann lieb . . . “, so beginnt eines der beliebtesten deutschen Märchen. Keine Frage, dass demjenigen, der Rotkäppchen seinen Kindern oder Enkeln vorträgt, eine gute Portion dramatisches Geschick zu eigen sein muss, um den Zuhörer ganz in die Grimm'sche Welt zu versetzen.

Diese Kombination aus Akzentuierung, Intonation und Pausensetzung, die ein Märchen erst wirklich erlebbar macht, wird unter dem Begriff Prosodie zusammengefasst und stellt heute eine der größten Schwachstellen in der automatisierten Sprachverarbeitung dar. Noch ist ein Computer weit davon entfernt, beliebige oder selbst aufbereitete Texte in einer Qualität in Sprache umwandeln zu können, die der eines semiprofessionellen Märchenerzählers nahe kommt.

Die Hauptprobleme der Prosodiegenerierung sind nach wie vor auftretende Monotonieeffekte und störende Betonungsfehler. Obwohl einfache Anwendungen heute durchaus zu realisieren sind, scheitern z. B. umfangreichere Mensch-Maschine-Dialoge an der fehlenden Flexibilität.

Für das Feld der Prosodie versprechen daher trainingsbasierte Systeme wie das Integrated Model of German Prosody (IGM) eine Verbesserung der Verständlichkeit. In dieser Arbeit sollen verschiedene Optimierungsansätze untersucht werden, um die Leistung des integrierten neuronalen Netzes zu verbessern.

Aufbau der Arbeit Kapitel 2 gibt einen kurzen Überblick über die Grundlagen von künstlichen neuronalen Netzen (KNN), die verwendeten Aktivierungsfunktionen und die Spezialform von mehrstufigen Feedforward-Netzen (MFN). Es erläutert die Entwicklung des der Arbeit zu Grunde liegenden Prosodie-Modells vom Fujisaki-Modell bis zum IGM. Zusätzlich wird das Datenmaterial in Form des Stuttgarter Nachrichtenkorpus vorgestellt und es werden die verwendeten Fehlermaße erläutert.

Kapitel 3 bildet den Hauptteil der Arbeit. Es wird zuerst versucht, Ansätze für Optimierungen des IGMs zu finden. Diese werden dann einzeln ausführlich analysiert und diskutiert.

Abschließend werden in Kapitel 4 eine Zusammenfassung der Ergebnisse gegeben sowie mögliche Richtungen für eine Fortführung der Arbeit aufgezeigt.

2. Literaturrecherche

2.1. Grundlagen neuronaler Netze

Künstliche Neuronale Netze (KNN) sind informationsverarbeitende, dynamische Systeme, die aus einer großen Anzahl einfacher, sich gegenseitig beeinflussender Einheiten bestehen [15].

Eine wichtige Eigenschaft von KNNs ist ihre Fähigkeit, selbstständig aus Trainingsbeispielen zu lernen, ohne dass dazu eine explizite Programmierung notwendig wäre.

Neuronale Netze bestehen aus:

- Zellen (Neuronen). Sie besitzen die Bestandteile:
  - Aktivierungszustand $a_j(t)$. Er gibt den Grad der Aktivierung des Neurons j an.
  - Schwellwert θ_j
  - Aktivierungsfunktion f_{act}. Sie berechnet den neuen Aktivierungszustand $a_j(t+1)$ aus der alten Aktivierung $a_j(t)$, der Netzausgabe $net_j(t)$ sowie dem Schwellwert des Neurons θ_j.
  - Ausgabefunktion f_{out}. Sie bestimmt die Ausgabe des Neurons in Abhängigkeit der Aktivierung $a_j(t)$.
- Verbindungsnetzwerk der Zellen. Die Gewichte w_{ij} für die Verbindung eine Neuron i zu einem Neuron j ergeben zusammen mit der Ausgabe vorhergehender Neuronen die Netzeingabe für ein nachfolgendes Neuron.
- Lernregel. Sie gibt an, wie das neuronale Netz lernt, zu einer Eingabe die gewünschte Ausgabe zu produzieren. Dazu werden die Schwellwerte der Neuronen und die Gewichte der Verbindungen verändert.

Je nach Aufgabenstellung gibt es unterschiedliche Strukturen für den Aufbau eines neuronalen Netzes. Bei den hier verwendeten so genannten mehrstufigen Feedforward-Netzen (MFN) werden die Neuronen in einer Schichtstruktur angeordnet. Dabei existieren zwischen den Neuronen unterschiedlicher Ebenen

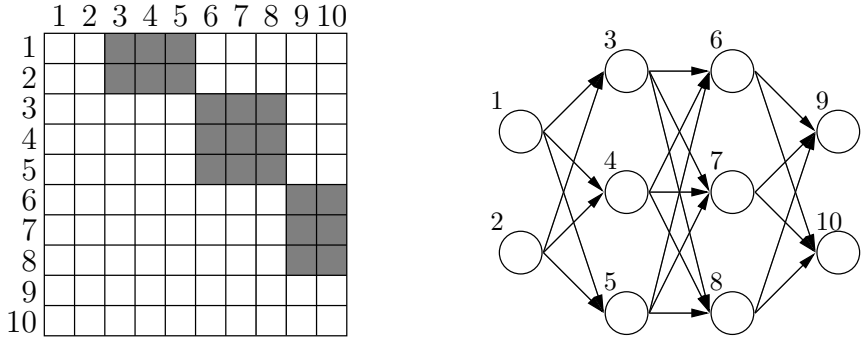

Abbildung 2.1: Beispiel eines 3-stufigen ebenenweise vollständig verbundenen Feedforward-Netzwerkes mit seiner Verbindungsmatrix

unidirektionale Verbindungen in Richtung der Eingabe- zu den Ausgabeneuronen.

Verbindungen, welche Ebenen überspringen, heißen shortcut connections. Bei Netzen, die Verbindungen zwischen allen Neuronen benachbarter Ebenen besitzen, spricht man von ebenenweise vollständig verbundenen Netzwerken (Abbildung 2.1).

2.1.1. Aktivierungsfunktionen

Die Aktivierungsfunktion f_{act} wird meist mit der Ausgabefunktion f_{out} zu einer Funktion f zusammengefasst, die wieder als Aktivierungsfunktion bezeichnet wird.

Aktivierungsfunktionen lassen sich in mehrere Gruppen einteilen:

- *Lineare Aktivierungsfunktionen* sind nur in einem einstufigen Netzwerk sinnvoll, alle Netze höherer Ordnung können auf ein solches Netz zurückgeführt werden.
- Die *Schrittfunktion*, auch binäre Schwellwertfunktion genannt, findet vor allem beim binären Perzeptron Verwendung und ist ein einfacher Schwellwertschalter.
- *Sigmoide Aktivierungsfunktionen*. Die zwei häufigsten Vertreter sind die logistische Aktivierungsfunktion und die Funktion $\tanh(x)$ (Abbildung 2.2 und 2.3 auf der nächsten Seite).

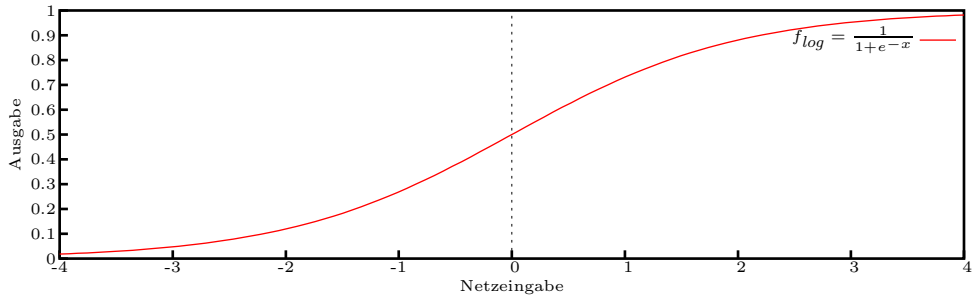

Abbildung 2.2: Logistische Aktivierungsfunktion $f_{log}(x)$

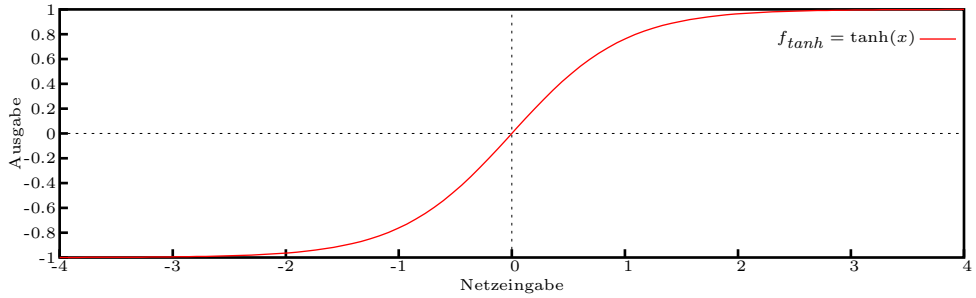

Abbildung 2.3: Aktivierungsfunktion Tangens hyperbolicus $f_{tanh}(x)$

2.1.2. Mehrstufige Feedforward-Netze

Die Anzahl der Schichten trainierbarer Verbindungen bestimmt maßgeblich über die Fähigkeit eines Netzes, eine gegebene Funktion realisieren zu können. Die folgenden Abbildungen gehen von binären Schrittfunktionen als Aktivierungsfunktionen aus, die gemachten Ausführungen gelten jedoch sinngemäß auch für andere Aktivierungsfunktionen.

Abbildung 2.4 auf Seite 6 zeigt ein einstufiges Netz sowie eine dafür realisierbare Funktion. Dieses klassische Beispiel erfordert eine lineare Trennbarkeit der zu lernenden Funktion, d. h. die Separierbarkeit mit Hilfe einer Hyperebene. Andere Funktionen, wie z. B. das XOR-Problem, lassen sich mit diesem Aufbau nicht repräsentieren.

Ein zweistufiges Netz wie das in Abbildung 2.5 dargestellte kann dagegen schon konvexe Polygone, oder allgemeiner formuliert, jede logische Verknüpfung von Halbebenen, darstellen. Die in der Abbildung dargestellte Funktion einer AND-Verknüpfung lässt sich für eine binäre Schrittfunktion am Ausgabeneuron z. B. durch einen Schwellwert erreichen, der geringfügig kleiner ist als die Summe

der Gewichte der Verbindungen zu diesem Neuron.

Das dreistufiges Netz (Abbildung 2.6) kann durch die logische Verknüpfung von konvexen Polygonen jede beliebige Funktion darstellen. Damit besitzen Netze mit weiteren Stufen keine zusätzlichen Fähigkeiten.

2.2. Fujisaki-Modell

Die Grundlage für die Analyse und Synthese von Grundfrequenzverläufen in dieser Arbeit ist das nach seinem Entwickler benannte Fujisaki-Modell (Fujisaki und Hirose, 1984). Im Gegensatz zu anderen Ansätzen für die Beschreibung von Grundfrequenzverläufen versucht es, die Entstehung sowohl aus physikalischer als auch aus physiologischer Sicht zu erklären.

Das Fujisaki-Modell geht vom physiologischen Aufbau des Kehlkopfs aus und leitet aus den zwei vorhandenen Freiheitsgraden der Bewegung beim Sprechen zwei verschiedene Mechanismen ab, die auf den Verlauf der Grundfrequenz Einfluss haben (Abbildung 2.7 auf Seite 7). Das ursprünglich für die japanische Sprache entwickelte Modell wurde durch Mixdorff an das Deutsche angepasst (siehe Abschnitt 2.3 auf Seite 8).

Es existieren zwei Arten von Eingangsgrößen: globale Phrasenkommandos (Impulssignale) und lokale Akzentkommandos (Rechtecksignale). Aus ihnen werden mit Hilfe der jeweiligen Steuereinheiten die Anteile am Grundfrequenzverlauf berechnet, welche dann mittels Superposition mit der Basisfrequenz F_b, die den minimal erreichbaren Grundfrequenzwert enthält, zum logarithmierten Grundfrequenzverlauf F_0 zusammengefügt wird (Gleichung 2.1).

$$\ln F_0(t) = \ln Fb + \sum_{i=1}^{I} Ap_i\, Gp(t - T_{0i}) + \sum_{j=1}^{J} Aa_j [Ga(t - T_{1j}) - Ga(t - T_{2j})] \tag{2.1}$$

$$Gp(t) = \begin{cases} \alpha^2 t \exp(-\alpha t) & \text{für } t \geq 0, \\ 0 & \text{für } t < 0. \end{cases} \tag{2.2}$$

$$Ga(t) = \begin{cases} \min[1 - (1 + \beta t)\exp(-\beta t), \gamma] & \text{für } t \geq 0, \\ 0 & \text{für } t < 0. \end{cases} \tag{2.3}$$

Phrasenkomponente $Gp(t)$ (Gleichung 2.2) beschreibt den generellen Verlauf der Grundfrequenz über eine ganze Phrase. Ein Impulssignal, welches zum Zeitpunkt T_{0i} den Phrasenbeginn markiert, erzeugt die Impulsantwort $Gp(t)$. Die

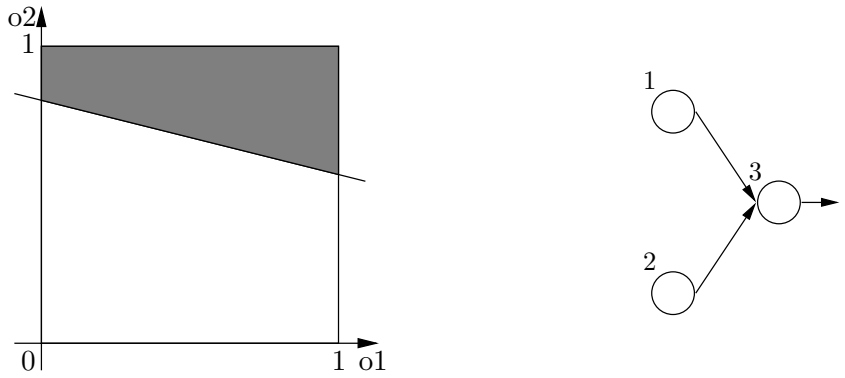

Abbildung 2.4: Von einem 1-stufigen Netz akzeptiertes Gebiet (nach [15])

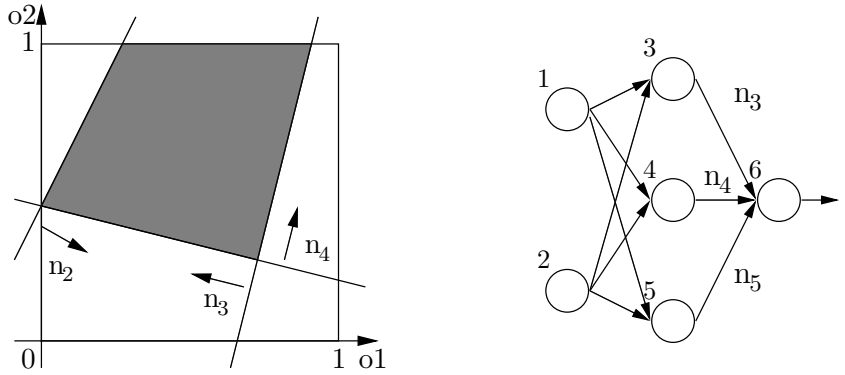

Abbildung 2.5: Von einem 2-stufigen Netz akzeptiertes Gebiet

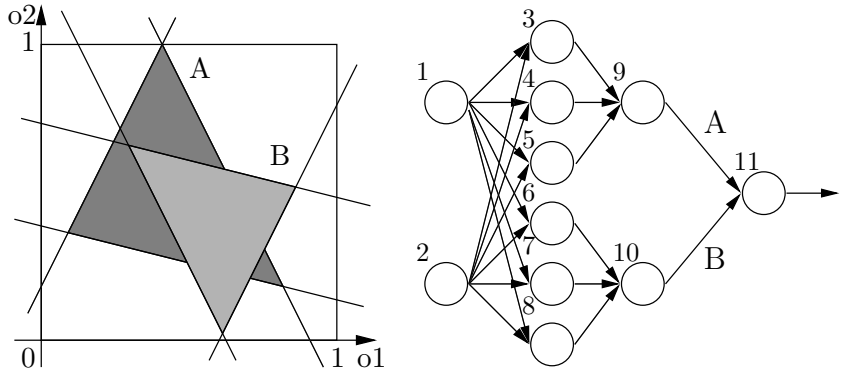

Abbildung 2.6: Von einem 3-stufigen Netz akzeptiertes Gebiet

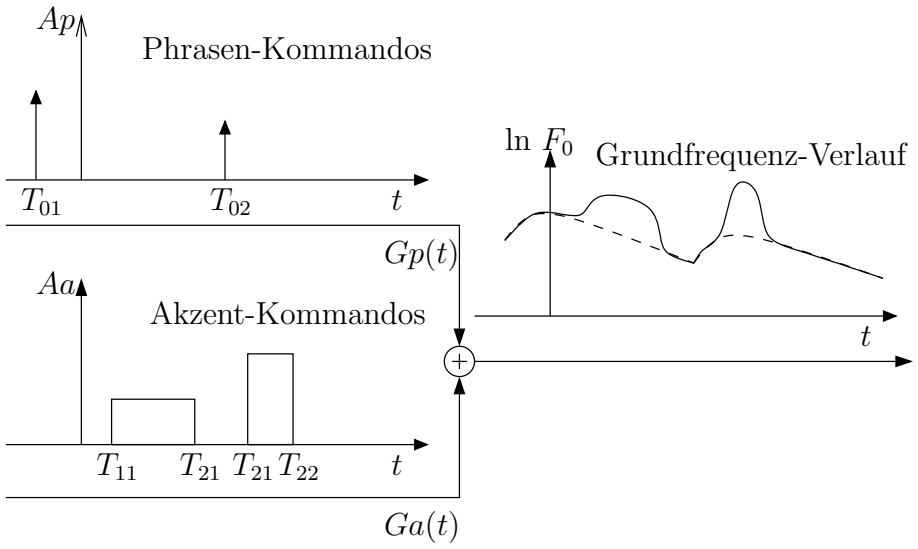

Abbildung 2.7: Vereinfachtes Blockdiagramm des Fujisaki-Modells

Zeitkonstante α bestimmt das Abfallen des Zeitverlaufes und liegt typischerweise im Bereich von 1,0/s. Sie wird über den gesamten Verlauf einer Äußerung als konstant angenommen. Die amplitudenbestimmende Konstante Ap_i des Impulssignals definiert die „Intensität“ der Phrase, d. h. den Grundfrequenzhub.

Akzentkomponente $Ga(t)$ (Gleichung 2.3 auf Seite 5) beschreibt die lokalen Akzente auf Silben, Worten oder Wortgruppen. Ein Akzentkommando in Form eines Rechtecksignals mit dem Einschalt-Zeitpunkt T_{1j} und dem Ausschalt-Zeitpunkt T_{2j} löst eine Sprungantwort der Akzent-Steuereinheit aus. Die Zeitkonstante β bestimmt den Verlauf der an- und absteigenden Flanken und liegt etwa bei 20,0/s. Sie wird über den gesamten Verlauf einer Äußerung als konstant angenommen. Mit Hilfe der Obergrenze in γ (meist $\gamma = 0{,}9$) wird die Zeit bis zum Erreichen des Maximalwertes der Amplitude begrenzt. die Konstante Aa_j bestimmt die Amplitude des Akzentkommandos.

Fb enthält die Basisfrequenz, d. h. den sprecherabhängigen asymtotischen Wert der Grundfrequenz in Abwesenheit von Akzent- oder Phrasenkommandos. Die Parameter *Ap*, T_0 und α für die Phrasen-Steuereinheit, *Aa*, T_1, T_2 und β für die Akzent-Steuereinheit sowie *Fb* werden als *Fujisaki-Parameter* bezeichnet.

Auf Grund ihrer geringen Varianz werden die Zeitkonstanten β und α im

weiteren Verlauf der Arbeit als konstant mit 1,0/s bzw. 20,0/s angenommen und nicht näher betrachtet.

2.3. Mixdorff-Fujisaki Model of German Intonation

Nach ersten grundlegenden Überlegungen zur Anwendung des Fujisaki-Modells für das Deutsche durch Möbius [12] erfolgte die Anpassung des eigentlich für das Japanische entwickelten Fujisaki-Modells an die deutsche Sprache durch das Mixdorff-Fujisaki Model of German Intonation (MFGI) von Mixdorff [9].

Ausgehend von Isačenko und Schädlich (1964) kann das Deutsche als Abfolge von Höhenwechseln aufgefasst werden. Perzeptive Experimente haben nachgewiesen, dass selbst mit einem stark vereinfachten Grundfrequenzverlauf, der nur aus Wechseln zwischen zwei konstanten Frequenzen besteht, sehr gute Ergebnisse in Hinsicht auf die zu übermittelnde syntaktische Funktion der Sätze erzielt werden können.

Stock und Zacharias (1982) verfeinern diesen Ansatz, indem sie den Begriff des Intonems einführen. Abhängig von ihrer syntaktischen Funktion unterscheiden sie drei Arten von Intonemen:

Informations-Intoneme I↓ sind fallende Wechsel der Grundfrequenz, die das Ende einer Aussage markieren.

Kontakt-Intoneme C↑ sind steigende Wechsel, die Fragen, welche nicht durch die Satzstruktur identifiziert werden können, kennzeichnen.

Nichtfinale Intoneme N↑ sind steigende Wechsel auf ein mittleres Niveau und signalisieren Unvollständigkeit.

Zusätzlich dazu werden zwei Klassen von Grenztönen definiert:

Verbindende Grenztöne B_{cat}↑ folgen direkt auf C↑ und signalisieren das Ende einer Frage.

Nichtverbindende Grenztöne B_{nocat}↑ folgen nach I↓ und kennzeichnen Fragen.

Das MFGI verbindet diese Ansätze mit dem Fujisaki-Modell, indem bei den Intonemen und Grenztönen nun neben den qualitativen Eigenschaften der steigenden oder fallenden Tonwechsel auch quantitative Eigenschaften hinzugefügt werden. Durch eine Zuordnung von Fujisaki-Akzentkommandos zu den einzelnen Intonemen gelingt eine vollständige linguistisch motivierte Analyse von

Grundfrequenzverläufen. Jede Äußerung wird dabei in sinnvolle Einheiten geteilt, die den Abschnitten des Grundfrequenzverlaufes zwischen zwei Phrasenkommandos entsprechen.

Zusätzlich wird das Fujisaki-Modell um eine Komponente ergänzt, die den bei einigen Äußerungen im Deutschen vorhandenen langsamen Anstieg der Grundfrequenz zum Ende eines Satzes hin modelliert.

2.4. Hybrider Ansatz zur Prosodiegenerierung

Da der Einsatz eines neuronalen Netzwerkes zur direkten Erzeugung des Grundfrequenzverlaufes meist schlechtere Ergebnisse liefert als ein vergleichbares regelbasiertes Modell, schlägt Jokisch in [4] einen hybriden daten- und regelbasierten Ansatz für die Prosodiekontrolle in einem TTS-System vor.

Ausgehend vom in [3] entwickelten hybriden Modell zur Dauerkontrolle beschreibt er ein Feedforward-Netzwerk (NN-MFGI), welches anstelle der Grundfrequenzwerte im Zeitbereich nur Parameter zur Erzeugung des Grundfrequenzverlaufes schätzt.

Die zuvor extrahierten Parameter des MFGI dienen als Eingabe für ein mehrstufiges neuronales Netz mit 14 Eingängen, welches zur Vorhersage der Fujisaki-Parameter von Akzentkommandos Aa, T_1 und T_2 sowie der Parameter von Phrasenkommandos Ap und T_0 dient.

2.5. Integrated Model of German Prosody (IGM)

Aufbauend auf dem NN-MFGI fasst Mixdorff im Integrated Model of German Prosody (IGM) neben den Fujisaki-Parametern noch weitere prosodische Kenngrößen in einem Netzwerk zusammen, um mögliche Synergieeffekte bei der Voraussage zu nutzen [11].

Die vom IGM zu bestimmenden Parameter sind:

- Start- und Endposition sowie Amplitude des Akzentkommandos,
- Position und Amplitude des Phrasenkommandos sowie die Länge der Pause vor der Phrase selbst,
- Silbendauer und mittlere Energie der Silbe.

Abbildung 2.8 auf der nächsten Seite zeigt das verwendete neuronale Netzwerk. Es handelt sich dabei um eine Feedforward-Netzwerk mit zwei Schichten verdeckter Neuronen mit jeweils 18 und 12 Neuronen, welches aus den 24 Eingängen die 8 gewünschten Parameter bestimmt.

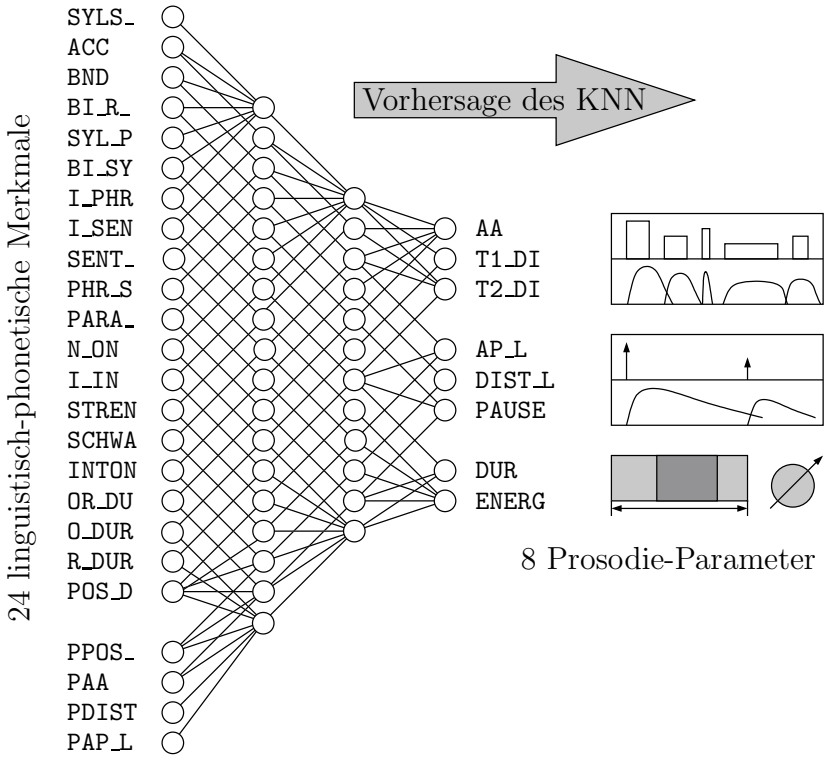

Abbildung 2.8: Modell des verwendeten neuronalen Netzes

2.6. Extraktion von Fujisaki-Parametern

Um aus einem gegebenen Grundfrequenzverlauf Fujisaki-Parameter zu extrahieren, sind einige Probleme zu lösen: die einzelnen Komponenten des Fujisaki-Modells sind durch Superposition miteinander verbunden, was eine direkte Ableitung einzelner Parameter schwierig macht, und die Abschätzung der Anzahl der notwendigen Akzent- und Phrasenkommandos erfordert die Abwägung von der Genauigkeit der Konturanpassung gegenüber der linguistischen Bedeutung solcher Fujisaki-Kommandos.

Mixdorff beschreibt in [10] ein Verfahren, dass sich in folgende Schritte gliedert:

1. Interpolation des Grundfrequenzverlaufes mittels quadratischer Splines

2. Hochpassfilterung und Auftrennung des Grundfrequenzverlaufes in eine „Hochfrequenzkontur“ und eine „Niederfrequenzkontur“

3. Initialisierung von Phrasen- und Akzentkommandos

4. Analyse-durch-Synthese zur Reduktion des mittleren quadratischen Fehlers mittels des Hill-Climb-Suchverfahrens

Dieses Verfahren ist die Grundlage für das in dieser Arbeit verwendete Datenmaterial.

Ein weiteres Verfahren beschreibt Kruschke in [7]. Dort erfolgt die Extraktion mit Hilfe von Wavelet-Analyse und evolutionärer Optimierung.

2.7. Trainingsdaten

Bei dem verwendeten Trainingsmaterial handelt es sich um Teile des Stuttgarter Nachrichtenkorpuses, welcher aufbereitete gesprochene Nachrichtentexte des Deutschlandfunkes von zwei Tagen des Jahres 1995 enthält. Er enthält Markierungen auf Laut-, Silben- und Wort-Ebene sowie linguistische Eigenschaften wie Part-Of-Speech (POS). Prosodische Markierungen entsprechend des Stuttgarter ToBI-Systems sind ebenfalls enthalten.

Die Datensätze untergliedern sich in Meldungen von Nachrichtensendungen im Abstand von 30 Minuten. Sätze, aber auch ganze Meldungen, kommen so teilweise mehrmals in fast unveränderter Form vor. Der Sprechstil ist über die gesamte Datenbasis konsistent.

Es werden nur die Meldungen verwendet, die von einem männlichen Sprecher gesprochen werden. Insgesamt ergibt sich damit ein Datenpool von 48 Minuten gesprochener Sprache in 356 Sätzen mit 5 726 Wörtern, 13 151 Silben bzw. 29 362 Phonemen. Im folgenden wird bei den 1 783 Datensätzen vom 28.07.1995 von Datensätzen von Tag 1, bei den restlichen 11 368 Datensätzen vom 21.11.1995 von Tag 2 gesprochen.

Diese Datenbasis wird durch Informationen vervollständigt, die Mixdorff im Zuge der Fujisaki-Parameter-Extraktion hinzugefügt hat.

Die Daten liegen in den verschiedensten Versionen in Tabellenform vor. Eine Übereinstimmung existiert dabei nur dort, wo die Daten aus dem Stuttgarter Korpus stammen. Die hinzugefügten Fujisaki-Parameter sowie die Akzent-Markierung wurden im Laufe der Zeit von Mixdorff mehrfach überarbeitet. Da es aus diesem Grund nicht möglich war, die ebenfalls zur Verfügung gestellten Muster, die in [6] verwendet wurden, zu reproduzieren, wurden die Trainingsdaten aus den Mustern regeneriert und mit weiteren Informationen über Dateiname, Wort, SAMPA-Notation u. ä. aus den vorhandenen Tabellen angereichert.

Es war leider ebenso unmöglich, die ursprünglichen Phrasen- und Akzentkommandos des Fujisaki-Modells, die direkt aus den Grundfrequenzverläufen der einzelnen Sprachsignale generiert wurden und so die Grundlage für die Generierung der Tabelle selbst bildeten, aufzufinden. Deswegen konnte eine geplante Ausweitung des neuronalen Netzes auf die Parameter α und β nicht vorgenommen werden.

2.8. Fehlermaße

Um die Netzleistung zu beurteilen, muss der Unterschied zwischen den gewünschten und den von dem Netz tatsächlich realisierten Ausgangswerten quantitativ erfasst werden. Als primäres Fehlermaß kommt dazu der quadratische Fehler (SE) zwischen erwartetem Ausgangswert T und erzieltem Ausgangswert I zum Einsatz.

Über alle infrage kommenden Muster N und Ausgänge n summiert, ergibt sich die Summe der quadratischen Fehler (*SSE*). Der mittlere Fehler pro Muster (*MSE*) ergibt sich bei einer Division durch die Anzahl der Muster N, die Summe der quadratischen Fehler pro Ausgang bei einer Division durch die Anzahl der Ausgänge n (*SSE/o-units*).

Alle drei Maße werden vom Stuttgarter Neuronale-Netze-Simulator (SNNS) verwendet. Zusätzlich wird die Wurzel aus dem mittleren quadratischen Fehler (*RMSE*) benutzt.

$$SSE_j = \sum_{i}^{N} (T_{j,i} - I_{j,i})^2 \tag{2.4}$$

$$SSE = \sum_{j}^{n} SSE_j \tag{2.5}$$

$$MSE_j = \frac{1}{N} SSE_j \tag{2.6}$$

$$MSE = \frac{1}{N} SSE \tag{2.7}$$

$$SSE/o\text{-}units = \frac{1}{n} SSE \tag{2.8}$$

$$RMSE_j = \sqrt{MSE_j} \tag{2.9}$$

$$RMSE = \sqrt{MSE} \tag{2.10}$$

Alle diese Fehlermaße messen den absoluten Fehler. Für eine Beurteilung

der Güte der Netzwerkschätzung pro Ausgabeparameter sind diese jedoch nur schlecht verwendbar.

Um eine Vergleichbarkeit der Netzleistung auch für unterschiedliche Gruppen von Datensätzen (z. B. unterschiedliche Sprecher) zu ermöglichen, werden relative Maße eingeführt. als Bezugsgröße für die Normierung kommt die Streuung S der Trainingswerte zum Einsatz, da sich die Mittelwerte wie in [2] für einige Parameter auf Grund des sich auch im Negativen erstreckenden Wertebereiches dafür nur bedingt eignen.

In den nachfolgenden Gleichungen bezeichnet S_j die Streuung, $rMSE_j$ den relativen mittleren quadratischen Fehler und $rRMSE_j$ die Wurzel aus dem relativen quadratischen Fehler eines Parameters j. $r\overline{T}_j$ beschreibt den Erwartungswert einer Ausgangsgröße, bezogen auf seine Streuung.

$$\begin{aligned} S_j &= \frac{1}{N-1}\sum_{i}^{N}(T_{j,i}-\overline{T}_j)^2 \\ &= \frac{1}{N-1}\left(\sum_{i}^{N}T_{j,i}^2-\frac{1}{N}\left(\sum_{i}^{N}T_{j,i}\right)^2\right) \end{aligned} \tag{2.11}$$

$$rMSE_j = \frac{MSE_j}{S_j^2} \tag{2.12}$$

$$rRMSE_j = \frac{RMSE_j}{S_j} = \sqrt{rMSE_j} \tag{2.13}$$

$$r\overline{T}_j = \frac{\overline{T}_j}{S_j} \tag{2.14}$$

3. Optimierungsansätze

Um für das IGM Optimierungsansätze zu finden, soll es in folgenden Teilaspekten näher untersucht werden:

- der Ablauf des verwendeten Trainingsprozesses und die Reproduzierbarkeit der Ergebnisse für das neuronale Netzwerk,
- die verwendete Netzwerkstruktur, d. h. der Typ des neuronalen Netzes sowie die Anzahl und Anordnung der Neuronen,
- die vorliegenden Trainingsdaten mit der Beachtung von Inkonsistenzen und der Möglichkeit, neue Parameter bereitzustellen.

3.1. Training eines neuronalen Netzes

Das Training und die Bewertung eines neuronalen Netzes erfordert zwei bzw. drei Gruppen von Mustern:

1. Die *Trainingsmenge* dient dem Training des Netzes beim überwachten Lernen in der *Lernphase*.
2. Mit Hilfe der *Testmenge* wird der bestmöglichste Trainingszustand des Netzes selektiert.
3. Das Ergebnis des Netzes mit der optionalen *Validierungsmenge* wird zur Bewertung der Netzleistung hinsichtlich Lernleistung und Generalisierung in der so genannten *Kannphase* herangezogen. Diese Aufgabe kann auch von der Testmenge übernommen werden.

Bei einem Verzicht auf eine Validierungsmenge ist das Ergebnis streng genommen nicht mehr unabhängig vom Training, da auch die Testmenge durch ihre Rolle bei der Selektion des optimalen Abbruchzeitpunkts am Training beteiligt ist. Meist wird jedoch trotzdem darauf verzichtet.

Um ein Training mit Hilfe eines Lernverfahrens wie Backpropagation zu ermöglichen, werden die Startgewichte bei der Initialisierung auf zufälligen Werte im Bereich von -1 bis 1 gesetzt. Dabei wird, um eine zufällig ungünstigere

Gewichtsinitialisierung auszuschließen, das Training in zehn voneinander unabhängigen Durchläufen mit Netzen wiederholt, die sich nur in der Gewichtsinitialisierung unterscheiden.

Die Auswahl der Trainings- und Testdatensätze wurde in [6] sequentiell vorgenommen, d. h. das Training erfolgte mit den ersten 10 000 Datensätzen, als Testmenge fungierten die übrigbleibenden 3 151 Datensätze. Um mögliche tages- und zeitabhängige Effekte auszuschließen, wird hier stattdessen auf eine zufällige Auswahl zurückgegriffen, wobei jedoch des Verhältnis von Trainings- zu insgesamt vorhanden Datensätzen von $\frac{10\,000}{13\,151} \approx 1{,}3 \ldots 1{,}4$ beibehalten wird.

Um eine Vergleichbarkeit zu gewährleisten, werden sofern möglich für die verschiedenen trainierten Netze für eine Tabelle oder ein Diagramm in diesem Bericht immer dieselben zufällig ausgewählten Trainings- und Test-Muster verwendet. Unter diesen Bedingungen können Trainings- und Testmengen verschiedener Netze auch voneinander unabhängig verglichen werden. Zwischen unterschiedlichen Versuchen können jedoch die Ergebnisse, z. B. des gegebenen Netzwerkes mit zwei Schichten verdeckter Neuronen, leicht schwanken.

Das Training und die Messung der Leistungsfähigkeit des Netzes erfolgt mit Hilfe eines Batchman-Skriptes [13] (Abbildung C.1 auf Seite 58 im Anhang). Dabei werden die selben Muster in mehreren Trainingsschritten zum Training des Netzes genutzt. Diese fortschreitende Adaption des Netzes an die Eingabedaten wird für eine konstante Schrittzahl durchgeführt.

Zur Auswahl des am besten trainierten Netzzustandes wird wiederholt nach einer festgelegten Anzahl von Trainingsschritten das Netz abgespeichert. Die Wertung der Netzleistung erfolgt nach vollständig durchgeführtem Training nach dem minimalen Fehler von Trainings- oder Testmenge von allen gespeicherten Netzen. Damit wird sichergestellt, dass der Zeitpunkt n_o des Minimums des Fehlers *SSE* der Testmenge nicht verfehlt wird und eine Überadaption des Netzes in den letzten Trainingsschritten für die Beurteilung der Leistungsfähigkeit folgenlos bleibt (Abbildung 3.1 auf der nächsten Seite).

3.2. Struktur

Bei dem verwendeten Netzwerk handelt es sich um ein ebenenweise vollständig verbundenes Feedforward-Netzwerk, wobei durch die Verwendung von Ausgangsinformationen vorheriger Silben Aspekte von Time-Delay-Netzen integriert werden [15].

Die genau Struktur des IGM ist relativ willkürlich gewählt. Es liegt nahe, das Verhalten ähnlich aufgebauter Netze zu untersuchen.

In [6] wurde der Ansatz verfolgt, mit Hilfe von evolutionärer Optimierung

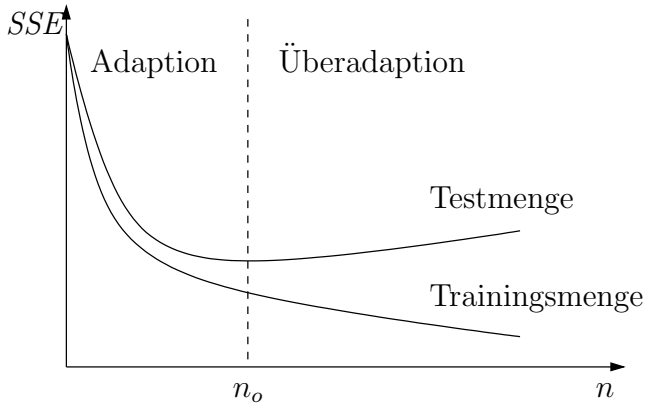

Abbildung 3.1: Vereinfachter Fehlerverlauf beim Training des neuronalen Netzes

(siehe auch [5]) die Struktur des IGM hinsichtlich einzelner Neuronen und Verbindungen zu optimieren. Im Gegensatz dazu soll nun untersucht werden, inwieweit

- die Anzahl der Schichten verdeckter Neuronen,
- die Anzahl der Neuronen pro Schicht sowie
- die Anzahl der Ausgabeneuronen

Einfluss auf die Lernleistung des Netzwerkes hat. Von einer Vertiefung des Ansatzes nach Kossebau wird abgesehen, da auf Grund der Lernfähigkeit des Netzes und der ebenenweise vollständig verbundenen Topologie keiner Verbindung und keinem Neuron einer verdeckten Schicht eine spezielle Aufgabe zufällt.

3.2.1. Anzahl der Schichten trainierbarer Verbindungen

Die Anzahl der Schichten hat direkten Einfluss auf die Menge der durch das Netz repräsentierbaren Funktionen. Die bisher gewählte Struktur eines dreistufigen Netzes ermöglicht das Erfassen beliebiger Eingabeverteilungen. Es ist zu untersuchen, inwieweit ein Netzwerk mit nur zwei Schichten trainierbarer Verbindungen signifikant schlechtere Ergebnisse liefert.

Abbildung 3.2 auf der nächsten Seite zeigt den prinzipiellen Aufbau eines solchen Netzwerks gegenüber dem ursprünglichen Netzwerk des IGM. Ein solches Netzwerk hätte wahrscheinlich eine geringere Anzahl Neuronen als ein vergleichbares dreistufiges Netz. Allerdings hängt die Anzahl der benötigten

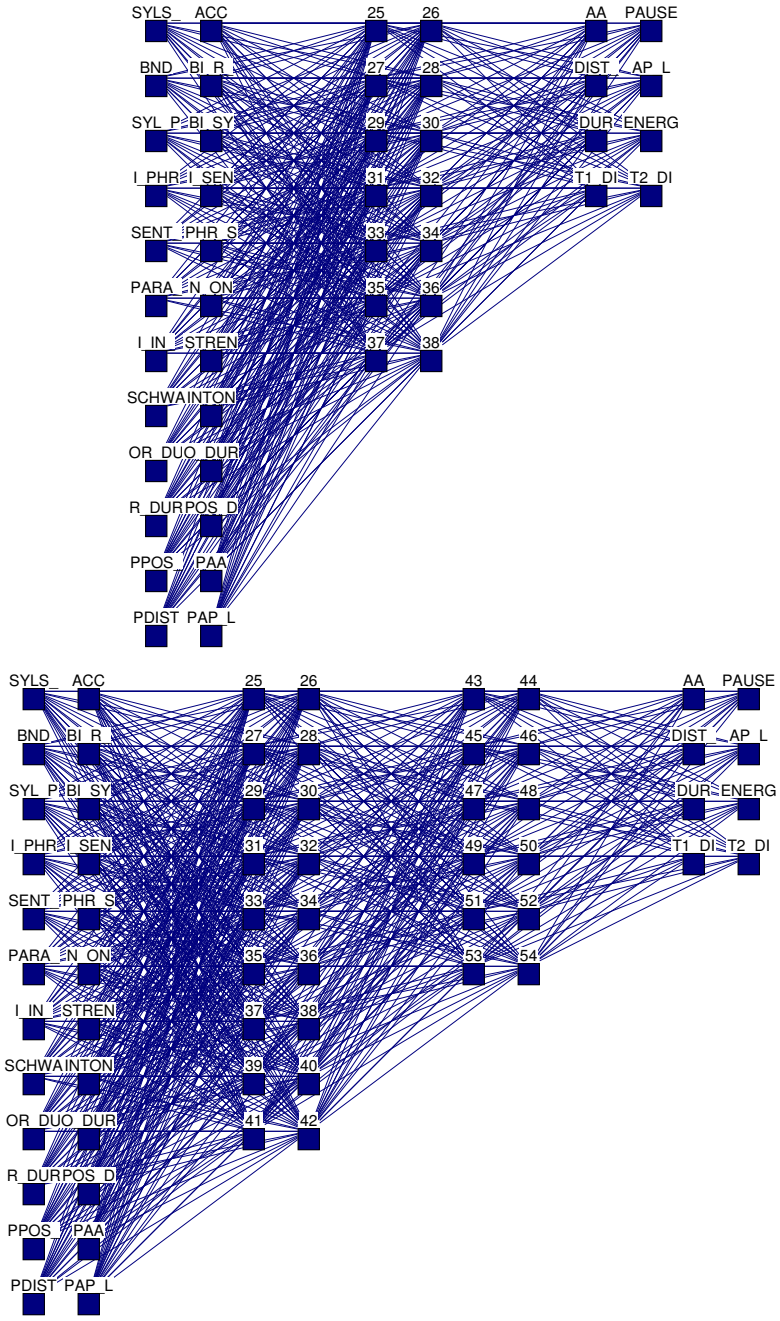

Abbildung 3.2: Beispiele für zwei- und dreistufige Netzwerke

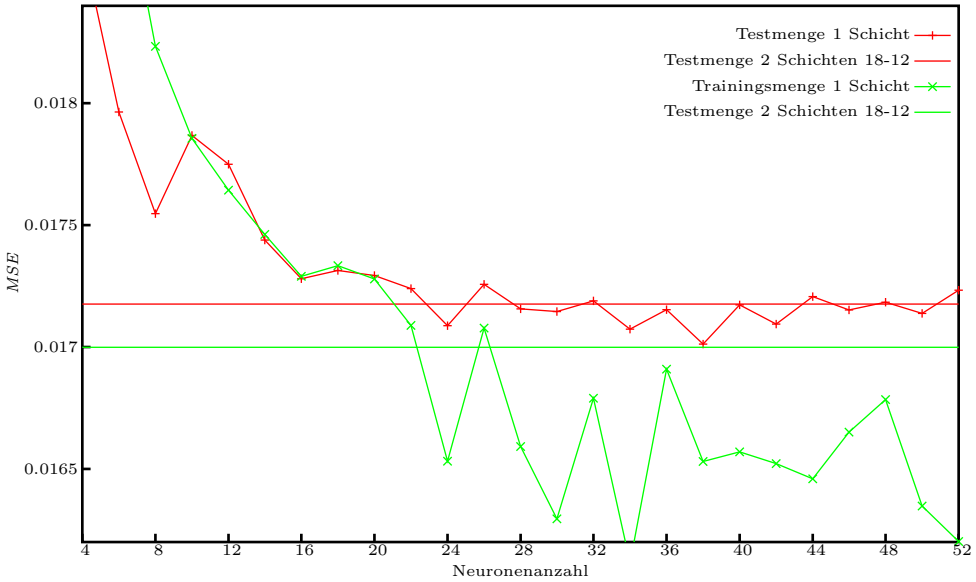

Abbildung 3.3: Fehlerwerte von zweistufigen Netzwerken für Tag 2

Verbindungen auch vom Aufbau des Netzwerks ab, so dass ein zweistufiges Netzwerk dann trotzdem durchaus mehr Verbindungen aufweisen kann.

Die Ergebnisse von Netzwerken mit nur einer Schicht verdeckter Neuronen im Vergleich zum vorgegebenen Netzwerk mit zwei verdeckten Schichten à 18 bzw. 12 Neuronen sind in Abbildung 3.3 zu sehen. Es ist erkennbar, dass mit steigender Neuronenanzahl der Lernerfolg für die Trainingsmenge steigt, ab einer Neuronenanzahl in der verdeckten Schicht, die in etwa der Zahl der Eingabeneuronen entspricht, ist ein besseres Ergebnis als mit dem bisherigen Netzwerk erzielbar. Für weniger als zehn Neuronen steigt der Fehler überproportional stark an, das Netzwerk ist nicht mehr fähig, die Eingabedaten ausreichend zu repräsentieren.

Anders sieht das Ergebnis jedoch bei der Testmenge aus. Die Verbesserung gegenüber dem Netzwerk mit zwei Schichten verdeckter Neuronen ab 24 Neuronen ist dort nicht erkennbar, vielmehr schmiegt sich das Ergebnis für höhere Neuronenzahlen an das Ergebnis des dreistufigen Netzwerks an.

Damit ist zu vermuten, dass das gegebene dreistufige Netzwerk den maximalen Lerngrad bei dem vorhandenen Datenmaterial darstellt. Ein ähnliches Ergebnis ist allerdings schon mit einem zweistufigen Netzwerk reduzierter Komplexität, d. h. von hier nur etwa 24 anstatt von 30 Neuronen, jedoch mit der gleichen Anzahl von etwa 750 Verbindungen dazwischen, realisierbar.

Netztyp	MSE_{Test}	Neuronen	Verbindungen
Original-Netz 18-12	$17{,}18 \cdot 10^{-3}$	30	744
Dreistufiges Netz 12-10	$17{,}31 \cdot 10^{-3}$	22	488
Zweistufiges Netz 24	$17{,}09 \cdot 10^{-3}$	24	768

Tabelle 3.1: Vergleich zwei- und dreistufiger Netzwerke für Tag 2

3.2.2. Anzahl der Neuronen pro Schicht

Um die Vermutung zu bestätigen, dass mit dem gegebenen Netzwerk der maximal erreichbare Grad an Lernleistung erreicht ist, werden Netzwerke mit zwei Schichten verdeckter Neuronen und unterschiedlicher Anzahl von Neuronen pro Schicht analysiert. Die Eingangsanzahl von 24 Neuronen sowie die halbe Ausgangsanzahl von vier Neuronen geben hierbei die Ober- bzw. Untergrenze für die Neuronenanzahl in den verdeckten Schichten vor (Abbildungen 3.4 bis 3.5 auf der nächsten Seite).

Auch hier ist ein ähnliches Verhalten wie bei Netzwerken mit einer Schicht verdeckter Neuronen sichtbar: Höhere Neuronenzahlen bedeuten zwar ein verbessertes Ergebnis beim Lernen der Trainingsmenge, jedoch ist die Generalisierungsfähigkeit weiterhin relativ unabhängig von der Neuronenanzahl. Erst bei einer sehr stark reduzierten Netzstruktur mit zehn oder weniger Neuronen pro Schicht steigt der Fehler der Testmenge signifikant an. Selbst eine hohe Zahl von Neuronen in einer Schicht kann den „Flaschenhals“ einer Schicht mit nur vier Neuronen nicht kompensieren.

Damit ist ein dreistufiges Netzwerk einem zweistufigen Netzwerk klar überlegen (Tabelle 3.1): Sowohl ein dreistufiges Netzwerk mit zwölf bzw. zehn Neuronen in den verdeckten Schichten als auch ein zweistufiges Netzwerk mit 24 Neuronen erzielten ähnliche Fehler in der Testmenge wie das ursprüngliche Netzwerk mit 18 bzw. zwölf Neuronen.

3.2.3. Spezialisierung des Netzes auf einen Ausgabeparameter

Um den Einfluss der Anzahl der Ausgabeneuronen abzuschätzen, werden Durchläufe mit jeweils nur einem Ausgabeneuron gemacht, Tabelle 3.2 auf Seite 21 zeigt die Ergebnisse.

Das Trainingsergebnis eines Netzes mit nur einem Ausgang ist durchschnittlich mehr als fünf Prozent besser als das des Netzwerkes mit allen acht Ausgängen. Dieses Ergebnis ist relativ unabhängig von Parameter und Tag zu beobachten. Zusätzlich zeigt sich, dass der größte Fehleranteil durch den Parameter

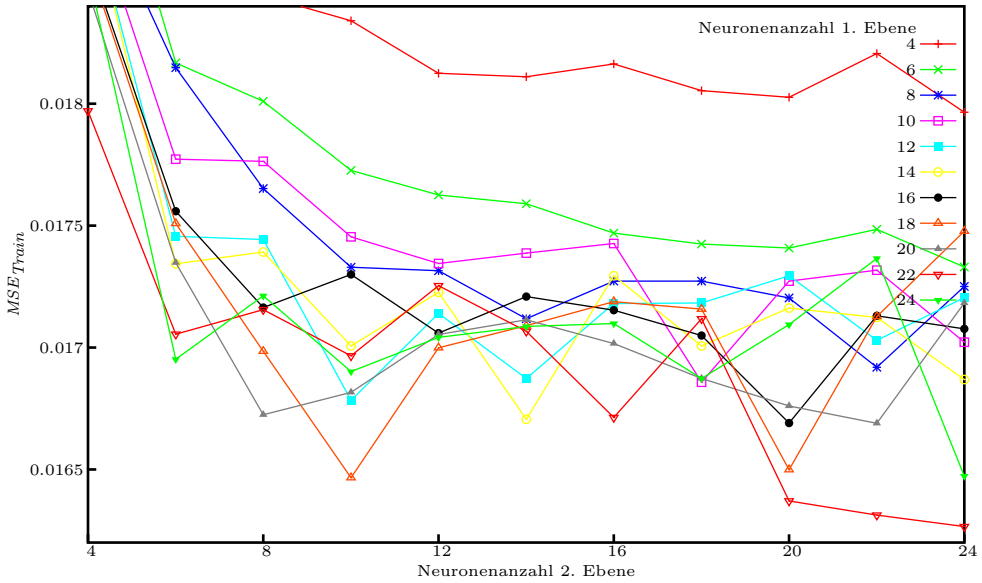

Abbildung 3.4: Trainingsfehler von dreistufigen Netzwerken für Tag 2

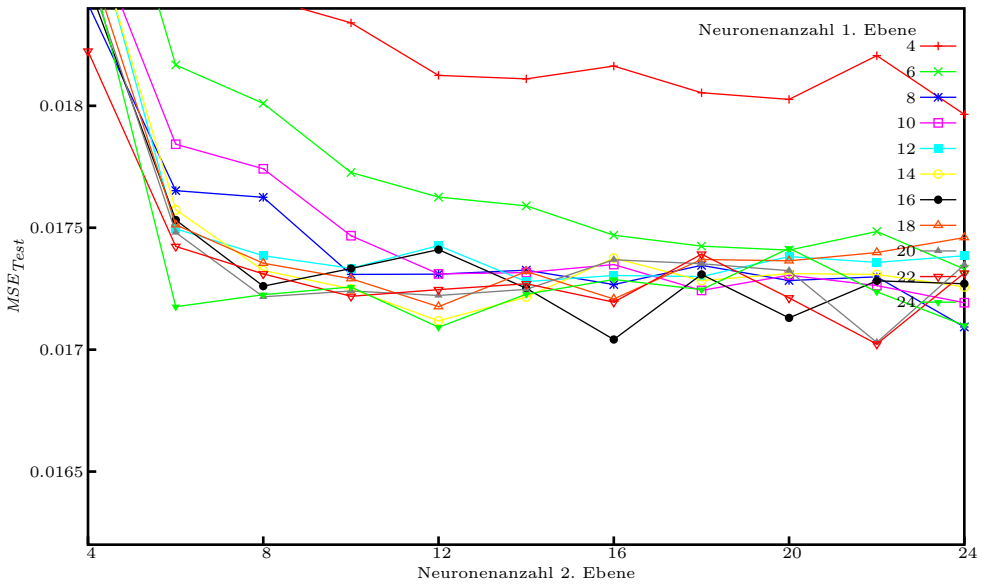

Abbildung 3.5: Testfehler von dreistufigen Netzwerken für Tag 2

	Tag 1 $MSE/10^{-3}$			Tag 2 $MSE/10^{-3}$		
Parameter	Training	Test	Gesamt	Training	Test	Gesamt
`AA`	1,576	1,512	1,559	0,637	0,577	0,621
`AA` einzeln	1,309	1,312	1,310	0,600	0,549	0,586
`T1_DI`	3,464	4,873	3,845	2,894	3,697	3,111
`T1_DI` einzeln	3,323	4,462	3,631	2,913	3,572	3,091
`T2_DI`	4,637	8,932	5,798	3,923	4,351	4,038
`T2_DI` einzeln	5,294	8,583	6,183	3,785	4,228	3,905
`AP_L`	1,336	1,281	1,321	1,082	1,164	1,104
`AP_L` einzeln	0,957	0,972	0,961	1,039	1,067	1,046
`DIST`	0,880	0,911	0,889	0,693	0,751	0,709
`DIST` einzeln	0,589	0,757	0,635	0,512	0,700	0,563
`PAUSE`	0,685	0,354	0,595	0,463	0,641	0,511
`PAUSE` einzeln	0,305	0,313	0,307	0,399	0,606	0,455
`DUR`	2,619	3,044	2,734	1,947	1,941	1,946
`DUR` einzeln	2,555	2,791	2,619	1,793	1,826	1,802
`ENERG`	14,22	13,71	14,08	4,662	4,750	4,686
`ENERG` einzeln	12,67	13,36	12,86	4,369	4,549	4,418
Gesamt	29,42	34,61	30,82	16,30	17,87	16,73
Gesamt einzeln	27,00	32,55	28,50	15,41	17,10	15,87
Verbesserung	8,2 %	6,0 %	7,5 %	5,5 %	4,3 %	5,1 %

Tabelle 3.2: Vergleich alle Ausgänge – nur ein Ausgang

`ENERG` entsteht, erst mit sehr viel kleinerem Fehler folgen `T1_DI` und `T2_DI`.

Um herauszubekommen, ob dieses Phänomen vielleicht nur an der relativ gesehen größeren Anzahl von Neuronen pro Schicht (also insgesamt $8 \times 18 = 144$ bzw. $8 \times 12 = 96$ Neuronen) liegt, werden einige Vergleichsläufe durchgeführt (Tabelle 3.3 auf der nächsten Seite), die dieses übergroße Netzwerk mit einem ebenfalls übergroßen zweistufigen sowie dem gegebenen Netzwerk vergleichen.

Dabei ergibt sich jedoch, dass selbst ein solches übergroßes Netzwerk zwar die Trainingsmenge besser lernt, sie jedoch nicht besser generalisiert und damit gleiche Werte für die Testmenge erzielt wie das gegebene Netzwerk mit 18 und zwölf Neuronen pro Schicht.

Netztyp	MSE_{Train}	MSE_{Test}	MSE_{Gesamt}
Dreistufiges Netz 18x12 Tag 1	$30{,}21 \cdot 10^{-3}$	$35{,}33 \cdot 10^{-3}$	$31{,}59 \cdot 10^{-3}$
Zweistufiges Netz 144 Tag 1	$26{,}68 \cdot 10^{-3}$	$35{,}28 \cdot 10^{-3}$	$29{,}00 \cdot 10^{-3}$
Dreistufiges Netz 144x96 Tag 1	$25{,}16 \cdot 10^{-3}$	$34{,}92 \cdot 10^{-3}$	$27{,}80 \cdot 10^{-3}$
Dreistufiges Netz 18x12 Tag 2	$18{,}05 \cdot 10^{-3}$	$18{,}35 \cdot 10^{-3}$	$18{,}13 \cdot 10^{-3}$
Zweistufiges Netz 144 Tag 2	$16{,}13 \cdot 10^{-3}$	$18{,}27 \cdot 10^{-3}$	$16{,}71 \cdot 10^{-3}$

Tabelle 3.3: Vergleich Netzwerk 144x96 mit 18x12

3.2.4. Zusammenfassung

Das bisherige Netzwerk ist überdimensioniert und kann ohne Probleme verkleinert werden. Obwohl ein zweistufiges Netzwerk mit 24 Neuronen in der verdeckten Schicht genügt, um ähnliche Erfolge wie mit dem bisherigen Netz zu erzielen, ist ein dreistufiges Netzwerk, z. B. mit zwölf und zehn Neuronen pro Schicht, angemessener. Dieses erfordert eine geringere Anzahl an Neuronen insgesamt und weist darüber hinaus nur etwa zwei Drittel der Verbindungen eines zweistufigen Netzwerkes auf.

Die Zerlegung des bisherigen Netzwerkes mit acht Ausgängen in acht Teilnetze mit jeweils einem Ausgang verringert den Fehler um ungefähr fünf Prozent. Scheinbar ist damit ein zielgerichteteres Training möglich.

3.3. Korrelationen zwischen den Parametern

Für einen Überblick über die verwendete Datenbasis zeigt Abbildung 3.6 auf der nächsten Seite das Ergebnis einer Korrelationsanalyse zwischen den Eingabeparametern und den vorgegebenen Fujisaki-, Dauer- und Energie-Parametern. „Heißere Farben“ geben einen höheren linearen Zusammenhang zwischen den Ein- und Ausgabeparametern an.

Gut sichtbar ist die schon in [11] und [6] erwähnte Abhängigkeit der Ausgänge von jeweils einigen wenigen Eingängen. Akzent-, Phrasen- und sonstige Parameter gliedern sich klar nach den Eingängen, mit denen eine Korrelation besteht.

3.4. Inkonsistenzen im Datenmaterial

Da Kossebau in [6] von Unterschieden zwischen den Daten von Tag 1 und Tag 2 berichtet, sollen die Daten unter diesem Gesichtspunkt näher untersucht

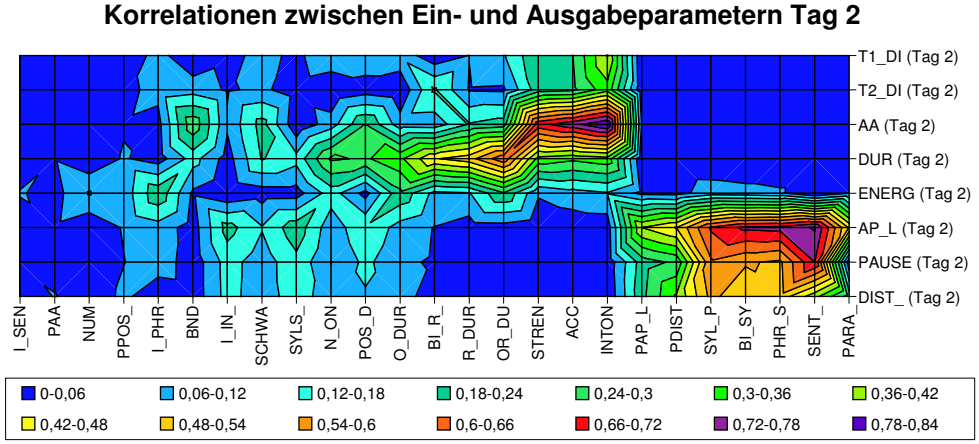

Abbildung 3.6: Korrelationen zwischen den Ein- und Ausgängen des IGM

Datensätze	Tag	Umfang	MSE_{Train}	MSE_{Test}	MSE_{Gesamt}
1 – 1783	Tag 1	klein	$29{,}99 \cdot 10^{-3}$	$34{,}53 \cdot 10^{-3}$	$31{,}22 \cdot 10^{-3}$
1 – 13151	Beide	groß	$19{,}96 \cdot 10^{-3}$	$22{,}13 \cdot 10^{-3}$	$20{,}55 \cdot 10^{-3}$
1 – 11368	Beide	mittel	$19{,}72 \cdot 10^{-3}$	$22{,}55 \cdot 10^{-3}$	$20{,}48 \cdot 10^{-3}$
1784 – 13151	Tag 2	mittel	$16{,}25 \cdot 10^{-3}$	$17{,}90 \cdot 10^{-3}$	$16{,}69 \cdot 10^{-3}$
1784 – 3566	Tag 2	klein	$15{,}34 \cdot 10^{-3}$	$16{,}96 \cdot 10^{-3}$	$15{,}77 \cdot 10^{-3}$

Tabelle 3.4: Einfluss des Umfangs der Trainings- und Testmengen

werden.

Eine erste subjektive perzeptive Untersuchung ergibt keine nennenswerten Unterschiede zwischen den Mitschnitten beider Tage.

3.4.1. Anzahl der Datensätze für Tag 1 und Tag 2

Die Daten von Tag 1 umfassen 1 783 Datensätze, die vom 2. Tag 11 368 Datensätze. Um auszuschließen, dass dies einen Einfluss auf die gemessenen Fehlerwerte hat, wird das Netzverhalten anhand mehrerer Trainings- und Testmengen in Tabelle 3.4 untersucht.

In Tabelle 3.4 ist dabei erkennbar, dass der Umfang der Trainings- und Testmenge nur geringen Einfluss auf die erzielten Fehlerwerte hat. Im Gegensatz dazu ist der Fehler primär durch die Zugehörigkeit zu Tag 1 bzw. Tag 2 bestimmt.

Datensätze	Tag	Teil	MSE_{Train}	MSE_{Test}	MSE_{Gesamt}
1 – 891	Tag 1	Teil 1	$30{,}18 \cdot 10^{-3}$	$37{,}76 \cdot 10^{-3}$	$32{,}23 \cdot 10^{-3}$
892 – 1783	Tag 1	Teil 2	$27{,}81 \cdot 10^{-3}$	$36{,}17 \cdot 10^{-3}$	$30{,}07 \cdot 10^{-3}$
1 – 1783	Tag 1	alle	$27{,}90 \cdot 10^{-3}$	$35{,}03 \cdot 10^{-3}$	$29{,}83 \cdot 10^{-3}$
1784 – 7467	Tag 2	Teil 1	$16{,}65 \cdot 10^{-3}$	$16{,}86 \cdot 10^{-3}$	$16{,}70 \cdot 10^{-3}$
7468 – 13151	Tag 2	Teil 2	$18{,}14 \cdot 10^{-3}$	$19{,}28 \cdot 10^{-3}$	$18{,}45 \cdot 10^{-3}$
1 – 13151	Tag 2	alle	$17{,}05 \cdot 10^{-3}$	$17{,}47 \cdot 10^{-3}$	$17{,}16 \cdot 10^{-3}$

Tabelle 3.5: Homogenität der Daten in Abhängigkeit vom Tag

3.4.2. Homogenität der Daten eines Tages

Um ein ähnliches Verhalten für Datensätze von unterschiedlichen Zeitpunkten eines Tages auszuschließen, wird stichprobenartig eine Überprüfung durchgeführt, ob sich bei einer Teilung der Datensätze eines Tages in zwei gleich große Teilmengen beide Hälften vergleichbar verhalten.

Tabelle 3.5 zeigt die Ergebnisse der Untersuchung. Wieder sind zwischen Versuchen des selben Tages keine Unterschiede zu finden, die kleineren Abweichungen zwischen den Datensätzen eines Tages lassen sich leicht mit der willkürlichen Auswahl der Trainings- und Testmengen erklären.

Eine Zusammenfassung dieser beiden Versuche zeigt Tabelle 3.6. Als einziges markantes Unterscheidungsmerkmal bleibt die Zugehörigkeit zu Tag 1 oder Tag 2.

	MSE_{Gesamt}		
Umfang	Tag 1	Tag 2	Beide
Teil 1	$32{,}23 \cdot 10^{-3}$	$16{,}70 \cdot 10^{-3}$	
Teil 2	$30{,}07 \cdot 10^{-3}$	$18{,}45 \cdot 10^{-3}$	
alle	$29{,}83 \cdot 10^{-3}$	$17{,}16 \cdot 10^{-3}$	
klein	$31{,}22 \cdot 10^{-3}$	$15{,}77 \cdot 10^{-3}$	
mittel		$16{,}69 \cdot 10^{-3}$	$20{,}48 \cdot 10^{-3}$
groß			$20{,}55 \cdot 10^{-3}$

Tabelle 3.6: Einflüsse von Homogenität und Umfang

3.4.3. Aufschlüsselung nach Ausgabeparametern

Um zu analysieren, bei welchen der acht Ausgabeparameter die Fehlerwerte nach dem Training sich am stärksten zwischen den beiden Tagen unterschieden, erfolgt eine Untersuchung auf Parameterebene. Die Tabellen 3.7 bis 3.9 auf der nächsten Seite zeigen eine statische Analyse der Datensätze.

Anzahl gesamt steht dabei für die Anzahl aller Datensätze an diesem Tag, *Anzahl möglich* ist die Nummer aller Akzent- bzw. Phrasenkommandos, *Anzahl gesetzt* bezeichnet die Datensätze, die diesen Parameter auch wirklich enthalten. Die nächsten zwei Zeilen enthalten diese Angabe noch einmal, auf alle bzw. alle möglichen Datensätze bezogen. Der *Mittelwert* berechnet sich nur über die Datensätze, in denen dieser Parameter verschieden von Null ist, gleiches gilt für die Streuung.

Besonders fallen die starken Unterschiede in den Mittelwerten und Standard-Abweichungen für `ENERG` und `AA`, ferner für `T2_DI` und `PAUSE` auf. Da bei diesen Werten ähnliche Unterschiede in den Fehlerwerten einhergehen (Tabelle 3.2 auf Seite 21), liegt es nahe, die Fehlerwerte relativ zu den Standard-Abweichungen zu betrachten. Die Mittelwerte eignen sich bei `DIST_`, `T1_DI` und `T2_DI` dafür nicht, da sich dort der Wertebereich auch in negativer Richtung erstreckt. Die Normierung erfolgt mit S^2, um der quadratischen Natur von *MSE* Rechnung zu tragen.

Die Werte nach der Normierung (Tabelle 3.10 auf Seite 27) liegen sehr viel näher beieinander als die ursprünglichen Werte. Besonders der Fehler von `ENERG`, der zu einem Großteil für die Differenz zwischen Tag 1 und Tag 2 verantwortlich war, ist relativ zur quadratischen Streuung gesehen für beide Tage gleich. Ähnliches gilt, jedoch mit größeren Abweichungen, auch für `AA` sowie alle anderen Parameter.

Damit ist eine Erklärung für die großen Abweichungen im Trainingsergebnis gefunden. Es existiert scheinbar kein qualitativer Unterschied zwischen Tag 1 und Tag 2, sondern nur ein quantitativer. Zugleich zeigt sich ein Schwachpunkt in der bisherigen Bewertung der Netzleistung. Auf Grund der hohen Anzahl von `ENERG`-Werten und dem damit auch höheren quadratischen Fehler wird dieser in der Gesamtbewertung überbetont.

Abhilfe könnte z. B. eine Berechnung von *MSE*-Werten bringen, die auf die wahre Anzahl von Datensätzen bezogen ist, die diesen Parameter auch wirklich gesetzt haben, anstatt immer auf die volle Anzahl Datensätze zu normieren. Da dies jedoch in SNNS nicht implementiert ist, bleibt dieser Ansatz auf die Auswertung beschränkt.

	AA		T1_DI		T2_DI	
	Tag 1	Tag 2	Tag 1	Tag 2	Tag 1	Tag 2
Anzahl gesamt	1783	11368	1783	11368	1783	11368
Anzahl möglich	422	2600	422	2600	422	2600
Anzahl gesetzt	422	2600	417	2569	422	2587
Anzahl/gesamt	23,7 %	22,9 %	23,4 %	22,6 %	23,7 %	22,8 %
Anzahl/möglich	100 %	100 %	98,8 %	98,8 %	100 %	99,5 %
Mittelwert	0,186	0,130	0,0090	0,0419	0,0285	0,0412
Streuung	0,0827	0,0598	0,163	0,151	0,219	0,174

Tabelle 3.7: Statistische Eigenschaften der Akzentparameter

	AP_L		DIST_		PAUSE	
	Tag 1	Tag 2	Tag 1	Tag 2	Tag 1	Tag 2
Anzahl gesamt	1783	11368	1783	11368	1783	11368
Anzahl möglich	189	1177	156	891	156	891
Anzahl gesetzt	156	891	156	891	100	493
Anzahl/gesamt	8,75 %	7,83 %	8,75 %	7,83 %	5,61 %	4,34 %
Anzahl/möglich	82,5 %	75,7 %	100 %	100 %	64,1 %	55,3 %
Mittelwert	0,268	0,277	0,0919	0,0861	0,152	0,174
Streuung	0,142	0,157	0,107	0,109	0,085	0,109

Tabelle 3.8: Statistische Eigenschaften der Phrasenparameter

	DUR		ENERG	
	Tag 1	Tag 2	Tag 1	Tag 2
Anzahl gesamt	1783	11368	1783	11368
Anzahl möglich	1783	11368	1783	11368
Anzahl gesetzt	1783	11368	1783	11368
Anzahl/gesamt	100 %	100 %	100 %	100 %
Anzahl/möglich	100 %	100 %	100 %	100 %
Mittelwert	0,189	0,189	0,305	0,198
Streuung	0,0854	0,0771	0,135	0,080

Tabelle 3.9: Statistische Eigenschaften der globalen Parameter

	$S^2/10^{-3}$		$MSE_{Te}/10^{-3}$		MSE_{Te}/S^2	
Parameter	Tag 1	Tag 2	Tag 1	Tag 2	Tag 1	Tag 2
AA	6,845	3,574	1,512	0,577	0,2209	0,1613
T1_DI	26,45	22,85	4,873	3,697	0,1842	0,1618
T2_DI	47,93	30,24	8,932	4,351	0,1864	0,1439
AP_L	20,06	24,59	1,281	1,164	0,0638	0,0473
DIST_	11,34	11,87	0,911	0,751	0,0803	0,0633
PAUSE	7,236	11,82	0,354	0,641	0,0490	0,0543
DUR	7,295	5,941	3,044	1,941	0,4173	0,3268
ENERG	18,41	6,402	13,71	4,750	0,7446	0,7420

Tabelle 3.10: Normierung der Fehler

Datei	RMS-Amplitude (in dB)
dlf950728.1200.n1	-17.23
dlf950728.1200.n2	-16.79
...	...
dlf951121.1200.n1	-22.15
dlf951121.1200.n2	-22.06
...	...

Tabelle 3.11: Ausgewählte Energiewerte von Tag 1 und Tag 2

3.4.4. Spektrum und RMS-Amplitude

Um einen Grund für diese quantitativen Unterschieden zwischen den Tagen zu finden, werden die eigentlichen Sprachsignale noch einmal untersucht. Abbildung 3.7 auf der nächsten Seite zeigt die unterschiedlichen Spektren der Datensätze von Tag 1 und Tag 2. Sehr gut ist ein Abkippen des Spektrums von Tag 1 bei 8 kHz zu sehen. Zusätzlich ist dessen Verlauf auch in den mittleren Frequenzen unruhiger und liegt ca. 5 dB unter dem der Datensätze von Tag 2.

Zur Bestätigung dieser Beobachtung wird die RMS-Amplitude der Signalpegel berechnet (Tabelle 3.11). Diese bestätigt die geschätzte Differenz zwischen den Amplituden.

Eine vergleichende Betrachtung der Zeitverläufe beider Tage sowie nochmaliges Probehören lässt nur den Schluss zu, dass an beiden Tagen unterschiedliche Aufnahmebedingungen beim Aufzeichnen der Signale geherrscht haben müssen. Die Signale von Tag 1 sind normiert, d. h. maximal laut. Zusätzlich scheint ein

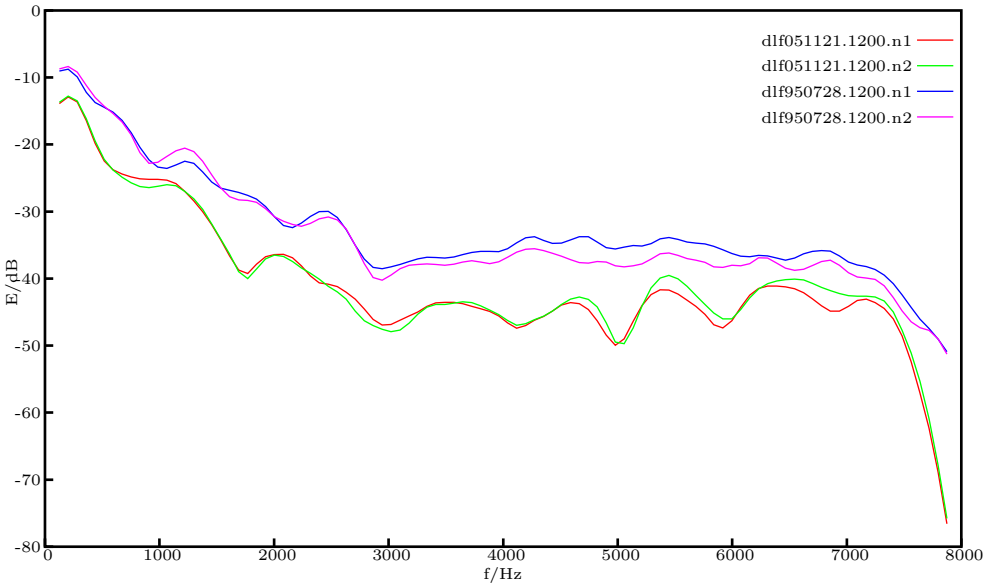

Abbildung 3.7: Vergleich der Spektren von Tag 1 und Tag 2

Tiefpass bei 8 kHz eingesetzt worden zu sein.

3.4.5. Zusammenfassung

Die unterschiedlichen Fehlerwerte beider Tage haben keine qualitativen Ursachen, d. h. die Datensätze von Tag 1 sind für das Netz nicht schwerer zu lernen als die von Tag 2.

Hauptsächliche Ursache sind die unterschiedlichen Bedingungen, unter denen die Aufzeichnung scheinbar stattfand. Die damit einhergehenden Unterschiede der Fehlerwerte des Parameters `ENERG` machen ungefähr zwei Drittel des Gesamtfehlerunterschiedes aus.

Der verbleibende Fehler kann damit jedoch nicht erklärt werden. Größtenteils entsteht dieser durch `T2_DI`, ferner auch durch `PAUSE` und `AA`. Diese Parameter sind jedoch von Aufnahmebedingungen unabhängig, womit Unterschiede nur über unterschiedliche Sprecher oder Sprechweisen erklärt werden können. Es war nicht möglich, diese Frage durch perzeptive Experimente zweifelsfrei zu entscheiden.

Das gefundene Problem der unterschiedlichen Wichtung der Fehler der einzelnen Parameter im Endergebnis und während des Trainings in Abhängigkeit der Anzahl „echter“ Datensätze erschwert die Gesamtbewertung des Netzes.

Um trotzdem aussagekräftige Ergebnisse zu erhalten, werden im weiteren die verschiedenen Ausgänge des IGM nur noch separat trainiert.

3.5. Vorhandene Eingabeparameter

Die vom Netz zu repräsentierenden Ausgabeparameter lassen sich in drei Gruppen einteilen:

1. *Akzentinformationen* in `AA`, `T1_DI` und `T2_DI`. Sie sind nur gesetzt, falls ein Akzent an der betreffenden Stelle im Trainingsmaterial enthalten ist.

2. *Phraseninformationen* in `AP_L`, `DIST_` und `PAUSE`. Sie sind nur gesetzt, wenn eine Phrase betont bzw. eine größere Pause davor markiert ist.

3. *Silbeninformationen* in `DUR` und `ENERG`. Sie sind für jede Silbe vorhanden und vom Fujisaki-Modell unabhängig.

Das Training erfolgt über alle Datensätze, unabhängig davon, ob z. B. überhaupt ein Phrasenstart vorliegt. Eine Verbesserungsmöglichkeit ist die Einschränkung der zu trainierenden Datensätze. Dabei ist zu überprüfen, ob diese Entlastung des Netzes eine Verringerung der Fehlerwerte mit sich bringt sowie durchgeführt werden kann, ohne zusätzliches Expertenwissen zur Verfügung zu stellen. Falls dies möglich ist, zeigt sich damit neben der Netzwerkstruktur selbst ein weiterer Ansatzpunkt, das IGM hinsichtlich der Geschwindigkeit von Lern- und Kannphase zu optimieren.

Akzentinformationen Alle Silben der Datenbasis sind im Feld `INTONEME` von Mixdorff auf ihre Akzentuierung gelabelt. Jeder Wert ungleich Null gibt an, dass sich an dieser Stelle ein Akzentkommando befindet. Diese Information kann im Vorfeld herangezogen werden, um das Netztraining nur auf akzentuierte Silben zu beschränken, da eine eindeutige Zuordnung von Akzentkommandos möglich ist. Nach Tabelle 3.7 auf Seite 26 existieren zwar einige Akzente, bei denen `T1_DI` oder `T2_DI` nicht gesetzt sind, dies wird jedoch auf Grund der geringen Anzahl hier vernachlässigt.

Phraseninformationen Phrasenkommandos treten nur am Anfang von markierten Phrasen auf. Nur ca. 80 % aller Phrasen tragen ein Phrasenkommando, wobei der Parameter `PAUSE` (Tabelle 3.8 auf Seite 26) bei etwa 60 % aller Phrasenkommandos gesetzt ist.

Name	Anzahl	*SSE*	MSE_{Ges}	$MSE_{Ges,korr}$
`AA` Alle Datensätze	11367	6,744		$0{,}593 \cdot 10^{-3}$
`AA` Akzentkommandos	2600	6,864	$2{,}640 \cdot 10^{-3}$	$0{,}604 \cdot 10^{-3}$
`T1_DI` Alle Datensätze	11367	34,83		$3{,}065 \cdot 10^{-3}$
`T1_DI` Akzentkommandos	2600	35,60	$13{,}69 \cdot 10^{-3}$	$3{,}132 \cdot 10^{-3}$
`T2_DI` Alle Datensätze	11367	47,26		$4{,}158 \cdot 10^{-3}$
`T2_DI` Akzentkommandos	2600	48,87	$18{,}80 \cdot 10^{-3}$	$4{,}300 \cdot 10^{-3}$

Tabelle 3.12: Training nur ausgewählter Akzent-Datensätze

Silbeninformationen Dauer und Energie einer Silbe sind für jeden Datensatz angegeben. Hier kann keine weitere Einschränkung der zu trainierenden Datensätze vorgenommen werden.

Die Abbildungen A.1 bis A.4 auf den Seiten 43 bis 44 im Anhang geben einen Eindruck von der Häufigkeitsverteilung über den Wertebereich der einzelnen Ausgabeparameter, wobei die Datensätze für die Darstellung der Akzent- und Phrasekommandoparameter jeweils auf solche beschränkt sind, wo auch wirklich ein Akzent- bzw. Phrasenkommando vorliegt.

3.5.1. Akzentkommandos

Tabelle 3.12 vergleicht die Fehlerwerte der Akzentparameter eines Trainings aller Datensätze des zweiten Tages mit denen eines Trainings, das auf die Datensätze mit Akzentkommandos beschränkt wurde. Die dritte Spalte MSE_{korr} gibt den *MSE* an, wenn als Anzahl die Gesamtheit aller Datensätze des zweiten Tages zugrunde gelegt wird.

Interessanterweise sind die Fehlerwerte bei einem Training mit reduzierter Anzahl ausnahmslos höher. Wegen der sehr kleinen Abweichung ist die Ursache jedoch wahrscheinlich nur die andere Auswahl an Datensätzen, die den Durchläufen zugrunde liegen.

3.5.2. Phrasenkommandos

In Tabelle 3.13 auf der nächsten Seite sieht man die Ergebnisse für ein Training, bei dem die Auswahl in verschiedenen Graden auf die Datensätze mit Phrasenkommandos beschränkt wurde:

- *Phrasenstarts* beschränkt die Auswahl auf alle im Originalmaterial gelabelten Phrasenstarts in `PHR_S`.

Name	Anzahl	SSE_{Ges}	MSE_{Ges}	$MSE_{Ges,korr}$
`AP_L` Alle Datensätze	11367	11,39		$1{,}002 \cdot 10^{-3}$
`AP_L` Phrasenstarts	1177	11,19	$9{,}508 \cdot 10^{-3}$	$0{,}984 \cdot 10^{-3}$
`AP_L` Phrasenkommandos	891	8,820	$9{,}899 \cdot 10^{-3}$	$0{,}776 \cdot 10^{-3}$
`DIST_` Alle Datensätze	11367	7,577		$0{,}667 \cdot 10^{-3}$
`DIST_` Phrasenstarts	1177	7,495	$6{,}368 \cdot 10^{-3}$	$0{,}659 \cdot 10^{-3}$
`DIST_` Phrasenkommandos	891	6,748	$7{,}573 \cdot 10^{-3}$	$0{,}594 \cdot 10^{-3}$
`PAUSE` Alle Datensätze	11367	5,183		$0{,}456 \cdot 10^{-3}$
`PAUSE` Phrasenstarts	1177	5,327	$4{,}526 \cdot 10^{-3}$	$0{,}469 \cdot 10^{-3}$
`PAUSE` Phrasenkommandos	891	5,079	$5{,}700 \cdot 10^{-3}$	$0{,}447 \cdot 10^{-3}$
`PAUSE` Nur Pausen	493	3,501	$7{,}102 \cdot 10^{-3}$	$0{,}308 \cdot 10^{-3}$

Tabelle 3.13: Training nur ausgewählter Phrasen-Datensätze

- *Phrasenkommandos* verwendet nur Datensätze, die ein Fujisaki-Phrasenkommando haben.
- *Nur Pausen* schränkt die Auswahl weiter ein, indem nur Phrasen mit einer Pause ungleich Null in das Training mit einbezogen werden.

Es ist keine echte Verbesserung erkennbar, wenn man das Training nur mit den Phrasenstarts durchführt. Erst eine Beschränkung auf die Fujisaki-Phrasenkommandos bringt eine Verbesserung.

Gleiches gilt für `PAUSE`, wo eine Einschränkung auf die Datensätze mit Phrasenpause eine deutliche Verbesserung bringt (Abbildung 3.8 auf der nächsten Seite). Sehr gut ist in der Abbildung sichtbar, dass das Netz erst ohne „Nicht-Pausen“ in der Lage ist, den Verlauf der Häufigkeitsverteilung mit den zwei lokalen Maxima rudimentär nachzubilden.

3.5.3. Verwendung anderer Aktivierungsfunktionen

Da bei den Versuchen mit allen Phrasenkommandos für `PAUSE` nur etwa reichlich die Hälfte aller Datensätze verschieden von Null ist, könnte die verwendete logistische Aktivierungsfunktion für das zugehörige Ausgabeneuron das Ergebnis negativ beeinflussen. Ähnliches trifft auch für alle anderen Phrasen- und Akzentparameter zu, die beim Training mit allen Datensätzen größtenteils nicht gesetzt sind.

Da der Arbeitspunkt bei der Funktion $f_{tanh} = \tanh(x)$ im Gegensatz zur logistischen Funktion f_{log} bei 0 anstelle von 0,5 liegt, werden einige Versuche

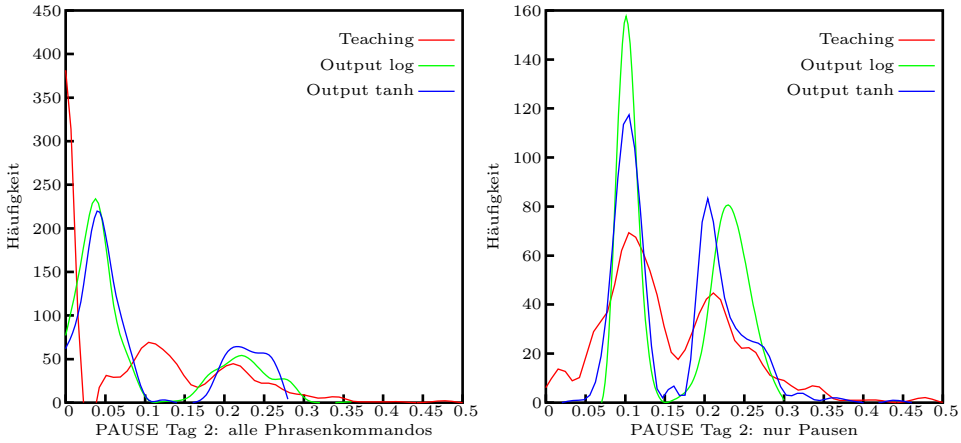

Abbildung 3.8: Histogramme `PAUSE` mit unterschiedlicher Auswahl an Datensätzen und Aktivierungsfunktionen

		$MSE_{Gesamt,korr}$	
Name	Anzahl	Logistisch	Tangens hyp.
`PAUSE` Alle Datensätze	11367	$0{,}456 \cdot 10^{-3}$	$0{,}406 \cdot 10^{-3}$
`PAUSE` Phrasenstarts	1177	$0{,}469 \cdot 10^{-3}$	$0{,}461 \cdot 10^{-3}$
`PAUSE` Phrasenkommandos	891	$0{,}447 \cdot 10^{-3}$	$0{,}450 \cdot 10^{-3}$
`PAUSE` Nur Pausen	493	$0{,}308 \cdot 10^{-3}$	$0{,}247 \cdot 10^{-3}$

Tabelle 3.14: Verschiedene Aktivierungsfunktionen für `PAUSE`

zum Vergleich dieser beiden Alternativen für den Parameter `PAUSE` durchgeführt (Tabelle 3.14).

Das Ergebnis ist in Abbildung 3.8 zu sehen. Die Unterschiede sind nicht signifikant, optisch gesehen ist jedoch die Häufigkeitsverteilung des Versuchs, der nur Pausen in das Training einschließt und als Aktivierungsfunktion des Ausgabeneurons den Tangens hyperbolicus nutzt, am besten.

3.5.4. Zusammenfassung

Die Reduzierung der Anzahl der zu trainierenden Datensätze auf die für den jeweiligen Ausgabeparameter informationstragenden hat keine oder eine nur unwesentliche Verbesserung des Netzergebnisses zur Folge.

Einzig eine Einschränkung der Datensätze für `PAUSE` auf diejenigen, die auch schon im Trainingsmaterial eine Phrasenpause markieren, zeigt Unterschiede.

Allerdings ist hier zu überprüfen, inwieweit die Information, ob eine Pause vorliegt, aus dem Datenmaterial gewonnen werden kann, ohne zusätzliche Eingabeparameter zur Verfügung zu stellen. Die Häufigkeitsverteilung dieses Parameters und die Reaktion des Netzes lässt darauf schließen, dass die Markierung von Phrasenpausen nicht für Werte kleiner 0,15 s durchgeführt wurde.

Der Verwendung einer anderen Aktivierungsfunktion, exemplarisch am Parameter `PAUSE` durchgeführt, zeigt ebenfalls keine Wirkung. Die Nutzung von f_{tanh} anstelle von f_{log} erzeugt allerdings eine optisch passendere Häufigkeitsverteilung für ein Training, das nur auf die Pausen beschränkt ist.

3.6. Zusätzliche Eingabeparameter

3.6.1. Automatisch erzeugte Eingangsdaten

Um den Effekt von zusätzlichen binären Eingabeparametern auf das Trainingsergebnis abzuschätzen, werden einige Testläufe gemacht, bei denen automatisch generierte Eingangsdaten hinzugefügt werden. Dabei soll überprüft werden, ob und in welcher Weise die Auszeichnung von bestimmten Silben mit Hilfe eines binären Eingabeparameters Einfluss auf das Ergebnis hat.

Das Training erfolgt für jeden Ausgang getrennt, um Seiteneffekte ausschließen zu können. Zum Einsatz kommt die Netzstruktur mit 18 und zwölf Neuronen in den verdeckten Schichten, die um ein oder zwei Eingänge erweitert wurde. Zur Auszeichnung der Datensätze dient die Abweichung vom Mittelwert. Dabei werden alle Datensätze markiert, die um mehr als ein bestimmtes Vielfaches der Streuung vom Mittelwert abweichen (Abbildung 3.9 auf der nächsten Seite). Es werden zwei Durchläufe durchgeführt: Im ersten Durchlauf wird nur markiert, ob der Wert genügend weit vom Mittelwert entfernt liegt; im zweiten Durchlauf wird über insgesamt zwei Eingänge zusätzlich die Richtung der Abweichung dem Netz zugeführt.

Tabelle 3.15 auf der nächsten Seite zeigt den Prozentsatz der markierten Datensätze pro Ausgang für ein bestimmtes Vielfaches der Streuung. Durchschnittlich liegen ca. 30 % aller Werte außerhalb der einfachen, 5 % außerhalb der zweifachen und noch 1 % außerhalb der dreifachen Streuung. Zu vermuten wäre, dass die Markierung für die einfache Streuung den Fehler am stärksten reduziert, da dort mehr Datensätze betroffen sind als bei den anderen beiden Möglichkeiten.

Die Ergebnisse der Durchläufe sind in Tabelle 3.16 auf Seite 35 zu sehen. Bis auf ein paar Ergebnisse, die durch ein ungünstiges Training aus dem allgemeinen Trend herausfallen (so z. B. $|3S|$ bei `T2_DI`), trifft die Vermutung zu, dass

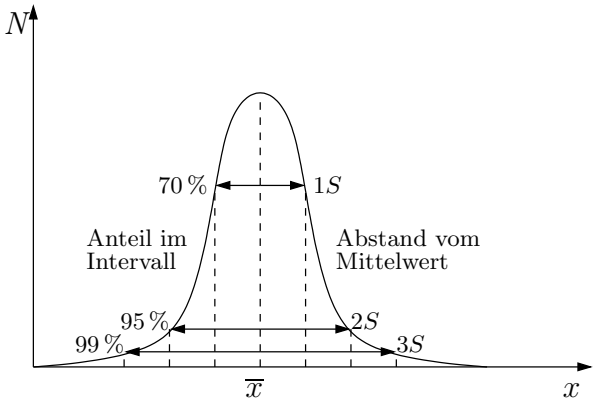

Abbildung 3.9: Qualitative Häufigkeitsverteilung der Parameter

	Datensätze Tag 2 außerhalb					
Parameter	$1S$		$2S$		$3S$	
AA	773	29.7 %	122	4.7 %	29	1.1 %
T1_DI	679	26.1 %	144	5.5 %	14	1.3 %
T2_DI	712	27.4 %	134	5.2 %	20	0.8 %
AP_L	295	33.1 %	38	4.3 %	5	0.6 %
DIST_	245	27.5 %	51	5.7 %	5	0.6 %
PAUSE	83	16.8 %	14	2.8 %	8	1.6 %
DUR	3541	31.1 %	490	4.3 %	102	0.9 %
ENERG	3435	30.2 %	478	4.2 %	101	0.9 %

Tabelle 3.15: Anzahl der markierten Datensätze in Abhängigkeit der Streuung

die größte Verbesserung bei Kennzeichnung außerhalb der einfachen Streuung auftritt.

Interessant ist ebenfalls, dass selbst eine Markierung von unter zehn Datensätzen wie für die Parameter AP_L, DIST_ und PAUSE einen deutlichen Effekt auf die Fehlerwerte hat. Dabei ist es nahezu unerheblich, ob nur ein Betragseingang oder zwei Eingänge verwendet werden. Es ist zu vermuten, dass die Suche nach fehlerhaften Ausreißern in der Datenbasis durchaus eine Verbesserung des generellen Ergebnisses des Netzes zur Folge haben würde.

Um einen Eindruck von der Art der Änderung der Ergebnisse durch die zusätzlichen Eingänge zu bekommen, werden die Histogramme der Verteilung über den Wertebereich der einzelnen Parameter für die unterschiedlichen Mar-

	$MSE_{Gesamt}/10^{-3}$						
Parameter	Orig	$\|1S\|$	$\|2S\|$	$\|3S\|$	$\pm 1S$	$\pm 2S$	$\pm 3S$
AA	2,549	2,138	1,901	2,342	0,863	1,883	2,293
T1_DI	12,73	9,480	9,527	9,700	4,751	7,022	10,04
T2_DI	17,57	13,42	15,57	19,68	5,776	12,09	15,83
AP_L	8,976	5,636	7,938	8,696	4,107	8,026	8,482
DIST_	7,735	5,831	6,006	6,468	2,830	5,283	6,623
PAUSE	5,574	4,512	3,065	3,601	3,670	3,029	3,555
DUR	1,809	1,289	1,548	1,676	0,991	1,512	1,678
ENERG	4,395	3,349	3,192	3,941	1,563	3,152	3,947

Tabelle 3.16: Vergleich unterschiedlicher Eingänge in Abhängigkeit der Streuung

kierungen berechnet (Abbildungen A.5 bis A.12 auf den Seiten 45 bis 48 im Anhang).

Sichtbar ist vor allem bei den Durchläufen mit zwei getrennten Eingängen für die einfache Streuung die Ausbildung von Nebenmaxima, die etwa im Abstand der Eingabeparameter (d. h. z. B. für Eingänge mit Kennzeichnung außerhalb der zweifachen Streuung Nebenmaxima bei $Mittelwert \pm 2S$) liegen. Dies ist ebenfalls teilweise auch bei nur einem Eingang der Fall. Obwohl die Fehlerwerte sinken, passen sich die Häufigkeitsverläufe nicht der Vorgabe an (Abbildung 3.10 auf der nächsten Seite).

Um die Theorie zu überprüfen, dass die Markierung von sehr extremen Werten dem Netz hilft, mit Ausreißern fertig zu werden, wird eine neuer Durchlauf gestartet. Es wird verglichen, wie sich die unterschiedliche Auszeichnung von Datensätzen mit extremen Werten für einen bestimmten Parameter auf den Gesamtfehler auswirkt. Folgende Möglichkeiten werden berücksichtigt:

- das ursprüngliche Netz (Orig)
- die Datensätze, die um mehr als die dreifache Streuung vom Mittelwert abweichen, werden aus Trainings- und Testmenge entfernt (Red)
- ein zusätzlicher Eingang, der gesetzt wird, falls der Datensatz um mehr als die dreifache Streuung vom Mittelwert abweicht ($|3S|$)
- zwei zusätzliche Eingänge, die jeweils gesetzt werden, falls der Datensatz in negativer oder positiver Richtung um mehr als die dreifache Streuung vom Mittelwert abweicht ($\pm 3S$)

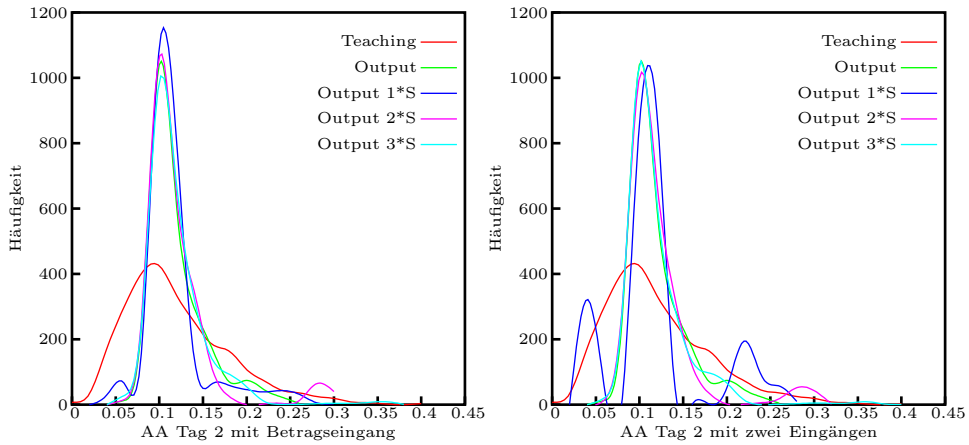

Abbildung 3.10: Histogramme `AA` mit Eingängen in Abhängigkeit der Streuung

Tabelle 3.17 auf der nächsten Seite zeigt die Ergebnisse. Bis auf einige Ausreißer beim Training ist sehr gut zu erkennen, dass alle drei Möglichkeiten ähnliche Verbesserungen der Fehlerwerte nach sich ziehen. Besonders markant ist dies bei `T1_DI`, `DIST_` und `PAUSE`, wo 20 % bis 30 % Reduktion in den MSE-Werten zu verzeichnen sind.

3.6.2. Manuell erzeugte Eingangsdaten

Um den Einfluss von eingängigen zusätzlichen Eingangsgrößen abzuschätzen, werden folgende Versuche mit zusätzlichen akzentbezogenen Eingangsgrößen durchgeführt:

- eine zusätzliche Eingangsgröße, die eine subjektiv als stark empfundene Betonung kennzeichnet
- eine zusätzliche Eingangsgröße, die einen Eigennamen kennzeichnet
- die zwei obigen Eingangsgrößen zusammen

Zur Markierung dieser Daten findet das Programm *mark.pl* Verwendung (Abschnitt C.7 auf Seite 69 im Anhang). Auf Grund des Umfangs der vorliegenden Datensätze wird nur Tag 1 markiert. Es werden ca. 40 % aller Datensätze als subjektiv stark betont und ca. 25 % aller Akzente als Eigennamen markiert.

Tabelle 3.18 auf der nächsten Seite zeigt die Ergebnisse. Für diese einfache Markierungen ist keine Verringerung der Fehlerwerte erkennbar. Scheinbar gibt es für diese zwei intuitiv gewählten Eingangsgrößen keinen direkten Zusammenhang zu den Ausgabeparametern.

	Anzahl		$MSE_{Gesamt}/10^{-3}$				Verbesserung (%)		
Parameter	Tag 2	$3S$	Orig	Red	$\|3S\|$	$\pm 3S$	Red	$\|3S\|$	$\pm 3S$
AA	2600	29	2.561	2.352	2,342	2,293	8.17	8.55	10.5
T1_DI	2569	14	13.13	10.31	9,700	10,04	21.5	26.1	23.5
T2_DI	2587	20	18.73	15.37	19,68	15,83	18.0	-5.07	15.5
AP_L	891	5	9.005	9.449	8,696	8,482	-4.93	3.43	5.81
DIST_	891	5	7.812	6.031	6,468	6,623	22.8	17.2	15.2
PAUSE	493	8	5.576	3.488	3,601	3,555	37.5	35.4	36.2
DUR	11368	102	1.797	1.676	1,676	1,678	6.74	6.73	6.62
ENERG	11368	101	4.375	3.946	3,941	3,947	9.81	9.92	9.78

Tabelle 3.17: Vergleich unterschiedlicher Markierung von $|3S|$

	$MSE_{Gesamt}/10^{-3}$ mit Markierung			
Parameter	keine	Betonung	Namen	beide
AA	5,771	5,681	5,876	5,649
DUR	2,537	2,532	2,566	2,527
ENERG	13,22	12,97	13,04	13,26
T1_DI	15,10	14,85	13,35	14,44
T2_DI	25,96	26,47	26,02	26,71

Tabelle 3.18: Ergebnisse mit manuell markierten Daten

3.6.3. Einzelfalluntersuchung extremer Parameterwerte

Zur exemplarischen Untersuchung von extremen Werten für die Ausgabeparameter werden die Ausreißer bei den Phrasen-Parametern AP_L, PAUSE und DIST_ näher betrachtet.

Phrasenamplitude

Es gibt sechs Datensätze mit einer Phrasenamplitude außerhalb der dreifachen Streuung, Tabelle 3.19 auf der nächsten Seite gibt einen Überblick. Dabei bezeichnet die erste Spalte die Nummer des Datensatzes, der den extremen Wert aufweist. Angeführte Nummern in den Bemerkungen beziehen sich auf andere identische Phrasen, die auf Grund der sich wiederholenden Nachrichten im Korpus existieren (siehe auch [8]).

Die extremen Werte sind ausnahmslos am Beginn von Sätzen zu finden. Größ-

Nr	Wert	Bemerkungen
1472	2.972	Ursache unbekannt
3745	3.485	im Vergleich zu *1986, 4577, 6745, 7331* fehlt das erste Akzentkommando, die Phrasenamplitude ist sehr viel höher (Abbildung 3.11 auf Seite 40)
3823	3.074	im Vergleich zu *2064, 6823* ist der erster Akzent zu niedrig, dafür die Phrasenamplitude zu hoch
4891	3.366	Ursache unbekannt
7645	3.233	im Vergleich zu *5076* ist die erste Akzentamplitude niedriger, dafür die Phrasenamplitude höher
8440	3.065	im Vergleich zu *10728* ist das erste Akzentkommando zu viel

Tabelle 3.19: Ursachen für Extremwerte für `AP_L`

tenteils handelt es sich um Verschiebungen von der Akzent-Ebene in die Phrasenebene, so z. B. weggelassene oder reduzierte Akzentkommandos auf dem ersten Wort. Teilweise ist es jedoch nicht nachzuvollziehen, ob diese Zuordnung fehlerhaft erfolgte oder nicht.

Position der Phrasenkommandos

Die sieben vorhandenen Extremwerte für die Position des Phrasenkommandos (Tabelle 3.20 auf der nächsten Seite) beruhen alle auf einer fehlerhaften Fujisaki-Parameter-Extraktion. Die Spalte *Länge* gibt die Länge der ausgezeichneten Phrase an. Teilweise setzen Phrasenkommandos sehr spät oder erst weit hinter den Phrasen ein, so dass diese Kommandos, falls sie gerechtfertigt sind, eher bei einer späteren Phrase trainiert werden sollten. Andere beginnen schon mehrere Sekunden vor dem Phrasenstart.

Phrasenpausen

Alle untersuchten Sprechpausen länger als 0,8 s (Tabelle 3.21 auf der nächsten Seite) rühren wahrscheinlich von einem Blattwechsel des Sprechers o. ä. her. Dafür spricht auch, dass derart lange Pausen trotz fehlender Motivation teilweise an ähnlichen Stellen vorkommen. Keine der untersuchten langen Pausen ist durch den Inhalt der Meldung gerechtfertigt.

Nr	Wert (s)	Länge (s)	Bemerkungen
1232	-1.482	0,98	das Phrasenkommando befindet sich außerhalb der Phrase
1251	-1.157	1,07	das Phrasenkommando befindet sich außerhalb der Phrase
2102	-1.594	0,62	vergleichbar mit *3861, 6596, 6861*: das Phrasenkommando befindet sich außerhalb der Phrase
5746	-1.421	2,15	vergleichbar mit *4089*: das Phrasenkommando befindet sich in der Mitte der Phrase
6285	2.301	1,36	vergleichbar mit *8945, 9496*: das Phrasenkommando befindet sich weit vor der Phrase
11816	-1.315	1,81	das Phrasenkommando befindet sich in der Mitte der Phrase
12612	4.184	1,38	das Phrasenkommando befindet sich weit vor der Phrase

Tabelle 3.20: Ursachen für Extremwerte für `DIST_`

Nr	Wert (s)	Bemerkungen
77	1,59	
917	1,30	
3479	1,82	
4037	1,87	vergleichbar mit *5694*: 0,19 s
6526	2,40	
6823	2,06	vergleichbar mit *2064*: 0,81 s, *3823*: 1,07 s
9737	1,86	
11746	2,43	vergleichbar mit *11391*: 0,71 s
12536	2,58	
12612	2,36	

Tabelle 3.21: Ursachen für Extremwerte für `PAUSE`

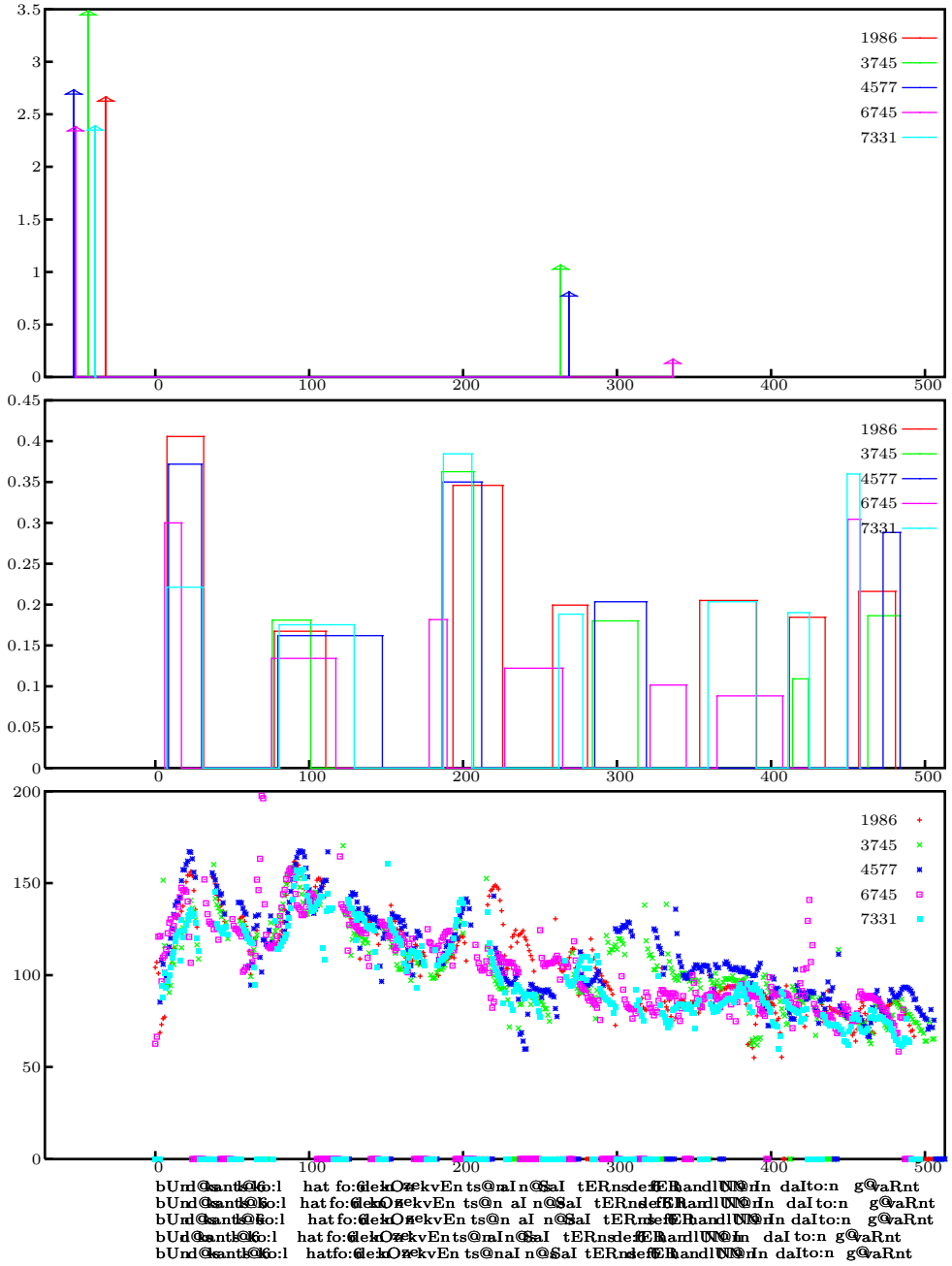

Abbildung 3.11: Beispiel für fehlerhafte Phrasen- und Akzent-Kommandos: „Bundeskanzler Kohl hat vor den Konsequenzen eines Scheiterns der Verhandlungen in Dayton gewarnt.“ – Phrase- und Akzentkommandos, Grundfrequenz

3.6.4. Zusammenfassung

Die Ergebnisse für die automatisch generierten Eingabedaten zeigen die Grenzen einer binären Auszeichnung. Obwohl sie zur Verringerung des Fehlers beiträgt, kann sie nur beschränkt dazu dienen, die bisherigen Probleme des IGM zu beseitigen. Trotz sinkender Fehlerwerte passen sich die Häufigkeitsverläufe nicht der Vorgabe an. Notwendig scheinen vielmehr Eingabeparameter zu sein, die mehr als eine Ja/Nein-Information enthalten.

Zusätzliche einfach motivierte Eingangsgrößen wie Eigennamen und subjektives Betonungsempfinden tragen nicht zu einem besseren Netzergebnis bei.

Selbst die Kennzeichnung oder das Aussortieren von wenigen Datensätzen (z. B. von maximal 1 %) als extrem in einem bestimmten Parameter kann sehr stark zur Verbesserung des Netzergebnisses beitragen.

Die exemplarische Untersuchung von Datensätzen mit extremen Werten in Phrasenkommandoparametern verdeutlicht die Grenzen des Korpus wie auch der Qualität der extrahierten Fujisaki-Parameter. Die Vermutung, dass das Netz diese „Ausreißer“ bei seinem Training gutmütig verkraftet, kann auf Grund der großen Unterschiede in den Fehlerwerten bei einer Markierung solcher Datensätze nicht bestätigt werden.

Obwohl das Training inklusive dieser extremen Werte das Netzergebnis für die restlichen Datensätze wahrscheinlich nicht stört, macht dieses jedoch die objektive Bewertung der Netzleistung schwer möglich.

Bei der Analyse identischer Sätze und Phrasen sind ebenso Inkonsistenzen in der Phrasen- und Satz-Auszeichnung ausgefallen. Bei ansonsten identischen Sätzen sind teilweise z. B. aufeinander folgende Phrasen zu einer Phrase zusammengefasst. Auch auf Satzebene sind die Markierungen nicht einheitlich.

Es kann angenommen werden, dass bei den Parametern der Akzentkommandos ähnliche Probleme wie bei den Phrasenkommandos existieren. Da teilweise mehrere Silben überspannende Akzentkommandos auf zwei Silben aufgeteilt wurden, sind auch da Inkonsistenzen zu erwarten.

Eine manuelle Bereinigung der Datenbasis ist notwendig, um dem Netz weniger widersprüchliche Informationen zu liefern. Dabei sollte vor allem die vorhandene Redundanz von mehrfach vorhandenen Phrasen, Sätzen und teilweise Meldungen ausgenutzt werden. Ein Vergleich schon bei der Erzeugung der Fujisaki-Parameter könnte helfen, die Verlässlichkeit der extrahierten Werte zu überprüfen.

4. Zusammenfassung und Ausblick

Diese Studienarbeit beschäftigte sich mit der Optimierung des im Dresdner Sprachsynthesesystems (DRESS) implementierten Modells zur Prosodiegenerierung. Der Fokus lag dabei auf der Evaluierung der bestehenden Netzstruktur des zum Einsatz kommenden neuronalen Netzes und der verwendeten Datenbasis.

Das bisher verwendete Netzwerk ist überdimensioniert und kann ohne Probleme verkleinert werden. Zweistufige Netzwerke sind jedoch nicht geeignet, um die benötigte Rechenleistung zu reduzieren.

Die beobachteten Unterschiede in der Netzleistung zwischen den beiden Aufnahmetagen der Datenbasis haben keine qualitativen Ursachen, sondern basieren auf den unterschiedlichen Aufnahmebedingungen. Von einem gemeinsamen Training wird jedoch selbst nach einer passenden Normierung abgeraten.

Auf Grund des unterschiedlichen Umfangs der im IGM integrierten Phrasen-, Akzent- und Silbenparameter ist das Netzergebnis, über alle Ausgänge betrachtet, nicht aussagekräftig. Dabei wird der Fehleranteil der Silbeninformationen Energie und Dauer überproportional stark betont.

Die Kennzeichnung oder das Aussortieren von Datensätzen mit sehr extremen Werten der zu schätzenden Parameter führt zu einer deutlichen Verbesserung des Netzergebnisses. Diese Datensätze machen eine objektive Bewertung des Netzergebnisses nur schwer möglich, trotzdem wird jedoch das Training der restlichen Datensätze wahrscheinlich nicht beeinflusst.

Da die dabei exemplarisch untersuchten Werte größtenteils auf ungenaue oder fehlerhafte Markierung zurückgehen, scheint eine manuelle Bereinigung der Datenbasis notwendig. Auch eine Untersuchung der extrahierten Fujisaki-Parameter und deren Abbildung auf die Silbenebene der Datenbasis könnte helfen, die Netzleistung zu verbessern.

Für eine weitergehende Optimierung ist eine Erarbeitung von zusätzlichen Eingabeparametern notwendig. Denkbar wären z. B. semantische Parameter, die das Auftreten unterschiedlicher fokaler Bedingungen besser erklären.

Nur durch die Ausdehnung der berücksichtigten Eingangsinformationen auf bisher nicht beachtete Ebenen der Sprachproduktion kann die Natürlichkeit der Prosodiegenerierung signifikant gesteigert werden.

A. Diagramme

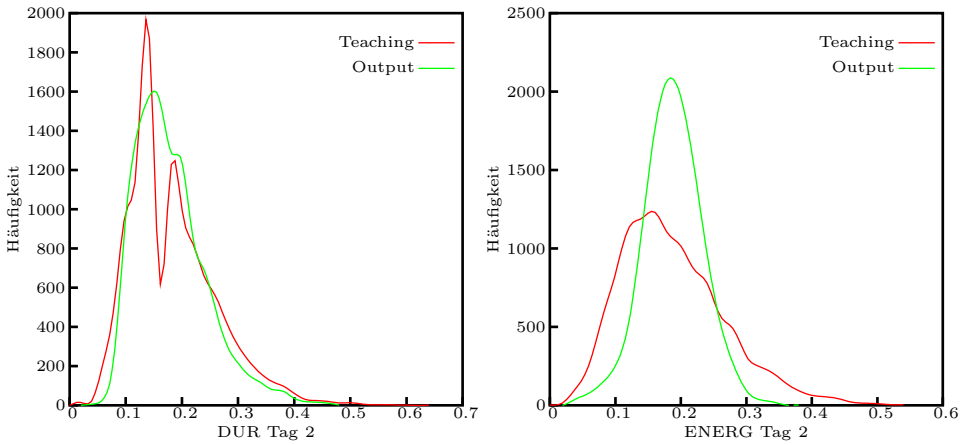

Abbildung A.1: Histogramme `DUR` und `ENERG`

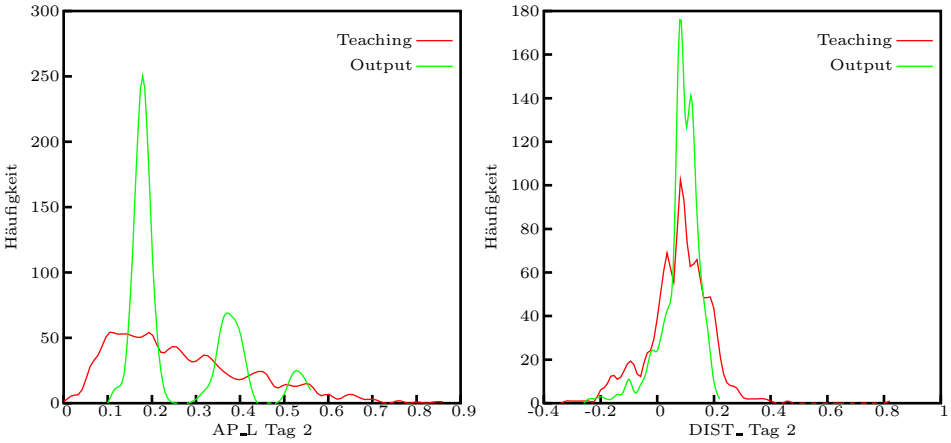

Abbildung A.2: Histogramme `AP_L` und `DIST_`

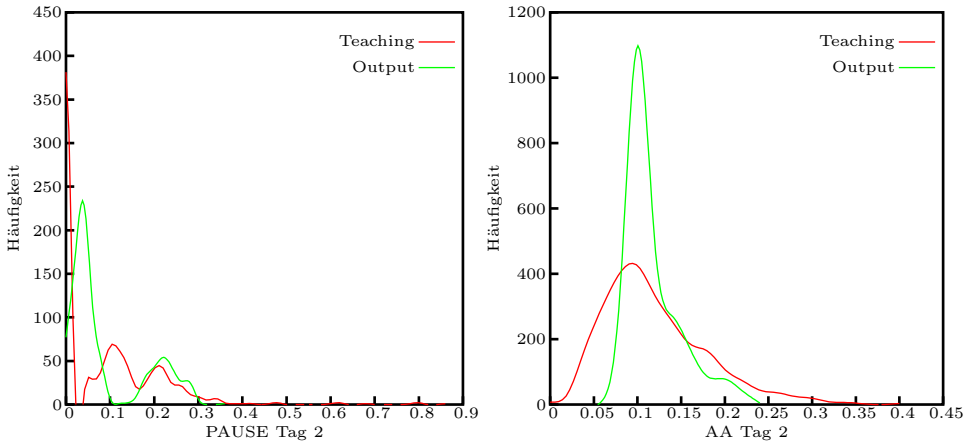

Abbildung A.3: Histogramme `PAUSE` und `AA`

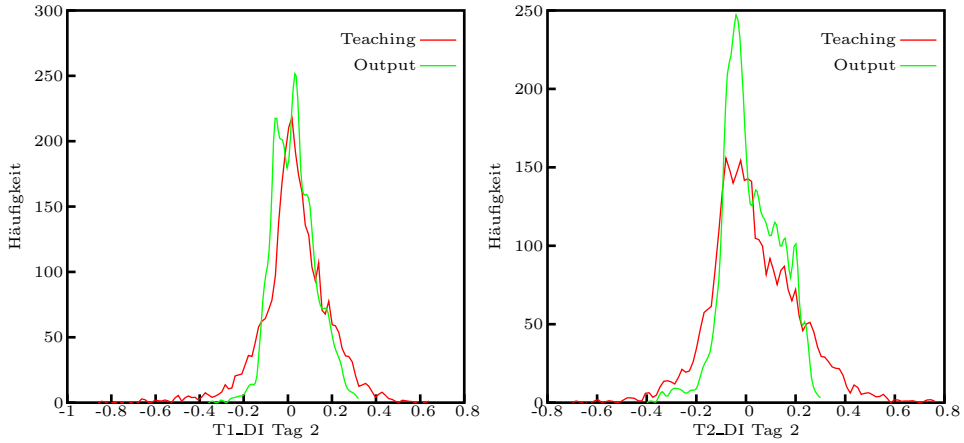

Abbildung A.4: Histogramme `T1_DI` und `T2_DI`

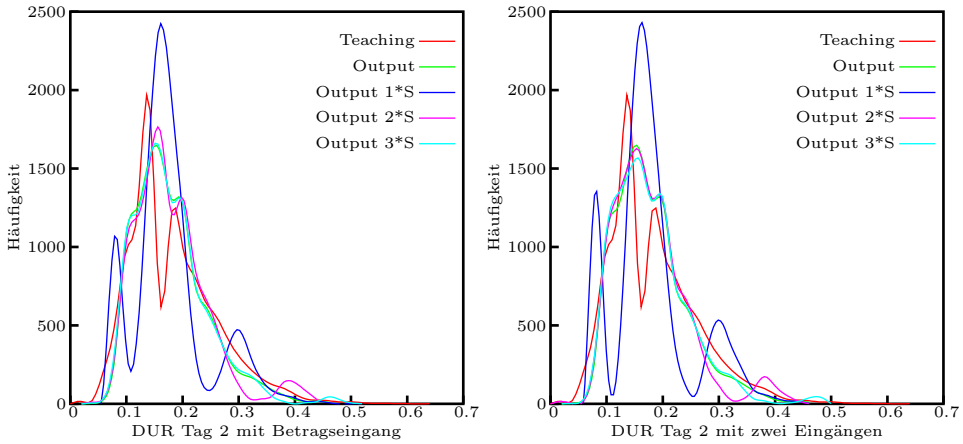

Abbildung A.5: Histogramme `DUR` mit Eingängen in Abhängigkeit der Streuung

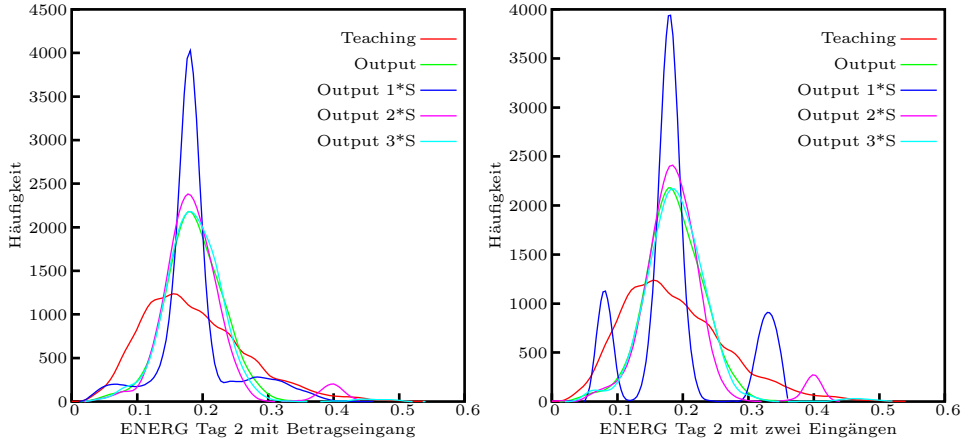

Abbildung A.6: Histogramme `ENERG` mit Eingängen in Abhängigkeit der Streuung

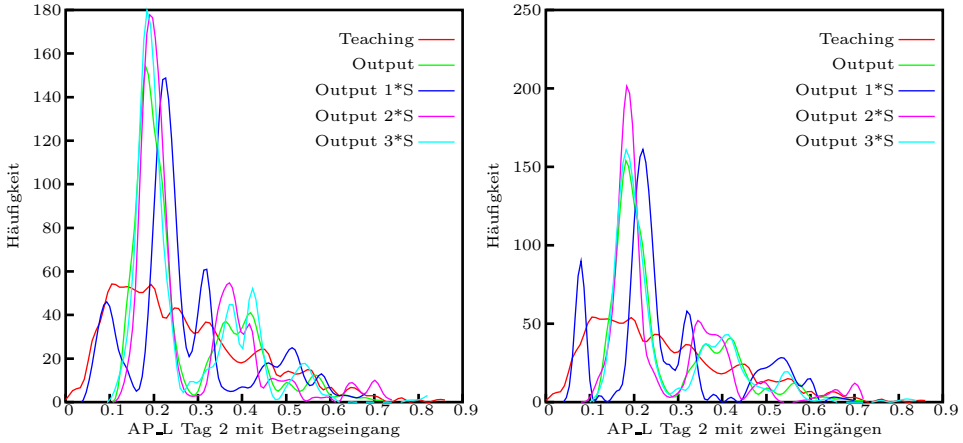

Abbildung A.7: Histogramme `AP_L` mit Eingängen in Abhängigkeit der Streuung

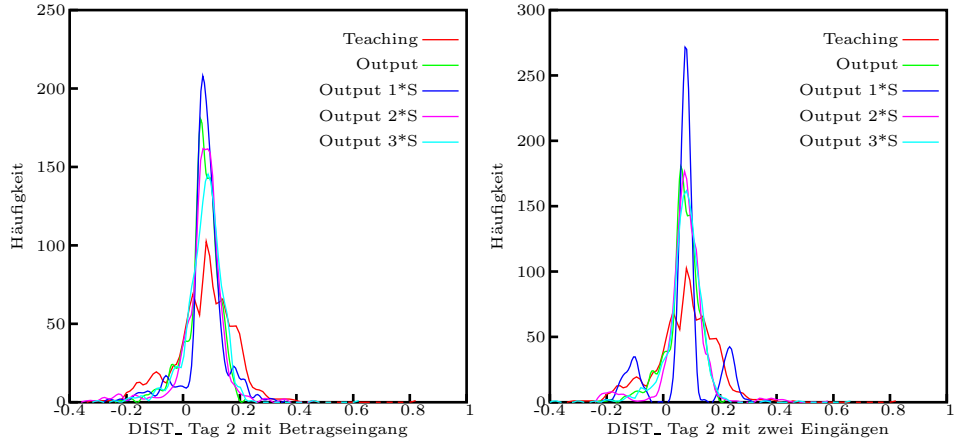

Abbildung A.8: Histogramme `DIST_` mit Eingängen in Abhängigkeit der Streuung

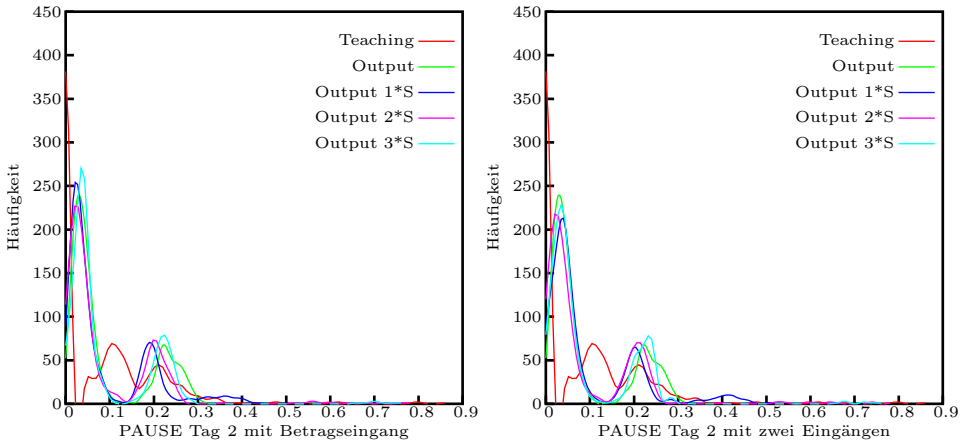

Abbildung A.9: Histogramme `PAUSE` mit Eingängen in Abhängigkeit der Streuung

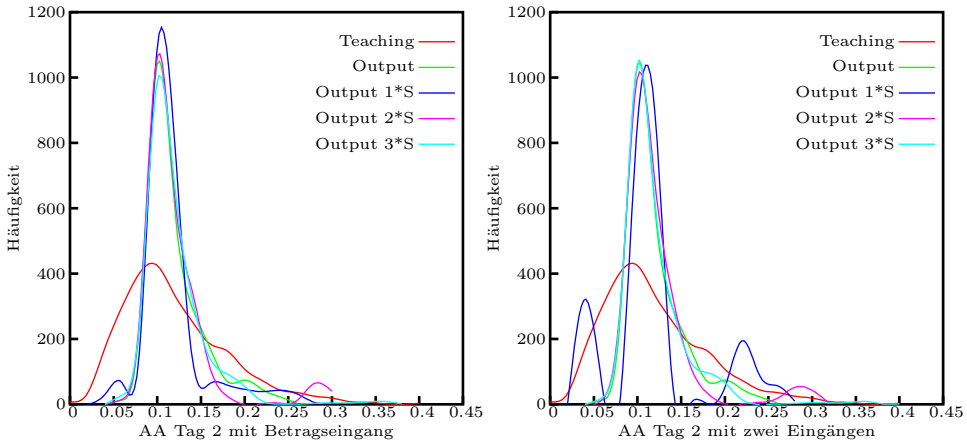

Abbildung A.10: Histogramme `AA` mit Eingängen in Abhängigkeit der Streuung

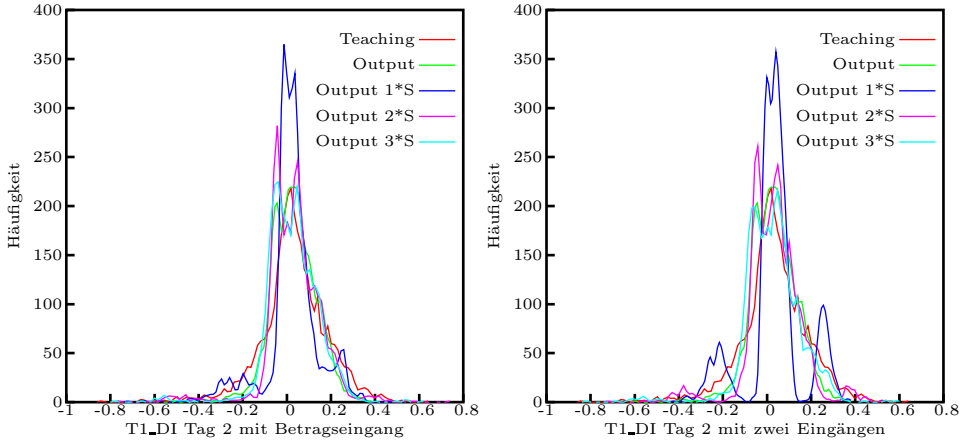

Abbildung A.11: Histogramme `T1_DI` mit Eingängen in Abhängigkeit der Streuung

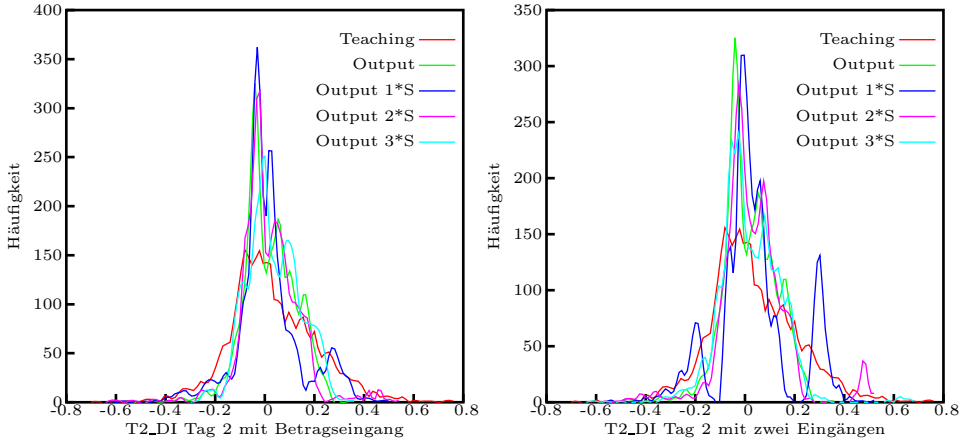

Abbildung A.12: Histogramme `T2_DI` mit Eingängen in Abhängigkeit der Streuung

B. Feldnamen der Datenbasis

Die Daten des Stuttgarter Nachrichtenkorpus liegen in Form einer Tabelle vor. Die Felder sind durch Tabulator getrennt, die erste Zeile enthält die Feldbezeichnungen. Nur ein Teil dieser Daten wird zum Netztraining verwendet, neben den MFGI-Daten sind dies vor allem die extrahierten Fujisaki-Parameter. Nachfolgend werden die wichtigsten Felder mit ihrer Funktion vorgestellt. Neben den beschriebenen Feldern existieren weitere, die jedoch für das Netztraining und die Vorhersage prosodischer Merkmale von Mixdorff als nicht bedeutsam eingestuft wurden.

In den Tabellen B.1 bis B.3 finden sich die Eingangsgrößen des IGM, sowie in Tabelle B.6 die Ausgangsgrößen. Tabelle B.4 enthält weitere Daten, die nicht zum Netztraining verwendet werden. In der zweiten Spalte findet sich der Wert, der zur Normierung der Eingangsgrößen für das neuronale Netz verwendet wurde. Tabelle B.5 enthält die Felder, die sich mit Informationen über die Daten selbst beschäftigen.

Zusätzlich gibt Tabelle B.7 einen Überblick über die verwendeten Part-Of-Speech-Bezeichnungen (POS). Die Nummer in der zweiten Spalte gibt den zum Tag gehörenden Wert des Feldes `POS_D` vor der Normierung für das MFGI an.

Feldname	M	Beschreibung
`BI_SY`	5	Stärke der syntaktischen Grenze links. Werte von 0 bis 4. Gesetzt nur am Anfang von Worten. Entspricht `BI_R_`, um 1 Zeitschritt verzögert.
`PPOS_`	38	Vorhergehende POS-Klasse bei einem Akzent. Ausschlaggebend ist der Wert beim vorherigen Akzentkommandos in `ACC`. Bei Phrasenstart wird zwar der „zwischengespeicherte“ Wert gelöscht, bis zum nächsten Akzentkommando ändert sich jedoch nichts. Erst dort schlägt die Löschung dann durch. Diese Eigenheit der Markierung hat jedoch keinen Effekt auf die Trainingsleistung.

Tabelle B.1: Eingangs-Daten aus einem vorhergehenden Datensatz

Feldname	M	Beschreibung
SYLS_	12	Anzahl Silben im aktuellen Wort. Zusammengesetzte Wörter werden insgesamt gezählt.
ACC	2	Im Korpus als akzentuiert gelabelt. Ein Wert von 1 markiert einen Hauptakzent, ein Wert von 0,5 bei zusammengesetzten Worten einen Nebenakzent (wurde scheinbar von Mixdorff hinzugefügt, stimmt teilweise nicht mit den anderen Feldern überein).
BND	2	Im Korpus als Grenzton gelabelt. Hier befindet sich ein Grenzton und signalisiert ein Phrasenende.
BI_R_	5	Break-Index (Stärke der syntaktischen Grenze) rechts. Gesetzt nur am Ende eines Wortes. Entspricht BI_SY, eine Silbe weiter.
SYL_P	32	Anzahl der Silben in der vorherigen Phrase. Nur bei Phrasenstart in PHR_S gesetzt.
I_IN_	11	Index der Silbe im Wort. Beginnt mit 0.
I_PHR	12	Index der Phrase im Satz. Beginnt mit 0.
I_SEN	9	Index des Satzes im Absatz. Beginnt mit 0.
N_ON	4	Anzahl der Phoneme im Onset.
STREN	3	Silbenstärke. Dabei bedeutet 0 unbetont, 1 betont, aber nicht akzentuiert und 2 betont mit Akzentkommando.
SCHWA	2	Silbenkern ist Schwa-Laut (@).
INTON	3	Intonem-Art. Normiert auf 3. Dabei entspricht -1 keinem Akzent, 0 einem deklarativen finalen Akzent I↓, 1 einem nonfinalen Akzent N↑ und 2 einem nonfinalen Akzent vor einer Phrasengrenze C↑.
OR_DUR	1	Summe der durchschnittlichen Dauern der Laute der gesamten Silbe (Laute haben in Onset/Reim unterschiedliche Dauern).
O_DUR	1	Summe der durchschnittlichen Dauern der Laute im Onset.
R_DUR	1	Summe der durchschnittlichen Dauern der Laute im Reim.
POS_D	38	POS-Klasse des Wortes.
PHR_S	2	Start einer neuen Phrase.
SENT_	2	Start eines neuen Satzes.
PARA_	2	Start eines neuen Absatzes.

Tabelle B.2: Eingangs-Daten

Feldname	M	Beschreibung
PAA_	2	Vorhergehende Akzent-Amplitude. Übernahmebedingung wie bei PPOS_.
PAP_L	4	Vorhergehende Phrasen-Amplitude. Nur bei Phrasenstart in PHR_S gesetzt.
PDIST	5	Vorhergehender Abstand des Phrasenkommandos vom segmentalen Onset der Phrase. Nur bei Phrasenstart in PHR_S gesetzt.

Tabelle B.3: Eingangs-Daten aus einem vorhergehenden Ausgangs-Datensatz

Feldname	Beschreibung
ACC_CMD	Akzent-Kommando. Geht einher mit AA ungleich 0.
T_ON	Beginn der Silbe im Absatz.
T_OFF	Ende der Silbe im Absatz.

Tabelle B.4: Eingangs-Daten, die nicht zum Netztraining verwendet werden

Feldname	Beschreibung
NUM	Aktuelle Datensatznummer. Sie liegt im Bereich von 1 bis 1 783 für Daten vom 28.07.1995 und im Bereich von 1 784 bis 13 151 für Daten vom 21.11.1995.
FILENAME	Dateiname der Nachricht. Er enthält den Sender (Deutschlandfunk), Datum und Zeit der Ausstrahlung sowie eine fortlaufende Nachrichtennummer.
WORD	Wort in normaler Schreibweise. Umlaute sind latex-mäßig mit Gänsefüßchen gekennzeichnet, z.B. "a=ä, "s=ß. Die Länge wurde auf 19 Zeichen beschränkt, einige Worte sind deshalb abgeschnitten.
SMPA	Lautumschrift in SAMPA-Notation.
SMPA_CHN	Lautumschrift wurde im Laufe der Dissertation bzw. Habilitation von Mixdorff manuell angepasst. Enthält 1 für Änderung, # für Originalversion.
SMPA_NEW	Von Mixdorff geänderte Lautumschrift in SAMPA-Notation, falls das Feld SMPA_CHN gesetzt ist.

Tabelle B.5: Meta-Daten

Feldname	M	Beschreibung
AA	2	Akzent-Amplitude.
T1_DI	1	Zeitpunkt des Anschaltens des Akzentkommandos.
T2_DI	1	Zeitpunkt des Abschaltens des Akzentkommandos.
AP_L	4	Phrasen-Amplitude. Nur bei Phrasenstart in PHR_S gesetzt.
DIST_	5	Abstand des Phrasenkommandos vom segmentalen Onset der Phrase. Nur bei Phrasenstart in PHR_S gesetzt. In s.
PAUSE	3	Länge der Pause vor der aktuellen Phrase. Nur bei Phrasenstart in PHR_S gesetzt.
DUR	1	Silben-Länge.
ENERG	8000	Mittlere Silbenenergie.

Tabelle B.6: Ausgangs-Daten

Tag	Nr	Beschreibung
ADJA	8	attributives Adjektiv
ADJD	25	adverbiales oder prädikatives Adjektiv
ADV	22	Adverb
APPR	11	Präposition
APPRART	10	Präposition mit Artikel
APPO		Postposition
APZR		Zirkumposition
ART	4	bestimmter oder unbestimmter Artikel
CARD	30	Kardinalzahl
FM		Fremdsprachliches Material
ITJ		Interjektion
KOUI		unterordnende Konjunktion mit „zu“ und Infinitiv
KOUS	21	unterordnende Konjunktion mit Satz
KON	15	nebenordnende Konjunktion
KOKOM	34	Vergleichspartikel, ohne Satz
NN	23	normales Nomen
NE	29	Eigennamen
PDS	26	substituierendes Demonstrativpronomen

Tabelle B.7: Part-Of-Speech-Tags (nach [14])

Tag	Nr	Beschreibung
PDAT	6	attribuierendes Demonstrativpronomen
PIS	17	substituierendes Indefinitpronomen
PIAT	7	attribuierendes Indefinitpronomen ohne Determiner
PIDAT	1	attribuierendes Indefinitpronomen mit Determiner
PPER	5	irreflexives Personalpronomen
PPOSS		substituierendes Possessivpronomen
PPOSAT	2	attribuierendes Possessivpronomen
PRELS	3	substituierendes Relativpronomen
PRELAT	24	attribuierendes Relativpronomen
PRF	28	reflexives Personalpronomen
PWS	35	substituierendes Interrogativpronomen
PWAT		attribuierendes Interrogativpronomen
PWAV		adverbiales Interrogativ- oder Relativpronomen
PAV	18	Pronominaladverb
PTKZU	9	„zu“ vor Infinitiv
PTKNEG	36	Negationspartikel
PTKVZ	37	abgetrennter Verbzusatz
PTKANT		Antwortpartikel
PTKA		Partikel bei Adjektiv oder Adverb
TRUNC	33	Kompositions-Erstglied
VVFIN	14	finites Verb, voll
VVIMP		Imperativ, voll
VVINF	20	Infinitiv, voll
VVIZU	27	Infinitiv mit „zu“, voll
VVPP	19	Partizip Perfekt, voll
VAFIN	13	finites Verb, aux
VAIMP		Imperativ, aux
VAINF	31	Infinitiv, aux
VAPP	12	Partizip Perfekt, aux
VMFIN	16	finites Verb, modal
VMINF	32	Infinitiv, modal
VMPP		Partizip Perfekt, modal
XY		Nichtwort, Sonderzeichen enthaltend

Tabelle B.7: Part-Of-Speech-Tags (fortgesetzt)

C. Programme

Dieses Kapitel gibt einen Überblick über die während dieser Studienarbeit entwickelten Programme und Skripte.

Sie sollten auf jedem einigermaßen POSIX-kompatiblen Unix-System ausführbar sein, getestet wurde unter SuSE-Linux 9.0 und 9.1.

Die Programme sind relativ flexibel einsetzbar, können jedoch für spezielle Aufgaben auch leicht modifiziert werden. Der Programmcode ist durchgehend dokumentiert.

Im Gegensatz zu den teilweise vorgefundenen Programmen auf den Rechnern des Computerkabinettes, die größtenteils nur in kompilierter Form vorlagen und auf Grund ihrer starren Struktur schlecht erweiterbar waren, wurden folgende Ziele für die zu entwickelnden Programme formuliert:

- vollständig in einer der Skriptsprachen *Perl*, *Awk* oder *Bash* geschrieben
- modularer Aufbau, um eine leichte Anpassung an eine spezifische Aufgabenstellung zu ermöglichen
- unkomplizierte Erweiterbarkeit
- grundlegende Dokumentation

Es werden folgende Aufgabengebiete abgedeckt:

- Erzeugung von SNNS-Trainings-Dateien aus der Stuttgarter Tabelle, wobei anstatt von festen Spaltennummern nur Klartext-Namen verwendet werden sollen
- Filterung von Datensätzen
- Automatisches Training von neuronalen Netzen
- Erzeugung beliebiger ebenenweise vollständig verbundener Feedforward-Netzwerke mit wählbaren Ein- und Ausgängen
- Einlesen von SNNS-Ergebnis-Dateien
- Verarbeitung von Fujisaki-Parameter-Daten, z. B. aus PAC-Dateien

C.1. cutsyltab.pl – Zufällige Auswahl von Mustern

Um die Datenbank von 13151 Datensätzen in Muster-Dateien umzuwandeln, die mit SNNS verwendet werden können, existieren die Programme der *cutsyltab*-Reihe.

Neben der Umwandlung ist es damit möglich, Datensätze auf verschiedene Arten zu selektieren sowie die zufällige Auswahl zu steuern. Es existieren Parameter, um

- Datensätze in Abhängigkeit ihrer laufenden Nummer zu selektieren (z. B. zur Beschränkung auf den ersten oder den zweiten Tag),
- zufällig eine bestimmte Anzahl Datensätze in zwei Gruppen auszuwählen (z. B. Trainings- und Testmenge), indem
  - zufällig eine bestimmte Anzahl in die eine Gruppe und der Rest in die andere Gruppe sortiert wird,
  - ein Zufallsstartwert und ein bestimmtes Verhältnis für die Gruppenstärke vorgegeben ist oder
  - die Auswahl der Datensätze für jede Gruppe aus einer Datei gelesen wird,
- die Menge der Eingabeparameter zu beschränken sowie
- spezielle Ausgabeparameter zu selektieren.

Die verwendeten Ein- und Ausgabeparameter sind für das Standardnetz fest im Perl-Quelltext kodiert und können bei Bedarf geändert und ergänzt werden.

Der Aufruf erfolgt mittels

```
cutsyltab.pl Start Anzahl Random [Ausgabe] [Eingabe] < Tabelle
```

Dabei werden folgende Parameter verwendet:

```
Start: Erster auszugebender Datensatz
Anzahl: Anzahl der auszugebenden Datensätze, fehlertolerant
Random: Spezifikation der Auswahl
  Leer: chronologische Ausgabe auf der Standard-Ausgabe
  Auswahl:Name1:Name2: zufällige Auswahl mit
    Dateiname|Anzahl|Seed,Teiler: Zufalls-Konfiguration
      Dateiname: Datei mit Nummern der Datensätze,
        welche in Datei1 und Datei2 geschrieben werden sollen. Eine
        Nummer pro Zeile, die Trennung erfolgt durch eine Leerzeile.
      Anzahl: Anzahl der Datensätze in der ersten Datei
```

```
      Seed,Teiler: Random-Seed, Float für Verhältnis der Anzahl
        Datensätze total/Datei1
    Name1: Name der ersten Datei
    Name2: Name der zweiten Datei
Ausgabe: Leer für die Ausgabe aller Parameter oder Liste in der
  Form (Feld1:Feld2:...)
Eingabe: Leer für die Nutzung aller Eingänge oder Liste in der
   Form (Feld1:Feld2:...)
Tabelle: Stuttgarter Super-Tabelle
```

C.2. only.awk – Filterung von Datensätzen

Um die Menge der Datensätze auf Akzente, Phrasenkommandos o. ä. zu reduzieren, existiert *only.awk*. Es selektiert Datensätze, die in einem bestimmten Feld verschieden von Null sind. Es bietet zusätzlich die Möglichkeit, die Auswahl mittels eines regulären Ausdrucks auf die Datensätze eines Tages zu beschränken. Die Ausgabe erfolgt auf die Standard-Ausgabe und kann damit direkt als Standard-Eingabe für *cutsyltab* verwendet werden.

Der Aufruf erfolgt mittels

```
only.awk Feld [Tag] < Tabelle | ....
```

Dabei werden folgende Parameter erkannt:

```
Feld: Name des Feldes, welches einen gültigen Datensatz
  markiert
Tag: Filterung eines bestimmten Tages, es reicht ein Teil
  des Datums wie ,,9507'' oder ,,1121''
Tabelle: Stuttgarter Super-Tabelle
```

C.3. allbut.awk – Inverse Filterung von Datensätzen

Das Programm *allbut.awk* folgt von Struktur und Aufrufkonventionen *only.awk*, filtert aber genau andersherum. Datensätze, für die ein bestimmtes Feld gesetzt ist, werden nicht mit ausgegeben.

Verwendet wird es z. B. bei der Unterdrückung von Datensätzen, für die ein bestimmtes Feld sehr extreme Werte annimmt. Dazu ist es nur notwendig, ein neues Feld hinzuzufügen, welches als Markierung für solche Datensätze dient.

Der Aufruf erfolgt analog zu *only.awk* mittels

```
allbut.awk Feld [Tag] < Tabelle | ....
```

Dabei werden folgende Parameter erkannt:

```
Feld: Name des Feldes, welches einen ungültigen Datensatz
  markiert
Tag: Filterung eines bestimmten Tages, es reicht ein Teil
  des Datums wie „9507“ oder „1121“
Tabelle: Stuttgarter Super-Tabelle
```

C.4. generatebatch.sh – Erzeugung von Trainings-Programmen

Zur Generierung der Trainings-Skripte für SNNS dient *generatebatch.sh*. Die erzeugten Skripte für den Batchman-Interpreter (Abbildung C.1 auf der nächsten Seite) können hinsichtlich

- der Anzahl der Trainingsiterationen sowie
- der Pfade für Netz-, Muster- und Ausgabedateien

frei konfiguriert werden. Sie trainieren mittels Backpropagation in normalerweise zehn Durchläufen und speichern im Verlauf des Trainings pro Durchlauf gleichmäßig über das gesamte Training verteilt 100 Netze ab, deren Dateinamen die erreichten Fehlerwerte enthalten. Bei jeder Durchlauf werden dabei die Startwerte für die Gewichte und Schwellwerte neu zufällig initialisiert.

Das Handbuch zu SNNS [16] enthält eine Übersicht über den Syntax und die zur Verfügung stehenden Befehle des Batchman-Interpreters.

Die erstellten Netze können leicht von Shell-Skripten ausgewertet werden, da sich alle wichtigen Kenngrößen im Dateinamen wiederfinden:

```
Ausgabe-MaxMSE-TrainMSE-TestMSE-Durchlauf-Zyklus.net
```

Dabei bezeichnet

- *Ausgabe* den frei wählbarer Stamm für den Dateinamen des Netzes,
- *MaxMSE* den Maximalwert von Trainings- und Testmengen-MSE,
- *TrainMSE* den gemittelten Fehler über alle Datensätze des Trainings,
- *TestMSE* den gemittelte Fehler über alle Datensätze der Testmenge,
- *Durchlauf* die Nummer des aktuellen Durchlaufs,
- *Zyklus* die Nummer des aktuellen Trainingsschrittes im Durchlauf.

Der Aufruf erfolgt mittels

```
#!/usr/lib/SNNS/tools/bin/i386-suse-linux/batchman -f
PatternSets             := "syltab-tag2"
NumberOfCycles          := 1500
NetBase                 := "generatenet-24-8"
OutputBase              := "tables/24-8"
NumberOfCaptures        := 100
NumberOfRandomStarts    := 10

Network                 := NetBase + ".net"
TrainingSet             := PatternSets + ".1.pat"
ValidationSet           := PatternSets + ".2.pat"
PrintEvery              := NumberOfCycles div NumberOfCaptures

setSeed ()
loadNet (Network); loadPattern (ValidationSet); loadPattern (TrainingSet)
setInitFunc ("Randomize_Weights", 1.0, -1.0)
setLearnFunc ("Std_Backpropagation")
setShuffle (TRUE); setSubPattern (1, 1, 1, 1)

for i := 1 to NumberOfRandomStarts do
    initNet (); setPattern (TrainingSet)
    print ("Run ", i, ": Training ", PAT, " Patterns")
    for j := 1 to NumberOfCycles do
        trainNet ()
        if j mod PrintEvery == 0 then
            # Statistik für Trainings-Set ausgeben
            testNet ()
            TrainingMSE = MSE
            print ("cycles = ", CYCLES, "  SSE = ", SSE, "  SSEPU = ", SSEPU);
            # Statistik für Validations-Set ausgeben
            setPattern (ValidationSet); testNet (); setPattern (TrainingSet)
            ValidationMSE = MSE
            print ("cycles = ", CYCLES, "  SSE = ", SSE, "  SSEPU = ", SSEPU);
            # Netz speichern
            if ValidationMSE > TrainingMSE then
                MaxMSE = ValidationMSE
            else
                MaxMSE = TrainingMSE
            endif
            saveNet (OutputBase + "-" + MaxMSE + "-" + TrainingMSE + "-" + \
                ValidationMSE + "-" + i + "-" +        CYCLES + ".net")
        endif
    endfor
endfor
```

Abbildung C.1: Beispiel-Programm zur Bewertung eines neuronalen Netzes

```
generatbatch.sh Pattern Zyklen Netz Skript Ausgabe
```

Dabei werden folgende Parameter erkannt:

```
Pattern: Stamm der Dateinamen der Muster-Dateien, wird zu
  „Pattern.1.pat“ und „Pattern.2.pat“ für Trainings- und Test-
  Muster ergänzt
Zyklen: Anzahl Trainingszyklen, die insgesamt zu durchlaufen sind
Netz: Netzname, wird zu „Netz.net“ ergänzt
Skript: Name des zu erzeugenden Skriptes, wird zu „Skript.sbf“
  ergänzt
Ausgabe: Stamm der Ausgabe-Netze, die insgesamt 100-mal während
  des Trainings erzeugt werden, der Name wird zu
  „Ausgabe-MaxMSE-TrainMSE-TestMSE-Durchlauf-Zyklus.net“ ergänzt
```

C.5. generatenet.pl – Erzeugung von neuronalen Netzen

Die Generierung der ebenenweise vollständig verbundenen Feedforward-Netze für SNNS erfolgt mit dem Programm *generatenet.pl*.

Dabei können folgende Aspekte frei konfiguriert werden:

- die Anzahl Schichten verdeckter Neuronen,
- die Neuronenanzahl und -anordnung pro verdeckter Schicht,
- der Netzname,
- die gewünschten Ausgangsgrößen,
- eine Anzahl Eingabeneuronen, die nicht mit dem Netz verbunden werden.

Dabei wird bei der Auswahl der Ausgänge die passende Aktivierungsfunktion beachtet (Tabelle C.1 auf der nächsten Seite). Die nicht mit dem Netz verbundenen Neuronen ermöglichen es, für verschiedene Netze die gleichen Muster-Dateien zu verwenden.

Der Aufruf erfolgt mittels

```
generatenet.pl Name Felder Dummy Größe ...
```

Dabei werden folgende Parameter erkannt:

```
Name: Der Name des zu erzeugenden Netzes
Felder: Leer für die Erzeugung aller Ausgabeneuronen oder Liste
  in der Form (Feld1:Feld2:...)
```

Parameter	Aktivierungsfunktion
AA	Act_Logistic
PAUSE	Act_Logistic
DIST_	Act_TanH_Xdiv2
AP_L	Act_Logistic
DUR	Act_Logistic
ENERG	Act_Logistic
T1_DI	Act_TanH_Xdiv2
T2_DI	Act_TanH_Xdiv2

Tabelle C.1: Aktivierungsfunktionen der Ausgabeneuronen

```
Dummy: Leer für die Verdrahtung aller Eingabeneuronen oder Liste
  der Form (Feld1:Feld2:...) der Eingänge, die nicht angeschlossen
  werden sollen
Größe: Angaben (Breite x Höhe) für die verdeckten Schichten
```

C.6. Perl-Framework

Zur Manipulation und Auswertung der Stuttgarter Korpus-Daten sowie der Ergebnis-Dateien von SNNS wurde ein objektorientiertes Perl-Framework entworfen.

Obwohl es weit davon entfernt ist, perfekt zu sein, leistete es unter anderem gute Dienste für das nachfolgend aufgeführte Programm *mark.pl*.

Das Framework ist objektorientiert aufgebaut und gliedert sich in folgende Klassen:

- *TableFile* importiert Daten aus Tabellen, die mittels Tabulator getrennt sind.
- *ResultFile* importiert Daten aus SNNS-Ergebnis-Dateien.
- *Pac* erleichtert die Arbeit mit Fujisaki-Parametern.
- *Syntax* erleichtert die Arbeit mit den Daten aus der Stuttgarter Tabelle.
- *Data* erleichtert die Arbeit mit SNNS-Ergebnis-Dateien.

C.6.1. Die Klasse TableFile

Die Klasse *TableFile* liest Tabellen, in denen die Felder mittels Tabulator getrennt sind, ein und stellt die Daten wahlweise auch über eine Aliastabelle zur Verfügung.

Methoden

`Object = sub new([Datei])`

Instantiiert ein Objekt der Klasse *TableFile*. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Tabelle, er wird in *file* gespeichert.

`sub load([Datei])`

Lädt eine Tabelle aus einer Datei. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Tabelle, er wird in *file* gespeichert.

`string = sub fieldByNum(Zeile,Spalte)`

Gibt den Wert eines bestimmten Feldes, gegeben durch die *Spalte* in der Tabelle, in einem bestimmten Datensatz, gegeben durch die *Zeile*, zurück. Sowohl Zeilen als auch Spalten beginnen mit 0.

`string = sub fieldByName(Zeile,Name)`

Gibt den Wert eines bestimmten Feldes, gegeben durch seine Spaltenbezeichnung *Name* in der Tabelle, in einem bestimmten Datensatz, gegeben durch die *Zeile*, zurück. Die Zeilen beginnen mit 0. Falls keine Spaltenbezeichnungen verfügbar sind, können diese mittels „Feld-??“ angesprochen werden, hier beginnt die Zählung bei 1.

`string = sub fieldByAliasName(Zeile,Alias)`

Gibt den Wert eines bestimmten Feldes, gegeben durch seinen Alias-Namen *Alias*, in einem bestimmten Datensatz, gegeben durch die *Zeile*, zurück. Die Zeilen beginnen mit 0. Die Aliase werden im Feld *alias* definiert.

`int = fieldAvailable (Name)`

Überprüft, ob eine Spalte *Name* vorhanden ist. Der Rückgabewert ist 1, wenn es eine solche Spaltenbezeichnung gibt und 2, falls es einen gültigen Alias dieses Namens gibt, ansonsten wird 0 zurückgegeben.

Schreib-Lese-Felder

`string file`

Enthält den Dateinamen der Tabelle.

`hashref alias`

Dieser Hash enthält die Spaltenbeschreibung und ihre Aliase. Zusätzlich kann auch eine Normierung angegeben werden, die Werte werden dann zur Weiterverarbeitung entnormiert:

```
$columns => {
  IstAkzentamplitude => {scale => 2, source => 'Feld-1'},
}
```

`boolean noheader`

Falls *noheader* vor dem Laden der Tabelle gesetzt ist, wird nicht versucht, die erste Zeile der Tabelle als Spaltenbeschriftungen zu interpretieren.

Nur-Lese-Felder

`int num`

Enthält die Anzahl der gelesenen Datensätze.

C.6.2. Die Klasse ResultFile

Die Klasse *ResultFile* liest Ergebnis-Dateien von SNNS, und stellt die Daten wahlweise auch über eine Aliastabelle zur Verfügung. Sie ist von der Klasse *TableFile* abgeleitet und unterstützt damit auch alle Methoden und Felder von *TableFile*.

Methoden

`Object = sub new([Datei])`

Instantiiert ein Objekt der Klasse *ResultFile*. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden SNNS-Ergebnis-Datei, er wird in *file* gespeichert.

`sub load([Datei])`

Lädt die SNNS-Ergebnisse aus einer Datei. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden SNNS-Ergebnis-Datei, er wird in *file* gespeichert.

Schreib-Lese-Felder

`string inputpref`

Enthält den Stamm für Felder, die von SNNS als Eingänge des Netzwerks gekennzeichnet werden, standardmäßig auf „input-“ gesetzt.

`string teachingpref`

Enthält den Stamm für Felder, die von SNNS als Sollwerte für die Ausgänge des Netzwerks gekennzeichnet werden, standardmäßig auf „teaching-“ gesetzt.

`string outputpref`

Enthält den Stamm für Felder, die von SNNS als Istwerte für die Ausgänge des Netzwerks gekennzeichnet werden, standardmäßig auf „output-“ gesetzt.

Nur-Lese-Felder

`boolean valid`

Gesetzt, falls die Datei erfolgreich gelesen werden konnte.

`string version`

Enthält nach erfolgreichem Lesen die Versionsnummer der SNNS-Ergebnis-Datei.

`string timestamp`

Enthält nach erfolgreichem Lesen das Datum und die Zeit der Erstellung der SNNS-Ergebnis-Datei.

`int inputnum`

Enthält nach erfolgreichem Lesen die Anzahl der Eingänge des verwendeten Netzes.

`int outputnum`

Enthält nach erfolgreichem Lesen die Anzahl der Ausgänge des verwendeten Netzes.

`boolean withinput`

Gesetzt, falls die Datei neben den realisierten Ausgabewerten auch die dafür verwendeten Eingabewerte enthält.

`boolean withoutput`

Gesetzt, falls die Datei neben den realisierten Ausgabewerten auch die Sollwerte für die Ausgabe enthält.

C.6.3. Die Klasse Pac

Die Klasse *Pac* erleichtert das Umgehen mit Fujisaki-Parametern. Sie ermöglicht das Lesen von so genannten PAC-Dateien, die als Format zur Speicherung von Fujisaki-Parametern zum Einsatz kommen. Zusätzlich kann der Grundfrequenzverlauf regeneriert werden.

Methoden

`Object = sub new([Datei])`

Instantiiert ein Objekt der Klasse *Pac*. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden PAC-Datei, er wird in *file* gespeichert.

`sub load([Datei])`

Lädt eine PAC-Datei. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Datei, er wird in *file* gespeichert.

`array = sub kontur`

Gibt den Grundfrequenzverlauf in einem eindimensionalen Array zurück.

Schreib-Lese-Felder

`string file`

Enthält den Dateinamen der PAC-Datei.

`int size`

Enthält die Anzahl Samples der zu erzeugenden Grundfrequenzkontur.

`float dt`

Enthält das Zeitintervall in Sekunden zwischen zwei Samples der Grundfrequenzkontur und ist standardmäßig auf 10 ms gesetzt.

`float alpha`

Enthält den Fujisaki-Parameter α und ist standardmäßig auf 1/s gesetzt. Dieser Wert kann für jede Phrase überschrieben werden.

`float beta`

Enthält den Fujisaki-Parameter β und ist standardmäßig auf 20/s gesetzt. Dieser Wert kann für jeden Akzent überschrieben werden.

`float fb`

Enthält den Fujisaki-Parameter *Fb* und ist standardmäßig auf 70 Hz gesetzt.

`array aa`

Enthält die Fujisaki-Parameter für die Akzente. Falls *Beta* nicht gesetzt ist, wird der Wert des globalen Feldes *beta* verwendet:

```
push @{$istfuji->aa}, {
  StartpositionAkzent => 12.6,
  EndpositionAkzent => 13.1,
  Akzentamplitude => 0.6,
  Beta => 18.5,
}
```

`array ap`

Enthält die Fujisaki-Parameter für die Phrases. Falls *Alpha* nicht gesetzt ist, wird der Wert des globalen Feldes *alpha* verwendet:

```
push @{$istfuji->ap}, {
  Phrasenposition => 10.2,
  Phrasenamplitude => 0.3,
  Alpha => 0.9,
}
```

C.6.4. Die Klasse Syntax

Die Klasse *Syntax* erleichtert den Zugriff auf die Syntax-Daten aus der Stuttgarter Tabelle. Dazu werden die Daten meldungs-, satz-, phrasen-, wort- sowie silbenweise aufbereitet.

Methoden

`Object = sub new([Datei])`

Instantiiert ein Objekt der Klasse *Syntax*. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Tabelle, er wird in *file* gespeichert.

`sub load([Datei])`

Lädt eine Tabelle aus einer Datei. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Tabelle, er wird in *file* gespeichert.

`sub parse`

Bereitet die Daten meldungs-, satz-, phrasen-, wort- sowie silbenweise auf und stellt sie in den Feldern *paragraphs*, *sentences*, *phrases*, *words* und *sylables* zur Verfügung.

Globale Felder

`hashref defaultFields`

Enthält einen Standard-Satz Alias-Daten für *fields*, die sich gut mit der meist verwendeten reduzierten Variante der Stuttgarter Tabelle verträgt. Dabei wird keine Normierung durchgeführt.

Schreib-Lese-Felder

`string file`

Enthält den Dateinamen der Tabelle.

`hashref fields`

Dieser Hash enthält die Spaltenbeschreibung und ihre Aliase, wie sie zum Zugriff auf die darunter liegende Tabellendatei mittels *TableFile* benötigt wird. Obwohl diese Tabelle von außen frei konfigurierbar ist, ist zu beachten, dass ein Teil der internen Funktionalität auf bestimmte Alias-Namen angewiesen ist (siehe auch *defaultFields*).

Nur-Lese-Felder

`array paragraphs`

Enthält eine Auflistung aller Absätze (Meldungen) in der Tabelle.

```
$currParagraphData = {
  id => <Nummer der Meldung in der Datenbank>,
  start => <Datensatznummer des Absatzstartes>,
  name => <Dateiname>,
  sentences => <untergeordnete Sätze>,
  phrases => <untergeordnete Phrasen>,
  words => <untergeordnete Wörter>,
  sylables => <untergeordnete Silben>,
  Laenge => <Länge des Absatzes>,
}
```

`array sentences`

Enthält eine Auflistung aller Sätze in der Tabelle.

```
$currSentenceData = {
  id => <Nummer des Satzes im Absatz>,
  start => <Datensatznummer des Satzstartes>,
  text => <Text des Satzes>,
  phrases => <untergeordnete Phrasen>,
```

```
  words => <untergeordnete Wörter>,
  sylables => <untergeordnete Silben>,
  paragraph => <zugehöriger Absatz>,
  Laenge => <Länge des Satzes>,
  StartpositionInAbsatz => <Start relativ zum Absatz>,
  EndpositionInAbsatz => <Ende relativ zum Absatz>,
}
```

`array phrases`

Enthält eine Auflistung aller Phrasen in der Tabelle.

```
$currPhraseData = {
  id => <Nummer der Phrase im Satz>,
  start => <Datensatznummer des Phrasestartes>,
  text => <Text der Phrase>,
  words => <untergeordnete Wörter>,
  sylables => <untergeordnete Silben>,
  paragraph => <zugehöriger Absatz>,
  sentence => <zugehöriger Satz>,
  Laenge => <Länge der Phrase>,
  StartpositionInAbsatz => <Start relativ zum Absatz>,
  EndpositionInAbsatz => <Ende relativ zum Absatz>,
  StartpositionInSatz => <Start relativ zum Satz>,
  EndpositionInSatz => <Ende relativ zum Satz>,
}
```

`array words`

Enthält eine Auflistung aller Wörter in der Tabelle.

```
$currWordData = {
  id => <Nummer des Wortes in der Phrase>,
  start => <Datensatznummer des Wortstartes>,
  text => <Text des Wortes>,
  sylables => <untergeordnete Silben>,
  paragraph => <zugehöriger Absatz>,
  sentence => <zugehöriger Satz>,
  phrase => <zugehörige Phrase>,
  Laenge => <Länge des Wortes>,
  StartpositionInAbsatz => <Start relativ zum Absatz>,
  EndpositionInAbsatz => <Ende relativ zum Absatz>,
  StartpositionInSatz => <Start relativ zum Satz>,
  EndpositionInSatz => <Ende relativ zum Satz>,
  StartpositionInPhrase => <Start relativ zur Phrase>,
  EndpositionInPhrase => <Ende relativ zur Phrase>,
}
```

`array sylables`

Enthält eine Auflistung aller Silben in der Tabelle.

```
$currSylableData = {
  id => <Nummer der Silbe im Wort>,
  start => <Datensatznummer der Silbe>,
  text => <SAMPA-Umschreibung der Silbe>,
  paragraph => <zugehöriger Absatz>,
  sentence => <zugehöriger Satz>,
  phrase => <zugehörige Phrase>,
  word => <zugehöriges wort>,
  Laenge => <Länge des Wortes>,
  StartpositionInAbsatz => <Start relativ zum Absatz>,
  EndpositionInAbsatz => <Ende relativ zum Absatz>,
  StartpositionInSatz => <Start relativ zum Satz>,
  EndpositionInSatz => <Ende relativ zum Satz>,
  StartpositionInPhrase => <Start relativ zur Phrase>,
  EndpositionInPhrase => <Ende relativ zur Phrase>,
  StartpositionInWort => <Start relativ zum Wort>,
  EndpositionInWort => <Ende relativ zum Wort>,
  Akzent => <ToBI-Akzent>,
  Wortakzent => <lexikalischer Akzent>,
  Grenzton => <Grenzton>,
  Intonem  => <Typ des Intonems>,
}
```

C.6.5. Die Klasse Data

Die Klasse *Data* erleichtert den Zugriff auf Ergebnis-Daten aus SNNS-Ergebnis-Dateien oder Tabellen.

Methoden

`Object = sub new([Datei])`

Instantiiert ein Objekt der Klasse *Syntax*. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Datei, er wird in *file* gespeichert.

`sub resLoad([Datei])`

Lädt die Ergebnisse aus einer SNNS-Ergebnis-Datei. Der optionale Parameter *Datei* liefert den Dateinamen der zu lesenden Tabelle, er wird in *file* gespeichert.

`sub parse`

Überträgt die Daten aller in *resFields* definierten Aliase in *sylables*.

Globale Felder

`hashref defaultResFields`

Enthält einen Standard-Satz Alias-Daten für *resFields*, die dem ursprünglichen 24-Eingänge-8-Ausgänge-Netz entspricht. Für alle anderen Netze müssen die Aliase angepasst werden. Bei der Konvertierung mit *parse* werden die Daten entnormiert.

Schreib-Lese-Felder

`string file`

Enthält den Dateinamen der Tabelle.

`hashref resFields`

Dieser Hash enthält die Spaltenbeschreibung und ihre Aliase, wie sie zum Zugriff auf die darunter liegende SNNS-Ergebnis-Datei mittels *ResultFile* benötigt wird.

Nur-Lese-Felder

`array sylables`

Enthält eine Auflistung aller Silben.

```
$currSylableData = {
  start => <Datensatznummer der Silbe>,
  ...   => <weitere Felder der Alias-Tabelle>,
}
```

C.7. mark.pl – Interaktive wortweise Markierung von Datensätzen

Um die Markierung von Wörtern nach Betonung oder Worttyp wie in Abschnitt 3.6.2 gefordert durchführen zu können, existiert *mark.pl*. Es ermöglicht es

- frei durch den kompletten Korpus zu navigieren,
- mit Hilfe einer übersichtlichen Oberfläche Datensätze wortweise zu markieren,
- den Wert der Fujisaki-Parameter für Akzente und Phrasen visuell zu veranschaulichen sowie

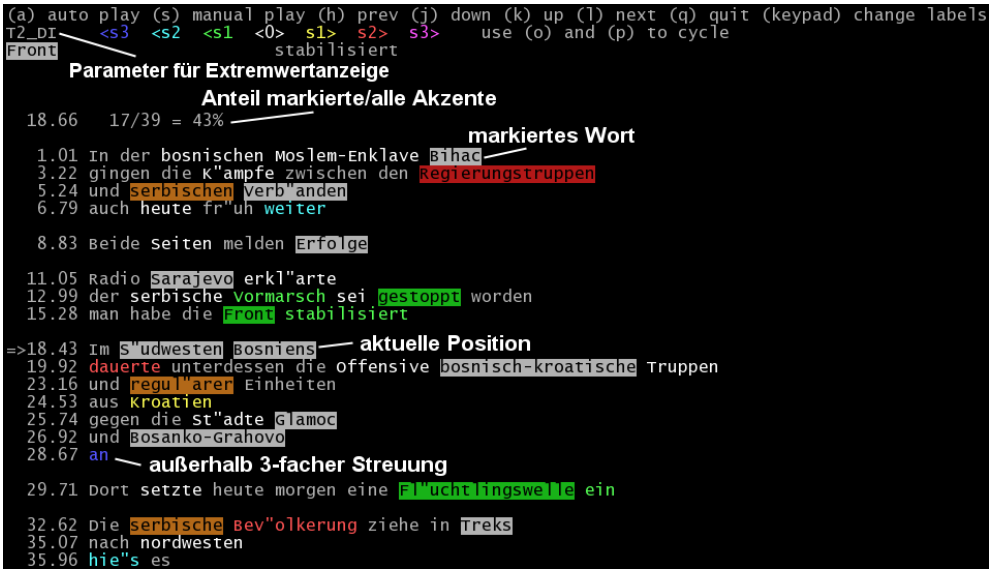

Abbildung C.2: Screenshot von mark.pl

- die Rohdaten meldungs- und phrasenweise über eine Soundkarte wiederzugeben.

Abbildung C.2 zeigt einen Screenshot von *mark.pl*. Das Programm ist mit obigem Perl-Framework geschrieben und greift für die Text-Ausgabe auf die Curses-Bibliothek zurück.

Die Bedienung des Programms erfolgt über die Tastenkombinationen in Tabelle C.2.

Zur Visualisierung der Fujisaki-Parameter werden Parameter, die um ein gewisses Vielfaches der Streuung vom Mittelwert abweichen, farbig unterlegt. Die

Taste	Funktion
a	Abspielen der kompletten Meldung, der Cursor folgt dem Abspielverlauf
s	Abspielen der aktuellen Phrase
h,l	zu einer anderen Meldung wechseln
j, k	eine andere Phrase auswählen
o, p	den Fujisaki-Parameter zur Visualisierung wechseln
q	das Programm beenden und die geänderten Markierungen speichern

Tabelle C.2: Tastenkombinationen zur Bedienung von mark.pl

Skala dazu befindet sich mit dem aktuell ausgewählten Fujisaki-Parameter in der zweiten Zeile.

Die Markierung einzelner Wörter erfolgt mit dem nummerischen Tastenblock. Jedem Wort mit Akzenten der aktuellen Phrase wird eine Taste des Nummernblocks zugewiesen, wobei die Zuordnung von Ziffern zu Worten im oberen Bildschirmteil visualisiert wird. Ein Tastendruck wechselt die Markierung des gewünschten Wortes. Falls die komplette Meldung über die Soundkarte wiedergegeben wird, werden auch die Zuweisungen von Tasten phrasenweise aktualisiert.

Zur Wiedergabe der einzelnen Meldungen wird auf *sox* bzw. dessen Alias *play* zurückgegriffen, der die Ausgabe auf der System-Soundkarte realisiert. Für andere Systeme als SuSE Linux ist hier möglicherweise eine Anpassung notwendig.

Der Aufruf erfolgt mittels

```
mark.pl Tabelle Ergebnis-Datei Markierungs-Datei
```

Dabei werden folgende Parameter erkannt:

```
Tabelle: Grunddaten der Stuttgarter Datenbasis in Tabellenform
Ergebnis-Datei: SNNS-Ergebnis-Datei mit den Daten für die Extrem-
  wertmarkierung
Markierungs-Datei: Datei mit den markierten Wörtern, wird neu
  angelegt, falls sie nicht existiert
```

C.8. wrd2syl.pl – Kovertierung von Wortindizes in Silbenmarkierungen

Da die Ausgabe von *mark.pl* in Form einer Liste der Indizes der markierten Wörter nicht direkt für das Training verwendet werden kann, konvertiert *wrd2syl.pl* diese in eine silbenbezogene Markierung, wobei nur Akzente berücksichtigt werden.

Der Aufruf erfolgt mittels

```
wrd2syl.pl Tabelle Markierungs-Datei [Feldname] > Silben-Datei
```

Dabei werden folgende Parameter erkannt:

```
Tabelle: Grunddaten der Stuttgarter Datenbasis in Tabellenform
Markierungs-Datei: Datei mit den markierten Wörtern, wird neu
  angelegt, falls sie nicht existiert
Feldname: Spaltenüberschrift
Silben-Datei: Silbeninformationen
```

Die erzeugte Datei kann z. B. mit Hilfe von GNU *paste* über

```
paste Original-Tabelle Silben-Datei > Neue-Tabelle
```

an die Trainingsdaten angefügt werden.

D. CD-Inhalt

Auf der beiliegenden CD-R befinden sich

- die LaTeX-Quellen des Belegs,
- die im Laufe der Arbeit entwickelten Programme,
- die Trainingsdaten der erwähnten Untersuchungen,
- die Literatursammlung.

Verzeichnisstruktur

/Beleg Der LaTeX-Quelltext und alle Grafiken und Bilder, die zur Erstellung des Berichtes benötigt werden.

/Programme Alle während der Studienarbeit erstellten Programme.

/Daten Nach Kapiteln sortierte Trainingsläufe, die in diesem Bericht Erwähnung finden. Auf Grund des Umfangs der im Laufe der Studienarbeit produzierten Daten von 14 GB wurde bei der Zusammenstellung der CD auf den Großteil der Trainingsdaten verzichtet. In allen Verzeichnissen finden sich jedoch Shell-Skripte, die eine Regenerierung der zum Training der Netze benutzen Daten möglich machen und eine Reproduktion der Ergebnisse zulassen.

/Literatur Alle im Literaturverzeichnis aufgeführten Werke, die in digitaler Form verfügbar sind.

/Zusatz Weitere Trainingsdaten, Programme und Untersuchungen, die z. B. wegen nicht verwertbarer Ergebnisse nicht Eingang in den Beleg fanden.

Literaturverzeichnis

[1] Grimm, Jacob: *Grimms Märchen*. Thienemann, 1989.

[2] Jokisch, Oliver, Diane Hirschfeld, Matthias Eichner, and Rüdiger Hoffmann: *Creating an individual speech rhythm: A data driven approach*. In *Proceedings of the 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, pages 115–119, Blue Mountains, NSW, Australia, 1998.

[3] Jokisch, Oliver, Diane Hirschfeld, Matthias Eichner, and Rüdiger Hoffmann: *Multi-level rhythm control for speech synthesis using hybrid data driven and rule-based approaches*. In *Proceedings ICSLP 1998*, pages 607–610, Sydney, Australia, 2000.

[4] Jokisch, Oliver, Hansjörg Mixdorff, Hans Kruschke, and Ulrich Kordon: *Learning the parameters of quantitative prosody models*. In *Proceedings ICSLP 2000*, volume 1, pages 645–648, Beijing, China, 2000.

[5] Kossebau, Friedrich W. H.: *Anwendung von Methoden aus dem Gebiet Evolutionäres Rechnen zum optimierten Entwurf von integrierten intelligenten Systemen*. Studienarbeit, Technische Universität Dresden, 2001.

[6] Kossebau, Friedrich W. H.: *Evolutionäre Optimierung einer trainingsbasierten Prosodiegenerierung*. Diplomarbeit, Technische Universität Dresden, 2002.

[7] Kruschke, Hans: *Parameter extraction of a quantitative intonation model with wavelet analysis and evolutionary optimization*. In *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 524–527, Hong Kong, 2003. ICASSP.

[8] Kühne, Marco: *Datenanalyse und Detaillierung eines Modells zur Intensitätssteuerung für die Sprachsynthese*. Studienarbeit, Technische Universität Dresden, 2003.

[9] Mixdorff, Hansjörg: *Intonation Patterns of German - Model-based Quantitative Analysis and Synthesis of F_0 contours*. Dissertation, Technische Universität Dresden, 1997.

[10] Mixdorff, Hansjörg: *A novel approach to to fully automatic extraction of fujisaki model parameters*. In *Proceedings of ICASSP 2000*, volume 3, pages 1285–1288, Istanbul, Turkey, 2000.

[11] Mixdorff, Hansjörg: *An Integrated Approach to Modelling German Prosody*. Habilitation, Technische Universität Dresden, 2001.

[12] Möbius, Bernd: *Components of a quantitative model of german intonation*. In *Proceedings of the 13th International Congress of Phonetic Sciences*, volume 2, pages 108–115, Stockholm, 1995.

[13] Pescheck, Markus: *kNN-Ansätze zur Prosodiegenrierung in Text-to-Speech (TTS) Systemen*. Studienarbeit, Technische Universität Dresden, 1996.

[14] Schiller, Anne, Simone Teufel, Christine Stockert und Christine Thielen: *Vorläufige Guidelines fur das Tagging deutscher Textcorpora mit STTS*. Universität Stuttgart, Institut für maschinelle Sprachverarbeitung und Universität Tübingen, Seminar für Sprachwissenschaft, November 1995.

[15] Zell, Andreas: *Simulation Neuronaler Netze*. Addison-Wesley, 1994.

[16] University of Stuttgart, Institute for Parallel and Distributed High Performance Systems (IPVR): *SNNS Stuttgart Neural Network Simulator User Manual, Version 4.2*, December 2003. `http://www-ra.informatik.uni-tuebingen.de/SNNS/`.