Technische Universität Dresden

Phonetische Transkription für ein multilinguales Sprachsynthesesystem

Horst-Udo Hain

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden
zur Erlangung des akademischen Grades eines

Doktoringenieurs
(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil Wilfried Sauer		
Gutachter:	Prof. Dr.-Ing. habil Rüdiger Hoffmann	Tag der Einreichung:	13.02.2004
	Prof. Dr. phil. nat. Harald Höge	Tag der Verteidigung:	23.09.2004
	Prof. Dr.-Ing. habil. Werner Zühlke		

Vorwort

Die vorliegende Arbeit beschäftigt sich mit einem datengetriebenen Verfahren zur Graphem-Phonem-Konvertierung für ein Sprachsynthesesystem. Die Aufgabe besteht darin, die Aussprache für beliebige Wörter zu bestimmen, auch für solche Wörter, die nicht im Lexikon des Systems enthalten sind. Die Architektur an sich ist sprachenunabhängig, von der Sprache abhängig sind lediglich die Wissensquellen, die zur Laufzeit des Systems geladen werden. Die Erstellung von Wissensquellen für weitere Sprachen soll weitgehend automatisch und ohne Einsatz von Expertenwissen möglich sein. Expertenwissen kann verwendet werden, um die Ergebnisse zu verbessern, darf aber keine Voraussetzung sein.

Für die Bestimmung der Transkription werden zwei neuronale Netze verwendet. Das erste Netz generiert aus der Buchstabenfolge des Wortes die zu realisierenden Laute einschließlich der Silbengrenzen, und das zweite bestimmt im Anschluß daran die Position der Wortbetonung. Diese Trennung hat den Vorteil, daß man für die Bestimmung des Wortakzentes das Wissen über die gesamte Lautfolge einbeziehen kann. Andere Verfahren, die die Transkription in einem Schritt bestimmen, haben das Problem, bereits zu Beginn des Wortes über den Akzent entscheiden zu müssen, obwohl die Aussprache des Wortes noch gar nicht feststeht. Zudem bietet die Trennung die Möglichkeit, zwei speziell auf die Anforderung zugeschnittene Netze zu trainieren.

Die Besonderheit der hier verwendeten neuronalen Netze ist die Einführung einer Skalierungsschicht zwischen der eigentlichen Eingabe und der versteckten Schicht. Eingabe und Skalierungsschicht werden über eine Diagonalmatrix verbunden, wobei auf die Gewichte dieser Verbindung ein Weight Decay (Gewichtezerfall) angewendet wird. Damit erreicht man eine Bewertung der Eingabeinformation während des Trainings. Eingabeknoten mit einem großen Informationsgehalt werden verstärkt, während weniger interessante Knoten abgeschwächt werden. Das kann sogar soweit gehen, daß einzelne Knoten vollständig abgetrennt werden. Der Zweck dieser Verbindung ist, den Einfluß des Rauschens in den Trainingsdaten zu reduzieren. Durch das Ausblenden der unwichtigen Eingabewerte ist das Netz besser in der Lage, sich auf die wichtigen Daten zu konzentrieren. Das beschleunigt das Training und verbessert die erzielten Ergebnisse. In Verbindung mit einem schrittweisen Ausdünnen der Gewichte (Pruning) werden zudem störende oder unwichtige Verbindungen innerhalb der Netzwerkarchitektur gelöscht. Damit wird die Generalisierungsfähigkeit noch einmal erhöht.

Die Aufbereitung der Lexika zur Generierung der Trainingsmuster für die neuronalen Netze wird ebenfalls automatisch durchgeführt. Dafür wird mit Hilfe der dynamischen Zeitanpassung

(DTW) der optimale Pfad in einer Ebene gesucht, die auf der einen Koordinate durch die Buchstaben des Wortes und auf der anderen Koordinate durch die Lautfolge aufgespannt wird. Somit erhält man eine Zuordnung der Laute zu den Buchstaben. Aus diesen Zuordnungen werden die Muster für das Training der Netze generiert.

Um die Transkriptionsergebnisse weiter zu verbessern, wurde ein hybrides Verfahren unter Verwendung der Lexika und der Netze entwickelt. Unbekannte Wörter werden zuerst in Bestandteile aus dem Lexikon zerlegt und die Lautfolgen dieser Teilwörter zur Gesamttranskription zusammengesetzt. Dabei werden Lücken zwischen den Teilwörtern durch die neuronalen Netze aufgefüllt. Dies ist allerdings nicht ohne weiteres möglich, da es zu Fehlern an den Schnittstellen zwischen den Teiltranskriptionen kommen kann. Dieses Problem wird mit Hilfe des Lexikons gelöst, das für die Generierung der Trainingsmuster aufbereitet wurde. Hier ist eine eindeutige Zuordnung der Laute zu den sie generierenden Buchstaben enthalten. Somit können die Laute an den Schnittstellen neu bewertet und Transkriptionsfehler vermieden werden.

Danksagung

Zu Beginn dieser Arbeit möchte ich mich bei denjenigen bedanken, die zu ihrem Gelingen beigetragen haben.

Mein erster Dank gilt meinem Doktorvater Prof. Rüdiger Hoffmann vom Institut für Akustik und Sprachkommunikation der TU Dresden. Seine Unterstützung war mir eine große Hilfe bei dieser Promotion. Ein nicht minder großer Dank geht an die beiden Gutachter, Prof. Harald Höge von der Siemens AG und Prof. Werner Zühle von der TU Ilmenau.

Weiterhin möchte mich bei den Menschen bedanken, die mich während der Promotion bei der Siemens AG begleitet haben. Sie haben mir immer mit Rat und Tat zur Seite gestanden. Dr. Martin Holzapfel danke ich für die fruchtbaren Diskussionen besonders zu Beginn meiner Promotion, Dr. Hans Georg Zimmermann für die ausführlichen Erläuterungen zum theoretischen Innenleben neuronaler Netze und vor allem für die Tips und Tricks zu deren Anwendung in der Praxis, Dr. Christoph Tietz für Hinweise zum Training der neuronalen Netze und die Unterstützung bei der Automatisierung der Trainingsläufe, Caglayan Erdem und Dr. Achim Müller für die Unterstützung bei der Erprobung diverser Netzwerkarchitekturen, Ute Ziegenhain und Dr. Herbert Tropf für die Hilfe bei phonetischen und linguistischen Problemen, Dr. Petra Witschel für die Diskusionen über die Informationstheorie, Thomas Volk für die Unterstützung bei der praktischen Anwendung der Ergebnisse dieser Arbeit und Dr. Josef Bauer, der mir mit vielen Ratschlägen zum Textsatzsytem LaTeX bei der Erstellung der schriftlichen Arbeit sehr geholfen hat.

Ein besonderer Dank geht an meine Mutter, die durch ihre Unterstützung meinen Werdegang und somit auch diese Promotion überhaupt ermöglicht hat.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Sprachsynthese ist ein Teilgebiet der Mensch-Maschine-Kommunikation (MMK). Sie kommt zur Anwendung, wenn Texte mit einem beliebigen Wortschatz akustisch ausgegeben werden sollen. Im Gegensatz zur reinen Sprachwiedergabe, bei der aufgezeichnete Spachsequenzen verwendet werden, muß die Sprachsynthese in der Lage sein, unbekannte, also nicht im Fundus enthaltene Wörter, bearbeiten zu können. Das wird in Zeiten, in denen man über Internet oder Mobilfunk immer mehr und immer größere Datenbanken erreichen kann, immer wichtiger.

Innerhalb der Sprachsynthese unterscheidet man zwischen „Text-to-Speech“ (TTS) und „Concept-to-Speech“ (CTS). Die Eingabe von TTS-Systemen besteht aus unbekanntem und zum Teil unformatiertem Text, der erst aufbereitet werden muß. Der Text muß linguistisch analysiert werden, um ihn dann prosodisch strukturieren zu können. Solchen Text findet man z. B. in elektronischer Post oder auf Internetseiten. Die Eingabe von CTS-Systemen besteht aus generiertem Text, der z. B. von einem Dialogsystem erstellt wurde, um dem Benutzer eine Frage zu beantworten. In diesem Fall ist bekannt, was gesagt werden soll. Man kennt die Struktur des Satzes und weiß auch, welches Wort im Satz wie betont werden soll.

Die heutigen Anforderungen an die Sprachsynthese sind extrem hoch. Es soll nicht nur eine gute Qualität erreicht werden, sondern auch eine hohe Portabilität. Das System soll nicht nur auf einem Server mit schnellen Prozessoren und viel Hauptspeicher laufen, sondern auch auf mobilen Geräten mit wenigen Kilo- oder Megabyte Speicher, und das mit immer noch akzeptabler Qualität. Außerdem besteht der Anspruch, mehrere Sprachen zu unterstützen, möglichst mit mehreren Sprechern in verschiedenen Sprechsituationen, wobei die Anpassung auf neue Sprachen und Sprecher in möglichst kurzer Zeit und mit wenig Aufwand vor sich gehen soll.

Die erste Anforderung, daß das System mit wenig Speicher auskommen soll, spricht für die Anwendung von regelbasierten Ansätzen. Die kurzfristige Anpassung an mehrere Sprachen verbietet jedoch die Verwendung von Regeln. Solche Regeln zu erstellen ist zeitaufwendig und erfordert Expertenwissen. Hier bieten sich datengetriebene Verfahren an, die nur noch an die neue Sprache oder den Sprecher angepaßt werden müssen.

Im weiteren Verlauf dieses Kapitels wird eine kurze Einführung zur Graphem-Phonem-Konvertierung gegeben und ihre Bedeutung innerhalb eines Sprachsynthese-Systems erläutert. Dazu wird kurz das Text-to-Speech (TTS) System „Papageno“ vorgestellt, für das die hier vorgestellte Graphem-Phonem-Konvertierung entwickelt wurde. Weiterhin werden bereits entwickelte Konvertierungssysteme vorgestellt und in Beziehung zu der vorliegenden Arbeit gesetzt. Im Kapitel 2 werden die Grundlagen datengetriebener Verfahren erläutert, die innerhalb dieser Arbeit ihre Anwendung finden. Kapitel 3 befaßt sich sowohl mit der Aufbereitung des phonetischen Lexikons als auch mit dem Lernen der im Lexikon enthaltenen Konvertierungsregeln. In Kapitel 4 werden die erzielten Ergebnisse vorgestellt und bewertet. In Kapitel 5 folgt eine Zusammenfassung der Arbeit und eine Diskussion nicht gelöster Probleme.

1.1 Übersicht über das TTS-System „Papageno“

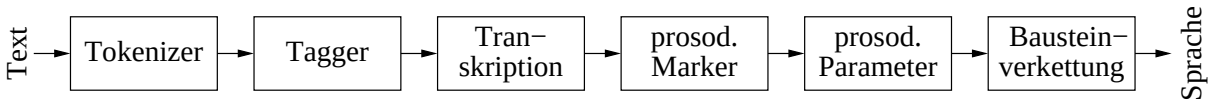

Abbildung 1.1: Architektur des TTS-Systems „Papageno“.

„Papageno“ ist ein Sprachsynthese-System, das in der Sprachgruppe der Zentralabteilung Technik der SIEMENS AG entwickelt wird. Es dient zur Generierung synthetischer Sprache aus beliebigem maschinenlesbarem ASCII-Text. Bei diesem Text kann es sich um Handbücher, elektronische Post oder ähnlichen Text handeln, der z. B. von einem Computer vorgelesen werden soll. Im Gegensatz zu einem Concept-to-Speech-System, bei dem bereits klar ist, was gesagt werden soll, muß bei einem TTS-System der zu sprechende Text erst analysiert werden.

In einem ersten Schritt wird der Text in sogenannte Token (Wörter, Zahlen, Abkürzungen, Akronyme, Satzzeichen usw.) eingeteilt [RSWH+99]. Ein Token beginnt mit einem Zeichen, das kein Trennzeichen ist, und endet mit dem Zeichen, nach dem ein Trennzeichen kommt. Diese Aufteilung kann in späteren Schritten wieder aufgehoben werden. Dabei werden z. B. im Englischen die Token *New* und *York* zu einem Token verschmolzen.

Für jedes Token wird im Anschluß die linguistische Kategorie bestimmt [Hain99b],[Sün02]. Zuerst wird das Wort in einem linguistischen Lexikon gesucht. Wird es dort nicht gefunden, so wird mit einer Ausnahmebehandlung eine Liste der möglichen Kategorien für dieses Wort ermittelt. Sind an dieser Stelle noch mehrere Kategorien möglich, so wird mit Hilfe eines Taggers eine Auswahl getroffen.

Daran schließt sich die Bestimmung der Transkription für jedes Token an. Dazu werden ein speziell aufbereitetes phonetisches Lexikon und neuronale Netze verwendet [Hain99a, Hain00a, Hain00c, HZ01a, HZ01b]. Diese Transkription enthält neben der Phonemfolge auch die Information über die Position der Wortbetonung und der Silbengrenzen.

Im nächsten Schritt wird versucht, an Hand der linguistischen Kategorien der Token auf prosodische Markierungen innerhalb eines Satzes zu schließen [Mül03]. Dabei geht es um die Bestimmung der Grenzen prosodischer Phrasen (z. B. Sprechpausen) sowie die Bestimmung von prosodischen Akzentstellen, also Akzentstellen auf Wortebene innerhalb einer prosodischen Phrase.

Die prosodischen Marken und die Transkription bilden die Eingabe für zwei nachfolgende Module. Zuerst werden für die gegebene Phonemfolge Lautdauer, Lautenergie und Grundfrequenzverlauf errechnet. Mit dieser Information wird aus einer Datenbank von Sprachbausteinen die Liste der Bausteine ausgewählt, die im phonetischen und prosodischen Kontext am besten zueinander passen. Im letzten Schritt werden die Bausteine miteinander verkettet, wobei sie entsprechend der berechneten prosodischen Parameter angepaßt werden [Holz00].

An dieser Stelle wird deutlich, welchen Einfluß die Qualität der erzeugten Transkription auf das TTS-System hat. Die Phoneme, Silbengrenzen und Wortbetonungen sind außerordentlich wichtig für alle nachfolgenden Module. Fehler in der Phonemfolge führen zwangsläufig zur Auswahl falscher Sprachbausteine, und Fehler in der Position von Silbengrenze und Wortbetonung können sich zusätzlich negativ auf die Errechnung der prosodischen Parameter wie Grundfrequenz oder Lautdauer auswirken.

1.2 Sprachsynthese im Projekt „Verbmobil“

Ein anderes Sprachsynthese-System wurde im Rahmen des vom Bundesministerium für Bildung und Forschung geförderten Projektes „Verbmobil“ [SWHK+00] entwickelt. In diesem Projekt wurde ein Dialog-System zum Vereinbaren von Terminen entwickelt, das von einem Benutzer durch Spracheingabe gesteuert wird und das seine Ausgabe an den Benutzer mit Hilfe von synthetischer Sprache realisiert. Dabei soll das Dialog-System als Übersetzer für Benutzer dienen, die jeweils die Sprache des anderen Teilnehmers nicht oder nur unzureichend sprechen. Es handelt sich hier also um ein CTS-System.

Die in diesem Projekt entwickelte Synthese teilt sich in drei Module auf. Im ersten Teil wird der Text aufbereitet und transkribiert. Die erzeugte Phonemfolge enthält wiederum prosodische Angaben über Silben- und Wortgrenzen, Betonungen auf Wort- und Phrasenebene sowie verschiedene Typen von Phrasengrenzen. Im zweiten Modul werden mit Hilfe dieser Information die akustischen Parameter Grundfrequenz, Lautdauer und -energie bestimmt. Das dritte Modul benutzt die Ausgaben der beiden vorhergehenden zur Bestimmung der Sprachbausteine und für die akustische Manipulation.

Dem ersten Modul der Verbmobil-Synthese entsprechen die ersten vier Module des Systems Papageno: Tokenizer, Tagger, Transkription und symbolische Prosodie. Der Unterschied zwischen den beiden Systemen ist, daß in der Verbmobil-Synthese regelbasiert vorgegangen wird, was die Anpaßbarkeit an andere Sprachen erschwert. Dagegen sind die vier Module der Papageno-Synthese datengetrieben und somit einfach an eine neue Sprache anzupassen.

1.3 Graphem-Phonem-Konvertierung

Bei der Graphem-Phonem-Konvertierung handelt es sich um ein Teilgebiet der Sprachverarbeitung. Die Aufgabe besteht darin, den Graphemen (Buchstaben) eines Wortes die entsprechenden Phoneme (Laute) zuzuordnen. Diese Zuordnung ist z. B. in der Spracherkennung nötig, um von einer erkannten Phonemfolge auf das erkennde Wort schließen zu können. Umgekehrt wird in der Sprachsynthese bestimmt, wie ein Wort ausgesprochen wird.

Es wurden verschiedene Theorien entwickelt, wie der Mensch diese Konvertierung der Buchstaben in die zu sprechenden Laute durchführt. Eine Zusammenfassung dazu ist in [DE97] zu finden. In einem Modell wird davon ausgegangen, daß der Mensch eine Zwei-Weg-Strategie [Colt78] verfolgt: für die bekannten Wörter holt er die Aussprache aus einem Lexikon, und für die Aussprache unbekannter oder neuer Wörter benutzt er Regeln, mit denen er die Buchstaben bestimmten Lauten zuordnet. Ein Argument für diese Annahme ist z. B. die Tatsache, daß der Mensch auch sog. Pseudowörter aussprechen kann, also Wörter, die in seiner Sprache keinen Sinn ergeben. Ein weiteres Argument ist, daß er unterschiedlich lange für die Aussprache von regulären Wörtern auf der einen und Pseudowörtern auf der anderen Seite braucht.

Es gibt aber auch Argumente für eine Ein-Weg-Strategie, die besagt, daß die Aussprache von Pseudowörtern durch die Ausnutzung von Analogien zu bekannten Wörtern bestimmt wird. Ein Pseudowort wird also in Bestandteile aus dem Lexikon zerlegt und die Gesamtaussprache mit Hilfe dieser Teile bestimmt. So hat Glushko [Glus79, Glus81] festgestellt, daß die Aussprache für „Ausnahme-Pseudowörter“ (z. B. engl. *tave*) langsamer bestimmt wird als die für „reguläre Pseudowörter“ (z. B. *taze*). *taze* wird als regulär angesehen, da alle „orthographischen Nachbarn“ (z. B. *raze, gaze, maze*) den regulären Vokal `/eI/` enthalten, während *tave* die Ausnahme *have* als orthographischen Nachbarn hat.

Man unterscheidet weiterhin noch zwischen expliziter und impliziter Anwendung dieser Analogien. Bei der expliziten Analogie nimmt man an, daß das Lexikon eine individuelle Repräsentation für jedes Wort enthält. Die Strategie besteht darin, die Aussprache eines unbekannten Wortes durch die Modifikation der Aussprache eines ähnlichen Wortes zu bestimmen. Bei der impliziten Anwendung wird die Aussprache durch die Anwendung von generalisiertem phonographischen Wissen über existierende Wörter erzeugt. Diese implizite Analogie hat Gemeinsamkeiten mit der Ein-Weg-Strategie der konnektionistischen Modelle, z. B. bei *NETtalk* von Sejnowski und Rosenberg [SR87] (siehe auch Abschnitt 1.5.3.3). Hier wird das Generalisierungswissen gelernt und in der Gewichtsmatrix eines neuronalen Netzes abgespeichert. Ein solches Netz hat keine Vorstellung von dem Konzept „Wort“, kann also die Zuordnungsregeln nur implizit anwenden.

In dieser Arbeit wird die Zwei-Weg-Strategie verfolgt. Zuerst wird die Aussprache eines Wortes in einem phonetischen Lexikon gesucht. Wird sie dort nicht gefunden, so wird versucht, das Wort in Bestandteile aus dem Lexikon zu zerlegen. Für die Teile, die nicht im Lexikon zu finden sind, wird die Phonemfolge mit Hilfe eines neuronalen Netzes bestimmt.

1.4 Wortbetonung

Eine weitere wichtige Aufgabe bei der Bestimmung der Transkription für ein Sprachsynthese-System ist die Ermittlung des Wortakzentes. Allgemein versteht man unter Akzent ein „ . . . suprasegmentales Merkmal zur Hervorhebung von Lauten, Silben, Wörtern, Wortgruppen und Sätzen durch Intensivierung der Muskelaktivität bei der Artikulation.“ [Buss83]. Er kann durch die Variation der akustischen Parameter Grundfrequenz, Lautdauer und -energie realisiert werden, wobei das Zusammenspiel dieser drei Parameter für den Eindruck der Akzentuierung verwantwortlich ist [AZ02].

Aufgabe in diesem System ist die Bestimmung der Akzentposition innerhalb eines Wortes. Auf Wortebene kann der Akzent bedeutungsunterscheidend sein, z. B. bei *umfahren* (*ich fahre um*) bzw. *umfahren* (*ich umfahre*). Ein Beispiel für Englisch wäre die Unterscheidung zwischen *record* (*die Aufname*) und *to record* (*aufnehmen*).

Die Veränderung der Akzentposition kann auch dazu dienen, einen Teil des Wortes gezielt zu betonen, z. B. um einen Kontrast darzustellen ([Meng99], S. 20):

Was ist das denn?	*Das ist eine Holztür.*
Ist das eine Stahltür?	*Nein, das ist eine Holztür.*
Ist das ein Holzfenster?	*Nein, das ist eine Holztür.*

Eigentlich liegt beim Wort *Holztür* der Akzent auf der ersten Silbe. Um aber im dritten Beispiel zu verdeutlichen, daß es sich um eine Tür, und nicht um ein Fenster handelt, wird die zweite Silbe betont.

Man unterscheidet zwischen drei Akzentstufen: hauptbetont, nebenbetont und unbetont. Normalerweise erhält nur eine Silbe pro Wort die Hauptbetonung. Bei Wörtern mit vielen Silben, z. B. bei Komposita, besteht aber auch die Möglichkeit, daß eine oder mehrere Silben eine Nebenbetonung erhalten. Dies könnte z. B. bei einem zusammengesetzten Wort die hauptbetonte Silbe des zweiten Bestandtteils sein. Weiterhin gibt es unbetonte Silben, z. B. Silben mit den unbetonbaren Vokalen `/@/` oder `/6/`. Es gibt aber auch Ausnahmen, bei denen zwei gleichberechtigte Akzente in einem Wort vergeben werden, z. B. bei einem Intensitäts- oder Kopulativakzent ([Meng99], S. 22 ff.). Beispiele hierfür sind *haarscharf*, *eiskalt*, *geistig-kulturell* oder *Nordwest*.

Bei der Bestimmung der Wortbetonung ist zu beachten, daß die Position der Betonung innerhalb des Wortes auf der Phonemebene erfolgen sollte [Jess99]. Außerdem gehen die meisten Untersuchungen auch davon aus, daß die Silbengrenzen bekannt sind (siehe Abschitt 1.5.4). Das bringt allerdings auch gewisse Schwierigkeiten mit sich, da eine falsch bestimmte Silbengrenze dementsprechend eine falsche Wortbetonung nach sich ziehen kann. Es gibt jedoch auch Sprachen, bei denen *vor* der Graphem-Phonem-Konvertierung die Position des Wortakzentes bestimmt werden muß, da die Aussprache der Grapheme von der Betonung des Wortes abhängt. Dies ist z. B. in einigen slawischen Sprachen der Fall [RKV02].

1.5 Verfahren zur Graphem-Phonem-Konvertierung und Wortbetonung

Verfahren zur Graphem-Phonem-Konvertierung werden schon seit längerem eingesetzt. Einige davon sollen hier vorgestellt werden, wobei zwischen regelbasierten und datengetriebenen Verfahren unterschieden wird. Regelbasierte Systeme haben den Vorteil, ausführlich auf Ausnahmen eingehen zu können. Dagegen sind datengetriebene Systeme flexibler und können einfacher an andere Sprachen angepaßt werden.

1.5.1 Graphem-Phonem-Zuordnung

In einer ersten Betrachtung werden Verfahren zur Aufbereitung des Aussprachelexikons für die Erstellung der Eingabe-Ausgabe-Muster vorgestellt. Jedes System, das die Graphem-Phonem-Konvertierung lernen soll, braucht eine eindeutige Zuordnung der Grapheme zu den Phonemen. Die üblicherweise verfügbaren Lexika enthalten nur eine Zuordnung der orthographischen Form zur Phonemfolge, z. B.

```
textlich
t E k s t l I C
```

Für die Erstellung der Graphem-Phonem-Muster ist aber die folgende Zuordnung erforderlich:

```
t e x   t l i c+h
t E k+s t l I C
```

Dabei fällt auf, daß sowohl ein Graphem mehreren Phonemen als auch ein Phonem mehreren Graphemen zugeordnet werden kann. Die Aufgabe besteht nun darin, die einzelnen Elemente der Graphem- und Phonemfolgen einander zuzuordnen.

In [LK86] ist ein regelbasiertes System für Englisch beschrieben. Die Aufgabe bestand darin, die Angaben für die Aussprache in einem englischen Lexikon automatisch zu erstellen. Der Abbildungsvorgang wurde mit Hilfe von Zuordnungslisten durchgeführt, die manuell erstellt wurden. Wenn ein Graphem kein Phonem generiert, so wird es einem Null-Phonem zugeordnet, um zu markieren, daß es nicht gesprochen wird. Um die Robustheit des Verfahrens zu erhöhen, wurden fünf Sonderregeln erstellt, die die Abbildung der Grapheme auf das Null-Phonem behandeln. Der Algorithmus vergleicht das Wort und die Phonemfolge mit der Zuordnungstabelle. Dabei werden die längsten Graphem- und Phonemfolgen zuerst betrachtet.

Ein HMM-basierter Ansatz wurde in [Coil90] vorgestellt (siehe Abbildung 1.2). Die Ausgabesymbole dieses Modells sind die Grapheme. Jedesmal, wenn Status 2, 3 oder 4 erreicht wird,

wird ein Graphem ausgegeben. Durch die Aneinanderreihung von Phonemmodellen entsprechend der Transkription wird ein Wortmodell erstellt, mit dessen Hilfe die Phonemfolge der Graphemfolge zugeordnet werden kann. Mit dem Viterbi-Algorithmus wird dann die Folge von Zuständen ausgesucht, die am häufigsten die orthographische Repräsentation erzeugt hat. Wenn die Phonemmodelle mit dem gleichen Viterbi trainiert wurden, dann kann man davon ausgehen, daß die gefundene Zustandsfolge mit der korrekten Zuordnung korrespondiert.

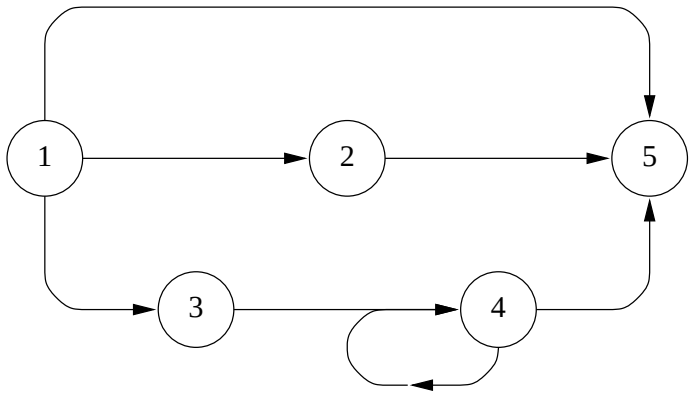

Abbildung 1.2: HMM-Architektur für die Graphem-Phonem-Zuordnung (aus [Coil90]).

Das Verfahren wurde für Holländisch getestet, zum einen mit einer Auswahl von 1000 aus den 10000 häufigsten Wörtern, und zum anderen mit 3000 selten auftretenden Wörtern. Auf dem ersten Testsatz wurden 98,6 % der Wörter richtig behandelt, auf dem zweiten 96,1 %. Bei diesen Wörtern wurde also die Graphemfolge richtig den Phonemen zugeordnet. Das so erstellte Lexikon ist der Ausgangspunkt für ein induktives Verfahren zum Erlernen der Transkriptionsregeln (siehe Abschnitt 1.5.3.1).

In [VPB98] wurde die dynamische Zeitanpassung (DTW , siehe Abschnitt 2.2) für die Zuordnung der Grapheme zu den Phonemen benutzt. Das Verfahren wurde auf ein amerikanisches (CMU [Weid98]), ein britisches (OALD [Mitt92]) und ein französisches (BRULEX [CMR90]) Lexikon angewendet. Es sollen die Wahrscheinlichkeiten bestimmt werden, mit denen ein bestimmtes Graphem einem bestimmen Phonem zugeordnet wird. Um die Aufgabe zu vereinfachen, wurden die Pseudophoneme `/K_S/` (engl. *fax*) und `/W_A/` (franz. *royal*) eingeführt. Somit gab es keine Fälle mehr, bei denen ein Graphem mehrere Phoneme erzeugt. In der Initialisierungsphase wurden die Wahrscheinlichkeiten für alle möglichen Graphem-Phonem-Zuordnungen errechnet. Damit wurde dann mit Hilfe des DTW der Pfad mit den wahrscheinlichsten Paaren gesucht. Die so erhaltenen Paare ergaben neue Zuordnungswahrscheinlichkeiten, die in der nächsten Iteration für die Bestimmung des besten Pfades verwendet wurden. Bereits nach fünf Iterationen konnte dieser Vorgang abgebrochen werden. Ähnliche Ansätze werden in [AD94] und [DYB95] verfolgt.

1.5.2 Regelbasierte Verfahren zur Graphem-Phonem-Konvertierung

Regelbasierte Systeme haben Vor- und Nachteile. Zum einen ist man bei Regelsätzen sehr gut in der Lage, auf Ausnahmen gezielt einzugehen, auf der anderen ist es jedoch schwierig, die Regeln so aufeinander abzustimmen, daß sie sich nicht widersprechen. Außerdem ist ein einmal mit viel Aufwand gewonnener Regelsatz auch nur für diese eine Anwendung gültig. Ändert sich die Anwendung, z. B. für eine neue Sprache, so beginnt die Arbeit größtenteils von vorn.

Für die Bestimmung der Transkription unbekannter Wörter bietet es sich an, diese zuerst in ihre morphologischen Bestandteile zu zerlegen, da es für die Zuordnung der Grapheme zu den Phonemen eine Rolle spielt, wo sich in der Graphemfolge Morphemgrenzen befinden. So wird z. B. <sch> in *täuschen* als /S/ ausgesprochen, wogegen es in *Häuschen* als /s C/ gesprochen wird, da sich zwischen <s> und <ch> eine Morphemgrenze befindet.

Für eine morphologische Zerlegung ist allerdings viel Expertenwissen nötig, und der Zeitaufwand ist enorm. Man benötigt nicht nur eine Liste der Morpheme einer Sprache, sondern zusätzlich einen Algorithmus, um bei mehrdeutigen Zerlegungen die „richtige“ auszusuchen [Hain96].

Ein System zur Graphem-Phonem-Konvertierung unter Verwendung einer morphologischen Zerlegung wurde von Wothke [Woth93] vorgestellt. Die Verarbeitung erfolgt in zwei Schritten. Zuerst wird das unbekannte Wort in seine morphologischen Bestandteile Präfix, Stamm und Suffix aufgeteilt, wobei bei den Suffixen zwischen drei Gruppen unterschieden wird: Suffixe deutschen Ursprungs, lateinischen oder griechischen Ursprungs und französischen oder englischen Ursprungs. Die Grenze zwischen Teilwörtern wird gesondert markiert, da es besondere Aussprachevarianten gibt, die nur am Wortanfang oder -ende auftreten.

In diesem System gibt es zwei primäre Wissensquellen für die morphologische Zerlegung: ein Morphemlexikon und eine Wortsyntax. Das Morphemlexikon enthält die Morpheme und ihre zugehörigen Klassen. Pro Eintrag sind bis zu 6 Klassen möglich; die durchschnittliche Anzahl von Klassen pro Morphem beträgt 1,87. Das Klassifikationsschema besteht aus 198 verschiedenen Klassen. Die Morpheme werden in

- Präfixe, Stämme und Suffixe sowie
- Verb-, Adjektiv- und Nomenmorpheme

eingeteilt. Zusätzlich werden noch die Eigenschaften Numerus, Kasus, Zeit, Modus, Komparation und Ablaut zugeordnet. Die Wortsyntax beschreibt die Sequenzen von Morphemen, die in der deutschen Sprache erlaubt sind. Damit kann z. B. die Zerlegung des Wortes *Walzer* in den Stamm *Wal* und das Präfix *zer* verhindert werden, da ein Wort nicht mit einem Präfix enden kann.

Die beiden Wissenquellen für die anschließende Bestimmung der Aussprache sind eine Liste von Graphem- bzw. Phonemgruppen und die eigentlichen Zuordnungsregeln der Grapheme auf

die Phoneme. Durch die Verwendung der Graphem- und Phonemgruppen wird die Anzahl der Regeln verringert und das Regelwerk somit übersichtlicher. So wird z. B. der Buchstabe <b> vor <t>, <s> und <k> immer als /p/ gesprochen. Dafür wären im Regelwerk die drei Einträge

```
[b]t -> p
[b]s -> p
[b]k -> p
```

nötig. Faßt man nun die drei Grapheme zur Gruppe /VLCONS/ (voiceless consonants = stimmlose Konsonanten) zusammen, so benötigt man nur noch die eine Regel

```
[b]/VLCONS/ -> p
```

Es gibt 31 solcher Gruppen, z. B. vokalische Buchstaben oder konsonantische Buchstaben, die üblicherweise stimmlosen Konsonanten zugeordnet werden. Die eigentlichen Konvertierungsregeln haben die Form

```
links [Zentrum] rechts -> Phonem1 Phonem2 ...
```

und geben an, welche Phoneme der Graphemfolge im Zentrum bei gegebenem Kontext zugeordnet werden soll. Eine Besonderheit dieses Systems ist, daß bei den Angaben auf der linken Seite der Regeln auch auf die verschiedenen Morphemgrenzen Bezug genommen werden kann. Somit kann für die Transkription eines Wortes auch seine morphologische Struktur in Betracht gezogen werden.

Da die Anwendung für das hier vorgestellte System die Erstellung eines Aussprachelexikons für die Spracherkennung war, wird allerdings eine wichtige Vorraussetzung für die Sprachsynthese nicht erfüllt. Es fehlt eine Entscheidung, welche der gefundenen Zerlegungen bzw. welche der damit generierten Transkriptionen die „richtige“ Aussprache darstellt. Das, was für die Erkennung gewünscht ist, nämlich möglichst viele plausible Aussprachevarianten für ein Wort, ist in der Synthese nicht erwünscht, da pro Wort nur eine Aussprache realisiert werden kann.

Ein etwas anderer regelbasierter Ansatz für eine deutsche Graphem-Phonem-Konvertierung wurde im TTS-System SVOX [Trab95] verfolgt. Zuerst wird jedes Wort in einem Lexikon gesucht, das neben der Aussprache auch syntaktische Eigenschaften enthält. Unterschiedliche Aussprachevarianten eines Wortes können in Abhängigkeit von diesen syntaktischen Eigenschaften gespeichert und zusätzlich noch mit Strafwerten versehen werden. Damit ist es z. T. möglich, das Problem mit Homographen (siehe Abschnitt 1.6.1) zu lösen. Unbekannte Wörter werden erst in ihre morphologischen Bestandteile zerlegt, bevor die Regeln zur Graphem-Phonem-Konvertierung angewendet werden. Diese Regeln sind reguläre Transformationen zwischen den beiden Stufen Oberfläche und Lexikon, und zwar in beide Richtungen. Es ist mit den hier verwendeten Regeln also auch möglich, eine Phonemfolge auf ihre graphemische

Repräsentation abzubilden. Jede Regel kann ebenso wie die Einträge im Lexikon mit einem Strafterm versehen werden, wodurch es möglich wird, mit Hilfe einer Grammatik aus einer Anzahl von mehrdeutigen Möglichkeiten die richtige auszusuchen. Damit stellt sich aber wieder sofort das Problem, diese Werte richtig einzustellen. Traber schreibt dazu: „Das Hauptproblem des grammatikbasierten Ansatzes ist das genaue Einstellen der Strafwerte der Grammatikregeln bzw. der Einträge im Lexikon. Da nur wenige Aussprachekombinationen der Graphemgruppen durch die Stammgrammatik verboten werden, sind die Strafterme sehr wichtig für die Auswahl einer plausiblen Aussprache. Eines der größten Probleme in diesem Kontext ist die Kompositabildung im Deutschen, weswegen es oft schwer ist, eine passende Segmentierung in Morpheme automatisch zu finden, wenn ein oder mehrere Stämme unbekannt sind: mehrsilbige fremde Stämme können oft als Folge von deutschen Stämmen analysiert werden, was zu einer deutlich anderen Aussprache führen kann. Die Auswahl der richtigen Lösung erfordert ein sehr hochentwickeltes System aus Wortstamm-Grammatikregeln und Straftermen.“ ([Trab95], S. 117).

Kommenda [Komm91] unterteilt sein System ebenfalls in die beiden Schritte morphologische Zerlegung und anschließende Bestimmung der Transkription. Sein morphologisches Lexikon enthält ca. 2500 Einträge, wobei es sich dabei nicht um Morpheme, sondern genauer gesagt um Morphe handelt, da es für ein Morphem mehrere Einträge geben kann, wenn es in verschiedenen Varianten auftritt (*Apfel - Äpfel*, *lauf - lief*). Bei vielen Stämmen wurde nicht die Nominativform abgespeichert, wie sie sonst im Wörterbuch zu finden ist, sondern eine reduzierte Form (*flamm* statt *flamme*). Dies erlaubt die ökonomische Beschreibung der flektierten Formen, Derivationen (*entflammbar*) und Kompositionen (*Flammpunkt*). Jeder Eintrag enthält morphologische, phonologische und syntaktische Eigenschaften des Morphs. Zur Angabe der Konjugations- und Deklinationsmuster werden noch grammatikalische Informationen gespeichert, die sich sowohl auf das Morph selber als auch auf die Morphe beziehen, die dem Morphem umittelbar folgen können. Um die Anzahl mehrdeutiger Zerlegungen möglichst gering zu halten, wurde ein sehr ausgefeiltes Klassifikationsschema erstellt. Somit konnten z. B. verschiedene Klassen von Endungen und Fugenelementen beschrieben werden, die ein Stamm oder ein Derivationssuffix bei der Komposition verlangt. Auf diese Weite können solche Fälle wie *erwerb-s-tätig* vs. *erwerbstätig* bzw. *Baum-strunk* vs. *Baum-s-trunk* (im Gegensatz zu *Abschied-s-trunk* vs. *Abschiedstrunk*) richtig behandelt werden.

Während der Zerlegung wird ein Wort so in Teilbuchstabenketten aufgeteilt, daß diese den Elementen aus dem Lexikon entsprechen und lückenlos aneinandergefügt das gesamte Wort ergeben. Die Reihenfolge der untersuchten Alternativen ist durch die Heuristik bestimmt, daß möglichst lange Segmente bevorzugt werden. Der Lösungsweg wird so lange verfolgt, bis das Wortende erreicht ist oder der Algorithmus in eine Sackgasse läuft.

Es entstehen dabei u. U. sehr viele falsche Zerlegungen. Um dies zu vermeiden, wurde zuerst mit einer grammatikalischen Analyse versucht zu entscheiden, ob eine gefundene Zerlegung möglich ist oder nicht. Diese Entscheidung wurde mit Hilfe des sehr ausgefeilten Klassifikationsschemas und einer Strukturformel für den Aufbau beliebiger deutscher Wörter getroffen:

$$\left[P_0^2 + S + D_0^5 + J\right]_0^\infty * P_0^2 + S + D_0^5 + I_0^1 \tag{1.1}$$

mit

X_a^b	mindestens a und maximal b Segmente des Typs X sind zulässig
$+$	Morphemgrenze
$*$	Teilwortgrenze
P	Partikel (i. a. identisch mit den sogenannten untrennbaren Präfixen, z. B. *ent*, *prä*)
S	Stamm
D	Derivationssuffix
J	Fugenelement
I	Flexionsendung

Diese Strukturformel drückt aus, daß ein deutsches Wort aus einem obligatorischen Stamm besteht. Davor können bis zu zwei Präfixe und danach bis zu fünf Derivationssuffixe bzw. eine Flexionsendung folgen. Zwischen den Teilwörtern tritt ein Fugenelement auf. Diese Vorgehensweise erwies sich allerdings als unpraktisch. Deswegen wurde ein Ansatz gewählt, bei dem die grammatikalische Analyse fast ausschließlich durch das Lexikon gesteuert wird. Dazu wird jedem Eintrag nicht nur die eigene grammatikalische Spezifikation mitgegeben, sondern auch die der möglichen Nachfolger. Diese Vorgehensweise ist vergleichbar mit der Übergangsfunktion eines Automaten, der für jeden Zustand die Menge der Folgezustände angibt.

Bei Wörtern, die nicht vollständig zerlegt werden können, springt ein Verfahren namens „Joker“ ein. Hier wird die Tatsache ausgenutzt, daß der Aufbau der Wortsämme selber gewissen Gesetzmäßigkeiten unterliegt. Er kann durch die Strukturformel

$$[K_i+]\,V\,[[+K_m]+V]\,\left[+K_f\right] \tag{1.2}$$

mit

$+$	Verkettung
K_i	initiales Konsonantenkluster
V	Vokalkluster
K_m	mediales Konsonantenkluster
K_f	finales Konsonantenkluster
$[\ldots]$	optionale Elemente

beschrieben werden. So besteht z. B. der Stamm *Strumpf* aus dem initialen Konsonantenkluster *str*, dem Vokal *u* und dem finalen Konsonantenkluster *mpf*. Um auf diese Weise auch die Aussprache bestimmen zu können, wird bei den Vokalklustern zwischen Kurzvokal und Langvokal (oder Diphthong) unterschieden und dementsprechend bei den medialen und finalen Konsonantenklustern zwischen solchen, die auf Kurzvokal bzw. Langvokal/Diphthong folgen können. Dadurch läßt sich z. B. entscheiden, daß das *u* vor dem finalen Konsonantenkluster *mpf* kurz zu sprechen ist. Mit dem Joker können natürlich nicht alle unbekannten Wörter, vor allem Fremdwörter aus dem Englischen oder Französischen, richtig behandelt werden. Er stellt aber zumindest sicher, daß Analyse und Umsetzung eines Wortes nicht global versagen, nur weil Teile davon nicht vollständig analysiert werden können.

Die Rohtranskriprion wird aus den phonemischen Angaben der jeweiligen Morphe zusammengesetzt. Sie wird aber zusätzlich noch in Abhängigkeit vom Kontext modifiziert. So sind z. B. die stimmhaften Plosive /b/, /d/ und /g/ am Ende eines Morphs durch die stimmlosen /p/, /t/ und /k/ zu ersetzen, wenn sie am Wortende oder vor einem konsonantisch beginnenden Morph stehen. Diese Anpassungen nehmen also ebenfalls Bezug auf die morphologische Struktur des Wortes. Für die Bestimmung der Wortbetonung wird ebenfalls auf die phonemische Repräsentation aus dem Lexikon zurückgegriffen. Dabei wird für jedes Teilwort die am weitesten rechts stehende Betonungsmarke beibehalten. Somit können Akzentverschiebungen in derivierten Fremdwörtern (*Form - Format - Formation*) auf einfache Weise erfaßt werden, nämlich durch die Betonungsmarkierung der akzentverschiebenden Suffixe (*-at*, *-ation*) im Lexikon.

Ein vollständig auf Graphemgruppen beruhender Ansatz wird in [Rook87] verfolgt. Diese Gruppen wurden an Hand von Wortlisten bestimmt, wobei nicht alle gefundenen Gruppen auch tatsächlich übernommen wurden (z. B. *ll* aus *Lloyd*), da mit steigender Anzahl der Gruppen auch die Anzahl der möglichen Zerlegungen steigt. Jedes Wort wird in initiale, mediale und finale Vokal- bzw. Konsonantengruppen zerlegt, wobei jedem Graphem die Eigenschaft Vokal oder Konsonant zugeordnet wird. Dabei gibt es Regeln für bestimmte Ausnahmen, z. B.

- nach <q> ist <u> ein Konsonant, sonst ein Vokal.
- <y> ist ein Konsonant:
  - am Wortanfang, wenn ein Vokal folgt, oder
  - zwischen Vokalen, wenn der folgende Vokal kein <e> ist;

  sonst ein Vokal.

Vokalgruppen stellen eine Besonderheit dar, weil sie ein eigenständiges Morphem sein können. Das führt zu Problemen, wenn solche Morpheme aufeinandertreffen, z. B. in *Papageieieigentümer* oder *zweieiig*. Deswegen wird die Länge einer Vokalgruppe auf 6 Grapheme beschränkt. Wird keine Zerlegung gefunden, so wird das Wort buchstabiert. Das System enthält ein Modul zur Affixerkennung, um Morphemgrenzen innerhalb des Wortes zu finden. Dafür werden Wortzerlegungshypothesen aufgestellt. Bei mehrdeutigen Zerlegungen entscheidet sich das Modul mit Hilfe des Kontextes und unter Verwendung von Prioritätswerten für das Affix mit dem höchsten Prioritätswert. Zusätzlich gibt es ein Lexikon mit Ausnahmen.

Die Wortzerlegungshypothesen gehen vom linguistischen Modell deutscher Wörter aus:

$$Wort = \sum_{i=0}^{n} Vorsilbe_i + Stamm + \sum_{k=0}^{m} Nachsilbe_k + \sum_{l=0}^{p} Endung_l \qquad m,n,l = 1,2,3,\ldots \quad (1.3)$$

$$Wortzusammensetzung = \sum_{z=1}^{r} (Wort_z + Fuge_z) + Wort_{r+1} \qquad r = 1,2,3,\ldots \quad (1.4)$$

Es sind maximal 4 Präfixe und 8 Suffixe möglich. Für die ersten 3 Präfixe sind entsprechend der Folgegrammatik alle Präfixe zugelassen; für die 4. Stelle sind nur *be-*, *ge-* und *zu-* erlaubt.

Die Zerlegung in Teilwörter wird sowohl vorwärts als auch rückwärts durchgeführt. Für die Beurteilung der verschiedenen Zerlegungen werden die Häufigkeitswerte der einzelnen Gruppen herangezogen, die an Hand eines Korpus bestimmt wurden. Dabei bevorzugt die Vorwärtszerlegung die Suffixe, die Rückwärtszerlegung dagegen die Präfixe. Eine Folgegrammatik der Präfixe gibt an, welches Präfix auf welches folgen darf. Sie besteht aus einer Liste, in der alle erlaubten Nachfolger aufgeführt sind. Ist die Liste leer, so muß auf das Präfix direkt ein Stamm folgen. Analog dazu gibt es auch eine Folgegrammatik für die Suffixe.

Ein Problem bei der Ermittlung der Wortstruktur sind falsch erkannte und somit zu lange Präfix- und Suffixketten oder die Vertauschung von Präfixen und Suffixen, die zueinander homographisch sind. Dafür werden Zwischenteilworthypothesen aufgestellt. Es erfolgt eine Betrachtung weiterer Teilwörter durch Modifikation der vorhandenen zu neuen Teilwörtern. So liefert die Vorwärtszerlegung *Werbesen - dung* und die Rückwärtszerlegung *Wer - besendung*. Durch Abtrennung des Präfixes *be-* vom zweiten Bestandteil der Rückwärtszerlegung erhält man das Teilwort *sendung*. Der Anfang *sen* dieses Teilwortes befindet sich am Ende des ersten Bestandteils der Vorwärtszerlegung. Nach Abtrennung von *sen* bleibt *Werbe* übrig. Die Folgegrammatik überprüft nun, ob *e* am Ende von *Werbe* als Suffix an dieser Stelle auftreten kann. Da dies möglich ist, erhält man nun die richtige Zerlegung *Werbe - sendung*. Anschließend wird noch eine Bewertung der verschiedenen Hyphothesen durchgeführt. Dafür werden insgesamt 17 Kriterien herangezogen, wobei für die Affixe eine spezielle Punktebewertung eingesetzt wird.

Die Graphem-Phonem-Konvertierung erfolgt auf der Basis der ermittelten Graphemgruppen, denen in einem Lexikon eine Lautschrift zugeordnet ist. Diese Lautschrift wird in Abhängigkeit von den Besonderheiten an den Schnittstellen zwischen den Gruppen bzw. bedingt durch die Position der Silbengrenzen noch angepaßt. Für die Vokale gibt es 3 Gruppen von Regeln: für die Vokaldauer (6 Regeln), für die Abhänigkeit der Vokale von ihrer Umgebung (2 Regeln) und für die Transkription des Graphems <e> (5 Regeln). Regeln für die Konvertierung der Konsonanten beziehen sich z. B. auf die Aussprache von <b>, <d> <g> und <s> in Abhängigkeit davon, ob sie am Silbenanfang stehen oder nicht, auf die Aussprache des Suffixes *-ig* und die Aussprache des Graphems <v> in deutschen und fremdstämmigen Wörtern.

Silbengrenze und Wortbetonung werden ebenfalls regelbasiert bestimmt. Für die Silbengrenzen gibt es 10 Regeln, wobei mögliche Positionen bereits in der Lautschrift des Lexikons enthalten sind. Die Wortbetonung wird zuerst für Präfixe, Suffixe und Wortstämme getrennt bestimmt und dann innerhalb der Teilwörter ermittelt. Im letzten Schritt erfolgt die Bestimmung des Wortakzents entsprechend der Teilwörter innerhalb des Wortes. Es wird zwischen Haupt- und Nebenakzent unterschieden.

1.5.3 Datengetriebene Verfahren zur Graphem-Phonem-Konvertierung

Wie man im vorigen Abschnitt gesehen hat, ist es sehr aufwendig, regelbasierte Systeme manuell zu erstellen. Außerdem erfordert es viel Expertenwissen. Die Idee ist nun, diese Regeln an Hand von Aussprachebeispielen aus einem phonetischen Lexikon automatisch zu erlernen. Das hat gewisse Vorteile. Zum einen hat man nur einmal einen hohen Aufwand, um das Lernsystem zu erstellen. Danach erfolgt die Erstellung des Regelsatzes automatisch, und der zeitliche Aufwand ist fast ausschließlich abhängig von der zur Verfügung stehenden Rechentechnik. Zum anderen kann man die Lernverfahren so konfigurieren, daß sie unter gegebenen Randbedingungen ein optimales System generieren. Kriterien bei der Erstellung eines Transkriptiossystems können z. B. der verfügbare Hauptspeicher oder die Prozessorleistung der Zielplattform sein. Unter diesen Vorgaben manuell ein System zu erstellen kann dagegen sehr aufwendig werden.

Die Aufgabe besteht nun darin, die Aussprachebeispiele aus dem Lexikon automatisch zu analysieren und danach Regeln aufzustellen, mit denen diese Beispiele reproduziert werden können. Dazu müssen die Grapheme wie in Abschnitt 1.5.1 den Phonemen zugeordnet worden sein. Die aus diesen Zuordnungen abgeleiteten Regeln werden abgespeichert und im Anwendungsfall benutzt, um für eine unbekannte Graphemfolge eine Phonemfolge zu erstellen.

Das Abspeichern der gefundenen Regeln kann auf verschiedene Arten geschehen. Im einfachsten Fall werden lediglich die während des Trainings gesehenen Beispiele abgelegt (siehe Abschnitt 1.5.3.1), wobei bei Widersprüchen das am häufigsten aufgetretene Beispiel verwendet wird. Im Anwendungsfall wird dann die Regel benutzt, deren Graphemfolge den geringsten Abstand zur unbekannten Graphemfolge hat. Hier ist das Training vergleichsweise einfach, Rechenaufwand und Speicherbedarf zur Laufzeit sind dagegen sehr hoch. Dafür ist dieser Ansatz gut in der Lage zu generalisieren, also unbekannte Wörter richtig zu transkribieren. Dadurch, daß alle Beispiele unverändert übernommen werden, geht keine Information verloren.

Wird dagegen versucht, aus den Beispielen ein abstraktes Modell zu bilden, ist es genau umgekehrt. Hier ist die Trainingsphase sehr aufwendig, dafür ist aber vor allem der Speicherbedarf deutlich geringer. Man kann dies auch als eine Kompression des verwendeten Trainingslexikons ansehen. Allerdings geht bei der Verallgemeinerung oft Information verloren, so daß es nicht mehr möglich ist, das Ausgangslexikon komplett zu rekonstruieren. Beispiele für diese Vorgehensweise sind die Erstellung von Entscheidungsbäumen (Abschnitt 1.5.3.2) oder das Training neuronaler Netze (Abschnitt 1.5.3.3).

In den folgenden drei Abschnitten werden einige dieser Verfahren näher vorgestellt.

1.5.3.1 Speicherbasierte Verfahren

Ein Beispiel für ein speicherbasiertes (engl. *instance based*) Verfahren ist IB1-IG [BBvdB99]. Der Ansatz besteht darin, ein gegebenes Eingabefenster aus 7 Graphemen einer Klasse zuzuordnen, die das auszugebende Phonem und dessen Betonung festlegt. Es gibt 159 dieser Klassen, also Kombinationen aus Phonemen und Betonungen. Aus dem englischen Celex [Burn90] mit

77565 Wörtern werden insgesamt 675745 Muster erzeugt. Ein solches Muster enthält einen Vektor, der die Eingabegrapheme beschreibt, und einen Wert für die Klassifikation dieses Eingabevektors (Phonem + Betonung). In der Trainingsphase werden die aus dem Lexikon gewonnenen Muster in einer Datenbank abgelegt. Im Anwendungsfall wird ein unbekanntes Muster einer der 159 Klassen zugeordnet, in dem über eine Eignungsfunktion das Muster in der Datenbank gesucht wird, das den geringsten Abstand zu diesem unbekannten Muster hat. Dazu muß für jedes der abgelegten Muster der Abstand berechnet werden, was sehr rechenintensiv ist. Der Abstand zwischen dem unbekannten Muster X und dem aus der Datenbank Y wird berechnet aus

$$\Delta(X,Y) = \sum_{i=1}^{N} w_i \delta(x_i, y_i)$$

mit

N	Anzahl der Werte des Vektors (Fenstergröße)
w_i	Gewicht für den Wert an Position i
$\delta(x_i, y_i)$	$= 0$ für $x_i = y_i$, sonst 1

Die Klassifikation erfolgt durch den k-NN Algorithmus, der die k „nächsten Nachbarn“ sucht, also die k Vektoren aus der Datenbank, die unter Verwendung von $\Delta(X,Y)$ dem unbekannten Vektor X am ähnlichsten sind. Diejenige Klasse, die unter den k Vektoren am häufigsten auftritt, wird dann als auszugebende Klasse angenommen. Meistens wird jedoch der Einfachheit halber $k = 1$ gesetzt, so auch in diesem Ansatz.

Die Gewichte w_i für die jeweiligen Werte des Eingabevektors werden während des Trainings über die Transinformation (engl. *information gain*, daher -IG) errechnet (siehe auch Abschnitt 2.3). Sie repräsentieren die Wichtigkeit bzw. die Bedeutung des Eingabegraphems innerhalb des Fensters. Je mehr Information das Graphem an einer bestimmten Position enthält, desto größer wird der Wert der Abstandsfunktion $\Delta(X,Y)$, wenn die Grapheme an einer Position nicht übereinstimmen.

Der reine speicherbasierte Ansatz liefert gute Ergebnisse im Vergleich zu anderen Verfahren [BBvdB99], ist dafür aber sehr speicher- und rechenintensiv. Dem kann man durch die Verringerung der abgespeicherten Muster entgegenwirken. Dazu gibt es zwei Ansätze:

Auslassen von Ausnahmefällen: Es kann auftreten, daß Regeln niemals als nächster Nachbar anderer Regeln ausgesucht werden oder die Klassifikation sogar stören. In diesem Falle können sie gelöscht werden. Diese beiden Möglichkeiten legen zwei Vorgehensweisen nahe:

- das Löschen aller Regeln, die entfernt werden können, ohne daß sich das Klassifikationsergebnis verschlechtert, und
- das Löschen aller Regeln, deren Klassifikation sich von der häufigsten Klasse der nächsten Nachbarn unterscheidet.

Für den zweiten Fall kann das Verhältnis herangezogen werden, wie oft eine Regel nächster Nachbar einer anderen Regel der gleichen Klasse ist, und wie oft sie der nächste Nachbar einer beliebigen Klasse ist. Ist dieses Verhältnis nahe Null, so sagt das aus, daß diese Regel ein schlechter Repräsentant dieser Klasse ist - sie ist ein Ausnahmefall.

Verallgemeinerung von Regeln: Hier werden Paare oder Gruppen von „nächster-Nachbar-Regeln“ der selben Klasse miteinander verschmolzen und ergeben eine einzige, allgemeinere Regel. Dieser Schritt wird aber nur dann durchgeführt, wenn feststeht, daß die Verschmelzung keinen negativen Einfluß auf die Generalisierungsfähigkeit hat. Obwohl der gesamte Speicherbedarf durch das Verschmelzen der Regeln verringert wird, kann mit den verbliebenen Regeln die gleiche k-NN-basierte Klassifikation durchgeführt werden. Die Abstraktion bei dieser Verschmelzung besteht darin, daß im Nachhinein nicht mehr die Möglichkeit besteht, die ursprünglichen Regeln zu rekonstruieren. Beispiele für diesen Ansatz sind NGE [Salz91], RISE [Domi96] und FAMBL [vdB99].

In [vdBD93] wird eine Methode für das Abspeichern der Lexikoneinträge vorgestellt, bei der nur so viel Kontextinformation berücksichtigt wird, wie auf der Trainingsmenge für eine eindeutige Entscheidung unbedingt nötig ist. Auf der einen Seite gibt es Grapheme, die unabhängig vom Kontext immer nur einem Phonem zugeordnet werden (z. B. `<q>` → `/k/`), aber auf der anderen Seite auch Grapheme, die mehrere Phoneme erzeugen können (z. B. `<c>` → `/C/`, `/x/`, `/k/` und `/S/`). Indem man nun nur die tatsächlich erforderliche Information in Betracht zieht, können Speicherbedarf und auch Rechenaufwand deutlich reduziert werden.

Ausgangspunkt ist erneut ein Lexikon, in dem die Grapheme den Phonemen zugeordnet sind. In diesem Lexikon wird zuerst nach den Graphem-Phonem-Zuordnungen gesucht, die unabhängig vom Kontext sind. Diese werden in einer Tabelle mit der Bezeichnung *0-1-0* abgelegt, was bedeutet, daß weder der linke noch der rechte Kontext vorhanden sind. Danach wird der Kontext erweitert. In welche Richtung diese Erweiterung erfolgt wird wiederum mit Hilfe der Transinformation bestimmt. Da das rechte Nachbargraphem nach dem Graphem im Zentrum den höchsten Wert hat, wird also zunächst dieses Graphem betrachtet. Nun werden alle Zuordnungen im Lexikon gesucht, die unter Betrachtung des Graphems im Zentrum und des rechten Nachbars eindeutig entschieden werden können. Diese Fälle werden in Tabelle *0-1-1* abgelegt. Danach wird der Kontext schrittweise erhöht (*1-1-1*, *1-1-2* usw.), bis alle Zuordnungen abgearbeitet sind. Bei dem holländischen Lexikon, das in der vorgestellten Arbeit verwendet wurde, lag der maximale Kontext bei *5-1-5*. Damit waren 99,5 % aller Zuordnungen abgedeckt.

Die Transkription unbekannter Wörter erfolgt nun durch schrittweises Suchen des gegebenen Graphemkontextes in den Tabellen. Zuerst wird das Graphem in der *0-1-0*-Tabelle gesucht. Wird es dort nicht gefunden, wird es zusammen mit dem ersten rechten Nachbar in der *0-1-1*-Tabelle gesucht usw. Die so gefundenen Phoneme werden dann zur Gesamttranskription zusammengefügt. Für den seltenen Fall, daß in keiner der Tabellen ein Eintrag gefunden wird, wird währdend des Trainings eine *1-1-1*-Tabelle erstellt, die das für diesen Kontext am häufigsten aufgetretene Phonem enthält. Eine dritte Tabelle mit dem Kontext *0-1-0* enthält das Phonem, daß einem Graphem am häufigsten ohne Berücksichtigung des Kontextes zugeorndet wurde.

Diese Tabelle wird verwendet, wenn auch in der *1-1-1*-Tabelle nichts gefunden wurde. Der gesamte Speicherbedarf der Tabellen einschließlich der beiden Ausnahmetabellen betrug 5,8 % des Speichers, der vom Trainingslexikon belegt wurde.

Eine ähnliche Vorgehensweise findet man in [Bags98]. Hier werden ebenfalls zuerst alle im Trainingsmaterial gefundenen Zuordnungen bei einer vorgegebenen Kontextbreite in einer Tabelle abgespeichert. Tabelle 1.1 gibt die Anzahl der Regeln und die erreichten Fehler bei der Graphem-Phonem-Zuordnung in Abhängigkeit von der Kontextgröße an.

Kontextgröße	Graphem-Phonem-Fehler		Anzahl Regeln
	in Prozent	Anzahl	
1	11,56	101723	57377
2	2,80	24662	242388
3	1,05	9264	429247
4	0,44	3908	583708
5	0,22	1895	695491
∞	0,04	384	880022

Tabelle 1.1: Abhängigkeit der Zuordnungsfehler und der Anzahl der Regeln von der Breite des Kontextes (aus [Bags98]). Die Zahl für den Kontext gibt die Anzahl der Grapheme links und rechts vom Zentrum an. Ein Kontext von 1 steht also für 3 Grapheme. Das Lexikon enthält ca. 110000 Wörter mit britischer und amerikanischer Aussprache. Selbst bei „undendlich“ breitem Kontext ist wegen der Homographe keine 100 %ig richtige Transkription zu erzielen.

Diese Regeln werden dann in mehreren Schritten reduziert. Zuerst wird versucht, den linken bzw. rechten Kontext zu verringern, ohne daß sich das Ergebnis verschlechtert. Danach werden Redundanzen in überspezifizierten linken und rechten Kontexten eliminiert. Anschließend werden diejenigen Regeln völlig entfernt, die bei einer Anwendung des Regelsatzes auf den Trainingstest niemals verwendet werden. Bei einem Kontext von 2 und mehr verbleiben danach ca. 80000 bis 110000 Regeln.

Für die Bestimmung der Transkription eines Wortes wird zuerst ein Netzwerk aus plausiblen Graphemzerlegungen erstellt. Dazu werden die Graphemkontexte der gespeicherten Regeln verwendet. Jedem Graphem wird ein Satz von Regeln zugeordnet, deren linker und rechter Kontext mit dessen Umgebung übereinstimmen. Die so gefundenen Regeln können noch reduziert werden, nachdem das Wort vollständig von links nach rechts abgearbeitet wurde. In einem zweiten Schritt wird das Wort von rechts nach links betrachtet und alle Regeln gelöscht, deren rechter Kontext nicht übereinstimmt. Falls in diesem Schritt Regeln entfernt wurden, so ist ein dritter Schritt nötig, der erneut von links nach rechts überprüft, ob noch alle linken Kontexte erüllt sind. Dieser bidirektionale Prozeß ist beendet, wenn keine Regeln mehr eliminiert werden können. Nun wird der beste Pfad durch das Netzwerk gesucht, das von den verbliebenen Regeln aufgespannt wird. Dabei werden jeder Verbindung zwei Werte zugewiesen: eine Bewertung S und ein Strafterm Z. Es wir der Pfad mit dem größten S und dem kleinsten Z gewählt, wobei die Präferenz auf dem kleinsten Z liegt. Die letztendlich generierte Transkription wird dann aus den Regeln entlang des besten Pfades zusammengesetzt.

Ein Verfahren zum induktiven Lernen der Ausspracheregeln wurde Anfang der 90er Jahre von *van Coile* [Coil91] vorgestellt. Das Lernen erfolgt schrittweise. Zuerst werden für jedes Graphem alle Aussprachebeispiele betrachtet und deren Zuordnungshäufigkeiten bestimmt. Danach wird eine Basis-Regel bestimmt, die angibt, welchem Phonem das Graphem ohne Berücksichtigung des Kontextes am häufigsten zugeordnet wurde. Im Anschluß daran wird der Kontext in Betracht gezogen. In einem iterativen Prozeß werden alle Zuordnungen für das betrachtete Phonem untersucht, die noch nicht erfolgreich behandelt werden können. Von diesen Zuordnungen wird diejenige als neue Regel hinzugefügt, bei der die maximale Anzahl der noch nicht erfolgten Zuordnungen richtig behandelt wird. Das Ziel dieser Regelgenerierung war, einen Grundstock an Regeln zu bilden, die im Anschluß an die automatische Verarbeitung manuell von einem Experten weiter verfeinert werden sollten.

1.5.3.2 Entscheidungsbäume

Ein anderer Ansatz für das Abspeichern der Zuordnungen des Trainingslexikons sind Entscheidungsbäume. Dieses Verfahren wird z. B. in [AD94] und [DvdB93] angewendet. Hier wird ähnlich wie bei der Erstellung der kontextabhängigen Tabellen [vdBD93] vorgegangen, mit dem Unterschied, daß die gewonnene Information in einer Baumstruktur abgelegt wird (siehe Abbildung 1.3).

Die Knoten auf der obersten Ebene repräsentieren das Graphem im Zentrum der zu untersuchenden Graphemfolge. Sie sind mit den Knoten der nächsten Ebene verbunden, die für das Graphem mit der zweithöchsten Transinformation stehen, also den rechten Nachbarn des Zentrumsgraphems. Die Knoten der dritten Stufe repräsentieren demzufolge das linke Nachbargraphem usw. Jeder Knoten enthält eine Information darüber, welches Phonem der entsprechenden Graphemfolge am häufigsten in der Trainingsmenge zugeordnet wurde. Verglichen mit dem tabellenbasierten Ansatz aus [vdBD93] entspricht z. B. die dritte Stufe dem Eintrag in der Tabelle *1-1-1*.

Im Anwendungsfall wird die unbekannte Graphemfolge in der Baumstruktur gesucht. So wird z. B. für die Bestimmung des Phonems für Graphem <a> im Word *behave* (siehe Abbildung 1.3) von dem Knoten <a> ausgegangen. Das Graphem mit der nächsthöheren Transinformation ist der erste rechte Nachbar <v>. Danach kommt der erste linke Nachbar <h>, dann der zweite rechte <e> und dann der zweite linke <e>. Die Suche wird abgebrochen, wenn ein Knoten keine weiteren Verzweigungen hat oder es keine Verzweigung für das nächste Graphem gibt. Dann wird das Phonem für das Zentrumsgraphem verwendet, welches in diesem Knoten gespeichert ist (in diesem Beispiel /eI/).

Eine interessante Untersuchung ist in [VPB98] zu finden. Hier wird ebenfalls (basierend auf ID3 [Quin93]) eine Baumstruktur zum Abspeichern eines Aussprachelexikons verwendet. Zur Bestimmung des Wortakzentes wird wie in [BBvdB99] für jeden betonbaren Laut zwischen einer betonten und einer unbetonten Variante unterschieden. Das Verfahren entscheidet sich also gleich an Hand der Graphemeingabe für den betonten oder unbetonten Laut, was bei dem vergleichsweise kleinen Eingabekontext fraglich erscheint. Außerdem widerspricht das der Aus-

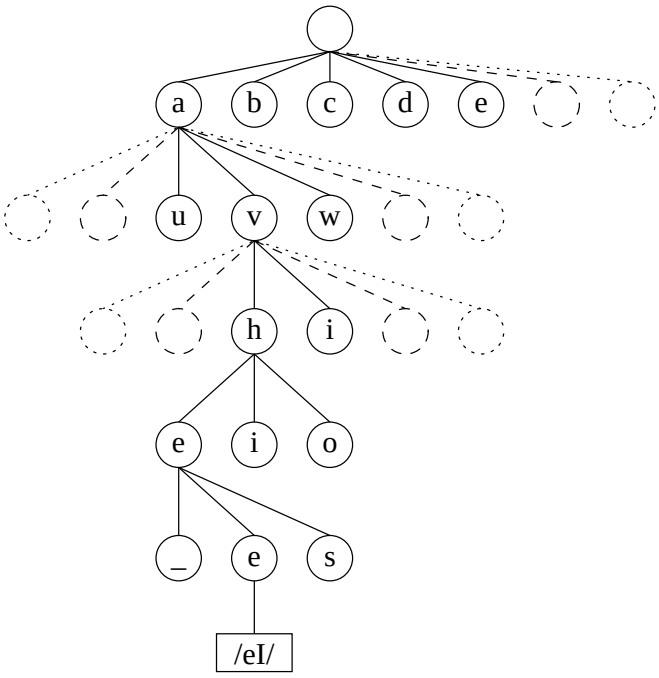

Abbildung 1.3: Beispiel für die Bestimmung des Phonems /eI/ für das Graphem <a> im englischen Wort *behave* mit Hilfe eines Entscheidungsbaumes (aus [DvdB93]). Die Knoten werden, ausgehend vom Graphem im Zentrum, entsprechend der Wichtigkeit des Kontextes durchlaufen, also in der Reihenfolge <a> - <v> - <h> - <e> - <e>.

sage aus Abschnitt 1.4, daß die Position der Betonung erst ermittelt werden kann, wenn die komplette Phonemfolge für das Wort bestimmt wurde. Der Vergleich mit einer schrittweisen Vorgehensweise zeigt aber, daß man auf diese Weise bessere Ergebnisse erzielen kann (siehe Tabelle 1.2).

Werden Phonemfolge und Betonung mit Hilfe zweier unabhängiger Bäume bestimmt, so ist zwar der Prozentsatz richtig transkribierter Wörter ohne Berücksichtigung des Akzentes höher (73,1 % bzw. 69,4 %), aber nach der Bestimmung der Akzentposition durch den zweiten Baum sind die Ergebnisse schlechter als bei der Variante, bei der Laut und Akzentstufe gleichzeitig bestimmt werden (54,6 % bzw. 69,3 %).

Das Insteressante an dieser Arbeit ist die Untersuchung, ob die Richtung, in der man ein Wort transkribiert, Einfluß auf die erzielten Ergebnisse hat. Da der Graphemkontext nicht immer für eine eindeutige Entscheidung ausreicht, wurde untersucht, ob sich die Ergebnisse verbessern, wenn man zusätzlich berücksichtigt, welche Phoneme bei den vorhergehenden Schritten ausgegeben wurden. Dadurch wird zum einen eine Art Rekurrenz erzielt, und zum anderen wird indirekt der Eingangskontext vergrößert. Nun kann man aber natürlich nur die Phoneme benutzten, die man bereits bestimmt hat. Wird das Wort von links nach rechts transkribiert, so stehen nur die Phoneme der linken Seite zur Verfügung. Wenn man aber der Meinung ist, die Phoneme der rechten Seite sind wichtiger als die der linken, dann muß man das Wort eben von rechts

	Phoneme ohne Betonung	Phoneme mit Betonung	Wörter ohne Betonung	Wörter mit Betonung	Baum-größe
2 getrennte Bäume	95,6%	-	73,1%	54,6%	24552 + 103
1 Baum	95,4 %	94,8 %	69,4 %	69,3 %	30368

Tabelle 1.2: Vergleich der Ergebnisse bei der Bestimmung der Phonemfolge mit Akzentpostion (aus [VPB98]). In der ersten Zeile sind die Ergebnisse für den Fall aufgeführt, daß im ersten Schritt nur die Phonemfolge generiert wird, in die dann im zweiten Schritt die Betonung eingefügt wird. Die zweite Zeile enthält die Ergebnisse für die gleichzeitige Bestimmung von Phonem mit Akzentstufe.

nach links transkribieren. Das Ergebnis der Untersuchung ist, daß es bei zwei von drei Lexika besser ist, das Wort von rechts nach links zu transkribieren (siehe Tabelle 1.3).

Lexikon	ohne Phoneme	von lins nach rechts	von rechts nach links
OALD	73,41	75,46	76,66
BRULEX	93,74	94,05	94,43
CMU	59,71	62,79	61,40

Tabelle 1.3: Vergleich der Ergebnisse in Abhängigkeit von der Richtung der Transkription für verschiedene Lexika: Oxford Advanced Learner's Dictionary OALD [Mitt92] (britisches Englisch), BRULEX [CMR90] (Französisch) und CMU [Weid98] (amerikanisches Englisch).

Zum einen ist zu sehen, daß sich die Ergebnisse verbessern, wenn man den Phonemkontext hinzunimmt. Zum anderen stellt sich heraus, daß es bei den Lexika OALD und BRULEX zu besseren Ergebnissen führt, wenn die Wörter von rechts nach links transkribiert werden.

1.5.3.3 Verwendung neuronaler Netze

Eine der ersten Anwendungen ist *NETtalk* von Sejnowski und Rosenberg [SR86] aus dem Jahre 1986. Sie verwendeten ein neuronales mehrschichtiges Perzeptron (*multi-layer perceptron*, MLP), das aus den am Eingang angelegten Graphemen den zu sprechenden Laut generierte. Die Eingabe bestand aus den 26 Buchstaben des Englischen und drei zusätzlichen Einheiten für Zeichensetzung und Wortgrenze. Die Phoneme wurden durch 21 Ausgabeknoten repräsentiert, die Eigenschaften wie Artikulationsstelle, Stimmhaftigkeit, Vokalhöhe usw. kodierten. Fünf zusätzliche Ausgabeknoten enthielten Informationen über Betonung und Silbengrenze. Es wurde ein zusätzlicher Laut verwendet, der ausdrückt, daß der angelegte Buchstabe nicht gesprochen wird wie z. B. in *phone* - `/f-on-/`. Wie dieser Zusatzlaut eingefügt wurde, ist in dem Report nicht beschrieben.

Ein neuronales Netz für Deutsch wurde in [Rose96] vorgestellt. Es handelt sich hier ebenfalls um ein MLP. Zur Generierung der Trainings- und Testmuster wurden das deutsche CELEX und das Lexikon aus dem Projekt ONOMASTICA verwendet. Dieses Lexikon enthält Vor- und Nachnamen sowie Städte- und Straßennamen. Allerdings wurden die Trainings- und Testdaten nicht strikt voneinander getrennt, da erst das Lexikon in zwei Teile zerlegt wurde und dann aus diesen Teilen jeweils die Trainings- und Testmuster erzeugt wurden. Somit können Testmuster auch in den Trainingsdaten enthalten sein und umgekehrt. Die Aufbereitung der Trainingsdaten, also die Zuordnung der Grapheme zu den Phonemen, erfolgte regelbasiert mit einer anschließenden manuellen Korrektur.

Es wurden verschiedene Kodierungsverfahren für die Graphemeingabe vorgestellt:

2 aus 12 Die Buchstaben wurden in 6 Kategorien eingeteilt (siehe Tabelle 1.4). Jede dieser Kategorien enthält bis zu 6 Buchstaben. Für die Kodierung wurde nun angebeben, zu welcher Kategorie der Buchstabe gehört (sechs Neuronen) und welche Position er innerhalb der Kategorie einnimmt (ebenfalls sechs Neuronen). Es sind also immer 2 von 12 Neuronen gesetzt.

1 aus 31 Pro Buchstabe steht ein Neuron zur Verfügung (1-aus-n Kodierung).

5 bit Die 31 Buchstaben werden binär kodiert. Dabei wurde auf eine gewisse Strukturierung geachtet, soweit dies möglich war.

var 7 Hier wurde bei der Kodierung berücksichtigt, daß die Buchstaben im Zentrum des Eingabefensters eine größere Bedeutung haben als die am Rand. Für ihre Kodierung wurden demzufolge mehr Eingabeknoten verwendet (14) als für die am Rand (12 bzw. 9). Die Verteilung der insgesamt 84 Knoten war 9-12-14-14-14-12-9. Die konkrete Kodierung der Positionen erfolgte analog zu *2 aus 12*.

var 9 wie *var 7*, aber mit 9 Neuronen und der Gewichtung 5-9-12-14-14-14-12-9-5.

var 11 wie *var 7*, aber mit 11 Neuronen und der Gewichtung 5-7-9-12-14-14-14-12-9-7-5.

Lautkategorie	zugeordnete Buchstaben
Plosive	b, d, g, k, p, t
Frikative	f, s, ß, v, w
Sonoranten	j, l, m, n, r
Affrikaten	c, q, x, z, ␣
Vordere Vokale	a, e, i, y, ü
Hintere Vokale	o, u, ä, ö, h

Tabelle 1.4: Zuordnung der deutschen Buchstaben zu Lautkategorien (aus [Rose96], Seite 60).

Für die Kodierung der ausgegebenen Phoneme wurden ebenfalls verschiedene Varianten ausprobiert:

6 bit Die 48 verwendeten Phoneme wurden binär mit 6 Bit kodiert. Es wurde ähnlich wie bei der Eingabekodierung *5 bit* versucht, bei der Kodierung eine gewisse Strukturierung zu erreichen.

1 aus 48 Die 1-aus-n Kodierung der 48 Phoneme.

Merkmal Die Phoneme wurden an Hand von 16 phonetischen Eigenschaften kodiert.

Merkmal* Die Kodierung *Merkmal* enthält sehr viele Nullen, die bei dieser Kodierung möglichst sinnvoll durch ± 0.5 ersetzt wurden.

Neben den verschiedenen Ein- und Ausgabekodierungen wurden weitere Parameter variiert, um die optimale Netzwerkarchitektur zu finden:

- die Anzahl der versteckten Neuronen,
- die Anzahl der versteckten Schichten und
- die Breite des Eingabefensters (Anzahl der am Eingang angelegten Buchstaben).

Als optimales System ergab sich ein Netzwerk mit zwei versteckten Schichten mit je 80 Knoten. Die Eingabe bestand aus dem zu transkribierenden Buchstaben mit jeweils 3 Vorgängern und Nachfolgern, also insgesamt 7 Buchstaben. Bei der Kodierung der Eingabeinformation wurde die unterschiedliche Wichtigkeit der einzelnen Buchstaben berücksichtigt (*var 7*). Als beste Ausgabekodierung erwies sich *Merkmal**. Bei diesem System sind 90,05 % der Netzwerkentscheidungen richtig (96,18% richtige Phoneme, 94,13 % richtige Betonungen und 97,36 % richtige Silbengrenzen). Bei einem Test auf 3111 Wörtern aus dem CELEX wurden 44,4 % der Wörter vollständig richtig transkribiert. Um dieses Ergebnis weiter zu verbessern, wurde der Eingabe über eine Rückkopplung die letzten zwei Entscheidungen des Netzes zugeführt. Unter Verwendung von zwei versteckten Schichten mit je 90 Neuronen konnten somit 59,7 % der Wörter richtig transkribiert werden. Zusätzlich konnte durch eine morphologische Vorverarbeitung der Wörter das Ergebnis auf 67,7 % gesteigert werden.

1.5.4 Wortbetonung

Datengetriebene Verfahren zur Bestimmung der Wortbetonung wurden bereits in den vorigen Abschnitten erläutert. Sie sind meistens eng mit der Graphem-Phonem-Konvertierung verbunden. In diesem Abschnitt soll ganz speziell auf den regelbasierten Ansatz eingegangen werden.

Regelbasierte Erklärungen zur Bestimmung der betonten Silbe innerhalb eines Wortes haben eine lange Tradition. Hier sollen beispielhaft einige Regeln für das Deutsche vorgestellt werden. Eine ausführliche Zusammenfassung dazu ist z. B. in [Jess99] zu finden.

Eine bedeutende Rolle bei der Festlegung, welche Silbe betont ist, spielt das sog. Silbengewicht. Man unterscheidet zwischen schweren und leichten Silben, wobei die Definition dieser Gewichtung bei den verschiedenen Autoren variiert. Kriterien dafür sind z. B., ob der Vokal in der Silbe kurz oder lang bzw. ein Diphthong ist, ob die Silbe offen (endet mit Vokal) oder geschlossen (endet mit Konsonant) ist, und sogar, wieviele Konsonanten am Ende der Silbe stehen. Diese Kriterien bekräftigen die Aussage aus Abschnitt 1.4, daß die Bestimmung der Wortbetonung erst nach der Graphem-Phonem-Konvertierung und der Festlegung der Silbengrenzen erfolgen kann[1].

Zudem verwenden einige Autoren getrennte Regelsätze für deutsche und fremdstämmige Wörter, wogegen andere Autoren die Meinung vertreten, es gäbe nur einen allgemeinen Regelsatz für alle Wörter. So stellt die erste Gruppe z. B. die Regel auf, daß deutsche Stämme auf der ersten (betonbaren) Silbe betont werden. Die Betonung der fremdstämmigen Wörter wird dagegen von rechts nach links bestimmt. Es wird also zuerst untersucht, ob die letzte Silbe den Akzent tragen kann. Ist dies nicht der Fall, so wird die vorletzte Silbe untersucht usw. Die zweite Gruppe geht hingegen grundsätzlich davon aus, daß die Bestimmung der Wortbetonung vom Ende des Wortes her festgelegt wird, unabhängig von der Herkunft des Wortes.

Für die in [Jess99] vorgestellten Regeln gilt die zweite Annahme. In dieser Arbeit werden für nicht abgeleitete Wörter (underived) die folgenden Regeln aufgestellt:

Beschränkung auf die letzten drei Silben Eine der am meisten akzeptierten Regeln ist, daß der Wortakzent auf eine der drei letzten Silben des Wortes fallen muß. Wenn der Akzent also weder auf die letzte noch auf die vorletzte Silbe fällt, so liegt er auf der vorvorletzten. Dies ist natürlich nicht der Fall, wenn diese Silbe nicht betonbar ist (wie z. B. in *Abenteuer*). Dann kann der Akzent auch auf der viertletzten Silbe liegen.

geschlossene Penultima[2] Diese Regel besagt, daß die Betonung nicht links von der Penultima liegen kann, wenn diese geschlossen ist (also auf einen Konsonanten endet). Das bedeutet nicht, daß die Penultima in diesem Fall betont sein muß, sondern nur, daß keine der vorhergehenden Silben den Akzent tragen kann. In dieser Betrachtung werden auch Silben, die auf einen Diphthong enden, als geschlossen angesehen. Der Gleitlaut am Ende des Diphthongs zählt also im Deutschen zu den Lauten, die eine Silbe als geschlossen markieren.

letzte Silbe enthält Schwa In diesem Fall wird die vorletzte Silbe betont. Eine Ausnahme für diese Regel liegt dann vor, wenn die letzte Silbe nicht mit einem Konsonanten beginnt (z. B. in *Akazie* oder *Statue*). Dann fällt der Akzent auf die Antepenultima.

Superschwere letzte Silbe und Diphthonge Endet das Wort auf eine „superschwere“ Silbe, d. h. sie enthält einen Langvokal und endet auf mindestens einen Konsonanten (*Komet*) bzw. einen Kurzvokal und endet auf mindestens zwei Konsonanten (*Konzert*), so wird

[1] In Abschnitt 3.4 wird jedoch beschrieben, daß es in der Praxis durchaus sinnvoll sein kann, auf die unsicher vorhergesagten Silbengrenzen bei der Bestimmung der Wortbetonung zu verzichten.

[2] Mit *Penultima* wird die vorletzte Silbe, mit *Antepenultima* die vorvorletzte Silbe bezeichnet.

diese betont. Ein Diphthong in der letzten Silbe (*Abtei*) hat ähnliche Auswirkungen auf die Wortbetonung wie superschwere Silben.

Es werden noch weitere Regeln aufgeführt, z. B. daß die Betonung auf die letzte Silbe fällt, wenn diese auf Vokal-Konsonant (-VC) bzw. auf einen Langvokal (-VV) endet. Jedoch gibt es viele Wörter, für die diese Regeln nicht zutreffen. Zum Teil können die Regeln genauer spezifiziert werden, wenn auf die Graphemebene Bezug genommen wird, was allerdings sehr umstritten ist. Eine mögliche zusätzliche Bedingung für die Betonung der letzten auf -VC endenden Silbe wäre, daß sie auf mehrere Konsonantengrapheme oder <ß> endet (*Barock, Kongreß*). Allerdings gibt es dafür auch wieder Ausnahmen (*Marschall*). Insgesamt ist die Anwendung dieser Regeln doch sehr fraglich.

Unabhängig von der Lautfolge kann die Betonung auch unter Ausnutzung morphologischer Informationen bestimmt werden. So können Affixe bei der Festlegung der Betonung behilflich sein. Die Verbpräfixe z. B. kann man in drei Gruppen einteilen (siehe [Jess99], Seite 535):

- Präfixe, die stets betont sind (*ab-, an-, empor-, nieder-*),
- Präfixe, die niemals betont werden (*be-, ent-, ver-, zer-*), und
- Präfixe, die sowohl betont als auch unbetont auftreten können (*durch-, über-, um-, unter-*).

Für die dritte Gruppe ist die Unterscheidung zwischen abtrennbaren und nicht abtrennbaren Präfixen ausschlaggebend. Damit ist gemeint, ob das Präfix bei der Verwendung des Verbs im Hauptsatz abgetrennt wird (*Ich fahre den Baum um.*) oder vor dem Verb steht (*Ich umfahre den Baum.*). Bei diesen Beispielen ist der semantische Unterschied durch die Orthographie gekennzeichnet. Im Nebensatz (*... der Baum, den ich umfahre.*) ist dies nicht mehr der Fall. Hier wird die Unterscheidung nur noch durch die Betonung realisiert, wobei das abtrennbare Präfix stets betont und das nicht abtrennbare stets unbetont ist. Es ist also zuerst eine semantische Analyse nötig, bevor entschieden werden kann, ob das Präfix betont ist oder nicht.

Eine quantitative Analyse von Wortbetonungsregeln für Deutsch ist in [Wagn01] zu finden, wobei auch der unterschiedliche Einfluß einzelner Regeln untersucht wurde. Ausgangspunkt war dabei die oben vorgestellte Arbeit von Jessen [Jess99].

Zuerst werden die unbetonbaren Affixe abgetrennt, da diese bei der Bestimmung der Akzentposition nicht berücksichtigt werden dürfen. Danach müssen u. U. die Silbengrenzen neu bestimmt werden, um eine richtige Einschätzung über das Silbengewicht durchführen zu können. Als Beispiel ist hier das Wort *geniales* aufgeführt, daß sich aus den Morphemen *genial* und *es* zusammensetzt. Die Transkription dieses Wortes ist `/ g E n - j a: - l @ s /`. Nach dieser Lesart wäre die zweite Silbe leicht. Nach Abtrennung des Suffixes und erneuter Bestimmung der Silbengrenzen ergibt sich `/ g E n - j a: l /` - nun ist die letzte Silbe also schwer.

Weiterhin werden die folgenden vier Regeln verwendet:

Approximantregel Diese Regel behandelt die Fälle, bei denen zwei Vokale an der Silbengrenze zwischen der vorletzten und der letzten Silbe zusammenstoßen, wobei der erste ein hoher Vokal ist (z. B. `/j u: - l i - a: /`). Dann wird dieser vermutlich als Approximant verwendet und steht somit am Anfang der letzten Silbe (`/ j u: l - j a: /`) und kann demzufolge nicht den Akzent tragen.

Schwere-Endsilben-Restriktion Endet ein Wort auf eine schwere Silbe, so wird diese betont.

Regeln der Penultimabetonung Ist die letzte Silbe nicht schwer, so wird die vorletzte betont, falls sie betonbar ist.

Regel der nächsten betonbaren Silbe Wenn keine der bisherigen Regeln zur Anwendung gekommen sind, so wird nächste (von rechts nach links gesehen) Silbe betont.

1.6 Weitere Aufgaben der Graphem-Phonem-Konvertierung

Das in dieser Arbeit vorgestellte System dient zur Bestimmung der Aussprache einzelner Wörter. Es gibt aber noch weitere, darüber hinausgehende Aufgaben.

1.6.1 Homographe

Homographe sind Wörter, für die es verschiedene Aussprachen gibt. So kann das Wort *Lache* mit kurzem oder langem a gesprochen werden, was dann den Unterschied zwischen dem Gelächter und der Pfütze macht. Bei einigen Wortpaaren im Deutschen kann man an Hand der Groß-Klein-Schreibung die Aussprache unterscheiden, z. B. bei *Weg - weg* oder *Sucht - sucht*. Da aber am Satzanfang alle Wörter groß geschrieben sind, kann man diese Information zumindest hier nicht anwenden.

Im Falle des englischen Beispiels *read* kann man mit Hilfe der linguistischen Kategorie die Aussprache festlegen. Handelt es sich um die Präsensform, so wird es mit langem `/i:/` gesprochen. Das Partizip hingegen wird mit kurzem `/E/` realisiert.

1.6.2 Aussprache von Zahlen

Bei der Aussprache von Zahlen gibt es zwei Probleme. Zum ersten muß analysiert werden, um welchen Zahlentyp es sich handelt, um die richtige Aussprache bestimmen zu können. So wird z. B. die Zeitangabe *20:15 Uhr* als *zwanzig Uhr fünfzehn* gesprochen, das Ergebnis oder Verhältnis *20:15* dagegen *zwanzig zu fünfzehn*. Bei langen Zahlenfolgen wie Telefon- oder Kontonummern ist es durchaus sinnvoll, die Zahl nicht einfach ziffernweise vorzulesen, sondern zu gruppieren.

Ein zweites Problem stellt in einigen Sprachen die Aussprache von Ordnungszahlen dar. Sie werden z. B. im Deutschen oder Russischen flektiert wie Adjektive, sind also in Genus, Numerus und Kasus kongruent zu dem bestimmten Substantiv, wodurch sich auch ihre Aussprache ändert. Im Deutschen wird zusätzlich noch zwischen stark und schwach flektiert unterschieden (Zitat Duden-Grammatik). So wird „1.“ in *ein 1. Versuch* als *erster* ausgeprochen, in *der 1. Versuch* hingegen *erste*.

1.6.3 Akronyme und Abkürzungen

Akronyme sind Wörter, die aus den Anfangsbuchstaben mehrerer, zusammengehörender Wörter oder den Teilwörtern längerer Komposita gebildet werden, z. B. *ZDF* aus *Zweites Deutsches Fernsehen*. Das Problem ist hier zu bestimmen, ob sie buchstabiert oder ausgesprochen werden. *ZDF* muß buchstabiert werden, da es nicht ausgesprochen werden kann – es ist kein Vokal enthalten. *UNO* hingegen kann ausgeprochen werden. Es gibt aber auch „aussprechbare“ Akronyme wie *USA*, die buchstabiert werden. Eine Untersuchung über die Aussprache von Akronymen ist z. B. in [TV97] zu finden.

Eine weitere Frage bei der buchstabierten Aussprache ist, wo die Betonung gesetzt werden soll. Für Deutsch und Englisch gilt hier die Regel, daß der letzte Buchstabe betont wird, wobei es natürlich auch Ausnahmen wie *Lkw* gibt.

Ein ähnliches Problem gibt es bei Abkürzungen. Diese stehen für ein Wort oder mehrere Wörter und enthalten die Anfangsbuchstaben der Wörter bzw. Silben. Hier kann die Aussprache nur mit Hilfe eines Lexikons bestimmt werden. Problematisch ist hier ebenso wie bei den Ordnungszahlen, daß Abkürzungen zwar bei der Aussprache flektiert werden, die Flexionsendungen aber nicht im Schriftbild enthalten sind. So kann *u. a.* als *und andere, und anderen, und anderer* usw. gesprochen werden.

1.6.4 Sprachenerkennung

Ein großes Problem für Deutsch ist die Angewohnheit, mögichst viele deutsche Wörter durch ihre englischen Entsprechungen zu ersetzen. Vor allem in der Computerindustrie, aber z. B. auch bei Unternehmensberatern, ist man ohne Grundkenntnisse in Englisch kaum noch in der Lage, die dort verwendeten Texte zu verstehen. Außerdem hatte auch Französisch lange Zeit einen großen Einfluß auf die deutsche Sprache. Während andere Sprachgebiete, z. B. der englischsprachige Raum, Fremdwörter in ihr eigenes Regelsystem einpassen und dann auch dementsprechend aussprechen, werden im Deutschen Fremdwörter möglichst unverändert übernommen, sowohl was die Schreibweise als auch die Aussprache betrifft.

Im Extremfall werden Deklination bzw. Konjugation von Fremdwörtern sogar mit deutschen Affixen durchgeführt, wobei auch noch die deutschen Regeln für die morphologische Zerlegung des Fremdwortes angewendet werden. Als Beispiel sei hier das englische Verb *update* genannt. Das Partizip Perfekt wird gebildet, indem zuerst das Wort in *up* und *date* zerlegt und dann

die deutschen Affixe *ge* und *t* eingesetzt werden. Das Ergebnis ist dann *upgedatet* – für jede Graphem-Phonem-Konvertierung ein nahezu unlösbares Problem.

Für die Textanalyse und somit auch für die Graphem-Phonem-Konvertierung heißt das, daß es bei Wörtern, die nicht im deutschen Aussprachelexikon gefunden werden, durchaus Sinn macht, sie im englischen bzw. im französischen Lexikon zu suchen. Ähnlich wie in der Spracherkennung [RPJ01] könnte man auch versuchen, die Graphemfolge zu untersuchen und die „wahrscheinlichste“ Aussprache zu bestimmen. So sind Wörter mit den Endungen *-heit*, *-keit*, *-lich* und *-ung* vornehmlich deutschen Ursprungs, wobei die Endungen *-lish* eher bei englischen bzw. *-eaux* bei französischen Wörtern vorkommen.

Dies hat aber auch noch ein ganz anderes Problem zur Folge. Wenn man z. B. den englischen Ursprung eines Wortes festgestellt und die englische Phonemfolge erstellt hat, dann muß das TTS-System natürlich auch in der Lage sein, die geforderten Laute zu erzeugen. Bei einem konkatenativen Ansatz müssen also auch diese Laute in der Datenbank enthalten sein. Ein solches System wurde in [THNP+99] vorgestellt.

1.7 Zielstellung

Gegenstand dieser Arbeit ist ein System, das ohne Änderungen in der Architektur in der Lage ist, eine Graphem-Phonem-Konvertierung für verschiedene Sprachen durchzuführen. Das wird dadurch erreicht, daß die sprachspezifischen Teile als externe Wissensquellen abgelegt sind und je nach Sprache in das System geladen werden. Hauptaugenmerk ist dabei die Automatisierung der Anpassung an eine neue Sprache, also die Generierung der ladbaren Wissensquellen ohne manuelle Arbeiten. Dabei soll so wenig wie möglich Zusatzwissen benötigt werden. Einziger Ausgangspunkt für eine neue Sprache ist ein phonetisches Lexikon, das die Wörter (Graphemfolgen) und die ihnen zugehörige Aussprache (Phonemfolgen) enthält. Zusatzwissen kann verwendet werden, um die Anpassung zu vereinfachen oder zu beschleunigen, soll aber keine notwendige Voraussetzung sein.

Ein weiterer Anspruch an das Verfahren ist die vollständige Automatisierbarkeit. Es soll ohne manuellen Eingriff oder Nachbearbeitungen funktionieren. Damit ermöglicht es die Graphem-Phonem-Konvertierung, ohne daß der Bearbeiter die zu behandelnde Sprache überhaupt beherrscht. Einzige Voraussetzung ist ein Aussprachelexikon für jede Sprache, das alle nötigen Informationen enthält, die der datengetriebene Ansatz braucht, um diese Sprache zu erlernen.

Die automatische Anpassung gliedert sich in zwei Teile. Der erste Teil beschäftigt sich mit der Aufbereitung des phonetischen Lexikons derart, daß es für jeden Eintrag die eindeutige Zuordnung der Phoneme zu den sie generierenden Graphemen enthält. Mit Hilfe dieser Zuordnungen können dann die Trainingsmuster für das datengetriebene Lernverfahren erstellt werden. Im zweiten Teil wird versucht, die sich daraus ergebenden Zuordnungsregeln zu lernen. Dabei ist zu beachten, daß im Anwendungsfall die Aufgabe besteht, die Phonemfolge auch für solche Wörter erstellen zu können, die nicht in dem phonetischen Lexikon enthalten sind. Das Le-

xikon darf also nicht auswendig gelernt werden, sondern der Algorithmus muß generalisieren können. Er muß also in der Lage sein, Graphemfolgen richtig zu behandeln, die in dieser Form nicht im Lexikon stehen.

Abgrenzung zu vorigen Arbeiten

Im Gegensatz zu den bisherigen Arbeiten auf dem Gebiet der Graphem-Phonem-Konvertierung wird in dieser Arbeit ein rein datengetriebenes System vorgestellt, welches vollautomatisch die Trainingsdaten aufbereitet, vollautomatisch die neuronalen Netze trainiert, und zusätzlich einen hybriden Ansatz unter Verwendung von Lexika und neuronalen Netzen benutzt, um die sichere Information der Wissensquellen mit der Flexibilität der Ausnahmeroutinen zu verbinden. Die neuronalen Netze werden nicht einfach nur einige Iterationen lang trainiert, sondern gezielt durch Pruningverfahren optimiert. Außerdem wird während des Trainings eine implizite Bewertung der Eingabeinformation durchgeführt, wodurch nicht nur die Ergebnisse verbessert, sondern auch die zum Training erforderliche Zeit deutlich verringert werden kann.

Kapitel 2

Grundlagen datengetriebener Verfahren

In der Einleitung wurde erläutert, warum in der vorliegenden Arbeit nur Algorithmen verwendet werden, die nicht auf manuell erstellten Regelsätzen beruhen. In diesem Kapitel werden die Grundlagen dieser Algorithmen vorgestellt.

In Abschnitt 2.1 werden neuronale Netze als ein Ansatz vorgestellt, mit dessen Hilfe die sprachabhängigen Regeln automatisch gelernt werden. An Hand von Beispielen, die für eine Sprache repräsentativ sind, werden Ausspracheregeln abgeleitet, die dann für die Wörter zur Anwendung kommen, die nicht im Lexikon enthalten sind. Im Abschnitt 2.2 wird kurz auf die dynamische Zeitanpassung eingegangen. Sie wird für die Aufbereitung des Aussprachelexikons benutzt, die Voraussetzung für die Erstellung der Trainingsmuster für die neuronalen Netze ist. In Abschnitt 2.3 werden die Formeln zur Berechnung der Transinformation zusammengestellt. Sie wird als Kriterium zur Bestimmung des optimalen Kontextes für die Eingabe der neuronalen Netze herangezogen.

2.1 Neuronale Netze

Die Theorie der neuronalen Netze wurde in mehreren Standardwerken (z. B. [Hayk99] oder [Zell94]) ausführlich erklärt. Deswegen soll hier nur kurz ihre Funktionsweise vorgestellt werden. Dafür wird ausführlicher auf die Möglichkeiten des Trainingswerkzeugs SENN(tm) [Siem98] eingegangen, das für das Training der Netze in dieser Arbeit verwendet wurde, sowie die Verfahren zum Ausdünnen der Netze und Bewerten der Eingangsinformation, wodurch die erzielten Ergebnisse deutlich verbessert werden konnten.

2.1.1 Grund für die Verwendung neuronaler Netze

Neuronale Netze wurden bisher für die verschiedensten Anwendungen verwendet. In dieser Arbeit wird ihre Fähigkeit zur Mustererkennung ausgenutzt. Während des Trainings lernt das

Netz, ein Graphemmuster auf ein Phonem abzubilden. Es handelt sich um eine Klassifizierungsaufgabe: welcher Phonemklasse soll das angelegte Graphemmuster zugeordnet werden.

Die Aufgabe des neuronalen Netzes für die Graphem-Phonem-Konvertierung ist die Erstellung einer phonetischen Transkription der Wörter oder Wortteile, die nicht im Aussprachelexikon enthalten sind. Ein Vorteil dabei ist, daß mit Hilfe neuronaler Netze hochdimensionale nichtlineare Zusammenhänge mit verhältnismäßig wenig Daten modellierbar sind. Außerdem haben sie eine gut ausgeprägte Generalisierungsfähigkeit. Diese Fähigkeit beschreibt die Klassifikationsleistung auf einem Datensatz, der nicht in der Trainingsmenge enthalten war. Neuronale Netze sind also auch dann in der Lage, Graphemfolgen der richtigen Phonemklasse zuzuordnen, wenn diese nicht während des Trainings angeboten wurden.

Um die Anforderung an das in dieser Arbeit entwickelte Verfahren, nämlich ein multilinguales System zu entwerfen, erfüllen zu können, müssen die verwendeten Algotithmen für jede zu behandelnde Sprache anwendbar sein. Das soll nicht bedeuten, daß die Vorgehensweise für jede Sprache identisch sein muß. Der Algorithmus muß aber einfach auf eine neue Sprache anzupassen sein, und das derart, daß das Laufzeitsystem trotzdem sprachenunabhängig bleibt. Neuronale Netze erfüllen diese Anforderung. Die sprachspezifischen Strukturen sind in der Netzwerkarchitektur selber enthalten, die je nach Sprache variieren kann. Das Laufzeitsystem erkennt an der Struktur des Netzes, welche Eingabe es erzeugen muß und welche Ausgabe es zu erwarten hat. So ist z. B. vorstellbar, daß es in einer Sprache, die keine Silbengrenzen kennt, eben keinen Ausgabeknoten für die Silbengrenze gibt. Das Laufzeitsystem erkennt dies am eingelesenen Netz und paßt sich daraufhin an.

Ein weiterer Vorteil neuronaler Netze ist, daß es leistungsfähige Verfahren zur Optimierung der Netzwerkarchitektur gibt (siehe Abschnitte 2.1.5 und 2.1.7). Damit ist es möglich, dem Netz während des Trainings die Informationen anzubieten, die für die Lösung der Aufgabe sinnvoll erscheinen. Die tatsächlich wichtigen Informationen werden aber während des Trainings automatisch separiert. Der möglicherweise negative Einfluß unnötiger Informationen wird somit verhindert. Dadurch wird auch die Bedeutung von Expertenwissen deutlich verringert.

Auf eine durch Expertenwissen motivierte Kodierung der Eingabedaten sollte verzichtet werden, da das Netz evtl. mit der Kodierung nichts anfangen kann (siehe Abschnitt 1.5.3.3 bei der Arbeit von [Rose96]). Es ist besser, die Information so einfach wie möglich zu präsentieren. Mit Hilfe geeigneter Verfahren (z. B. Weight Decay, siehe Abschnitt 2.1.7) kann während des Trainings eine „Vorverarbeitung“ der Daten durchgeführt werden, die zudem noch optimal an die Daten und die Netzwerkarchitektur angepaßt ist.

2.1.2 Funktionsweise neuronaler Netze

Neuronale Netze bestehen aus einzelnen Neuronen, die zu einem Netzwerk zusammengesetzt sind. Die künstlichen Neuronen sind ihren biologischen Vorbildern nachempfunden. Ein solches Neuron ist in Abbildung 2.1 dargestellt.

Ein Neuron hat ein oder mehrere Eingänge, an denen die Eingabewerte x_i angelegt werden.

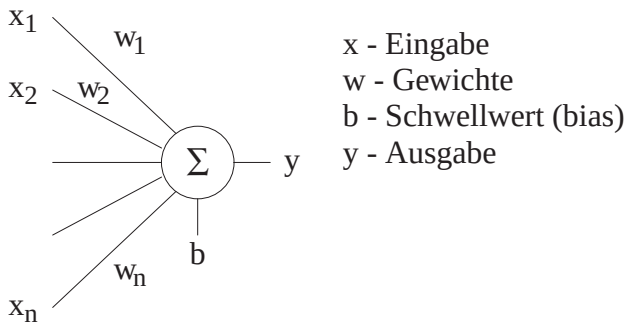

Abbildung 2.1: Darstellung eines Neurons.

Diese Eingabewerte werden mit den zugehörigen Gewichten w_i multipliziert und aufsummiert. Dazu wird der Schwellwert b addiert. Die Ausgabe y des Neurons ist der Wert der Aktivierungsfunktion für diese Gesamtsumme:

$$y = f_{act}\left(b + \sum x_i \cdot w_i\right) \tag{2.1}$$

Durch die Aktivierungsfunktion f_{act} kann man den Wertebereich der Ausgabe beschränken. Mit einer nichtlinearen Aktivierungsfunktion (z. B. tangens hyperbolicus oder die Logistik-Funktion) ist man auch in der Lage, in den Raum, der durch die Muster aufgespannt wird, eine nichtlineare Trennfunktion zu legen.

Die Formel (2.1) wird auf jedes Neuron einer Schicht angewendet. Die Reihenfolge, in der die einzelnen Schichten eines Netzes behandelt werden, wird durch die Verknüpfung der Ausgänge der einen Schicht mit den Eingängen einer anderen Schicht festgelegt.

2.1.3 Allgemeines zum Training neuronaler Netze

Die Aufgabe des Netzes ist, aus den angelegten Eingabewerten Ausgabewerte zu errechnen, die dann von der jeweiligen Anwendung entsprechend interpretiert werden. Während des Trainings lernt das Netz, welche Ausgabewerte es für welche Eingabewerte errechnen soll. Beim Training unterscheidet man zwischen überwachtem und unüberwachtem Lernen.

Beim unüberwachten Lernen gibt es keine Zielmuster, die das Netz lernen soll. Vielmehr besteht die Aufgabe darin, die Eingabemuster in gewisse Gruppen mit ähnlichen Eigenschaften einzuteilen. Im Anwendungsfall legt die Ausgabe des Netzes fest, welcher Gruppe die angelegte Eingabe zugeordnet werden soll. Zu den unüberwachten Lernverfahren zählen z. B. die selbstlernenden Karten von KOHONEN [Koho84].

Beim überwachten Lernen gibt es für jedes Eingabemuster ein Zielmuster, welches für das angelegte Muster errechnet werden soll. Der Trainingsalgorithmus stellt eine Differenz zwischen dem Zielmuster und der vom Netz berechneten Ausgabe fest und paßt mit Hilfe dieser Differenz

die Gewichte der Verbinungen so an, daß die gewünschte Ausgabe erreicht wird. Dazu werden Optimierungsverfahren verwendet, die basierend auf der Gradienteninformation das Minimum in der Fehlerfläche suchen. Die Gewichte werden um einen Bruchteil des negativen Gradienten der Fehlerfunktion verändert:

$$\Delta W = -\eta \nabla E(W) \tag{2.2}$$

mit W als Gewichtsmatrix, η als Lernschrittweite und E als Fehlerfunktion. Die Fehlerfunktion kann z. B. der quadratische Abstand zwischen Zielwert und errechneter Ausgabe sein.

Das Finden des Minimums ist ein Optimierungsproblem, für das es verschiedene deterministische Methoden gibt, z. B. die Methode des steilsten Abstiegs (engl. steepest descent), das Conjugate-Gradient-Verfahren oder das Newton-Verfahren und seine Ableger. Diese Methoden sind aber für das Training neuronaler Netze nur bedingt geeignet, zumindest am Anfang des Trainings, da sie nicht in der Lage sind, einmal gefundene lokale Minima zu verlassen. Sie könnten in dem linken Minimum stecken bleiben und das rechte niemals erreichen. Zu Beginn des Trainings bieten sich deswegen stochastische Verfahren an, die eher dazu in der Lage sind, lokale Minima zu verlassen. Dies sind z. B. *MomentumBackProp* und *VarioEta*.

Die Optimierung aus mathematischer Sicht beschränkt sich auf die Bestimmung der Schrittrichtung und evtl. noch auf die der Schrittweite. Das ist aber für die Optimierung neuronaler Netze oft nicht ausreichend. Die Frage bei der Optimierung neuronaler Netze ist, wie man mit dem Fehler umgeht, den man pro Muster erhält. Bei einer stochastischen Optimierung hat man mehr Möglichkeiten, auf Besonderheiten in der Fehlerfläche einzugehen. Man kann z. B. den Fehler Muster für Muster auswerten, wodurch sich eine sehr gute Anpassung auf lokale Eigenschaften ergibt. Eine andere Variante ist *VarioEta*, bei der die lokalen Eigenschaften nicht so stark berücksichtigt werden, dafür aber globale Eigenschaften stärker in die Optimierung einfließen (siehe Abschnitt 2.1.4.2).

Das Ziel des Trainings muß nicht unbedingt sein, das globlae Minimum zu finden, wenn es sich um ein tiefes, aber steiles Minimum handelt. Dies gilt vor allem dann, wenn man nur wenige Trainingsmuster hat, um das Minimum zu bestimmen. Es besteht die Gefahr, daß ein solches Minimum nur das Optimum auf der Trainingsmenge repräsentiert, denn bereits bei der kleinsten Abweichung befindet man sich weit außerhalb des Minimums an einer vielleicht sehr ungünstigen Stelle in der Fehlerfläche. Für ein Netz, von dem eine gute Generalisierungsfähigkeit erwartet wird, ist die Wahl eines flachen Minimums unter Umständen besser [HS97].

2.1.3.1 Gewinnung der Trainingsmuster

Wichtig für ein erfolgreiches Training ist eine gut durchdachte Aufbereitung der Trainingsdaten. Sie müssen das Problem genau beschreiben, es sollte möglichst wenig Rauschen enthalten sein. Wichtig ist auch, daß sich die Muster für das Training nicht widersprechen, daß es also keine Muster mit identischer Eingabe aber unterschiedlicher Ausgabe gibt.

Weder ein datengetriebener noch ein auf Regeln basierender Algorithmus hat die Mögichkeit, mit sich widersprechenden Mustern umzugehen. Während aber bei einem Regelapparat dann je nach Implementierung die erste oder letzte zutreffende Regel angewendet wird, stört man bei einem datengetriebenen Ansatz das Training. Das Lernverfahren wird versuchen, den Fehler, der durch die Widersprüche entsteht, zu minimieren. Der Fehler kann aber nie Null werden!

Deswegen ist es wichtig, schon während der Erstellung der Lernmuster auf solche Widersprüche zu achten. Sollten sich Muster widersprechen, so kann man versuchen, die Eingabeinformation zu verändern. Eine Möglichkeit wäre, zusätzliche Information zu verwenden, z. B. bei variabler Kontextgröße den Kontext zu erweitern.

Durch eine Kontexterweiterung kann man in den meisten Fällen solche Widersprüche auflösen. Das funktioniert allerdings nicht immer (siehe Tabelle 1.1, Seite 17). So kann es passieren, daß es auch bei einem größeren Kontext noch Widersprüche gibt. Da aber eine Vergrößerung des Kontextes auch eine steigende Zahl von Mustern zur Folge hat und die Widersprüche nicht linear mit der Musteranzahl steigen, verringert sich das Verhältnis von Widersprüchen zur Anzahl der Muster, wodurch man den Einfluß der Widersprüche auf das Training verringern kann. Dies soll am Beispiel eines neuronalen Netzes zur Entscheidung über das Satzende erläutert werden.

Für das TTS-System „Papageno“ wurde ein neuronales Netz erstellt, mit dessen Hilfe entschieden werden kann, ob es sich innerhalb eines Textes beim Satzzeichen „Punkt“ um ein Satzende handelt oder nicht [Hain99b]. Dazu wird vom Tokenizer das spezielle Token „mögliches Satzende“ (engl. possible end of sentence, PEOS) ausgegeben. Das Netz entscheidet nun an Hand der linguistischen Kategorien der Nachbartoken, ob es sich um ein Satzende handelt oder nicht.

Ein Problem bei diesem Netz ist die große Anzahl der sich widersprechenden Eingabe/Ausgabe-Kombinationen. So tritt z. B. die Eingabe

```
INTJ PUNCT NOMEN PEOS NOMEN PREP DET
```

in den 20000 Sätzen des Textkorpus insgesamt 97mal auf, wobei in 83 Fällen das PEOS ein Satzende markiert und in 14 Fällen nicht. Wenn sich das Netz nun für ein Satzende entscheidet, kommen 14 Prozent der „Fehler“ allein aus dem Korpus. In Tabelle 2.1 ist die Anzahl der widersprüchlichen Muster in Abhängigkeit der Größe des linken und rechten Kontextes angegeben.

Das Verhältnis der sich widersprechenden Muster zur Gesamtanzahl der Muster sinkt mit größer werdendem Kontext. Bei einem Kontext von einem linken und einem rechten Token treten insgesamt 183 verschiedene Muster auf, wobei es bei 29 Mustern Widersprüche gibt. 16 Prozent der vorhandenen Muster sind also nicht eindeutig. Bei einem Kontext von 4 linken und 5 rechten Token gibt es 17.098.182 verschiedene Muster mit 12480 Widersprüchen. Die absolute Anzahl von Widersrüchen ist zwar deutlich gestiegen, der prozentuale Anteil aber deutlich auf gerade einmal 0,07 Prozent gesunken. Ihr Einfluß auf das Trainingsergebnis wird somit stark reduziert.

Kontext links	Kontext rechts	Anzahl der Muster	Anzahl der Widersprüche	Verhältnis in Prozent
1	1	183	29	15,85
1	2	1615	112	6,93
1	3	9805	434	4,43
1	4	38658	1100	2,85
1	5	104728	1754	1,67
2	1	1447	122	8,43
2	2	13269	311	2,34
2	3	80069	1336	1,67
2	4	301072	3333	1,11
2	5	747902	4569	0,61
3	1	9248	415	4,49
3	2	82953	810	0,98
3	3	496371	3231	0,65
3	4	1830297	7457	0,41
3	5	4384497	9230	0,21
4	1	40464	1521	3,76
4	2	336875	1863	0,55
4	3	1984368	5318	0,27
4	4	7240661	10806	0,15
4	5	17098182	12480	0,07

Tabelle 2.1: Anzahl und Verhältnis der widersprüchlichen Muster in Abhängigkeit vom Kontext. Je größer der Kontext wird, desto mehr Widersprüche gibt es zwar, aber ihr relativer Anteil sinkt.

2.1.3.2 Kodierung der Ausgabeinformation

Während des Trainings wird an den Eingabeknoten ein Muster angelegt und mit den aktuellen Gewichten die Ausgabe berechnet. Der daraus resultierende Fehler wird vom Lernverfahren zur Adaption der Gewichte verwendet. Das Netz wird sozusagen dafür bestraft, daß es einen Fehler gemacht hat. Die Größe der Strafe ist abhängig vom gemachten Fehler. Wenn aber die Ausgabe mehr als eine Information enthält, z. B. Knoten für Gruppierung, Phonem und Silbengrenze, und diese Informationen durch eine unterschiedliche Anzahl von Knoten repräsentiert werden, dann besteht die Gefahr, daß einige Knoten den Fehler dominieren, und andere bei der Berechnung des Fehlers nicht ausreichend berücksichtigt werden. Das kann man z. B. durch eine explizit vorgegebene Gewichtung umgehen. Eine andere Möglichkeit ist, die bisher unterrepräsentierten Knoten zu vervielfachen, bis ihre Anzahl der Anzahl der anderen Knoten entspricht. Damit vergrößert man allerdings das Netz u. U. beträchtlich.

2.1.4 Training neuronaler Netze im SENN

Bei der Minimierung des Trainingsfehlers hat man im SENN drei Parameter. Dies ist zum ersten die Art und Weise, wie aus der Menge der vorhandenen Muster diejenigen ausgewählt werden, die zur Berechnung des Fehlers herangezogen werden sollen. Zum zweiten stellt sich die Frage, wie man die Richtung bestimmt, in der das Minimum der Fehlerfläche gesucht werden soll, und drittens muß eine sinnvolle Schrittweite η festgelegt werden.

2.1.4.1 Auswahl der Trainingsmuster

Im SENN sind dafür zwei Verfahren vorgesehen. Das erste ist das *TrueBatch*-Verfahren, bei dem der „echte“ Gradient berechnet wird, da wirklich alle vorhandenen Trainingspattern zur Berechnung der Suchrichtung genutzt werden. Das zweite ist das *Stochastic*-Verfahren, bei dem immer nur ein Teil M der Muster zur Berechnung des Gradienten verwendet wird:

$$\nabla E^M(w) = \frac{1}{|M|} \sum_{t \in M} \nabla E^t(w) \tag{2.3}$$

wobei $|M|$ die Anzahl der Elemente von M angibt. Dieser Parameter wird im SENN als *Batchsize* bezeichnet. Bei dieser Art der Musterauswahl wird schon nach der Behandlung von $|M|$ Mustern die Gewichtsmatrix angepaßt.

Für die Bestimmung der Menge M aus der Menge T der Trainingsmuster stehen wiederum mehrere Möglichkeiten zur Auswahl:

Sequential: hier werden im ersten Schritt die ersten $|M|$ Muster ausgewählt, im zweiten Schritt die zweiten $|M|$ usw.

Permute: bei der ersten Auswahl werden per Zufallsgenerator die ersten $|M|$ Muster ausgewählt, die bei der nächsten Auswahl für diese Iteration aber nicht mehr zur Verfügung stehen. Bei der zweiten Auswahl werden also $|M|$ Muster aus $|T| - |M|$ Trainingsmustern ausgewählt usw. Dieses Verfahren nennt man auch „Ziehen ohne Zurücklegen“.

Subsample: es wird eine bestimmte Prozentzahl von Muster gleichverteilt ausgewählt.

Random: hier werden ebenfalls zufällig $|M|$ Muster ausgewählt, aber im Unterschied zu **Permute** „mit Zurücklegen“.

ExpRandom: funktioniert ähnlich wie **Random** mit dem Unterschied, daß die jüngsten Muster mit einer höheren Wahrscheinlichkeit ausgewählt werden als die älteren.

2.1.4.2 Bestimmung der Suchrichtung

OnlineBackProp verwendet in der Variante *TrueBatch* den negativen Gradienten $-\nabla E(w^i)$ und bei *Stochastik* dessen Approximanten $-\nabla E^M(w^i)$. Dieses Verfahren wird als Methode des steilsten Abstiegs (engl. steepest descent) bezeichnet.

MomentumBackProp Hier wird die Suchrichtung als Linearkombination aus dem aktuellen Gradienten und der letzen Suchrichtung bestimmt. Die neue Suchrichtung ist also nicht nur abhängig von der Stelle, an der man sich gerade befindet, sondern auch von dem Weg, auf dem man zu dieser Stelle gelangt ist.

VarioEta Dieses Verfahren ist eine stochastische Approximation des Quasi-Newton-Verfahrens. Es berücksichtigt den Charakter der Fehlerfunktion als Mittel von K Teilfehlern stärker als die anderen Verfahren. Man kann den Fehler, den ein einzelnes Muster erzeugt, als eine „Kraft“ interpretieren, die an einem Gewicht „zieht“, um es zu ändern. Wenn diese Werte für die Muster sehr unterschiedlich sind, sich also die Muster in der Bewertung eines Gewichtes uneinig sind, so ist die Information, die dieser Fehler liefert, schwächer zu beurteilen. In diesem Fall scheint es sinnvoll, die Anpassung des Gewichtes geringer ausfallen zu lassen. Auf *VarioEta* wird im nächsten Abschnit noch genauer eingegangen.

2.1.4.3 Bestimmung der Lernschrittweite

Üblicherweise wird vom Benutzer eine feste Lernschrittweite η eingestellt. Bei einigen Verfahren zur Bestimmung der Suchrichtung bietet es sich aber an, η in Abhängigkeit der vorgefundenen Fehlerfläche optimal zu bestimmen. Dieses Verfahren zur Bestimmung der Lernschrittweite wird im SENN als *LineSearch* bezeichnet.

Im Gegensatz zur festen Schrittweite testet man hier verschiedene Werte für η, um eine optimale Schrittweite zu bestimmen. Entlang des negativen Gradienten werden iterativ verschiedene Werte für η ausprobiert. Dies ist ein sehr rechen- und zeitaufwendiges Verfahren, das durch mehrere Parameter gesteuert werden kann. Zudem hat es noch den Nachteil, daß man mit LineSearch niemals ein lokales Minimum verlassen kann, da es die Gewichte *immer* in Richtung des Minimums verändert. Das bedeutet aber auf der anderen Seite, daß es garantiert das lokale Minimum findet. Im SENN-Handbuch steht dazu: „Es bohrt sich quasi in ein Minimum hinein.“

Wie in Abschnitt 2.1.4.2 schon angedeutet wurde, ist *VarioEta* auch ein Verfahren, bei dem die Lernschrittweite an die aktuelle Fehlerfläche angepaßt wird. Dies soll in Abbildung 2.2 verdeutlicht werden.

Die Fehlerfläche soll hier durch ein Tal dargestellt werden, das in Richtung w_1 schwach und in Richtung w_2 stark abfällt. Ziel ist es, das Minimum in der Mitte des Tales zu finden. Am Startpunkt ist der Gradient in Richtung w_2 groß, der in Richtung w_1 dagegen klein. Bei einem

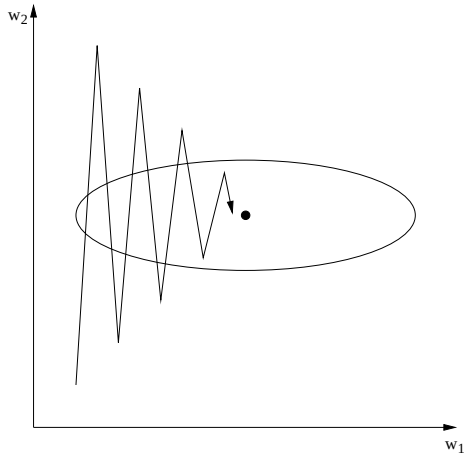

Abbildung 2.2: *VarioEta* (aus [NZ98], S. 396).

konstanten η würde das dazu führen, daß eine „starke“ Kraft in Richtung w_2 wirkt und der Algorithmus so über das Ziel hinausschießt. Nach der Anpassung der Gewichte würde man sich auf der anderen Seite des Tales wiederfinden. Die Anpassung in Richtung w_1 ist dagegen gering, obwohl eigentlich in diese Richtung ein viel weiterer Weg zurückzulegen wäre.

Abhilfe wird dadurch geschaffen, daß die Lernschrittweite mit einem Korrekturterm β

$$\beta = \frac{1}{\sqrt{\sum (g_t - g)^2}} \tag{2.4}$$

mulitpliziert wird. Dabei sind g_t der Gradient für ein bestimmtes Muster und g der Gradient über alle Muster. Somit kann der Zickzack-Kurs in Richtung des Minimums abgeschwächt werden, da sich die Gradienten in Richtung w_1 aufaddieren, die in Richtiung w_2 jedoch einander aufheben.

2.1.4.4 Weitere nützliche Parameter

Es gibt noch zwei weitere Parameter, die während des Trainings verwendet werden:

Tolerance gibt bei Ausgabeknoten an, bis zu welchem absoluten Wert die errechnete Ausgabe von der Zielausgabe abweichen kann, ohne daß der Fehler zu einer Gewichtsanpassung führt. So kann man z. B. bei einer 1-aus-n Kodierung der Ausgabe mit den Zielwerten 1 bzw. -1 durchaus zulassen, daß ein Wert $> 0,9$ bzw. $< -0,9$ immer noch als richtig gewertet wird (siehe Abschnitt 3.6.2).

DerivEps hilft zu verhindern, daß im Sättigungsbereich der Aktivierungsfunktion deren erste Ableitung Null wird, indem er zum Wert der ersten Ableitung addiert wird.

2.1.5 Ausdünnverfahren für neuronale Netze im SENN

Das Ausdünnen (engl. Pruning) neuronaler Netze dient dazu, die Komplexität des Netzwerks durch Optimierung der Netzwerkstruktur zu verringern. Wichtig dabei ist, daß diese Verringerung gezielt erfolgt, um die bereits erlernte Struktur nicht zu zerstören. Die hier vorgestellten Ausdünnverfahren sind ausführlich in [RZ94] oder [NZ98] beschrieben.

In einer Netzwerkstruktur gibt es grundsätzlich drei Klassen von Gewichten:

1. Gewichte, die etwas Sinnvolles gelernt haben und somit einen Beitrag zur Entscheidung des Netzes liefern,
2. Gewichte, die vermeintlich etwas Sinnvolles gelernt haben und nur die Überanpassung an die Trainingsdaten repräsentieren, und
3. Gewichte, die (auf der Trainingsmenge) nicht stören und somit gelöscht werden können.

Aufgabe des Trainings ist nun, die Gewichte der Klassen 2 und 3 zu eliminieren. Durch das Ausdünnen werden die Gewichte aus dem Netzwerk entfernt, die nur vermeintlich etwas Sinnvolles gelernt haben (Klasse 2). Verfahren dafür werden in den nachfolgenden Abschnitten vorgestellt. In Abschnitt 2.1.7 wird die Behandlung der Gewichte der Klasse 3 erläutert.

2.1.5.1 Node-Pruning

Bei diesem Verfahren werden Eingabeneuronen oder Neuronen aus der versteckten Schicht gelöscht. Für jedes Neuron wird eine Sensitivitätsanalyse durchgeführt. Das Ziel des Verfahrens ist, ein Neuron zu finden, dessen Entfernung aus der Struktur eine möglichst große Senkung des Netzwerkfehlers hervorruft.

2.1.5.2 Weight-Pruning

Das Ausdünnen von Gewichten dient zur Begrenzung der Zahl der freien Parameter des Netzes, um die Überanpassung an Datenrauschen zu verhindern. Es wird versucht, die relevanten von den irrelevanten Gewichten zu trennen. Dazu werden alle Gewichte mit einem Testwert versehen, der ihre Bedeutung für das Netz widerspiegelt. Ein hoher Wert steht für eine große Bedeutung, ein kleiner für eine geringe Bedeutung. Die Gewichte mit kleinen Testwerten werden somit zuerst entfernt.

Bei einigen Methoden kann es passieren, daß negative Testwerte vergeben werden. Für aktive Gewichte läßt sich dann keine Aussage über dieses Gewicht treffen; es wird grundsätzlich nicht entfernt. Für bereits gelöschte Gewichte jedoch ist dies ein Hinweis darauf, daß sie nun doch wieder benötigt werden. Im SENN besteht bei einigen Ausdünnverfahren die Möglichkeit, gelöschte Gewichte mit negativen Testwerten wieder zu aktivieren.

Im folgenden werden einige der im SENN implementierten Weight-Pruning-Verfahren kurz vorgestellt.

Wt-Pruning Dieses Verfahren beruht auf der Annahme, daß nur die Gewichte mit großen Werten eine Rolle spielen, weil über sie „starke“ Signale vermittelt werden. Kleinere Gewichte führen dagegen zu Instabilität oder repräsentieren das Rauschen in den Daten. Sie werden deswegen gelöscht. Zur Berechnung des Testwertes wird das Gewicht quadriert. Dadurch ist das nachträgliche Einfügen bereits gelöschter Gewichte natürlich nicht möglich, da es keine negativen Testwerte geben kann.

S-Pruning Das statistische Pruningverfahren berücksichtigt für die Bestimmung des Testwertes nicht nur das Gewicht selber, sondern setzt es in Beziehung zu seiner Stabilität im Backprogagation-Schritt. Dies bedeutet mathematisch gesehen, daß auch die Verteilung der Gewichte berücksichtigt wird. Dafür wird für jedes Gewicht, auch für die bereits gelöschten, seine jeweilige Signifikanz errechnet. Dieses Verfahren bietet demnach die Möglichkeit, bereits ausgeschaltete Gewichter wiederzubeleben, wenn dies sinnvoll erscheint.

OBD-Pruning OBD steht für Optimal Brain Damage. Hier wird die zweite Ableitung der Fehlerfunktion berechnet, um über die Entfernung eines Gewichtes zu entscheiden. Dabei wird für jedes Gewicht separat mit Hilfe der Taylorentwicklung berechnet, welche Veränderung des Fehlers durch eine kleine Verschiebung Δw des Gewichtes hervorgerufen würde. Eine Voraussetzung für dieses Verfahren ist die Annahme, daß man sich bereits im Minimum befindet, was zu Fehlentscheidungen führen kann, wenn dies nicht der Fall ist. Bei diesem Verfahren gibt es wiederum keine Möglichkeit, bereits gelöschte Gewichte wiederzubeleben.

EBD-Pruning EBD-Pruning (Early Brain Damage) setzt nicht voraus, daß man sich im Minimum befindet und kann deswegen zu einem früheren Zeitpunkt als beim OBD-Pruning eingesetz werden. Hier wird mit Hilfe der Taylorapproximation berechnet, wie groß der Fehler bei Nullsetzen des Gewichtes ist und wie groß er bei der bestmöglichen Wahl des Gewichtes wäre [Hain00a]. Die Differenz der beiden Werte wird als Testgröße genutzt. EBD-Pruning erlaubt das Wiederbeleben von ausgeschalteten Gewichten.

2.1.5.3 Node-Merging

Bei diesem Verfahren werden hochkorrelierte Neuronen miteinander verschmolzen, um die Netzwerkstruktur zu optimieren. Es kann sowohl für die Eingabe- als auch für die versteckten Neuronen durchgeführt werden. Bei der Anwendung auf die Eingabeknoten werden die Trainingsdaten daraufhin untersucht, ob sie redundante Neuronen enthalten.

Eine zu starke Korrelation der Eingabewerte kann die Modellbildung erheblich behindern, da der Lernalgorithmus schlechter konvergiert und auch die Ausdünnverfahren zur Optimierung der Netztopologie nicht so gut funktionieren. Während des Trainings kann sich herausstellen, daß einige Neuronen der versteckten Schicht zu stark korrelieren. Dies kann z. B. auftreten, wenn zuviele Neuronen für die versteckte Schicht verwendet wurden oder zuwenig Trainingsdaten vorhanden sind. Node-Merging bietet nun die Möglichkeit, solche Neuronen zu einem Neuron zu verschmelzen.

2.1.6 Vorgehensweise beim Ausdünnen

Die Aufgabe des Trainings ist das Finden des globalen Minimums in der Fehlerfläche. Diese Suche kann man sich so vorstellen, daß man ausgehend von einem Startpunkt, der durch das Initialisieren der Gewichte mit Zufallswerten festgelegt wird, den Punkt sucht, der auf der Trainingsmenge den geringsten Fehler erzeugt. Der gefundene Punkt muß aber nicht zwangsläufig das globale Minimum für alle verfügbaren Daten sein. Das gesuchte Minimum soll aber nicht nur die Trainingsdaten repräsentieren, sondern zur besten Generalisierungsfähigkeit auf allen Daten führen.

Das Ausdünnen kann dazu verwendet werden, die optimale Netzwerkstruktur zu finden. Das Ziel besteht darin, in der Netzwerkstruktur die Elemente zu ermitteln, die keinen oder evtl. sogar einen negativen Einfluß auf das Ergebnis haben. Dabei gibt es grundsätzlich zwei Möglichkeiten.

Beim „early stopping“ beginnt das Pruning bereits nach wenigen Iterationen, solange die Gewichte noch kleine Werte haben. Ein möglicher Zeitpunkt dafür ist, wenn sich der Fehler auf der Testmenge verschlechtert während er auf der Trainingsmenge weiter fällt. Die Information, daß das Netz auswendig lernt, dient als Hinweis dafür, daß die Netzstruktur zu groß gewählt wurde. Demzufolge scheint es sinnvoll, einige Gewichte zu löschen, um die Komplexität zu verringern. Nach dem Ausdünnen werden die verbliebenen Gewichte initialisiert, und das Training wird neu gestartet. Ist das Netz auch mit dieser Anzahl von Gewichten in der Lage, auswendig zu lernen, werden erneut Gewichte ausgedünnt. Diese Prozedur wird fortgesetzt, bis es nicht mehr zum Übertrainieren kommt. Das Netz, das am besten generalisieren kann, wird als bestes Netz beibehalten.

Beim „late stopping“ wird das Netz gezielt übertrainiert. Das Pruning beginnt, wenn sich der Fehler auf der Trainingsmenge nicht mehr verringert. An dieser Stelle hat das Netz zwar typischerweise auswendig gelernt. Das heißt aber auch, daß das Netz nun die maximale Nichtli-

nearität gelernt hat, die aus den Daten ableitbar ist. Es hat also auch versucht, nichtlineare Zusammenhänge zu erlernen, was ja eigentlich auch gewünscht ist. Für lineare Zusammenhänge braucht man keine neuronalen Netze zu trainieren.

Nach einem Ausdünnschritt dürfen die verbliebenen Gewichte nicht initialisiert werden. Durch das Training wurden in die Gewichte eine gewisse Struktur gebracht, in der einige Verbindungen überflüssig sind. Diese Verbindungen werden durch das Ausdünnen gelöscht. Die gelöschten Verbindungen sind aber nur für die ausgebildete Struktur überflüssig. Wenn die Gewichte neu initialisiert werden, müßte das Netz wieder genau zu dieser Struktur kommen. Das ist unwahrscheinlich, und ob die gelöschten Verbindungen auch bei einer anderen Struktur sinnvollerweise fehlen ist fraglich.

2.1.7 Gewichtezerfall

Der Gewichtezerfall (engl. weight decay) ist eine Möglichkeit, unnötige Gewichte (Klasse 3 in der Beschreibung in Abschnitt 2.1.5) in einem Netzwerk zu finden und zu eliminieren. Dadurch wird die Generalisierungsfähigkeit des Netzes verbessert [KH95]. Das Prinzip besteht darin, daß Gewichte, die innerhalb des Netzes nicht benötigt werden, gegen Null gedrückt werden, wodurch sie ihren Einfluß verlieren. Das geschieht durch das Hinzufügen eines Strafterms zur eigentlichen Fehlerfunktion E_0 :

$$E(w) = E_0(w) + \frac{\lambda}{2}\sum w^2 \tag{2.5}$$

Wird für die Anpassung der Gewichte ein Gradientenverfahren genutzt, so ergibt sich nun der Wert w_+ für das neue Gewicht aus

$$w_+ = w - \eta\frac{\partial E}{\partial w} \tag{2.6}$$

und mit Gleichung (2.5)

$$\frac{\partial E}{\partial w} = \frac{\partial E_0}{\partial w} + \lambda w \tag{2.7}$$

errechnet sich w_+ aus

$$w_+ = w - \eta\lambda w - \eta\frac{\partial E_0}{\partial w} = (1-\eta\lambda)w - \eta\frac{\partial E_0}{\partial w} \tag{2.8}$$

Wenn sich während des Trainings herausstellt, daß ein Gewicht keinen Beitrag zur Entscheidung liefert, wenn also der zweite Teil der Differenz

$$\eta \frac{\partial E_0}{\partial w}$$

sehr nahe Null ist, dann gilt für w nach n Adaptionsschritten

$$w_n = (1 - \eta\lambda)^n w_0. \tag{2.9}$$

w wird also ständig mit einer Zahl im Bereich $0 \ll x < 1$ multipliziert und somit schrittweise gegen Null gedrückt. Diese Verringerung auf Null wird nur verhindert, wenn die partielle Ableitung betragsmäßig deutlich größer als Null ist, wenn das Gewicht also „gebraucht“ wird.

Einen zweiten Vorteil dieser Verschiebung gegen Null stellt die Tatsache dar, daß diese kleinen Gewichte bei einem Pruning einen kleinen Testwert liefern und somit bevorzugte Kandidaten für das Löschen sind.

Mit Weight Decay kann die dritte Klasse der Gewichte eliminiert und die zweite für das Pruning „vorbereitet“ werden. Diese zweite Klasse könnte ebenfalls gelöscht werden. Allerdings wäre dafür ein relativ großes λ nötig, was ganz allgemein zu Gewichten nahe Null und somit zu einem linearen Modell führen würde, was natürlich nicht gewünscht ist.

Ein Nachteil vom Ansatz nach Gleichung (2.5) ist die Tatsache, daß korrelierte Eingabeknoten gleichmäßig gegen Null gedrückt werden, statt einen der Knoten auszublenden und den anderen beizubehalten. Das liegt daran, daß bei der Minimierung der Fehlerfunktion zwei gleich große Gewichte ein besseres Ergebnis liefern als ein großes und ein kleines Gewicht. Das soll am Beispiel eines Netzes mit zwei Gewichten näher erläutert werden.

Bei einem solchen Netz berechnet sich die Ausgabe y aus

$$y = \tanh(x_1 w_1 + x_2 w_2). \tag{2.10}$$

Für die Fehlerfunktion E ergibt sich bei zwei ähnlich großen Gewichten

$$E = w_1^2 + w_2^2. \tag{2.11}$$

Würde sich das Verfahren dafür entscheiden, die beiden Gewichte zusammenzulegen und zu einem Gewicht zu vereinigen, wobei das zweite Gewicht auf Null gesetz wird, so ergibt sich

$$E = (w_1 + w_2)^2 + 0^2 \tag{2.12}$$

Ein Vergleich von (2.11) und (2.12) zeigt, daß die erste Variante den kleineren Wert liefert[1]:

[1] Dies gilt natürlich nur unter der Annahme, daß die hoch korrelierten Gewichte das gleiche Vorzeichen haben.

$$w_1^2 + w_2^2 < w_1^2 + 2w_1w_2 + w_2^2 \tag{2.13}$$

Da es um die Minimierung der Fehlerfunktion geht, wird die Variante mit den beiden ähnlich großen Gewichten bevorzugt. Das ist aber bei korrelierten Eingabeknoten nicht erwünscht. Besser wäre es, wenn nur einer der Knoten erhalten bliebe.

Das kann man erreichen, indem man beim Strafterm eine Potenz kleiner als 1 verwendet. Allgemein kann man schreiben

$$E = E_0 + \frac{\lambda}{p}\sum |w|^p \tag{2.14}$$

Nun ergibt ein Vergleich der beiden Varianten für korrelierte Eingabeknoten (wiederum bei gleichem Vorzeichen) genau das Gegenteil:

$$|w_1|^p + |w_2|^p > |w_1 + w_2|^p + 0^p \tag{2.15}$$

Hier wird das Minimum erreicht, wenn sich das Verfahren auf einen Knoten konzentriert und den anderen ausblendet, indem er mit einem Gewicht von 0 multipliziert wird. Die Information fließt also nur noch über einen Knoten in das Netz ein, die anderen (somit überflüssigen) Knoten spielen keine Rolle mehr. Das Netz kann sich auf einen Knoten konzentrieren und liefert somit bessere Ergebnisse.

Zu beachten ist, daß dieses Verfahren dann versagt, wenn es sich nur um vermeintlich korrelierte Eingabeknoten handelt. Bei einer 1-aus-n Kodierung mit großen binären Eingabevektoren, bei denen u. U. nur wenige Knoten auf 1, dafür viele Knoten auf 0 gesetzt sind, sind scheinbar alle Knoten hoch korreliert.

Im Gegensatz dazu hat sich bei neuronalen Netzen zur Bestimmung der Lautdauer [EZH01] bzw. der Grundfrequenz [EZ02a] die Verwendung von $p = 0,6$ als Optimum erwiesen. An den Eingangsknoten werden hier vorwiegend kontiunierliche Werte angelegt. Somit kann es oft vorkommen, daß Eingabevektoren ähnlich, aber nicht identisch sind. In diesem Fall ist es durchaus sinnvoll, bei hoch korrelierten Knoten einen auszuwählen und den bzw. die anderen aus dem Netzwerk zu entfernen.

2.2 Dynamische Zeitanpassung

Bei der Mustererkennung kommt es oft vor, daß für die Klassifikation eines Musters nicht einfach ein Merkmal zu einem bestimmten Zeitpunkt ausreicht, sondern daß eine Folge von Merkmalen zur Entscheidung herangezogen werden muß. Eine solche Merkmalsfolge stellen z. B.

die Phoneme eines Wortes bei der Spracherkennung dar. Hier genügt es nicht, einzelne Phoneme zu klassifizieren. Die Frage ist, zu welcher *Folge von Phonemen* das zu erkennende Sprachsignal am ähnlichsten ist. Dazu wird aus dem unbekannten Sprachsignal in einem bestimmten Zeitraster (z. B. 20ms) ein Merkmalsvektor errechntet, und die so erhaltene Vektorfolge wird mit denen der zu erkennenden Wörter verglichen. Als erkanntes Wort wird dasjenige gewählt, zu dem die unbekannte Vektorfolge den geringsten Abstand hat.

Das Prinzip besteht darin, ein unbekanntes Muster in Komponenten zu zerlegen und diese Komponenten mit denen der Repräsentanten (sozusagen einem „typischen Vertreter“) der einzelnen Klassen zu vergleichen. Dazu wird eine globale Unterscheidungsfunktion genutzt, die sich aus den lokalen Bewertungen der einzelnen Komponenten zusammensetzt. Diese Einzelbewertungen und deren Zusammensetzung zu einer Gesamtbewertung sind so zu wählen, daß diese Gesamtbewertung im Sinne einer Extremierung der globalen Unterscheidungsfunktion optimal wird. Sie soll also beim Beispiel der Worterkennung für die ähnlichste Phonemfolge den geringsten Abstand liefern.

Dazu kann das Verfahren der dynamischen Zeitanpassung (engl. dynamic time warping, DTW) verwendet werden. Bei diesem Verfahren handelt es sich um einen Spezialfall der dynamischen Programmierung [Hoff98, RJ93]. Dabei wird eine Matrix aufgespannt, deren eine Achse durch die Komponenten der unbekannten Folge und deren andere Achse durch die Komponenten der Referenzfolge aufgespannt wird. In die Felder dieser Matrix werden die Abstände zwischen den jeweiligen Komponenten (die lokale Bewertung) eingetragen. Nun wird ein Weg durch diese Matrix gesucht, bei dem die Verknüpfung der einzelnen Bewertungen entlang dieses Weges ein Optimum (z. B. minimaler Abstand oder maximale Ähnlichkeit) ergibt. Das unbekannte Muster wird der Klasse mit der besten Gesamtbewertung zugeordnet.

In dieser Arbeit wird die dynamische Zeitanpassung für die Aufbereitung des Trainingslexikons für die Graphem-Phonem-Konvertierung verwendet (siehe Abschnitt 3.1.2). Für die Erstellung der Trainingsmuster braucht man eine eindeutige Aussage, welche Grapheme bzw. Graphemfolgen welche Phoneme generieren. Hier besteht die Aufgabe also nicht darin, die ähnlichste Referenzfolge zu finden, sondern die Komponenten der einen Folge (die Grapheme) den Komponenten der anderen Folge (die Phoneme) zuzuordnen. Die als Nebenprodukt erhaltene Gesamtbewertung sagt bei dieser Anwendung aus, wie sicher diese Zuordnung ist.

2.3 Entropie und Transinformation

Die Transinformation (engl. mutual information) kann genutzt werden, um bei der Graphem-Phonem-Konvertierung zu bestimmen, wie groß der Graphemkontext sinnvoll gewählt werden sollte. Sie kann mit Hilfe der Entropie berechnet werden. Die dazu verwendeten Formeln sollen hier kurz vorgestellet werden.

Die Entropie (mittlerer Informationsgehalt) $H(x)$ einer Zufallsgröße x berechnet sich aus

$$H(x) = -\sum P(x) \cdot \mathrm{ld} P(x) \tag{2.16}$$

Ist x von einer weiteren Zufallsgröße y abhängig, dann kann man die Verbundentropie

$$H(x,y) = -\sum P(x,y) \cdot \mathrm{ld} P(x,y) \tag{2.17}$$

und die bedingte Entropie

$$H(x|y) = -\sum P(x,y) \cdot \mathrm{ld} P(x|y) \tag{2.18}$$

berechnen. Weiterhin gilt

$$H(x,y) = H(x) + H(y|x) \tag{2.19}$$

Die Transinformation T ist in [Frie95] folgendermaßen definiert: „Zwischen zwei Zufallsgrößen ... kann die Transinformation (mutual information) definiert werden, die angibt, wieviel Information eine der beiden Zufallsgrößen über die andere vermittelt." Sie errechnet sich aus

$$T = \sum P(x,y) \cdot \mathrm{ld} \frac{P(x,y)}{P(x) \cdot P(y)} \tag{2.20}$$

oder durch Einsetzen der Gleichungen (2.16) und (2.18) aus

$$T = H(x) - H(x|y) \tag{2.21}$$

Für weitere Bemerkungen zu diesen Formeln sei hier auf [Gall68] oder [Pehl98] verwiesen.

Kapitel 3

Transkription

3.1 Aufbereitung des Lexikons

In Abschnitt 1.5.1 wurde bereits auf die Problematik der Aufbereitung der Trainingsmuster hingewiesen. In diesem Abschnitt sollen nun zwei sprachenunabhängige Möglichkeiten aufgezeigt werden.

3.1.1 Suche nach Graphemgruppen

In einem ersten Versuch wurden den Phonemen Graphemgruppen zugeordnet (siehe [Hain99a, Hain01]). Ausgangsbasis für diese Zerlegung ist neben dem Aussprachelexikon nur eine Liste der Phoneme, wobei zu jedem Phonem angegeben ist, ob es sich um einen Vokal oder einen Konsonanten handelt. Diese Information wird für die Zusammenlegung von Phonemen benötigt. Die Zuordnung erfolgt in sechs Schritten:

1. Zuerst wird bei allen Wörtern, bei denen die Anzahl der Grapheme gleich der der Phoneme ist, jedes Graphem dem entsprechenden Phonem zugeordnet.

2. Nun wird versucht, die im ersten Schritt gefundenen Zuordnungen auf die anderen Wörter anzuwenden. Wenn dabei alle Phoneme bis auf das letzte zugeordnet werden können und Grapheme übrig sind, so werden die verbleibenden Grapheme dem letzten Phonem zugeordnet.

3. Wenn ein Eintrag nicht vollständig abgebildet werden kann, so wird am Ende der gefundenen Zuordnung das nächste Graphem an die bisherige Graphemgruppe angehängt und dann die Zuordnung erneut versucht. Dieses Anhängen wird solange probiert, bis nur noch soviel Grapheme wie Phoneme übrig sind.

4. Kann das Wort immernoch nicht vorwärts abgebildet werden, so wird die Zuordnung rückwärts probiert. Wenn dann nur ein nicht abbildbares Phonem übrig bleibt, so werden alle verbliebenen Grapheme diesem Phonem zugeordnet.

5. Im nächsten Schritt wird versucht, Phonemgruppen zu bilden. So ist z. B. bei dem Wort *Fax* (`/f a k s/`) klar, daß Phoneme zusammengefaßt werden müssen, da vier Phoneme auf drei Grapheme abgebildet werden müssen. Die Kandidaten für die Zusammenfassung werden über die Häufigkeiten der möglichen Zuordnungen gewonnen. In diesem Falle werden `/k/` und `/s/` zu `/k+s/` verbunden. Zusätzlich zu den Häufigkeiten wird hier auch die oben erwähnte Information ausgenutzt, ob es sich bei einem Phonem um einen Vokal oder einen Konsonanten handelt, denn es werden zuerst nur Vokale mit Vokalen oder Konsonanten mit Konsonanten verbunden. Erst wenn keine sinnvolle Verbindung zu finden ist, werden die Laute auch vermischt zusammengefaßt.

6. Im letzten Schritt wird in den Wörtern, für die keine eindeutige Entscheidung getroffen werden konnte, nach Phonemgruppen gesucht, die im vorigen Schritt gefunden wurden. Ist es unter Verwendung einer solchen Gruppe möglich, das Wort zu zerlegen, so wird die Zuordnung übernommen.

Jeder der sechs Schritte wird erst für alle Einträge im Lexikon durchgeführt bevor der nächste Schritt zur Anwendung kommt. Nach jedem Schritt wird allerdings noch die Häufigkeit jeder Zuordnung getestet. Ist sie viel geringer als die Zuordnung des Graphems zu einem der Nachbarphoneme, so wird diese Zuordnung gelöscht.

Alle Schritte werden solange ausgeführt, bis keine neue Zuordnung mehr gefunden wird. Dann wird die Häufigkeitskontrolle ausgeführt und zum nächsten Schritt übergegangen. Nach dem letzten Schritt werden das abgebildete Lexikon und die Häufigkeitstabelle abgespeichert.

3.1.2 Zuordnung mit dynamischer Zeitanpassung

Die Zuordnung der Grapheme auf die Phoneme ist ein Vorgang, den man auch mit Hilfe der dynamischen Zeitanpassung durchführen kann [Hain00c, Hain03c]. Ähnlich wie in der Spracherkennung, bei der aus dem Sprachsignal abgeleitete Muster auf die möglichen Phoneme abgebildet werden, wird hier versucht, jedes Graphem einem Phonem zuzuordnen (siehe Abbildung 3.1). Der wohl größte Unterschied in diesem Fall ist, daß beide Folgen schon sicher bekannt sind. Außerdem muß jedes Graphem einem Phonem zugeordnet werden, Auslassungen sind nicht möglich.

Wie in Abschnitt 2.2 beschrieben wurde, braucht man für die Suche des besten Pfades Übergangshäufigkeiten bzw. Kostenfunktionen, mit deren Hilfe man die Entscheidung treffen kann, welcher Weg sinnvoll ist und welcher nicht. Die Übergangshäufigkeiten werden hier aus dem Lexikon selber gewonnen, für die Kostenfunktionen wird phonetisches Wissen ausgenutzt.

In den folgenden Abschnitten werden verschiedene Häufigkeiten zur Berechnung des besten

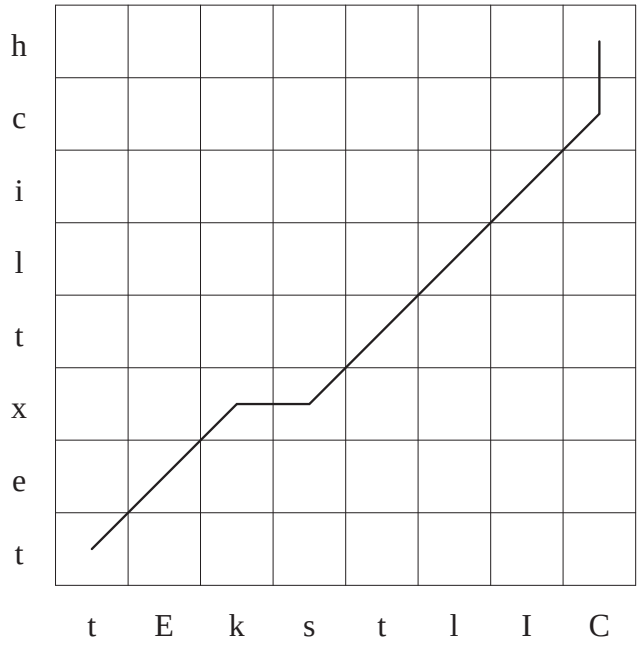

Abbildung 3.1: Zuordnung der Buchstaben des Wortes *textlich* auf die Phoneme.

Pfades verwendet. Diese werden nun vorgestellt. In der gesamten Arbeit wird, wenn nicht anders angegeben, der Begriff Häufigkeit synonym für relative Häufigkeit verwendet.

Es werden zwei Häufigkeiten verwendet. Die Häufigkeit F_p[1], mit der das Phonem /p/ aus dem Graphem <g> entstanden ist, berechnet sich aus

$$F_p(g \to p) = \frac{Z(g \to p)}{N(p)} \tag{3.1}$$

Dabei sind $Z(g \to p)$ die Anzahl der Zuordnungen des Graphems <g> zu dem Phonem /p/ und $N(p)$ die Anzahl aller Zuordnungen zu diesem Phonem /p/. Für die Häufigkeit F_g wird dagegen als Bezug die Anzahl der Zuordnungen eines bestimmten Graphems zu den Phoneme zu Grunde gelegt:

$$F_g(g \to p) = \frac{Z(g \to p)}{N(g)} \tag{3.2}$$

In den folgenden Abschnitten werden noch weitere Häufigkeiten benötigt, da zur endgültigen Entscheidung der Zuordnungen die einfache Übergangshäufigkeit nicht ausreicht. Deswegen werden die positionsabhängige Häufigkeit sowie die Vorgänger- und die Nachfolgerhäufigkeit eingeführt.

[1]Da das Symbol H schon für die Entropie vergeben ist, wird für die Häufigkeit das Symbol F (engl. frequency) verwendet.

Unter der positionsabhängigen Häufigkeit F^{pos} versteht man die Häufigkeit, mit der das Graphem an einer bestimmten Position innerhalb einer Graphemgruppe einem Phonem zugeordnet wurde. So stehen z. B. in Abbildung 3.1 bei der Zuordnung der Graphemgruppe <ch> zum Phonem /C/ <c> an erster und <h> an zweiter Position. Die Häufigkeit F^{pos} berechnet sich aus

$$F^{pos}(g \rightarrow p|Pos) = \frac{Z(g \rightarrow p|Pos)}{N(p)} \tag{3.3}$$

Die Vorgängerhäufigkeit F^{v} bzw. die Nachfolgerhäufigkeit F^{n} beschreiben die Zuordnung eines Graphems zu einem Phonem unter Berücksichtigung des Vorgänger- bzw. Nachfolgergraphems:

$$F^{v}(g \rightarrow p|g_{-1}) = \frac{Z(g \rightarrow p|g_{-1})}{N(p)} \tag{3.4}$$

$$F^{n}(g \rightarrow p|g_{+1}) = \frac{Z(g \rightarrow p|g_{+1})}{N(p)} \tag{3.5}$$

Bei der Zuordnung <ch> → /C/ würde also für die Nachfolgerhäufigkeit gezählt werden, wie oft das Graphem <c> dem Phonem /C/ zugeordnet wurde, wenn das nächste Graphem ein <h> war.

3.1.2.1 Initialisierung der Übergangshäufigkeiten

Zur Initialisierung der Überganghäufigkeiten werden die Einträge im Lexikon verwendet, bei denen die Anzahl der Grapheme mit der Anzahl der Phoneme übereinstimmt. Es wird angenommen, daß jedes Graphem dem entsprechenden Phonem zugeordnet ist (siehe Abbildung 3.2).

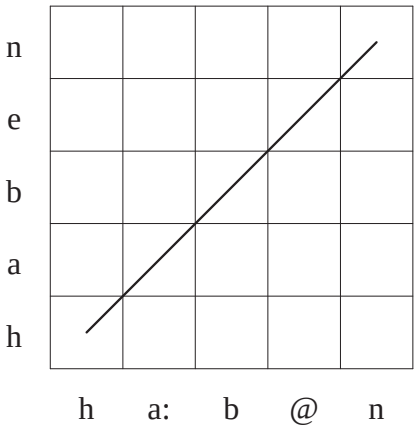

Abbildung 3.2: 1:1 Zuordnung der Buchstaben zu den Lauten.

Das ist natürlich nicht immer richtig, wie das Beispiel in Abbildung 3.1 zeigt. Da aber die Ausnahmen selten auftreten, werden sie bei der Anwendung der so gewonnenen Statistik auch entsprechend gering gewichtet. Außerdem werden alle Häufigkeiten, die einen bestimmten Schwellwert unterschreiten, bei der Generierung der neuen Statistik nach einer Iteration nicht übernommen (siehe Abschnitt 3.1.2.8).

3.1.2.2 Berechnung des optimalen Pfades

Mit der im vorhergehenden Schritt gewonnenen Statistik wird für jeden Eintrag im Lexikon die Matrix mit den Übergangshäufigkeiten aufgefüllt (siehe Abbildung 3.3).

n	**0**	0.013	0.949	0	**1**
e	0	0	0	0.986	0
n	0	0.013	0.949	0	0.949
n	0	0.013	0.949	0	0.949
ö	0	0.920	0	0	0
k	**1**	0	0	0	**0**
	k	9	n	@	n

Abbildung 3.3: Anwendung der ersten Statistik für das Wort *können*.

Ausnahmen sind die vier markierten Felder. Die Felder links unten und rechts oben müssen immer durchlaufen werden, da sie Start- bzw. Endpunkt sind. Sie werden deswegen mit 1 initialisiert. Dagegen können die Felder links oben und rechts unten niemals durchlaufen werden. Sie werden mit 0 initialisiert. Alle anderen Felder enthalten nun die entsprechenden Graphemhäufigkeiten $F_g(g \rightarrow p)$ (siehe Formel (3.2)). Die Graphemhäufigkeiten werden deswegen verwendet, weil die Aufgabe darin besteht, die Grapheme den Phonemen zuzuordnen. Die Frage ist, welches der zur Auswahl stehenden Phoneme von dem Graphem „bevorzugt“ wird, und nicht, welche anderen Grapheme dieses Phonem ebenfalls generieren können.

Offensichtlich gab es bei der ersten Initialisierung Zuordnungen von `<n>` auf `/9/`. Daher steht in den entsprechenden Feldern keine 0, sondern $0,013$. Man sieht aber, daß diese Häufigkeiten viel kleiner sind als die der „richtigen“ Zuordnungen. Sie fallen demnach kaum ins Gewicht.

Für die Berechnung des Pfades werden nun die einzelnen Häufigkeiten F jeweils mit dem Maximum der akkumulierten Nachbarhäufigkeiten A multipliziert. Da nur die Bewegungen nach oben, nach rechts oder nach oben rechts erlaubt sind, werden nur die Werte links, unten und

links unten betrachtet (siehe Abbildung 3.4). Die erste Spalte und die unterste Zeile stellen Sonderfälle dar, da es keinen linken bzw. unteren Nachbarn gibt. Hier wird die aktuelle Häufigkeit also stets mit der unteren bzw. der linken multipliziert. Für das Maximum $M_{p,g}$ der Nachbarfelder kann man also schreiben

$$M_{p,g} = \begin{cases} A_{1,g-1} & : \quad p = 1 \\ A_{p-1,1} & : \quad g = 1 \\ max(A_{p-1,g}, A_{p-1,g-1}, A_{p,g-1}) & : \quad sonst \end{cases} \tag{3.6}$$

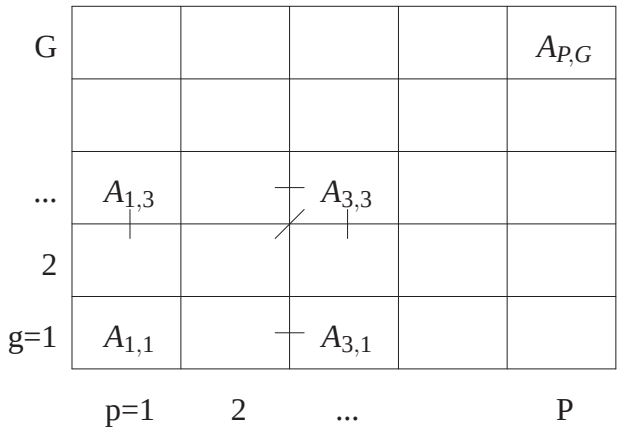

Abbildung 3.4: Berechnung der akkumulierten Häufigkeiten.

Für alle Felder gilt damit für die akkumulierten Häufigkeiten

$$A_{p,g} = F_{p,g} \cdot M_{p,g} \tag{3.7}$$

p gibt hier den Phonemindex und g den Graphemindex an. Sie laufen jeweils bis P (Anzahl der Phoneme) bzw. G (Anzahl der Grapheme).

Bei dem Beispiel aus Abbildung 3.3 sind $P = 5$ und $G = 6$. $A_{2,2}$ berechnet sich aus $0,92 \cdot max(0;1;0) = 0,92$, $A_{2,3} = 0,013 \cdot max(0;0;0,92) = 0,012$ und $A_{3,3} = 0,949 \cdot max(0,012;0,92;0) = 0,873$ usw.

Die akkumulierte Häufigkeit am Endpunkt $A_{P,G}$ ist somit das Produkt der Häufigkeiten auf dem optimalen Weg vom Startpunkt zum Endpunkt:

$$A_{P,G} = \prod_{t=1}^{T} F_{p(t),g(t)} \tag{3.8}$$

Dabei ist T die Anzahl der Schritte, die vom Startpunkt zum Endpunkt durchlaufen werden, und $p(t)$ bzw. $g(t)$ geben den Phonem- und Graphemindex des Feldes an, das in Schritt t die

maximale Häufigkeit enthält. Für das Beispiel aus Abbildung 3.3 gilt $A_{5,6} = 1 \cdot 0,92 \cdot 0,949 \cdot 0,949 \cdot 0,986 \cdot 1 = 0,817$. Die einzelnen Produkte sind noch einmal in Abbildung 3.5 zu sehen.

n	**0**	0	0	0	**0.817**
e	0	0	0	0.817	0
n	0	0.000	0.828	0	0
n	0	0.012	0.873	0	0
ö	0	0.920	0	0	0
k	**1**	0	0	0	**0**
	k	9	n	@	n

t	p(t)	g(t)
1	1	1
2	2	2
3	3	3
4	3	4
5	4	5
6	5	6

Tabelle mit $p(t)$ und $g(t)$ des optimalen Pfades

Abbildung 3.5: Matrix nach Multiplikation der Häufigkeiten.

Bei der Bestimmung von $M_{p,g}$ wird die Richtung abgespeichert, in der das Maximum liegt. Damit kann man, nachdem alle Berechnungen durchgeführt wurden, ausgehend vom Endpunkt den optimalen Pfad bestimmen, also den Pfad entlang der maximalen akkumulierten Häufigkeiten.

3.1.2.3 Kostenfunktionen

Zusätzlich zu den Übergangshäufigkeiten werden auch noch Kostenfunktionen verwendet, die aus Expertenwissen gewonnen werden.

Vokal-Konsonant-Zuordnung: Zum einen wird ausgenutzt, daß einige Grapheme vorwiegend oder sogar ausschließlich Vokalen (z. B. <a>) bzw. Konsonanten (z. B. <t>) zugeordnet werden. Es gibt im Deutschen keinen Fall, bei dem das Graphem <a> einem Konsonanten zugeordnet wird. Genausowenig wird ein <t> zum Vokal. Das ist bei anderen Sprachen nicht so der Fall, z. B. im Englischen, außer man verwendet ein sogenanntes Nullphonem. Durch dessen Verwendung könnte man z. B. verhindern, daß das <h> in *Uhr* dem Vokal /u:/ zugeordnet wird. Dann gäbe es nur noch die Möglichkeiten, daß ein Graphem entweder Vokal und Nullphonem oder Konsonant und Nullphonem zugeordnet wird. Die Variante Vokal und Konsonant würde somit nicht mehr auftreten.

Diese Kostenfunktion liefert den Wert 1, wenn sowohl Graphem als auch Phonem den Vokalen bzw. den Konsonanten zugerechnet werden. Anderenfalls liefert sie einen Wert, der anfangs 0 ist und nach jeder Iteration schrittweise erhöht wird, bis er schließlich auch den Wert 1 erreicht (siehe Abschnitt 3.1.2.7).

Vorgänger-Nachfolger-Information: Im Augenblick wird hierfür nur die Abhängigkeit vom Vorgängergraphem in Betracht gezogen (siehe Gleichung (3.4)).

Es wird allerdings nicht die Häufigkeit selber, sondern die Wurzel der Häufigkeit verwendet. Der Grund dafür ist in Abbildung 3.6 zu sehen. Durch die Verwendung der Wurzel werden die kleinen Werte etwas angehoben. Eine seltene Zuordnung wird dadurch nicht so stark bestraft wie bei der Verwendung der linearen Abbildung.

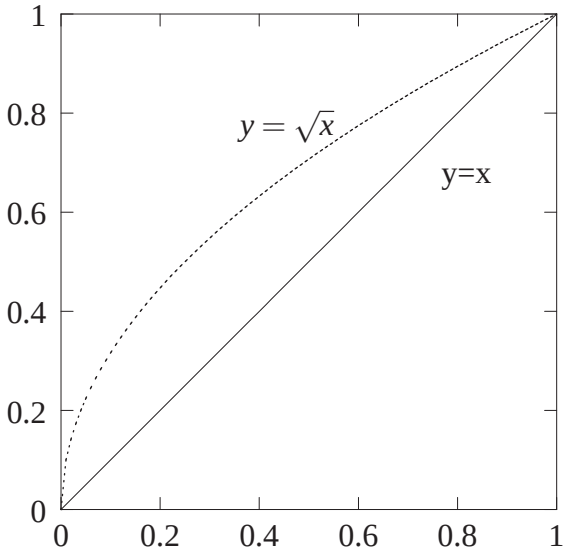

Abbildung 3.6: Die Funktionen $y = x$ und $y = \sqrt{x}$.

Gesamtwert der Kostenfunktion: Der Gesamtwert der Kostenfunktion ist das Produkt aus seinen Einzelkomponenten. Somit wird erreicht, daß die Zuordnung als schlecht bewertet wird, sobald ein Kriterium schlecht erfüllt ist (UND-Verknüpfung).

3.1.2.4 Nachbehandlung mit Hilfe von Zuordnungshäufigkeiten

Die Nachbehandlung dient zum Überprüfen der getroffenen Entscheidungen. Wenn zwei ähnliche Laute direkt aufeinanderfolgen, so ist die Information, wie oft das Graphem den beiden Lauten zugeordnet wurde, meist nicht ausreichend. Auch die in Abschnitt 3.1.2.3 beschriebene Vorgänger-Nachfolger-Information hilft bei einigen Fällen nicht. Zur Entscheidung dieser Fälle wird die Information herangezogen, mit welcher Häufigkeit das Graphem in einer bestimmten Position dem Phonem zugeordnet wurde. Dies ist erst in diesem Schritt möglich, da erst jetzt der (vorläufige) optimale Pfad bekannt ist. Bei der Nachbehandlung wird unterschieden, ob sich das Graphem am Anfang der Zuordnung zum Phonem befindet oder nicht.

Am Anfang einer Zuordnung: Hier wird überprüft, ob das aktuelle Graphem besser zum aktuellen oder zum vorhergehenden Phonem paßt. Dieser Test ist natürlich nur sinnvoll, wenn dem aktuellen Phonem mehrere Grapheme zugeordnet wurden. Ein Beispiel dafür ist in Abbildung 3.7 zu sehen.

n	0	0	0	0.052
e	0	0.005	0.052	0
e	0	0.017	0.052	0
i	0	0.052	0	0
r	0.078	0	0	0
	r	i:	@	n

Abbildung 3.7: Ausschnitt der Matrix für das Wort *schrieen* mit dem vorläufigen besten Pfad.

Das Graphem `<e>` wird häufiger dem Phonem `/@/` als dem Phonem `/i:/` zugeordnet. Deswegen erfolgt ohne die Berücksichtigung der Positionen eine Zuordnung von `<i>` zu `/i:/` und `<ee>` zu `/@/`.

Zur Überprüfung, ob dies sinnvoll ist, werden zwei Bewertungen errechnet, eine für den Fall, daß das zu untersuchende Graphem zum vorhergehenden Phonem gehört (B_{-1}), und eine dafür, daß es dem aktuellen Phonem zugeordnet wird (B_0).

B_{-1} wird mit der Häufigkeit der Zuordnungen des aktuellen Graphems zum Vorgängerphonem initialisiert:

$$B_{-1} = F(g \rightarrow p_{-1}) \tag{3.9}$$

In die Bewertung fließt auch mit ein, ob das aktuelle Graphem am Ende der vorhergehenden Zuordnung stehen kann:

$$B_{-1} = B_{-1} \cdot F^{pos}(g \rightarrow p_{-1} | Pos = ende) \tag{3.10}$$

Die zweite Bewertung wird mit der Häufigkeit initialisiert, mit der das aktuelle Graphem dem aktuellen Phonem zugeordnet wurde:

$$B_0 = F(g \rightarrow p) \tag{3.11}$$

Sie wird noch mit der Häufigkeit multipliziert, mit der das aktuelle Graphem an Position 1 zugeordnet wurde:

$$B_0 = B_0 \cdot F^{pos}(g \rightarrow p|Pos = 1) \tag{3.12}$$

Wenn es Vorgänger- und Nachfolgergrapheme gibt, dann werden die beiden Bewertungen noch ergänzt:

$$B_{-1} = B_{-1} \cdot F^{n}(g_{-1} \rightarrow p_{-1}|g) \tag{3.13}$$

bzw.

$$B_0 = B_0 \cdot F^{n}(g \rightarrow p|g_{+1}) \tag{3.14}$$

Wenn es eine Nachfolgerhäufigkeit gibt, so wird diese auch noch zur Berechnung der zweiten Bewertung herangezogen:

$$B_0 = B_0 \cdot F^{pos}(g_{+1} \rightarrow p|Pos = 2) \tag{3.15}$$

Wenn B_{-1} größer ist als B_0, was hier der Fall ist, so werden die Zuordnungen und der beste Pfad entsprechend korrigiert.

Innerhalb der Zuordnung: Ein ähnliches Problem kann auch innerhalb von Graphemgruppen entstehen. So tritt z. B. das Graphem `<e>` beim Phonem `/i:/` nie an erster Stelle auf, sondern immer nur an zweiter. Das kann beim Wort *Alliierte* für die Entscheidung ausgenutzt werden, daß das zweite `<i>` auch zum zweiten `/i:/` gehört, obwohl es nur unter Beachtung der Häufigkeiten dem ersten `/i:/` zugeordnet werden würde (siehe Abbildung 3.8).

Das Problem entsteht dadurch, daß an dieser Stelle durch das Auffüllen der Matrix mit den reinen Zuordnungshäufigkeiten zwei Wege mit identischer Bewertung existieren. Da es aber häufiger auftritt, daß zwei aufeinander folgende gleiche Grapheme gemeinsam ein Phonem generieren, als daß sie das gleiche Phonem zweimal generieren, wird hier der Weg nach oben dem diagonalen Weg vorgezogen. Das funktioniert allerdings nicht immer, weswegen manchmal eine nachträgliche Überprüfung des Weges nötig wird.

Ohne weitere Bewertung würden in diesem Beispiel die Zuordnungen `<ii>` → `/i:/` und `<e>` → `/i:/` getroffen werden. Aus anderen Einträgen, bei denen `<ie>` eindeutig `/i:/` zugeordnet wurde, kann man aber ablesen, daß `<e>`, wenn es `/i:/` zugeordnet wurde, stets an zweiter Position der Graphemgruppe stand. Daraus kann die positionsabhängige Häufigkeit

$$F^{pos}(g \rightarrow p|Pos) \tag{3.16}$$

r	0	0	0	0.928
e	0.011	0.186	0.186	0.182
i	0.025	0.694	0.694	0.011
i	0.025	0.694	0.694	0.011
l	0.931	0	0	0
	l	i:	i:	r

Abbildung 3.8: Ausschnitt der Matrix für das Wort *Alliierte* mit dem vorläufigen besten Pfad.

errechnet werden. In solchen Zweifelsfällen werden dann also die Produkte der F^{pos} verglichen. Für den Fall, daß das zweite `<i>` zum ersten `/i:/` gehört, ergibt sich das Produkt

$$F^{pos}(i \rightarrow i : |Pos_{<i>} = 2) \cdot F^{pos}(e \rightarrow i : |Pos_{<e>} = 1) \tag{3.17}$$

und für den anderen Fall, daß es zum zweiten `/i:/` gehört, ist das Produkt

$$F^{pos}(i \rightarrow i : |Pos_{<i>} = 1) \cdot F^{pos}(e \rightarrow i : |Pos_{<e>} = 2) \tag{3.18}$$

Da $F^{pos}(e \rightarrow i : |Pos_{<e>} = 1)$ Null oder im Falle falscher Zuordnungen zumindest extrem klein ist, wird das Produkt aus (3.18) größer als das aus (3.17). Demnach wird der optimale Pfad entsprechend geändert.

Zusätzlich kann an dieser Stelle bestraft werden, wenn eine Graphemgruppe zu lang ist. Je länger sie ist, desto schlechter ist das für die Architektur des neuronalen Netzes, da die Anzahl der Knoten ansteigt. Außerdem werden diese Zerlegungen nur selten auftreten und sind damit schlecht lernbar. Hier kann also darauf Einfluß genommen werden, wie lang die längste Graphemgruppe sein kann.

3.1.2.5 Nachbehandlung mit Hilfe von Graphem- und Phonemgruppen

Die im vorigen Abschnitt verwendeten Häufigkeiten reichen in einigen Fällen nicht für eine richtige Entscheidung aus. Das ist der Fall, wenn die positionsabhängigen Häufigkeiten ähnlich groß sind. Deswegen werden in einem nächsten Schritt Graphem- und Phonemgruppen miteinander verglichen und der Pfad durch die Matrix gegebenenfalls korrigiert. Diese Gruppen werden an Hand des aktuellen besten Pfades gewonnen. Es wird in diesem Fall nicht einfach

betrachtet, an welcher Position ein Graphem einem Phonem zugeordnet wird, sondern wie oft welche Graphemkombination welchem Phonem zugeordnet wird.

So wird z. B. bei dem Wort *Junggeselle* die Einteilung <n> → /N/ und <gg> → /g/ getroffen, da <g> viel häufiger /g/ zugeordnet wird als /N/. Nun werden analog zu den vorhergehenden Abschnitten beide Möglichkeiten der Zuordnung durchgespielt:

$$F(n \rightarrow N) \cdot F(gg \rightarrow g) = 0,276 \cdot 0,007 = 0,001 \tag{3.19}$$

und

$$F(ng \rightarrow N) \cdot F(g \rightarrow g) = 0,724 \cdot 0,993 = 0,718 \tag{3.20}$$

Damit fällt die Entscheidung zu gunsten der Zuordnungen <ng> → /N/ und <g> → /g/.

3.1.2.6 Überspringen von Zuordnungslücken

Da die Häufigkeiten mit Hilfe der Wörter initialisiert werden, bei denen die Anzahl der Grapheme gleich der Anzahl der Phoneme ist, kann es vor allem während der ersten Iterationen vorkommen, daß einige Zuordnungen noch nie auftraten und deswegen die entsprechenden Häufigkeiten den Wert 0 haben. Dies tritt vorwiegend bei langen Graphemgruppen wie z. B. <sch> oder <tsch> auf (sie Abbildung 3.9).

e	**0**	0.314	0	**1**
h	0	0	0	0
c	0	0	0	0
s	0	0	1.0	0
i	0	0.686	0	0
n	**1**	0	0	**0**
	n	i:	S	@

e	**0**	0.314	0	**1**
h	0	0	0.001	0
c	0	0	0.001	0
s	0	0	1.0	0
i	0	0.686	0	0
n	**1**	0	0	**0**
	n	i:	S	@

Abbildung 3.9: Zuordnungen für *Nische* mit Lücken (linkes Bild) und nach dem Auffüllen der unbesetzen Häufigkeiten (rechtes Bild).

In diesem Falle wäre es niemals möglich, das Ende des Wortes zu erreichen, da das Gesamtprodukt immer 0 ist. Das ist in den ersten Iterationen auch sinnvoll. Bei den späteren Iterationen muß aber dennoch ein Weg gefunden werden, um auch die Zuordnungen für die langen Graphemgruppen zu finden. Dazu werden die Häufigkeiten innerhalb einer Lücke auf einen sehr geringen Wert (z. B. 0,001) gesetzt, was eine Fortsetzung des Pfades bis zum Wortende ermöglicht.

3.1.2.7 Iterative Anpassung der Restriktionen

Die verwendeten Häufigkeiten stellen in den ersten Iterationen nur Schätzungen dar und sind noch relativ ungenau. Um zu verhindern, daß ein anfangs gemachter Fehler das Gesamtergebnis verfälscht, werden zu Beginn harte Restriktionen eingeführt, die bei zunehmender Anzahl der Iterationen schrittweise aufgeweicht werden. Während der ersten Iterationen ist die Kostenfunktionen für die Vokal-Konsonant-Zuordnung so eingestellt, daß sie den Wert 1 liefert, wenn sowohl Graphem als auch Phonem zu den Vokalen bzw. Konsonanten gehören, sonst liefert sie 0. Bei den späteren Iterationen wird dieser Wert dann auf 10^{-5} gesetzt und fortan mit 10 multipliziert, bis er 1 erreicht. Somit wird gegen Ende der Abarbeitung auch gestattet, daß ein Vokal einem Konsonanten zugeordnet wird und umgekehrt. Wenn man dies von Anfang an erlauben würde, würden unter Umständen viele falsche Zuordnungen gefunden.

Weiterhin gibt es zwei Schwellwerte, mit denen die Übernahme der gefundenen Zuordnungen in die neue Statistik gesteuert werden kann. Der erste Wert legt fest, wie groß die akkumulierte Häufigkeit am Endpunkt $A_{P,G}$ (Formel (3.8)) mindestens sein muß, damit der gefundene Pfad überhaupt akzeptiert wird. Der Schwellwert wird mit 0,1 initialisiert und schrittweise durch Multiplikation mit 0,1 verringert. Der zweite Wert bestimmt, wie groß die relative Häufigkeit einer Zuordnung mindestens sein muß, damit sie in die neue Statistik übernommen wird. Er errechnet sich aus der Wurzel des Schwellwertes für die akkumulierte Häufigkeit.

Schließlich gibt es noch einen Wert, der festlegt, bis zu welcher Größe Lücken innerhalb der Zuordnungen übersprungen werden können. Am Anfang sind keine Lücken zugelassen, später wird dieser Wert auf 2 gesetzt und bis zur maximal zulässigen Gruppierung erhöht.

3.1.2.8 Generieren der neuen Statistik

Die Zuordnungen, bei denen das Produkt der Übergangshäufigkeiten den vorgegebenen Schwellwert überschritten hat, werden zur Gewinnung der neuen Statistik gewonnen. Schon bei der ersten Auswertung der so gewonnenen Statistik sind die meisten Fehler verschwunden, die durch die eins zu eins Initialisierung der Häufigkeiten gemacht wurden. Dieser absolut festgelegte Schwellwert hat noch eine andere Funktion. Da die Häufigkeiten kleiner Eins sind, wird ihr Produkt um so kleiner, je mehr Werte multipliziert werden. Das bedeutet, daß am Anfang vorwiegend kurze Wörter in die Generierung der neuen Statistik eingehen, da bei langen Wörtern kaum der Schwellwert überschritten wird.

Außerdem wird noch überprüft, wie häufig jede Graphem-Phonem-Zuordnung auftrat. Wenn das Verhältnis einen Schwellwert unterschreitet, so wird diese Zuordnung gelöscht und somit beim nächsten Auffüllen der Matrizen nicht verwendet.

3.1.2.9 Ergebnis der Zuordnungen

Im Ergebnis der Lexikonaufbereitung steht ein neues Lexikon, das nun eine Zuordnung der Grapheme zu den Phonemen enthält. Dieses Lexikon kann im nächsten Schritt zur Erzeugung der Muster für das Training der neuronalen Netze verwendet werden. Hier folgen die Ergebnisse für die Wörter, die in diesem Abschnitt als Beispiele verwendet wurden.

```
A - ll i  - ,ie r - t @
a - l  i: - "i: r - t @

J ,u ng - g e - s e - ll e
j "U N  - g @ - z E - l  e

N ,i  - sch e
n "i: - S   @

h ,a  - b e n
h "a: - b @ n

k ,ö - nn @ n
k "9 - n  @ n

sch r ,ie - e n
S   r "i: - @ n

t ,e x  t - l I ch
t "E ks t - l I C
```

Dabei steht der Bindestrich für eine Silbengrenze. Die Wortbetonung wird bei den Graphemen durch ein Komma, bei den Phonemen dagegen durch ein Anführungszeichen markiert. Quasi als Nebenprodukt erhält man auf diese Weise ein Lexikon, das die Informationen über Silbengrenze und Wortbetonung auch auf der Graphemebene enthält.

Aus diesem Lexikon kann man auch ableiten, welche Grapheme das letzte Phonem des Wortes generieren. Damit ist man in der Lage, bei der Zusammensetzung der Gesamttranskription eines Wortes aus Teilwörtern die Phoneme am Übergang zwischen den Teiltranskriptionen zu überprüfen und nötigenfalls zu korrigieren (siehe Abschnitt 3.5.1). Aus einem solchen Lexikon kann man auch eine Tabelle erstellen, die für jedes Phonem alle Graphemgruppen enthält, aus denen es generiert werden kann. Diese Tabelle wird in Abschnitt 3.5.2 für die Korrektur möglicher Fehlentscheidungen des neuronalen Netzes verwendet.

3.2 Transkription mit neuronalen Netzen

Neuronale Netze für die Bestimmung der Transkription wurden bereits in Abschnitt 1.5.3.3 vorgestellt. Diese Architekturen enthalten in der Ausgabe neben den Phonemen und der Silbengrenze auch die Betonung. Die Entscheidung über die Betonung wird also für jedes Phonem getroffen, obwohl dafür kein ausreichender Kontext zur Verfügung steht. In [SR86] wird überhaupt kein Phonemkontext berücksichtigt, und in [Rose96] nur ein oder mehrere Vorgängerphoneme. Nachfolgende Phoneme fließen somit nicht in die Entscheidung über die Betonung ein.

In der hier vorgestellten Arbeit wird die Transkription mit zwei neuronalen Netzen bestimmt. Das erste Netz entscheidet über die Phoneme und Silbengrenzen, das zweite Netz fügt dann in die Phonemfolge die Betonung ein. Damit fällt die Entscheidung auf Wortebene, oder bei langen Wörtern zumindest unter Verwendung des Wortanfangs. Die Aufteilung führt zu deutlich besseren Ergebnissen, da zum einen das erste Netz die Entscheidung aufgrund des fehlenden Kontextes nicht treffen kann, und zum anderen sich das zweite Netz voll auf die Betonung spezialisieren kann.

Zu Beginn der Arbeiten wurde auch eine andere Unterteilung untersucht. Viele der verfügbaren Aussprachelexika enthalten weder Angaben über Silbengrenzen noch über die Wortbetonung, da sie für die Spracherkennung erstellt wurden und dort diese Informationen nicht von Interesse sind. Deswegen wurde das erste Netz so gestaltet, daß es nur die Entscheidung über das Phonem zu treffen hat, und das zweite Netz hat in die generierte Phonemfolge die Silbengrenzen und Betonungen eingefügt. Die Ergebnisse waren nicht so gut wie die bei der anderen Aufteilung. Der Grund dafür ist darin zu suchen, daß die Bestimmung der Silbengrenzen durch das zweite Netz natürlich stark von den Fehlern auf Phonemebene bestimmt wird. In eine falsch generierte Phonemfolge kann das zweite Netz keine richtige Silbengrenze setzen. Wenn sich das erste Netz z. B. bei der Graphemfolge `<sch>` für die Phonemfolge `/S/` statt für `/s C/` entschieden hat, so ist auch automatisch die zu setzende Silbengrenze falsch. Indem man das erste Netz die Silbengrenze auch lernen läßt, wird es für einen falschen Laut an dieser Stelle doppelt bestraft, weil auch noch die Silbengrenze falsch ist, was zu einem höheren Fehler führt. Wegen der schlechteren Ergebnisse wurde diese Aufteilung nicht weiter untersucht.

3.3 Neuronales Netz für Phonem und Silbengrenze

Die Aufgabe für dieses neuronale Netz besteht darin, aus der Folge von Graphemen eine Folge von Phonemen zu generieren. Gleichzeitig wird entschieden, ob vor dem Phonem eine Silbengrenze eingefügt wird oder nicht. Im Verlauf der Arbeiten wurden zwei ähnliche Architekturen ausprobiert, da das zuerst entwickelte Netz einige Nachteile aufwies. Die Architektur wurde daraufhin geringfügig verändert.

3.3.1 Topologie des Ausgangsnetzes

Die Topologie des Netzes ist in Abbildung 3.10 zu sehen. Am Eingang der Netze werden die Grapheme angelegt. Im Zentrum (*z*) steht das Graphem, für das ein Phonem bestimmt werden soll. Weiterhin werden die linken und rechten Nachbargrapheme angegeben (*l* und *r*).

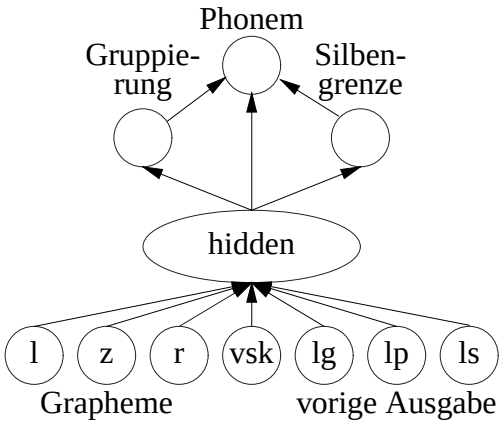

Abbildung 3.10: Architektur des Netzes zur Bestimmung von Phonem und Silbengrenze.

Die Ausgabe besteht aus dem Phonem und der Aussage über die Silbengrenze. In [SR86] und [Rose96] wird jedes Graphem am Eingang angelegt, auch wenn es kein Phonem generiert (z. B. `<c>` und `<h>` bei der Konvertierung von `<sch>` zu `/S/`). Dafür wurde ein zusätzlicher Ausgabeknoten verwendet, der markiert hat, daß dem angelegten Graphem kein Phonem zugeordnet wird.

In der vorgestellten Architektur wird dieses Problem anders gelöst. Hier gibt es Ausgabeknoten, mit deren Hilfe bestimmt wird, wieviele Grapheme zur Generierung des Phonems verwendet wurden. Sie werden also zu einer Graphemgruppe zusammengefaßt. Der Wert, der die Anzahl der zusammengefaßten Grapheme angibt, wird im weiteren als Gruppierung bezeichnet. Er wird als Schrittweite verwendet, um die Grapheme, für die die Entscheidung bereits getroffen wurde, zu überspringen. Das verringert nicht nur die Anzahl der Trainingsmuster, sondern auch die der auszuführenden Schritte im Anwendungsfall.

In [Rose96] wurde nachgewiesen, daß sich die Ergebnisse verbessern, wenn man dem Netz als zusätzliche Eingabe die Ausgabe der vorhergehenden Entscheidung für das entsprechende Wort verwendet. Allerdings wird hier keine explizite Rekurrenz verwendet, sondern die Werte der Ausgabeknoten werden an drei zusätzlichen Eingabeknoten bei der nächsten Entscheidung angelegt (Gruppierung *lg*, Phonem *lp* und Silbengrenze *ls*). Außerdem enthält die Eingabe einen Knoten, der angibt, ob es sich bei dem Phonem der vorhergehenden Entscheidung um einen Silbenkern handelt oder nicht (*vsk*). Diese Angabe ist für die Silbengrenzenentscheidung hilfreich.

3.3.2 Weiterentwickeltes Netz

Das Problem der vorigen Architektur besteht in der Trennung von Ausgabephonem und Ausgabegruppierung. Während des Trainings entsteht nur ein geringer Fehler, wenn sich das Netz zwar für das richtige Phonem, nicht aber für die richtige Gruppierung entscheidet. Der Fehler ist stärker, wenn Phonem und Gruppierung kombiniert werden, so wie es in [UZW98] vorgeschlagen wird. Deswegen wird jetzt durch die Ausgabeneuronen nicht mehr nur das Phonem, sondern auch die zugehörige Gruppierung kodiert. Es gibt nun z. B. zwei Neuronen für das Phonem /a:/, eines mit der Gruppierung 1 und eines mit der Gruppierung 2.

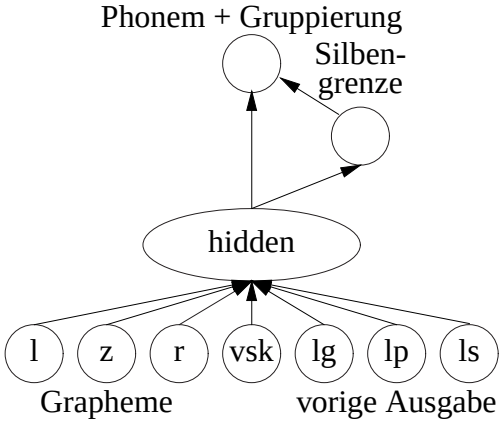

Abbildung 3.11: Architektur des Netzes mit der Kombination Phonem+Gruppierung.

Diese Architektur hat aber nicht nur während des Trainings Vorteile, sondern auch in der Anwendung. Durch die explizite Vorgabe der Gruppierungen, die für ein Phonem möglich sind, ist eine unmögliche Gruppierung ausgeschlossen. Während sich das erste Netzwerk bei der Graphemfolge <ahn> für das Phonem /a:/ und einer Gruppierung von 3 entscheiden konnte, hat das neue Netz nicht die Möglichkeit, diesen Fehler zu machen - es gibt kein Phonem /a:/ mit der Gruppierung 3.

Außerdem ist es mit dem neuen Netz möglich, nicht nur das beste, sondern auch die nächstbesten Phoneme zu bestimmen. Das war bisher nicht der Fall, da das Phonem nicht nur vom Phonemknoten, sondern auch vom Gruppierungswert abhängig war. Es wäre damit denkbar, nicht ausschließlich das beste Phonem als das richtige anzunehmen, sondern z. B. eine Liste der 3 besten Werte zu erstellen, die dann in einem anschließenden Schritt überprüft werden.

3.3.3 Bestimmung der Eingabekontextes

Untersuchungen zur Bestimmung des Informationsgehaltes des Kontextes (Transinformation, siehe Abschnitt 2.3) wurden u. a. in [AD94], [Rose96] und [DvdB93] durchgeführt. Das Ergebnis ist, daß nach dem Graphem im Zentrum der erste rechte Nachbar am wichtigsten ist, dann

folgt der erste linke Nachbar, danach der zweite rechte Nachbar, dann der zweite linke Nachbar usw. (siehe Tabelle 3.1). Die Transinformation für die drei in dieser Arbeit verwendeten Lexika ist in Tabelle 3.2 zu sehen.

i	Eng	Fre	Kor	Dut	Spa
-3	0,173	0,213	0,262	0,196	0,174
-2	0,303	0,305	0,485	0,347	0,341
-1	0,725	0,720	1,153	0,891	0,966
0	3,144	2,899	3,516	3,537	3,824
1	0,912	1,100	1,227	0,980	0,958
2	0,323	0,545	0,443	0,462	0,367
3	0,179	0,284	0,260	0,242	0,178

Tabelle 3.1: Transinformation $T(P;G_i)$ zwischen den Phonemen und den Graphemen in der Entfernung i für die Sprachen Englisch, Französisch, Koreanisch, Holländisch und Spanisch (aus [Coil93], Seite 1458). Je größer der Wert ist, desto mehr Information trägt das Graphem dieses Kontextes zur Entscheidung über das Phonem bei.

i	Deutsch	Englisch	Holländisch
-5	0,048	0,037	0,030
-4	0,084	0,051	0,056
-3	0,136	0,090	0,102
-2	0,268	0,197	0,246
-1	0,861	0,684	0,769
0	3,549	2,958	3,396
1	1,138	0,934	0,956
2	0,447	0,341	0,398
3	0,195	0,168	0,185
4	0,102	0,098	0,089
5	0,062	0,059	0,043

Tabelle 3.2: Transinformation für die in dieser Arbeit verwendeten Lexika.

In [Rose96] wurde auch untersucht, welchen Einfluß der Phonemkontext hat. Ohne Vorgängerphonem waren die Ergebnisse deutlich schlechter als mit einem Vorgänger. Durch Hinzufügen eines zweiten Vorgängers verbesserten sich die Ergebnisse nochmals. Ein dritter Vorgänger brachte keine Verbesserung mehr. Beim Phonemkontext ist zusätzlich zu beachten, daß während der Anwendung die eingegebenen Phoneme das Ergebnis der Berechnungen des neuronalen Netzes sind. Die bestimmten Phoneme können also fehlerhaft sein, was dazu führen kann, daß ein falscher oder zumindest ungünstiger Kontext angelegt wird. Deswegen wird hier nur ein Vorgängerphonem betrachtet.

3.3.4 Untersuchung des Informationsgehaltes mit Weight Decay

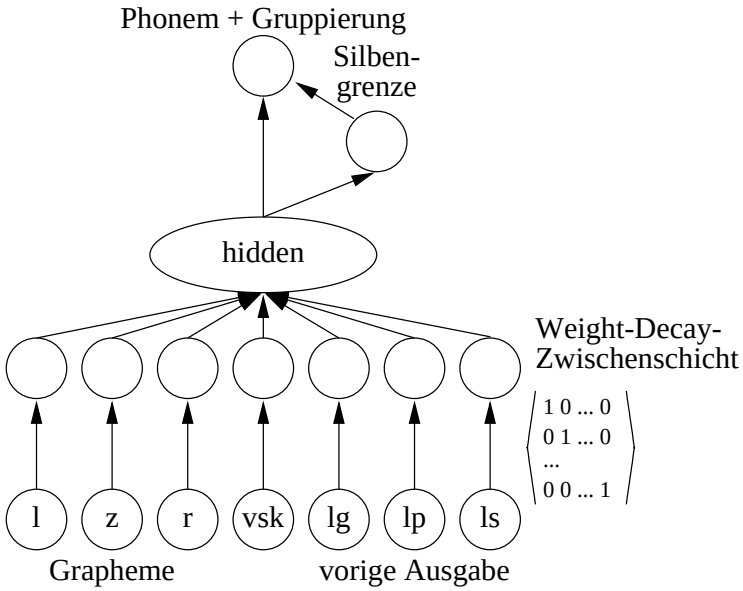

Abbildung 3.12: Architektur des Netzes mit der Kombination Phonem+Gruppierung und der Weight-Decay-Schicht. Die Eingabeknoten sind mit den WD-Knoten nur über eine Diagonalmatrix verbunden.

In Abschnitt 2.1.7 wurde Weight Decay als ein Verfahren vorgestellt, mit dem die Gewichte so eingeschränkt werden, daß sie nicht unnötig große Werte annehmen. Es wird der kleinste Gewichtsvektor gesucht, mit dem das Problem optimal zu lösen ist. Gewichte ohne Einfluß werden gegen Null gedrückt, was einem Ausdünnen entspricht, und die benötigten Gewichte werden nur so groß wie unbedingt erforderlich eingestellt. Damit kann man die Größe eines Gewichtes als ein Maß dafür heranziehen, welchen Einfluß es auf die Entscheidung des Netzes hat. Je größer ein Gewicht ist, desto wichtiger ist es.

Das Netzwerk wurde nun um eine zweite versteckte Schicht erweitert (siehe Abbildung 3.12). Die eigentliche Eingabe wurde mit dieser neuen Zwischenschicht über eine Diagonalmatrix verbunden. Die Werte auf der Diagonale werden mit 1 initialisiert, alle anderen mit 0. Damit ist jedes Eingabeneuron über genau ein Gewicht mit dem Netzwerk verbunden. Nur auf diese Diagonalverbindungen wird das Weight Decay angewendet, wobei als Aktivierungsfunktion für die Zwischenschicht der tangens hyperbolicus (*tanh*) verwendet wird.

Um zu überprüfen, ob die Größe dieser Gewichte tatsächlich die Bedeutung der mit ihnen verbundenen Grapheme widerspiegelt, wurden sie mit den berechneten Transinformationen verglichen (siehe Abbildungen 3.13 und 3.14). Der Verlauf in Abbildung 3.14 entspricht mit Ausnahme des Kontextes -1 (erster linker Vorgänger) in etwa dem aus Abbildung 3.13. Die Gewichte des rechten Kontextes haben größere Werte als die des linken, werden also analog zur berechneten Transinformation als wichtiger eingeschätzt.

Für die Abweichung des ersten linken Kontextes gibt es zwei Erklärungen. Zum einen treten nicht alle Grapheme als direkter Vorgänger des Zentrumgraphems auf, da sie ausschließlich am Beginn einer längeren Graphemgruppe stehen. So kommen z. B. die Grapheme <c> (aus <ch> und <ck>) und <q> (aus <qu>) nie als direkter Vorgänger des Graphems im Zentrum in Frage, sondern nur in größeren Kontexten. Deswegen ergeben sich für die Transinformation andere Werte, wenn man sie mit Hilfe der Muster errechnet, mit denen das neuronale Netz trainiert wird (siehe Tabelle 3.3). Außerdem werden die Muster für das Training eindeutig sortiert. Jedes Muster kommt also, im Gegensatz zum Lexikon, genau einmal vor.

i	Lexikon	Trainingsmuster	Mittelwerte
-4	0,084	0,047	0,20
-3	0,136	0,091	0,26
-2	0,268	0,153	0,38
-1	0,861	0,486	0,26
0	3,549	2,785	1,00
1	1,138	0,846	0,87
2	0,447	0,328	0,55
3	0,195	0,136	0,31
4	0,102	0,060	0,21

Tabelle 3.3: Transinformation für Deutsch. Die zweite Spalte gibt die Werte an, die an Hand der Einträge im Lexikon errechnet wurden. Die Werte der dritten Spalte wurde mit Hilfe der Trainingsmuster errechnet, die sich aus dem Lexikon ergeben. In der vierten Spalte sind zum Vergleich die Mittelwerte der Gewichte der Weight-Decay-Matrix für den entsprechenden Kontext angegeben.

Zum anderen wird bei der Berechnung der Transinformation nicht die Korrelation der Grapheme untereinander berücksichtigt. $T(P;G_i)$ gibt eben nur die direkte Abhängigkeit des Phonems P vom Graphem G des Kontextes i an. Das neuronale Netz hat während des Trainings immer alle Graphemkontexte gleichzeitig zur Verfügung, so daß Abhängigkeiten unter den Graphemen in die Gewichtsanpassung mit einfließen. Durch die Verwendung von Weight Decay werden die Gewichte ja gerade auf das maximal notwendige Maß begrenzt. Sind zwei Eingabeknoten korreliert und liefern somit die gleiche Information, so kann sich das Netz entscheiden, welches der zugehörigen Gewichte verstärkt und welches abgeschwächt wird. Da der rechte Kontext tendenziell den größeren Informationsgehalt hat und somit mehr über die zu treffende Entscheidung aussagt, werden die Gewichte des rechten Kontextes größer eingestellt als die des linken. Der linke Kontext wird vorwiegend für die Entscheidungen herangezogen, bei denen der rechte Kontext nicht genügend Information für eine sichere Entscheidung liefert.

In Tabelle 3.4 ist auch zu sehen, daß einige der Weight-Decay-Verbindungen sehr kleine Werte annehmen, z. B. der Kontext -1 beim Graphem <c> oder alle Werte außer dem im Zentrum für das Graphem <j>. Dies bedeutet, daß diese Verbindungen keinen Einfluß auf die Entscheidung des Netzes haben, da der Wert am Eingabeknoten mit Null multipliziert wird. Für die Einstellung dieser Werte kann es zwei Gründe geben. Wie bereits erwähnt, tritt das Graphem <c> im

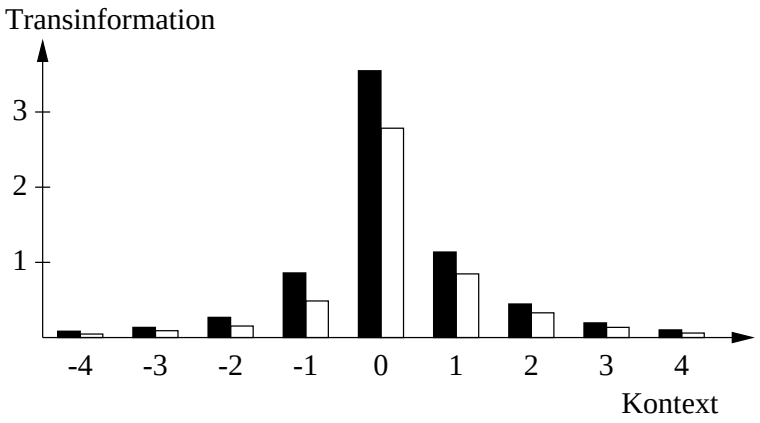

Abbildung 3.13: Die Transinformation in Abhängigkeit vom Kontext für Deutsch. Die schwarzen Balken stellen die Werte für das gesamte Lexikon dar, die weißen Balken stehen für die Werte, die anhand der Trainingsmuster für das neuronale Netz beestimmt wurden (vgl. auch Tabelle 3.3).

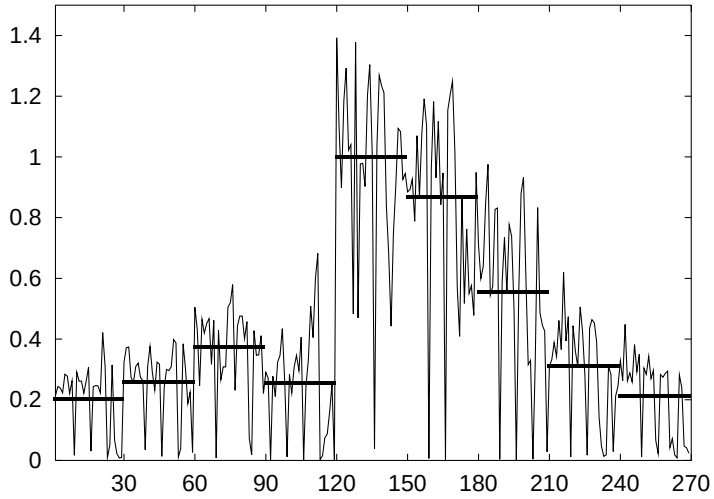

Abbildung 3.14: Die Werte der Diagonalmatrix für die Weight-Decay-Verbindung des neuronalen Netzes zur Graphem-Phonem-Konvertierung. Die x-Achse gibt die Nr. des Eingabeknotens an. Die Knoten 1 bis 30 repräsentieren den vierten Vorgänger des zentralen Graphems (Kontext -4 in Abbildung 3.13), 31 bis 60 den dritten Vorgänger usw. Das Graphem im Zentrum wird durch die Knoten 120 bis 150 repräsentiert. Die y-Achse gibt die Werte der jeweiligen Gewichte an (siehe auch Tabelle 3.4). Die horizontalen Linien stellen jeweils den Mittelwert der Gewichte für den entsprechenden Kontext dar.

Graphem	linker Kontext				Zentrum	rechter Kontext				Mittelwert
	-4	-3	-2	-1	0	1	2	3	4	Graphem
a	0,24	0,32	0,50	0,29	1,39	0,89	0,71	0,31	0,33	0,55
b	0,21	0,37	0,43	0,27	1,09	0,89	0,60	0,33	0,26	0,49
c	0,24	0,37	0,25	0,00	0,90	0,93	0,64	0,39	0,45	0,46
d	0,24	0,27	0,46	0,28	1,19	0,79	0,84	0,34	0,26	0,52
e	0,22	0,26	0,42	0,21	1,29	1,07	0,98	0,46	0,29	0,58
f	0,28	0,31	0,45	0,32	1,02	0,87	0,54	0,37	0,26	0,49
g	0,28	0,32	0,47	0,35	1,04	1,08	0,57	0,62	0,38	0,57
h	0,22	0,27	0,32	0,43	0,48	1,19	0,83	0,39	0,29	0,49
i	0,26	0,26	0,46	0,26	1,38	1,10	0,83	0,47	0,35	0,60
j	0,02	0,03	0,01	0,01	0,47	0,01	0,00	0,01	0,01	0,06
k	0,29	0,31	0,43	0,28	0,98	0,96	0,59	0,44	0,30	0,51
l	0,26	0,38	0,27	0,22	0,98	1,18	0,73	0,36	0,28	0,52
m	0,26	0,29	0,31	0,31	0,90	0,93	0,55	0,31	0,34	0,47
n	0,22	0,23	0,31	0,35	1,20	1,12	0,78	0,51	0,27	0,55
o	0,26	0,32	0,50	0,30	1,30	0,84	0,74	0,43	0,30	0,55
p	0,31	0,32	0,52	0,41	1,01	0,95	0,50	0,28	0,07	0,49
q	0,03	0,01	0,58	0,00	0,04	0,00	0,00	0,02	0,02	0,08
r	0,24	0,24	0,23	0,24	1,03	1,15	0,55	0,44	0,28	0,49
s	0,25	0,30	0,44	0,33	1,27	1,21	0,88	0,46	0,27	0,60
t	0,25	0,30	0,48	0,51	1,23	1,25	0,93	0,45	0,29	0,63
u	0,22	0,31	0,48	0,40	1,21	1,01	0,62	0,39	0,29	0,55
v	0,42	0,40	0,40	0,60	0,84	0,57	0,32	0,14	0,04	0,41
w	0,31	0,39	0,46	0,68	0,68	0,41	0,33	0,05	0,07	0,38
x	0,01	0,01	0,07	0,00	0,44	0,87	0,00	0,01	0,02	0,16
y	0,05	0,04	0,02	0,02	0,75	0,52	0,32	0,02	0,01	0,19
z	0,31	0,38	0,43	0,07	0,91	0,76	0,83	0,31	0,28	0,48
ä	0,07	0,30	0,35	0,09	1,09	0,55	0,49	0,28	0,24	0,38
ö	0,02	0,19	0,35	0,16	1,08	0,58	0,45	0,03	0,05	0,32
ü	0,01	0,22	0,41	0,25	0,93	0,48	0,43	0,21	0,04	0,33
ß	0,01	0,03	0,22	0,00	0,95	0,95	0,03	0,24	0,02	0,27
Mittelwert Kontext	0,20	0,26	0,37	0,25	0,97	0,84	0,55	0,30	0,21	

Tabelle 3.4: Gewichte der Weight-Decay-Matrix für Deutsch. Die letzte Spalte gibt den Mittelwert der Gewichte für das entsprechende Graphem an, die letzte Zeile den Mittelwert für den gegebenen Kontext.

Kontext -1 gar nicht auf. Somit wird das Gewicht niemals benötigt und ausgedünnt. Auf der anderen Seite sagt das Graphem <j> an anderer Position als im Zentrum offenbar nichts über das zu generierende Phonem aus. Die Entscheidung ist also unabhängig von diesem Graphem. Somit werden diese Verbindungen ebenfalls nicht benötigt und auf Null gesetzt. Eine ausführlichere Diskussion zu diesem Thema ist in Abschnitt 3.4.2.2 zu finden.

3.3.5 Kodierung der Ein- und Ausgabe

In [Rose96] werden verschiedene Kodierungsverfahren für die Graphemeingabe vorgestellt, von denen einige eine sehr kompakte Kodierung darstellen. Die besten Ergebnisse wurden erzielt, wenn die Grapheme zum einen nach ihren Eigenschaften in sechs Gruppen eingeteilt (siehe Tabelle 1.4, Seite 21) und zum anderen je nach Kontext unterschiedlich gewichtet werden.

Solche Kodierungen sind aber stark von der Sprache abhängig, was gegen die Verwendung in einem multilingualen System spricht. Außerdem ist es z. B. im Englischen nur schwer möglich, eine ähnliche Zuordnung der Grapheme zu Phonemeigenschaften zu treffen. Deswegen erfolgt die Angabe der Werte in einer 1-aus-n Kodierung. Das heißt, daß ein Eingabeknoten soviele Neuronen hat, wie Werte möglich sind, wobei jedes Neuron genau einen Wert repräsentiert. Diese Kodierung wurde auch in [SR86] benutzt.

In [SR86] werden die Phoneme unter Verwendung ihrer artikulatorischen Eigengschaften, wie z. B. Ort der Artikulation, Stimmhaftigkeit, Vokalhöhe usw. kodiert. Damit wurden auch in [Rose96] die besten Ergebnisse erzielt. Eine solche Kodierung hat den Vorteil, daß die Komplexität des Netzes verringert wird, da weniger Ausgabeknoten und somit weniger Gewichte nötig sind. Sie hat aber auch einen entscheidenden Nachteil. Da sich einige Phoneme nur in einer Eigenschaft unterscheiden (z. B. der Länge bei /a/ und /a:/), unterscheiden sich auch die Ausgabevektoren für die beiden Laute nur geringfügig. Bei einer 1-aus-n Kodierung erreicht man, daß zwei verschiedene Laute durch zwei vollständig verschiedene Vektoren repräsentiert werden, was bei einer Fehlentscheidung des Netzes zu einem größeren Fehler führt und somit das Lernen erleichtert. Außerdem ist bei der Verwendung der phonetisch motivierten Kodierung keine Kombination von Phonem und Gruppierung möglich.

Auf der einen Seite ist eine kompakte Kodierung sicherlich von Vorteil, um die Komplexität des Netzes zu verringern. Auch eine Gewichtung der Eingabeknoten, z. B. über die Anzahl der verwendeten Neuronen, kann dafür nützlich sein. Je weniger Gewichte der Trainingsalgorithmus anzupassen hat, desto schneller läuft das Training. Man sollte sich aber immer vor Augen halten, daß eine solche Kodierung erstens ein gewisses Maß an Expertenwissen erfordert, und man zweitens dem Netz evtl. die Information in einer Art und Weise anbietet, die für den Menschen sinnvoll erscheint, mit dem das Netz aber vielleicht nichts anzufangen weiß.

Bei einer einfachen und übersichtlichen Kodierung ist der Algorithmus selber in der Lage, die seiner Meinung nach interessante Information entsprechend zu gewichten. Wenn die Graphemnachbarn des weiter entfernt liegenden Kontextes nicht soviel zur Entscheidung über das zu generierende Phonem beitragen, dann werden während des Trainings ihre Gewichte entsprechend

klein eingestellt. Außerdem stellt das Pruningverfahren selber auch fest, welche Verbindungen keine Information vermitteln oder schlimmstenfalls die Entscheidungen sogar negativ beeinflussen. Eine explizite Gewichtung der Knoten kann in diesem Zusammenhang eher schaden als nützen.

3.4 Bestimmung der Wortbetonung

Eine phonetische Silbe besteht aus einem Silbenkern, vor oder nach dem noch weitere Laute auftreten können. In vielen europäischen Sprachen können nur Vokale Silbenkerne sein. Ausnahmen bilden hier einige slawische Sprachen, z. B. Tschechisch, in dem auch die stimmhaften Konsonanten /r/ und /l/ Silbenkerne sein können. Das Wort *zmrzlina* (Eis) besteht aus drei Silben, wobei in der ersten *zmr* das /r/ den Silbenkern bildet. Im Slovakischen ist es sogar möglich, daß stimmhafte Konsonanten betont werden. Um also einen multilingualen Ansatz zu gewährleisten, darf man die Laute nicht in Vokale und Konsonaten einteilen, sondern muß viel allgemeiner unterscheiden, ob sie Silbenkern sein können oder nicht und ob sie betonbar sind oder nicht.

Die Aufgabe für den Algorithmus zur Wortbetonung besteht nun darin, den Silbenkern auszuwählen, der im Wort die Hauptbetonung trägt. Für die Bestimmung der Wortbetonung wurden zwei ähnliche Architekturen ausprobiert. Die erste versucht, die Theorie nachzubilden, nach der die letzte schwere Silbe betont ist (siehe Abschnitt 1.5.4). Diese Architektur wird im folgenden als silbenbasierter Ansatz bezeichnet.

Die zweite Architektur verzichtet auf die Silbeninformation. Die Eingabe besteht hier ausschließlich aus den ersten n Phonemen des Wortes. Dieser Ansatz geht davon aus, daß nicht die Verteilung der Phoneme auf die Silben, sondern die Anordnung der Phoneme hintereinander die Wortbetonung festlegt.

Was beiden Architekturen fehlt, ist wiederum die morphologische Information über die Wortbestandteile. So gibt es z. B. im Deutschen Vorsilben und Endungen, die nie betont werden. Dabei handelt es sich u. a. um die Präfixe *ge-*, *ent-*, *be-* usw. Die Phoneme dieser Affixe werden aber nur dann nicht betont, wenn sie eben aus diesen Graphemfolgen entstanden sind. Dafür ist es wichtig, diese Präfixe als solche zu erkennen, um die Entscheidung über die Betonbarkeit der Laute treffen zu können.

3.4.1 Silbenbasierter Ansatz

Bei diesem Ansatz wird einem neuronalen Netz die Silbenstruktur des Wortes gezeigt. Um den Kontext möglichst einheitlich gestalten zu können, wird für jede Silbe neben einem Knoten für den Silbenkern nur jeweils angegeben, ob die Silbe mit einem Konsonanten beginnt bzw. endet oder nicht. Dadurch gibt es für jede Silbe immer einen definierten Kontext. Anderenfalls hätte man das Problem, daß man für jede Silbe am Anfang und am Ende Knoten für mehrere

Konsonaten anlegen müßte, die dann aber vor allem an den Rändern nur selten besetzt wären. Das würde zu einem sehr großen Netz führen. Außerdem war ja die Idee, daß nur die Information über die An- oder Abwesenheit der Konsonanten eine Rolle spielt, nicht aber, um welche Konsonanten es sicht handelt.

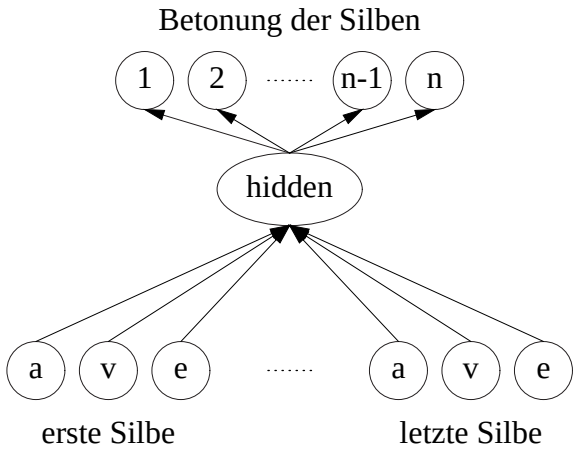

Abbildung 3.15: Architektur des Netzes zur Wortbetonung mit Silbeneingabe.

Die realisierte Architektur ist in Abbildung 3.15 zu sehen. Die Eingabe besteht aus 3 Knoten pro Silbe: einem Knoten, der den Silbenkern kodiert, und zwei Knoten, die jeweils auf 1 oder -1 gesetzt sind, wenn die Silbe mit einem Konsonanten beginnt bzw. endet oder nicht. Die Ausgabe besteht aus soviel Knoten, wie am Eingang Silben angelegt werden. Der Knoten mit der maximalen Ausgabe legt die betonte Silbe fest. Es werden natürlich nur die Ausgabeknoten untersucht, deren zugehöriger Eingabeknoten betonbar ist.

Der Nachteil dieser Architektur gegenüber der im nächsten Abschnitt beschriebenen phonembasierten Vorgehensweise liegt in der Abhängigkeit des Netzes von der Silbengrenzeninformation. In dem verwendeten TTS-System erfolgt die Bestimmung der Wortbetonung im Anschluß an die Bestimmung der Phonemfolge inklusive der Silbengrenzen. Die Ergebnisse zeigen zwar, daß Fehler bei den Silbengrenzen seltener auftreten als bei den Phonemen, aber je korrekter die Eingabe ist, desto weniger Fehler treten bei der Ausgabe auf. Wenn z. B. für das Wort *Ölofen* nicht die Transkription `/2: l - o: - f @ n/`, sondern `/2: - l o: - f @ n/` angelegt wird, so ist die Information sowohl über das Ende der ersten als auch über den Anfang der zweiten Silbe falsch, was durchaus zu einer falschen Betonung des Wortes führen kann.

Bestimmung des Kontextes

Zur Bestimmung des Kontextes wird eine Statistik über das phonetische Lexikon erstellt, die eine Aussage über die Verteilung der Silben und deren Betonungen trifft. Für den silbenbasierten Ansatz wird gezählt, wie oft die wievielte Silbe in einem Wort mit wieviel Silben betont wird. Das Ergebnis für das deutsche Lexikon ist in Tabelle B.1 (Seite 113) zu sehen. Aus dieser

Tabelle ergibt sich, daß im deutschen Lexikon Wörter mit bis zu 11 Silben auftreten (z. B. Angestelltenversicherungsgesetze). Von den 5 Wörtern mit 11 Silben wurden 4 auf der ersten Silbe und 1 auf der fünften Silbe betont. Die Betonung liegt maximal auf der achten Silbe (z. B. bei Disproportionalität[2]), nachfolgende Silben sind nicht betont. Berücksichtigt man die ersten fünf Silben, so sind bereits 99,6 Prozent der betonten Silben erfaßt. Deswegen wurde der Kontext für das Netz auf Silbenebene auf fünf Silben festgelegt.

3.4.2 Phonembasierter Ansatz

Um das Problem mit der potentiell falschen Information über die Silbengrenze zu vermeiden, wurde auf die Einteilung in Silben verzichtet und nur die Phonemfolge als Eingang für ein Netz gewählt. Diese Architektur hat die Nachteile, daß durch die Kodierung mehrerer Phoneme das Netz sehr groß wird und daß nun doch die Laute am Eingang angelegt werden, die nicht betonbar sind und auch keinen Silbenkern bilden können. Dadurch wird dem Netz die Möglichkeit geboten, einzelne Phonemfolgen zu lernen. Tests auf jeweils dem gleichen Lexikon haben aber ergeben, daß die Ergebnisse mit diesem Netz deutlich besser sind.

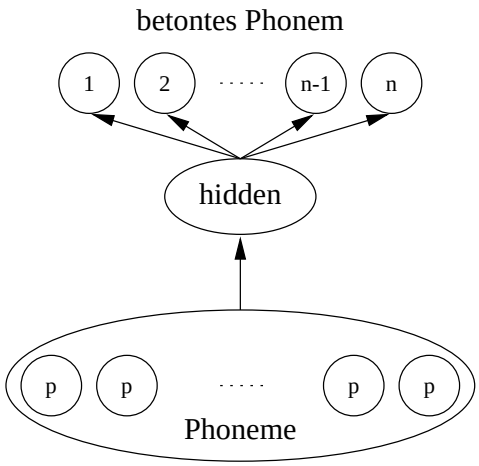

Abbildung 3.16: Architektur des Netzes zur Wortbetonung mit Phonemeingabe.

Die Architektur des Netzes ist in Abbildung 3.16 zu sehen. Die Eingabe besteht aus den ersten n Phonemen des Wortes. Die Ausgabe hat soviele Knoten, wie Phoneme eingegeben werden. Wiederum legt der Ausgangsknoten mit dem maximalen Wert das betonte Phonem fest, an dessen zugehörigem Eingabeknoten ein betonbarer Laut anliegt.

[2] Im deutschen Celex wird das Derivationssuffix *tion* in die Silben `/ts i:/` und `/o: n/` aufgeteilt.

3.4.2.1 Bestimmung des Kontextes

In Tabelle B.4 (Seite 115) ist die Verteilung der Betonungen auf Phonemebene angegeben. Die zweite Spalte enthält die Anzahl der Wörter mit n Phonemen, die vierte gibt an, wie oft das Phonem n betont war. Während also 46105 Wörter 9 Phoneme haben, ist nur 4293 mal die neunte Silbe betont. Die dritte und fünfte Spalte geben das prozentuale Verhältnis für die jeweilen Werte an.

Aus dieser Tabelle kann man ablesen, daß bei 98,4 Prozent der Wörter die Betonung innerhalb der ersten neun Phoneme liegt. Spätere Phoneme tragen nur noch weniger als ein Prozent bei. Deswegen werden bei dem Betonungsnetz auf Phonemebene die ersten 9 Phoneme betrachtet.

3.4.2.2 Bewertung der Eingangsinformation durch Weight Decay

Im vorigen Abschnitt wurde beschrieben, warum für eine Sprache eine bestimmte Kontextgröße gewählt wurde. Dabei handelt es sich aber um eine „harte“ statistische Entscheidung. Es stellt sich nun die Frage, ob die angebotene Information überhaupt genutzt werden kann. Außerdem bietet sich die Möglichkeit, durch das Ausblenden unnützer Eingabeknoten die Ergebnisse zu verbessern.

Dazu wurde das in Abschnitt 2.1.7 vorgestellte Weight Decay implementiert [HZ01a, HZ01b]. Es wurde analog zu Abschnitt 3.3.4 eine zusätzliche versteckte Schicht (`wd_input`) in das Netz eingefügt, die eine Zwischeneingabe realisiert. Die neue Architektur ist in Abbildung 3.17 zu sehen.

Die eigentliche Eingabe und die Zwischeneingabe sind über eine Diagonalmatrix miteinander verbunden. Die Werte auf der Diagonale werden mit 1 initialisiert, alle anderen mit 0. Damit ist jeder Eingabeknoten nur mit dem korrespondierenden Knoten der Zwischenschicht verbunden. Während des Trainings wird nun ausschließlich für diese Diagonalmatrix das Weight Decay aktiviert, also der Strafterm zur Fehlerfunktion addiert. Die restliche Architektur bleibt unverändert. Als Aktivierungsfunktion für die Zwischenschicht wird wieder *tanh* verwendet.

Bei einem Traning für Deutsch mit dem maximalen Kontext von 16 werden diese Diagonalverbindungen beobachtet. Der Verlauf der Gewichte ist in Abbildung 3.18 zu sehen, die dazugehörgien Werte stehen in Tabelle 3.5.

Während die auf den Trainingsmustern errechnete Transinformation der prozentualen Verteilung der Betonungen innerhalb des Lexikons ähnelt, haben die Mittelwerte einen anderen Verlauf. Ihr Maximum liegt bei Kontext 4, und auch die Kontexte 3, 5 und 6 haben größere Werte als Kontext 2. Es ist aber bei beiden Verläufen gut zu sehen, daß, wie erwartet, die weiter hinten liegenden Knoten weniger Information vermitteln. Die Statistik über die Betonungsverteilung ist also gut dafür geeignet, die Größe des sinnvollen Kontextes zu bestimmen.

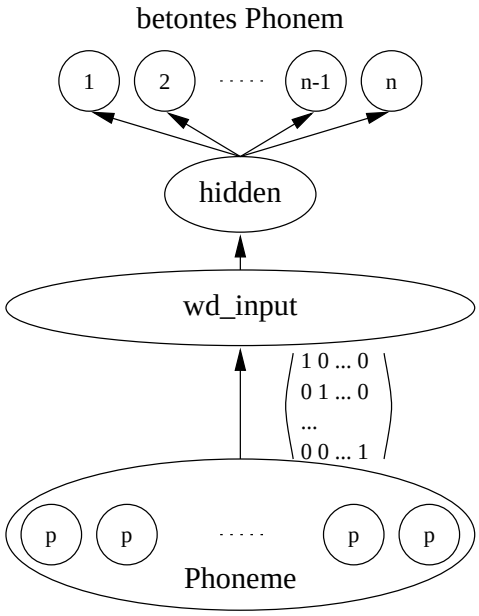

Abbildung 3.17: Neuronales Netz zur Bestimmung der Wortbetonung mit der Weight-Decay-Verbindung.

3.5 Verbesserung der Transkription durch Auswertung zusätzlicher Wissensquellen

In Abschnitt 3.1.2.9 wurde beschrieben, daß im Ergebnis der Lexikonaufbereitung zwei Wissensquellen entstehen, die man zur weiteren Verbesserung der Transkriptionsergebnisse verwenden kann. Zum einen kann man mit Hilfe des aufbereiteten Lexikons die Transkription an Wortenden korrigieren, und zum anderen kann man mit der Liste der Zuordnungen zwischen Phonemen und Graphemgruppen die Ausgabe des neuronalen Netzes überprüfen und somit grobe Fehler vermeiden.

3.5.1 Kombination von Lexikon und neuronalen Netzen

Mit diesem Verfahren sollen die Vorteile des sicheren aber unflexiblen Lexikons auf der einen Seite mit der Flexibilität aber Unsicherheit des neuronalen Netzes auf der anderen Seite verbunden werden [Hain00a, Hain00c, Hain00b, Hain03b]. Die Transkription im Lexikon kann als sicher angenommen werden. Allerdings ist es unmöglich, alle Wörter einer Sprache im Lexikon halten zu können. Ein neuronales Netz hingegen kann für jede Graphemfolge eine Transkription bestimmen. Diese kann aber fehlerhaft sein.

Es wird zuerst versucht, das Wort aus Einträgen des Lexikons zusammenzusetzen. Um sicherzustellen, daß die gefundene Transkription repräsentativ ist, muß die gefundene Graphemfolge

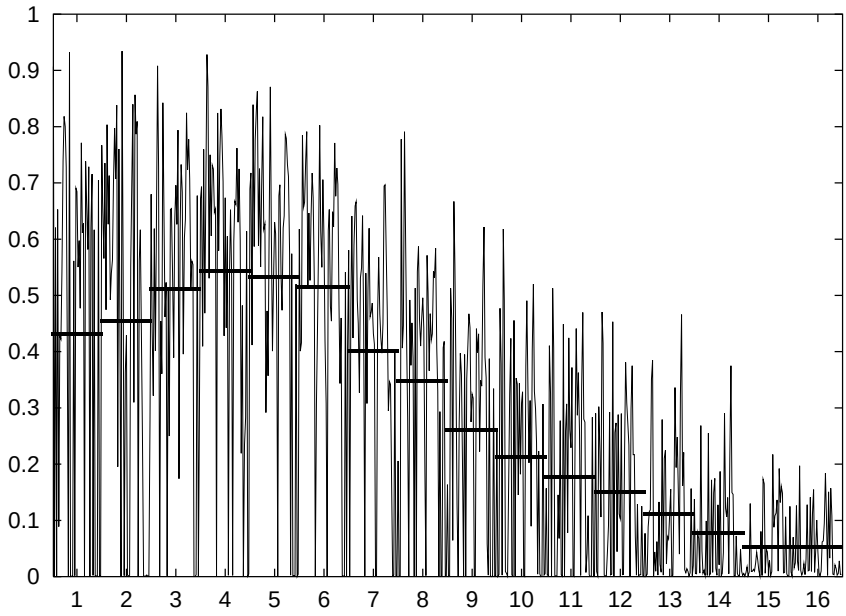

Abbildung 3.18: Verlauf der Gewichte der Weight-Decay-Verbindungen für Deutsch. Die horizontalen Linien geben wiederum die Mittelwerte für den entsprechenden Kontext an.

Kontext	Transinformation	Verteilung der Betonungen (in %)	Mittelwerte
1	0,643	19,16	0,431
2	0,707	33,93	0,455
3	0,463	11,75	0,511
4	0,405	10,55	0,543
5	0,321	9,48	0,533
6	0,285	6,16	0,514
7	0,256	3,72	0,402
8	0,223	2,22	0,348
9	0,194	1,39	0,260
10	0,169	0,87	0,212
11	0,149	0,48	0,177
12	0,117	0,17	0,151
13	0,095	0,076	0,111
14	0,080	0,031	0,078
15	0,076	0,012	0,053
16	0,096	0,005	0,052

Tabelle 3.5: Transinformation, prozentuale Verteilung der Betonungen (siehe Tabelle B.4) und Mittelwerte der Weight-Decay-Verbindungen für Deutsch für den gegebenen Kontext.

eine gewisse Mindestlänge haben. Wird für den Anfang des Wortes kein Eintrag mit mindestens dieser Länge gefunden, so werden gleich die ersten Grapheme übersprungen und innerhalb des Wortes weitergesucht. Die Anzahl der zu auszulassenden Grapheme ist dabei abhängig von der Qualität der Netze (siehe Abschnitt 4.4.1). Nachdem alle Teile gefunden wurden bzw. das Wortende erreicht ist, wird die Transkription ausgewählt, für die die wenigsten und längsten Teile genutzt werden. Eventuell auftretende Lücken zwischen den Bestandteilen werden mit den neuronalen Netzen aufgefüllt.

Ein Problem bei dieser Zusammensetzung ist, daß die Transkriptionen aus dem Lexikon nur mit Einschränkungen verwendbar sind. Ein Grund dafür ist die Auslautverhärtung im Deutschen. Die Transkription von *Schlag* ist `/S l a: k/`, die von *Schlages* jedoch `/S l a: - g @ s/`. Zum einen wird bei *Schlages* das *g* als `/g/` gesprochen, da es am Silbenanfang steht, zum anderen befindet sich die Silbengrenze innerhalb der gefundenen Transkription. Eine Verbindung der Transkriptionen von *Schlag* und *es* würde `/S l a: k - E s/` statt `/S l a: - g @ s/` ergeben. Ohne besondere Behandlung solcher Fehler ist es nicht meist nicht möglich, die Transkription des gesamten Wortes aus den im Lexikon gefundenen Teiltranskriptionen zusammenzusetzen.

Die Lösung für dieses Problem ist, sich für jeden Eintrag zu merken, aus welchen Graphemen das letzte Phonem generiert wurde [Hain00c, Hain00a, Hain03a]. Diese Information erhält man aus dem Lexikon, in dem die Graphem-Phonem-Zuordnung enthalten ist. Dieses Lexikon wird mit dem im Abschnitt 3.1 beschriebenen Verfahren gewonnen. Somit kann man Phonem und Silbengrenze am Ende des linken Bestandteiles einer Zusammensetzung mit Hilfe des neuronalen Netzes neu bestimmen.

Dies soll an dem Wort *überflüssigerweise* erläutert werden. Dieses Wort steht nicht im Lexikon, dafür aber die beiden Einträge *überflüssig* und *erweise*. Sie repräsentieren aber nicht die morphologische Zusammensetzung von *überflüssigerweise*, weswegen die reine Verkettung der Teiltranskriptionen

```
          / y: - b 6 - f l Y - s I C - E r - v aI - z @ /
statt
        / y: - b 6 - f l Y - s I - g 6 - v aI - z @ /
```

liefert. Hier sind also Phoneme und Silbengrenzen falsch. Die Aufgabe für das neuronale Netz besteht nun darin zu entscheiden, welches Phonem am Ende des linken Teilwortes verwendet werden soll und ob sich davor eine Silbengrenze befindet oder nicht.

Der Eintrag für *überflüssig* im aufbereiteten Lexikon ist

```
,ü  - b er - f l ü - ss i g
"y: - b 6  - f l Y - s  I C
```

Damit wird die Eingabeinformation für das neuronale Netz erstellt (siehe Abbildung 3.12). Das letzte Phonem `/C/` ist aus dem Graphem `<g>` entstanden. `<g>` wird also im Zentrum *z* angelegt. Der linke Graphemkontext besteht aus den letzten vier Graphemen vor `<g>` (`<üssi>`), der rechte

aus den ersten vier Graphemen von *erweise* (<erwe>). Die Information für die Netzwerkknoten, die die vorhergehende Entscheidung repräsentieren, wird mit Hilfe des Phonemes vor /C/ gewonnen. Bei diesem Phonem /I/ handelt es sich um einen Silbenkern, die Variable *vsk* wird auf 1 gesetzt. Das vorhergehende Phonem *lp* ist /I/, die vorhergehende Gruppierung *lg* ist 1, und vor dem /I/ befindet sich keine Silbengrenze, *ls* ist demzufolge 0. Tabelle 3.6 gibt noch einmal eine Übersicht über die an den Eingabeknoten angelegte Information.

l	c	r	vsk	lg	lp	ls
üssi	g	erwe	1	1	I	0

Tabelle 3.6: Eingabe für das neuronale Netz zur Entscheidung über das letzte Phonem.

Die Ausgabe besteht aus dem Phonem /g/ mit gesetzter Silbengrenze. Die korrigierte Phonemfolge ist nun

/ y: - b 6 - f l Y - s I - g E r - v aI - z @ /

statt

/ y: - b 6 - f l Y - s I - g 6 - v aI - z @ /

An dieser Stelle wäre noch eine weitere Korrektur nötig. Statt des Phonems /6/ werden die Phoneme /E r/ verwendet. Dies ist zwar immer noch ein Fehler, aber bei weitem nicht so gravierend wie die Verwechslung von /C/ und /g/. Außerdem ist diese Korrektur nicht mehr so einfach zu handhaben, da erst entschieden werden müßte, wieviele Phoneme am Anfang des rechten Teilwortes korrigiert werden sollen. Meist ist aber das Problem nur das Phonem am Ende des linken Teilwortes, und am Anfang des rechten sind keine Änderungen nötig.

3.5.2 Bewertung der Netzausgabe mit Hilfe der Graphem-Phonem-Zuordnungen

Aus dem im Abschnitt 3.1.2.9 beschriebenen aufbereiteten Lexikon läßt sich eine Liste der Zuordnungen gewinnen, welche Graphemgruppen welche Phoneme generieren können bzw. welche Phoneme aus welchen Graphemgruppen generiert werden können. Diese werden in der Form

```
Phonem, Graphem-Gruppe Graphem-Gruppe ... Graphem-Gruppe
```

abgespeichert und zur Bewertung der Ausgabe des neuronalen Netzes herangezogen. Damit kann eine offensichtliche Fehlentscheidung des Netzes verhindert werden.

Aus der Zuordnungsliste wird eine Liste der Phoneme erstellt, die aus der jeweiligen Graphemgruppe generiert werden kann. Im folgenden werden dann nur diese Phoneme überprüft und die entsprechenden Ausgabeknoten ausgewertet. Ausgangspunkt ist die Annahme, daß der vom

Netz ausgegebene Wert für die Gruppierung weniger fehleranfällig ist als der für das Phonem. Dies geschieht in folgenden Schritten:

1. Entsprechend der Netzgruppierung werden die ersten Zeichen der zu transkribierenden Zeichenkette in der Zuordnungsliste gesucht. Wird eine solche Graphemgruppe gefunden, so wird diese übernommen.
2. Wird im ersten Schritt keine Gruppe gefunden, so wird die Gruppierung schrittweise bis auf Eins verringert und nach der so verkürzten Graphemgruppe gesucht.
3. Ist auch nach der Verringerung der Gruppierung auf eins keine Gruppe zu finden, so wird die Gruppierung schrittweise bis zum maximal möglichen Wert erhöht und nach längeren Graphemgruppen gesucht.

Wenn in allen drei Schritten keine Graphemgruppe gefunden werden kann, so werden wie sonst auch alle Phonemknoten auf ihr Maximum untersucht und das Phonem, das diesem Knoten entspricht, als das generierte Phonem ausgewählt. Falls aber eine Gruppe gefunden wurde, so werden nur die möglichen Phonemknoten untersucht.

Dies soll an einem Beispiel kurz erläutert werden. Angenommen, die zu transkribierende Zeichenkette sei *sch* und die Ausgabe für die Gruppierung sei zwei. Damit wird im ersten Schritt in der Zuordnungsliste nach der Graphemgruppe *sc* gesucht. Diese wird aber nicht gefunden. Daraufhin wird die Gruppierung auf eins verringert. Die neue Graphemgruppe *s* wird gefunden. Die möglichen Phoneme sind `/s/` , `/S/` und `/z/`. Nun werden also nur diese drei Ausgabeknoten auf ihr Maximum untersucht.

3.6 Training der Netze

3.6.1 Die verwendete Trainingsprozedur

Um das Training automatisch ablaufen lassen zu können, wurde nicht die graphische Oberfläche benutzt, sondern die SENN-Funktionen wurden über die TCL-Schnittstelle angesprochen. Dadurch bot sich neben rein praktischen Vorteilen die Möglichkeit, den Trainingsablauf genauer zu steuern und in den Trainingsablauf eingreifen zu können, ohne das Training anhalten zu müssen. Ein Ablaufplan des Trainings ist in Abbildung 3.19 dargestellt.

Zu Beginn werden alle Gewichte mit zufälligen Werten initialisiert. Das gilt natürlich nicht für die Diagonalmatrizen zwischen der Eingabe- und der Weight-Decay-Schicht. Hier werden die Gewichte auf der Diagonale auf 1 und der Rest auf 0 gesetz. Der Startwert für die Lernschrittweite η ist 0,1.

Nach jeder Iteration werden die Signale der Trainingsmenge („late stopping“, siehe Abschnitt 2.1.6) ausgewertet und bei einer Verbesserung der Werte die aktuellen Gewichte abgespeichert.

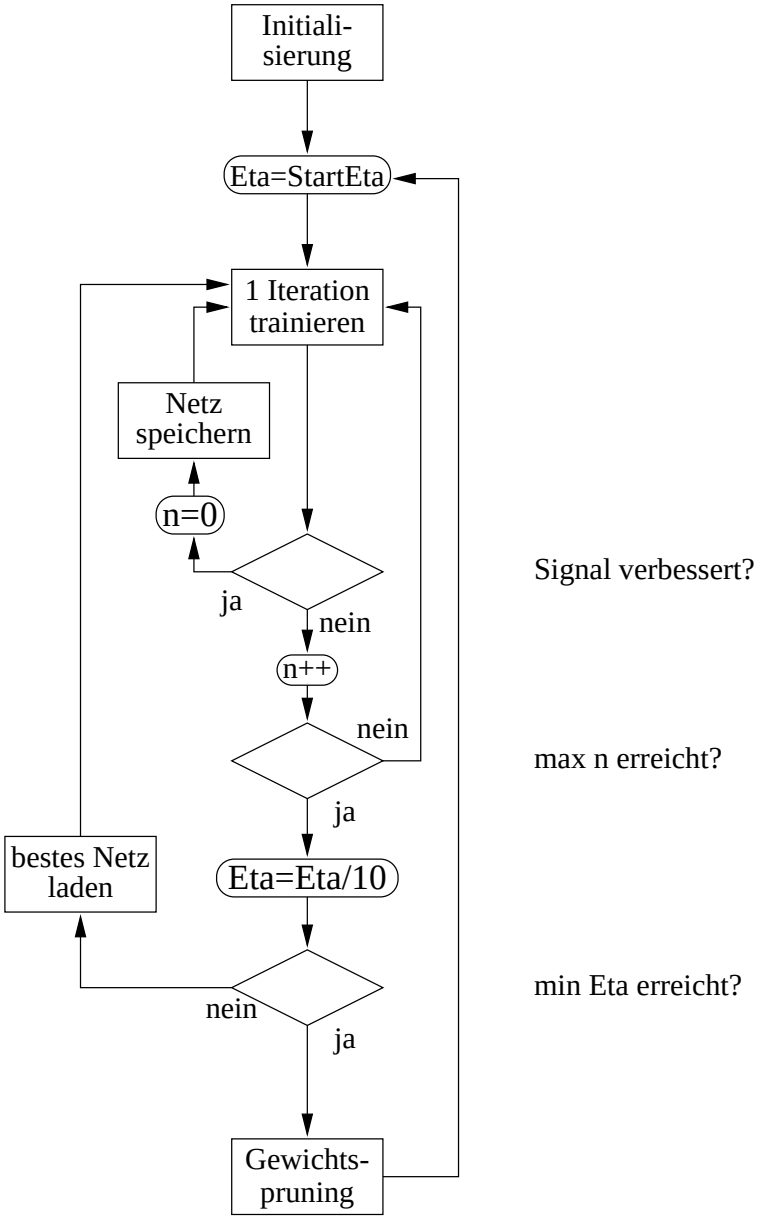

Abbildung 3.19: Ablaufplan der Trainingsprozedur.

Unter Signalen versteht man im SENN die Möglichkeit, nach einer Iteration die Anzahl der richtig berechneten Muster auszuwerten, sowohl für die Trainings- als auch für die Testmenge. Damit kann man zusätzlich zum errechneten Fehler auf diesen beiden Mengen abschätzen, wie gut das Netz die eigentliche Aufgabe gelernt hat. Ein sinkender Fehler bedeutet nicht gleichzeitig, daß die Anzahl der richtig gelernten Muster steigt. Durch die Betrachtung der Signale als Abbruchkriterium wird immer das Netz gespeichert, das der gestellten Aufgabe am besten gerecht wird.

Wenn sich die Signale innerhalb einer gewissen Anzahl von Iterationen nicht verbessert haben, so wird das zuletzt gespeicherte Netz geladen und η auf $0,01$ verringert. Nun wird wieder trainiert, bis sich die Signale nicht mehr verbessern. η wird dann auf $0,001$ verringert und weiter trainiert. Wenn sich dann die Signale nicht mehr verbessern, beginnt das Ausdünnen.

Zum Ausdünnen wird das EBD-Pruning verwendet (siehe Abschnitt 2.1.5.2). Dabei wird immer ein gewisser Prozentsatz der Gewichte ausgeschaltet. Gewichte mit einem negativen Testwert werden nicht wiederbelebt. Im Anschluß an das Ausdünnen wird eine Iteration mit einem η von $0,001$ trainiert, dann wird η auf $0,01$ gesetzt. Ein η von $0,1$ ist zu groß, damit würde man die gelernte Struktur nur wieder zerstören (siehe Abschitt 4.3.2.1). Es wird wieder trainiert, bis sich die Signale nicht mehr verbessern, η wird auf $0,001$ gesetzt, es wird weiter trainiert und ausgedünnt. Das Training wird beendet, wenn sich das beste Signal vor dem Ausdünnen nicht mehr verbessert.

Diese „gewissen“ Werte und Prozentsätze sind wichtige Parameter für das Training und Ausdünnen. Die Anzahl der Iterationen, die man abwartet, ohne daß sich die Signale verbessern, hat einen direkten Einfluß auf die erzielten Ergebnisse. Bei einem zu schnellen Verringern der Schrittweite gibt man dem Netz nicht genug Zeit, ein besseres Minimum zu finden. Auf der anderen Seite erhöht sich damit die Trainingszeit erheblich, da viel mehr Iterationen durchlaufen werden. Ein guter Kompromiß für die beiden verwendeten Netze ist hier ein Wert von 10. Wenn sich die Signale also nicht innerhalb von 10 Iterationen verbessern, wird η verringert bzw. das Ausdünnen gestartet.

Bei dem Prozentsatz der auszudünnenden Gewichte ergibt sich ein ähnliches Problem. Werden zuwenig Gewichte auf einmal ausgedünnt, so dauert das Training insgesamt sehr lange, da viele Iterationen bis zum Erreichen der besten Werte nötig sind. Werden zuviel Gewichte ausgeschaltet, wird u. U. die bereits gelernte Struktur zerstört und es wird nur ein schlechtes Endergebnis erzielt. Dieses Verhalten wird in Abschnitt 4.3.2.1 näher untersucht.

3.6.2 Netzwerkparameter

In diesem Abschnitt werden die Einstellungen beschrieben, die für das Training der Netze verwendet wurden. Beide Netze haben eine einfache Multilayer-Perceptron-Architektur (MLP-Architektur), bei der der Informationsfluß in nur eine Richtung geht (feed forward). Es gibt keine direkte Rückkopplung. Nur bei dem Netz zur Bestimmung von Phonem und Silbengrenze gibt es eine implizite Rückkopplung, indem die Ausgabe bei der nächsten Entscheidung für

das gleiche Wort am Eingang wieder angelegt wird. Das hat aber auf das Training selber keinen Einfluß.

Für die Eingabeneuronen wird als Aktivierungsfunktion die Identität eingestellt; sie geben also die angelegten Werte ohne Änderung weiter. Für die versteckte Schicht und auch für die Ausgabe wird als Aktivierungsfunktion *tanh* verwendet. Die Nichtlinearität auch bei der Ausgabe hat den Vorteil, daß das Lernen der auszugebenden Werte vereinfacht wird.

Die Ausgabewerte sind 1-aus-n kodiert. Der Zielwert für das Neuron, welches das gesetzte Phonem repräsentiert, ist $+1$, alle anderen Neuronen werden auf -1 gesetzt. Das Neuron für die Silbengrenze wird ebenfalls auf $+1$ gesetzt, wenn eine Silbengrenze markiert werden soll, sonst auf -1.

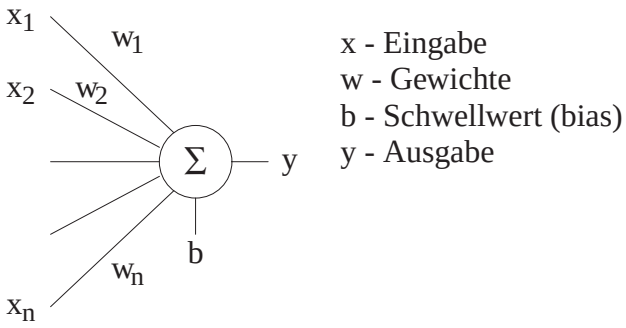

Abbildung 3.20: Darstellung eines Neurons.

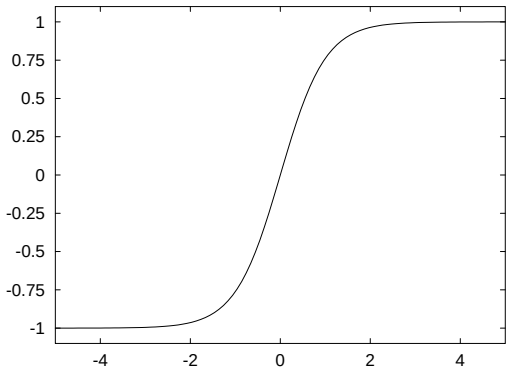

Abbildung 3.21: Tangens hyperbolicus.

Die Ausgabe y eines Neurons, wie es in Abbildung 3.20 dargestellt ist, errechnet sich aus

$$y = f_{act}\left(b + \sum x_i \cdot w_i\right) \tag{3.21}$$

mit der Aktivierungsfunktion f_{act}. Ohne die Aktivierungsfunktion wäre es schwierig für den Trainingsalgorithmus, für alle Pattern die Summe $b + \sum x_i \cdot w_i$ auf $+1$ oder -1 einzustellen. Das gilt besonders für das Netz zur Bestimmung der Phoneme. Das Netz für Englisch hat z. B. 165 Ausgabeknoten. Die Aufgabe besteht also darin, die Summe aus 165 Produkten und dem Bias genau auf 1,0 einzustellen. Wird die Summe allerdings mit *tanh* „bewertet“, so ist die Aufgabe viel einfacher, da $tanh(x)$ für Werte größer 3 nahe 1 ist (siehe Abbildung 3.21). Der Trainingsalgorithmus muß also nicht erzwingen, daß die Summe 1 wird, sondern er muß nur Sorge dafür tragen, daß sie größer als 3 wird. Das vereinfacht die Aufgabe erheblich.

Eine ähnliche Vereinfachung kann durch die Verwendung der *Tolerance* erzielt werden (siehe Abschnitt 2.1.4.4). Durch die Vorgabe einer Toleranz von 0,1 kann man erreichen, daß die Summe im Bereich von 0,9 bis 1,1 liegen muß, ohne daß dies als Fehler gewertet wird. Das erleichtert ebenfalls die Einstellung der Netzwerkparameter.

Zu Beginn wurden die Gewichte in einem Wertebereich von $-0,01$ und $0,01$ initialisiert. Danach beginnt das Training mit der Musterauswahl *Stochastic* (siehe Abschnitt 2.1.4.1) und dem Suchverfahren *VarioEta* (siehe Abschnitt 2.1.4.2). Die Lernparameter werden auf die Werte in Tabelle 3.7 gesetzt.

Parameter	Wert
Musterauswahl	Stochastic
Auswahl der Untermenge	Permute
Suchverfahren	VarioEta
Eta	0,1 bis 0,001
Batchsize	50
DerivEps	0,001

Tabelle 3.7: Lernparameter des Trainings.

Kapitel 4

Ergebnisse und Auswertung

In diesem Kapitel werden die erreichten Ergebnisse dargestellt und bewertet.

4.1 Aufbereitung des Lexikons

In diesem Abschnitt werden die in 3.1 vorgestellten Verfahren verglichen.

4.1.1 Suche nach Graphemgruppen

Das Verfahren zur Suche nach Graphemgruppen wurde speziell für Deutsch entwickelt. Für diese Sprache wurden auch gute Ergebnisse erzielt. Bei der Anwendung auf andere Sprachen wie Englisch und Französisch stellte sich aber heraus, daß es sich nur bedingt für diese Sprachen eignet. Das liegt daran, daß in diesen Sprachen die Graphemgruppen deutlich größer sein können und es viel mehr Möglichkeiten gibt, bei denen Grapheme nicht unmittelbar zur Aussprache beitragen. Deswegen wurde dieser Ansatz nicht weiter verfolgt und dafür das Verfahren mit der dynamischen Programmierung weiterentwickelt.

4.1.2 Dynamische Programmierung

Da das Verfahren zur Suche nach Graphemgruppen für andere Sprachen außer Deutsch nur unzureichende Ergebnisse liefert, wurde ein Verfahren entwickelt, das die dynamischen Programmierung verwendet. Dadurch konnten die Ergebnisse deutlich verbessert werden. Die dynamische Programmierung ist viel besser in der Lage, mit der z. T. sehr großen Anzahl von Abbildungsmöglichkeiten umzugehen.

Für eine Beurteilung der Ergebnisse wurden für jede Sprache jeweils 300 zufällig ausgewählte Einträge manuell überprüft. Die Resultate sind in Tabelle 4.1 zu sehen.

Sprache	Phoneme	Fehler	Fehlerrate	
			Phonemebene	Wortebene
Deutsch	2897	0	0	0
Englisch	2110	33	3,1	11
Holländisch	2995	5	0,3	3,3

Tabelle 4.1: Ergebnisse der Lexikon-Aufbereitung mit dynamischer Programmierung.

Für Deutsch ist es schwer, die Fehlerrate zu bestimmen, da in den 300 Beispielen kein einziger Fehler auftrat. Einige Fehler gibt es bei Wörtern wie

```
R hei n
r aI  n ,
```

bei denen sehr seltene Zuordnungen vorkommen. Hier wird <h> dem Phonem /aI/ zugeordnet, weil es noch unwahrscheinlicher ist, daß es zum /r/ gehört, als daß es an Position 1 dem /aI/ zugeordnet wird.

Die 33 Fehler im Englischen ergeben 66 falsche Graphem-Phonem-Zuordnungen. Das entspricht einer Fehlerrate von 3,1 Prozent auf Phonemebene. Damit enthalten aber 11 Prozent der Wörter mindestens einen Fehler. Dies könnte auch ein Grund dafür sein, daß die Graphem-Phonem-Konvertierung für Englisch nicht so gute Ergebnisse liefert wie für Deutsch.

Im Holländischen treten 5 Fehler auf. Die Fehlerrate auf Phonemebene beträgt demnach 0,3 Prozent, die auf Wortebene 3,3 Prozent.

Einträge, die nicht behandelt werden können, werden in einem separaten Lexikon zusammen mit ihren DTW-Matrizen abgespeichert. Dabei handelt es sich zum einen um sehr lange Einträge und zum anderen um Einträge mit einer fehlerhaften Transkription. Dieses Lexikon kann nun weiter untersucht werden, um Transkriptionsfehler zu finden oder herauszubekommen, an welcher Stelle das Verfahren bei einem bestimmten Wort nicht weiterkam. Dadurch kann es Schritt für Schritt verbessert werden.

Um Transkriptionsfehler einfacher korrigieren zu können, wurde ein Programm geschrieben, mit dessen Hilfe alle Zuordnungen anzeigen werden können. Man kann also z. B. nach der Zuordnung <hei> zu /aI/ suchen und bekommt das Wort *Rhein* angezeigt. Nun kann man selber entscheiden, ob die Transkription falsch ist oder das Verfahren einen Fehler gemacht hat.

4.2 Netz für Phonemfolge und Silbengrenze

Die in Abschnitt 3.3 vorgestellten Netze wurden für die Sprachen Deutsch, Englisch und Holländisch trainiert. Tabelle 4.2 gibt einen Überblick der erzielten Ergebnisse.

Verfahren	Deutsch	Englisch	Holländisch
mit Gruppierung, ohne WD	89,3 %	64,9 %	87,3 %
ohne Gruppierung, ohne WD	92,6 %	71,7 %	88,7 %
ohne Gruppierung, mit WD	94,3 %	74,0 %	89,6 %

Tabelle 4.2: Ergebnisse der neuronalen Netze für Phonemfolge und Silbengrenze für Deutsch, Englisch und Holländisch. Angegeben ist der Prozentsatz der richtig transkribierten Wörter auf dem gesamten Lexikon (ohne Auswertung der Graphem-Phonem-Zuordnungen, siehe Abschnitt 4.4.2).

Dabei wird zum einen zwischen den Netzen mit (Abschnitt 3.3.1) und ohne (Abschnitt 3.3.2) Gruppierungsknoten unterschieden. Bei der ersten Architektur gibt es einen Ausgabeknoten, der die Anzahl der Grapheme angibt, welche das Ausgangsphonem generiert haben. Bei der zweiten Architektur hingegen ist die Gruppierung Teil der Phonemkodierung. Zum anderen wird bei den Netzen ohne Gruppierungsknoten noch zwischen einem Training mit und ohne Weight Decay unterschieden.

Die Ergebnisse haben sich mit den Weiterentwicklungen stets verbessert. Der Schritt von der Verwendung eines gesonderten Ausgangsknotens für die Gruppierung hin zur Kombination aus Phonem und Gruppierung hat die Ausgabe des Netzes stabilisiert. Ein während des Trainings gemachter Fehler (richtiges Phonem bei falscher Gruppierung) wird jetzt stärker bestraft, da bei einer Fehlentscheidung sowohl Phonem als auch Gruppierung falsch sind. Außerdem ist es nun gar nicht mehr möglich, für ein Phonem eine unmögliche Gruppierung auszugeben (z. B. Gruppierung 4 für Phonem `/t/`), da beide Informationen direkt miteinander verbunden sind.

Der Einsatz des Weight Decay hat die Ergebnisse noch einmal etwas verbessert. Durch die Gewichtung der Eingabe nach ihrem Informationsgehalt ist das Netz besser in der Lage, die für das Training hilfreiche Information von dem störenden Rauschen in den Daten zu trennen. Eingabeknoten, die keine Information in das Netz vermitteln, werden mit einem sehr kleinen Gewicht „bestraft“ und somit quasi ausgeblendet. Ein positiver Nebeneffekt des Weight Decay ist die Beschleunigung des Trainings. Es sind viel weniger Iterationen bis zum Erreichen des besten Ergebnisses nötig.

Am Beispiel des Netzes für Deutsch soll im nächsten Abschnitt eine genauere Fehleranalyse durchgeführt werden. Die Netze für Englisch und Holländisch werden in Abschnitt 4.2.2 verglichen.

4.2.1 Fehleranalyse des Netzes für Deutsch

In diesem Abschnitt werden die Fehler des neuronalen Netzes für Deutsch näher untersucht. Dabei werden zuerst die drei Varianten gegenübergestellt. Im Anschluß daran erfolgt eine genauere Untersuchung des Netzes ohne Gruppierungsknoten und mit Weight Decay, da es die besten Ergebnisse liefert.

4.2.1.1 Vergleich der drei Netzvarianten

Tabelle 4.3 zeigt etwas ausführlicher die Ergebnisse, die mit den drei verschiedenen Netzwerkvarianten erzielt werden. Der Test wurde auf dem Lexikon mit ca. 310000 Wörtern durchgeführt, das zur Erstellung der Trainings- und Testmuster verwendet wurde.

	mit Gruppierung	ohne Gruppierung	
		ohne WD	mit WD
Wörter	89,3 %	92,6 %	94,3 %
Muster	97,2 %	98,0 %	98,5 %
Phoneme	97,9 %	98,5 %	98,8 %
Gruppierungen	99,5 %	99,6 %	99,7 %
Silbengrenzen	98,6 %	98,9 %	99,2 %

Tabelle 4.3: Gegenüberstellung der Ergebnisse für die drei Netzwerkvarianten für Deutsch.

In der ersten Zeile sind noch einmal die richtig transkribierten Wörter angegeben. Bei diesen Wörtern sind also alle Phoneme und alle Silbengrenzen korrekt. Die zweite Zeile enthält den Prozentsatz richtig behandelter Muster. Für die Transkription der 310000 Wörter sind ungefähr 3 Millionen Eingabemuster durchzurechnen (9,6 Phoneme pro Wort). Die Zahl gibt an, bei wievielen Berechnungen Phonem, Gruppierung und Silbengrenze richtig bestimmt wurden. In den letzten drei Zeilen sind die Ergebnisse für die drei Einzelentscheidungen Phonem, Gruppierung und Silbengrenze aufgeführt.

Es ist deutlich zu sehen, welchen Einfluß die Folgefehler auf das Gesamtergebnis haben. Bei der Variante ohne Gruppierung ohne Weight Decay sind nur 0,8 Prozent mehr Muster richtig, was aber zu 3,3 Prozent mehr richtigen Wörtern führt. Analog dazu liegt der Prozentsatz richtig behandelter Wörter bei der Variante mit Weight Decay 1,7 Prozent über der ohne Weight Decay, obwohl der Unterschied bei den Mustern nur 0,5 Prozent beträgt.

Die Netze ohne separaten Gruppierungsknoten machen deutlich weniger Fehler bei den Phonemen, auch die Ergebnisse bei den Silbengrenzen sind besser. Bei den Gruppierungen gibt es kaum Unterschiede, wobei es für alle drei Netze eine einfach zu lernende Information zu sein scheint. Die Unterschiede zwischen den beiden Varianten ohne Gruppierung sind in den einzelnen Kategorien nicht so gravierend, aber insgesamt ist die mit Weight Decay um fast zwei Prozent besser.

4.2.1.2 Trainingsmuster und Lexikon

Im vorigen Abschnitt wurden die Ergebnisse auf Lexikonebene gegenübergestellt. Hier soll nun genauer auf die Fehler bei den einzelnen Mustern eingegangen werden.

Aus dem Lexikon mit ca. 310000 Einträgen werden insgesamt 714239 verschiedene Muster erzeugt. Diese werden im Verhältnis 70:30 in Trainings- und Testmenge eingeteilt. Damit ist

eine ausreichend große Testmenge garantiert. In anderen Arbeiten wird oft das Lexikon im Verhältnis 70:30 geteilt und dann die Trainings- bzw. Testmuster aus den separaten Lexika erzeugt. Dadurch kommt es aber dazu, daß Muster der Testmenge auch in der Trainingsmenge enthalten sind und umgekehrt. Ein Versuch mit dieser Einteilung hat gezeigt, daß 92 Prozent der Testmuster auch in der Trainingsmenge enthalten waren. Die auf diese Weise erhaltene Testmenge ist deutlich zu klein und eine Überanpassung des Netzwerkes an die Trainingsmuster vorprogrammiert. Das Lexikon wird somit auswendig gelernt, was sicherlich während der Tests gute Ergebnisse liefert. Für eine Behandlung von Wörtern, die nicht im Lexikon enthalten sind, ist ein solches Netz aber kaum geeignet.

Der Verlauf der richtig behandelten Muster während des Trainings ist in Abbildung 4.1 zu sehen. Während die Ergebnisse auf der Testmenge schnell die Sättigung erreichen, verbessern sie sich auf der Trainingsmenge auch noch bei höheren Iterationen. Das beste Ergebnis auf der Trainingsmenge ist 99,69 Prozent bei Iteration 1059, auf der Testmenge 98,96 Prozent bei Iteration 802.

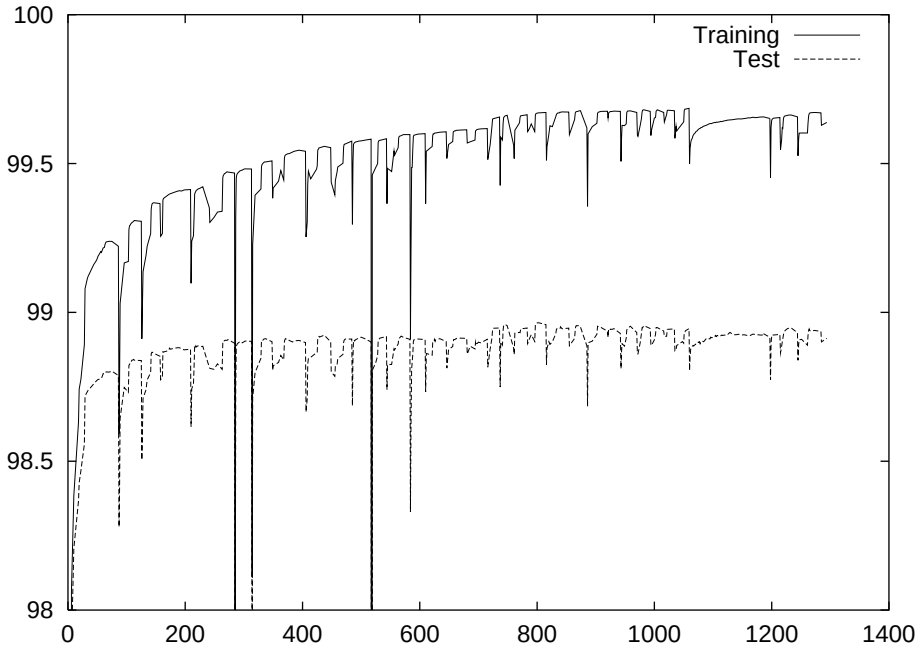

Abbildung 4.1: Verlauf der richtig errechneten Muster während des Trainings in Prozent.

Das beste Ergebnis auf dem gesamten Lexikon wird bei Iteration 1023 erreicht. Hier liegen die Ergebnisse von Trainings- und Testmenge bei 99,67 Prozent bzw. 98,96 Prozent. Bei einer durchschnittlichen Phonemelänge von 9,63 Phonemen pro Wort und der Annahme, daß alle Fehler voneinander unabhängig und gleichverteilt sind, kann man den Prozentsatz der richtig transkribierten Wörter abschätzen mit $(0,9968 * 0,7 + 0,9894 * 0,3)^{9,63} = 0,949$. Demzufolge müßten ca. 95 Prozent der Wörter korrekt transkribiert werden. Da die Fehler aber nicht gleichverteilt sind und auch Folgefehler nach sich ziehen können, ist im Gegensatz zur Abschätzung

bei 94,28 Prozent der Wörter die Transkription fehlerfrei, was verglichen mit anderen Arbeiten ein sehr gutes Ergebnis ist. Dabei ist hier zu berücksichtigen, daß die Ausgabe aus den Phonemen und den Silbengrenzen besteht. Bei mehr als 290000 Wörtern sind also alle Phoneme und alle Silbengrenzen korrekt. Aus dieser Berechnung ersieht man aber auch, warum es problematisch ist, bessere Ergebnisse zu erzielen. Um z. B. 99 Prozent der Wörter richtig transkribieren zu können, wäre eine Transkriptionsrate von $\sqrt[9.63]{0,99} = 0,999$, also 99,9 Prozent nötig, und das sowohl bei der Trainings- als auch bei der Testmenge. Hier sind dem Verfahren also Grenzen gesetzt.

Die während der Transkription eines Wortes gemachten Fehler kann man noch weiter unterteilen in die Fehler bei Gruppierung, Phonem und Silbengrenze. Bei der Transkription der 310000 Wörter des Lexikons sind ca. 3 Millionen Entscheidungen des Netzwerkes auszuführen. Bei diesen Entscheidungen sind 99,7 Prozent der Silbengrenzen, 99,2 Prozent der Gruppierungswerte und 98,8 Prozent der Phoneme richtig. Diese drei Entscheidungen werden pro Muster getroffen. Insgesamt werden 98,5 Prozent der Muster richtig behandelt.

Diese Ergebnisse haben alle relativ große Werte. Allerdings reicht ein einziger Teilfehler aus, um einen Fehler auf Wortebene zu erzeugen. Bei durchschnittlich mehr als 9 Entscheidungen pro Wort ergibt das eine Fehlerrate von fast 6 Prozent. Dazu kommt, daß eine falsche Gruppierung weitere Fehler nach sich zieht, da im weiteren Verlauf für das Wort eine falsche Grapheme-ingabe erfolgt.

4.2.1.3 Phonemverwechslungen

Die bisher behandelten Zahlen geben nur an, wieviele Fehler insgesamt bei der Transkription gemacht werden, ohne diese Fehler genauer einzuordnen. Eine falsch gesetzte Silbengrenze ist sicherlich nicht so schlimm wie ein falsch ausgewähltes Phonem, und auch bei den Phonemverwechslungen gibt es Unterschiede. Die Verwechslung der Phoneme /a:/ und /a/ ist geringer zu bewerten als die Verwechslung von /a:/ und /u:/. Um einen Überblick zu erhalten, welche Fehler bei den Phonemen gemacht werden, erfolgt nun eine Untersuchung der Phonemverwechslungen.

Die Tabellen 4.4 und 4.5 zeigen die Phonemverwechslungen bei den Vokalen für Deutsch, summiert für Trainings- und Testmuster. Besonders häufig werden die Phoneme /a:/, /a/, /e:/, /E/, /@/, /o:/ und /O/ verwecheselt, wobei die meisten Verwechslungen zwischen Kurz- und Langvokal erfolgen. Bei diesen ablsoluten Zahlen muß man aber beachten, daß die Muster für die einzelnen Phoneme unterschiedlich oft auftreten. Diese Angaben sind in Tabelle A.1 zu finden.

Die häufigsten Verwechslungen bei den Konsonanten (siehe Tabelle 4.6) treten zwischen /n/ und /N/, /t/ und /ts/ sowie /s/ und /S/ auf. Auch /g/ und /Z/ werden häufig verwechselt, wobei fast immer (in 69 von 70 Fällen) statt eines /Z/ ein /g/ ausgegeben wird. Das /Z/ wurde überhaupt schlecht gelernt. Nur 52 Prozent der Muster, bei denen dieses Phonem ausgegeben werden soll, werden richtig behandelt. Es tritt im Deutschen sehr selten auf. Es gibt 147 Muster

	aI	aU	OY	a:	a	E:	E	e:	6	@	o:	∑
aI	.	.	.	4	4	.	.	.	.	1	.	9
aU	.	.	.	1	.	.	.	.	.	.	1	2
OY	.	.	.	.	.	.	.	.	.	.	14	14
a:	2	.	.	.	387	1	1	.	.	.	.	391
a	9	.	.	305	.	1	.	.	.	.	.	315
E:	16	1	5	8	11	.	45	.	.	.	.	86
E	.	.	2	.	1	56	.	99	71	209	.	438
e:	45	.	14	.	.	.	102	.	15	244	.	420
6	.	.	.	.	.	.	31	16	.	28	.	75
@	6	.	.	.	.	.	73	109	17	.	.	205
o:	.	3	.	.	.	.	.	1	.	.	.	4
∑	78	4	21	318	403	58	252	225	103	482	15	

Tabelle 4.4: Phonemverwechslungen für Deutsch, Teil 1. Die Zeilen geben an, wie oft an Stelle des Phonems in der ersten Spalte vom neuronalen Netz ein anderes Phonem ausgegeben wurde.

	i:	I	o:	O	2:	9	y:	Y	OY	∑
i:	.	127	.	.	.	.	27	3	.	157
I	113	.	.	.	.	.	.	.	.	113
o:	.	.	.	81	.	.	.	.	.	81
O	.	.	162	.	.	.	.	.	.	162
2:	.	.	.	.	.	10	.	.	4	14
9	.	.	.	.	14	.	.	.	.	14
y:	.	.	.	.	.	.	.	45	.	45
Y	.	.	.	.	.	.	22	.	.	22
OY	.	.	14	6	2	.	.	.	.	22
∑	113	127	176	87	16	10	49	48	4	

Tabelle 4.5: Phonemverwechslungen für Deutsch, Teil 2.

	f	v	p	b	n	N	t	ts	S	s	Z	g	k	∑
f	.	19	1	.	.	.	.	.	.	.	.	.	.	20
v	18	.	.	.	.	.	.	.	.	.	.	.	.	18
p	4	.	.	23	.	.	.	.	.	.	.	.	.	27
b	.	.	15	.	.	.	.	.	.	.	.	.	.	15
n	.	.	.	.	.	36	1	.	.	.	.	.	.	37
N	.	.	.	.	73	.	1	.	.	.	.	.	.	74
t	.	.	.	.	.	.	.	39	.	.	.	.	.	39
ts	.	6	2	.	.	.	37	.	.	.	1	.	1	47
S	.	.	.	.	.	.	.	.	.	31	.	.	1	32
s	.	1	2	1	.	.	.	11	86	.	1	.	.	102
Z	.	.	.	.	.	.	.	.	.	.	.	69	.	69
g	.	1	.	.	.	.	.	.	.	.	7	.	10	18
k	.	4	2	.	.	.	.	2	.	2	5	22	.	37
∑	22	31	22	24	73	36	39	52	86	33	14	91	12	

Tabelle 4.6: Phonemverwechslungen für Deutsch, Teil 3, die häufigsten Verwechslungen bei den Konsonanten.

für /Z/ und 21102 Muster für /g/. Von diesen 147 Mustern sind 94 in der Trainingsmenge enthalten. Das entspricht weniger als 0,02 Prozent der Trainingsmuster.

4.2.2 Fehleranalyse für Englisch und Holländisch

Tabelle 4.7 zeigt die Ergebnisse für Englisch auf einem Lexikon mit ca. 61000 Wörtern. Die Werte sind deutlich schlechter als für Deutsch, liegen aber im Bereich anderer Veröffentlichungen (siehe Abschnitt 1.5.3). Ein Grund dafür ist die Vielfalt der möglichen Zuordnungen zwischen den Graphemen und Phonemen. So gibt es z. B. im Englischen 14 Graphemkombinationen, die zum Phonem /i:/ konvertiert werden können, im Deutschen dagegen nur 7, und während die 7 deutschen Graphemkombinationen noch 6 weitere Phoneme generieren können, sind des bei den 14 im Englischen 25 weitere Phoneme. Im Englischen gibt es also viele mehrdeutige Graphem-Phonem-Zuordnungen, was das Training erschwert.

Tabelle 4.8 zeigt die Ergebnisse für Holländisch auf einem Lexikon mit ca. 310000 Wörtern. Die Werte sind nicht ganz so gut wie für Deutsch, aber deutlich besser als für Englisch. Ein Grund dafür könnte sein, daß es im Gegensatz zum Deutschen überhaupt keine Korrekturen an dem holländischen Lexikon gegeben hat. Während im deutschen Lexikon mehrere Tausend Einträge korrigiert wurden, ist das holländische Lexikon unverändert verwendet worden.

	mit Gruppierung	ohne Gruppierung	
		ohne WD	mit WD
Wörter	64,9 %	71,7 %	74,0 %
Muster	90,0 %	91,4 %	92,2 %
Phoneme	91,5 %	92,5 %	93,1 %
Gruppierungen	97,8 %	97,5 %	97,8 %
Silbengrenzen	97,7 %	97,8 %	98,2 %

Tabelle 4.7: Gegenüberstellung der Ergebnisse für die drei Netzwerkvarianten für Englisch.

	mit Gruppierung	ohne Gruppierung	
		ohne WD	mit WD
Wörter	87,3 %	88,7 %	89,6 %
Muster	96,8 %	97,2 %	97,4 %
Phoneme	97,7 %	98,1 %	98,0 %
Gruppierungen	99,5 %	99,5 %	99,4 %
Silbengrenzen	98,5 %	98,6 %	98,9 %

Tabelle 4.8: Gegenüberstellung der Ergebnisse für die drei Netzwerkvarianten für Holländisch.

4.3 Wortbetonung

4.3.1 Silbenbasierter Ansatz

Tabelle 4.9 zeigt die Ergebnisse für die drei Sprachen. Es wurden in allen drei Sprachen jeweils die ersten fünf Silben betrachtet.

	Deutsch	Englisch	Holländisch
Wörter	311360	61223	309668
eine Silbe	5250	7985	6125
Betonung außerhalb	1100	21	1316
Pattern	305010	53217	302227
richtig	276031 (90,5 %)	47063 (88,4 %)	256608 (84,9 %)
falsch	28979 (9,5 %)	6154 (11,6 %)	45619 (15,1 %)
gesamt	90,2 %	88,4 %	84,5 %

Tabelle 4.9: Tabelle der Ergebnisse für die Wortbetonung.

In der ersten Zeile ist die Anzahl aller Wörter im Lexikon angegeben. Die zweite Zeile enthält die Anzahl der Wörter, die nur eine Silbe haben. Für diese Wörter muß keine Betonungsentscheidung getroffen werden. Die dritte Zeile enthält die Anzahl der Wörter, bei denen die Betonung hinter der fünften Silbe liegt. Für diese Wörter ist mit den Netzen also keine richtige

Entscheidung möglich. Die vierte Zeile gibt die Anzahl der Pattern an, die durch das Netz bearbeitet werden müssen. Dabei fallen während des Tests die Wörter heraus, deren Betonung nicht innerhalb der ersten fünf Silben liegt. Die beiden folgenden Zeilen geben jeweils die Anzahl der Pattern an, bei denen sich das Netz richtig oder falsch entschieden hat. In der letzten Zeile ist angegeben, wieviel Prozent aller zu treffenden Betonungsentscheidungen richtig waren. In diesem Verhältnis sind also auch die Pattern enthalten, die das Netz nicht richtig entscheiden konnte, da die richtige Betonung hinter der fünften Silbe liegt.

4.3.2 Phonembasierter Ansatz

Mit dieser Architektur wurden bessere Ergebnisse erzielt als mit dem silbenbasierten Ansatz. Bietet man dem neuronalen Netz also die komplette Phonemfolge an, ist es für alle drei Sprachen besser in der Lage, die Wortbetonung zu lernen.

Für diesen Ansatz wurden ausführliche Experimente durchgeführt. Zum einen wurde der Einfluß des Gewichtsprunings untersucht, zum anderen wurde überprüft, inwieweit sich die Ergebnisse durch die Anwendung des Weight Decay verbessern.

4.3.2.1 Einfluß des Ausdünnens der Gewichte

Auf die Gewichte zwischen der Eingabeschicht (bzw. der Zwischeneingabe bei Verwendung des Weight Decay) und der Ausgabeschicht wurde ein Gewichtspruning angewendet. Wie schon in Abschnitt 2.1.5.2 beschrieben wurde, kann dieses Verfahren zum Löschen der Gewichte der Klasse 2 benutzt werden, also der Gewichte, die nur vermeintlich etwas Sinnvolles gelernt haben. Dadurch können sich die Ergebnisse verbessern.

Dies ist bei der vorliegenden Architektur zu beobachten. Ohne Gewichtspruning werden bei 75 Knoten in der versteckten Schicht 95,1 Prozent der englischen Wörter richtig betont. Das Netz hat in diesem Fall 40500 Gewichte für 35859 Trainingsmuster, ist also in der Lage, auswendig zu lernen. Bei der Verwendung von 200 versteckten Knoten erhöht sich die Anzahl der freien Parameter auf 108000, was mehr als dem Dreifachen der Trainingsmuster entspricht.

Während des Gewichtsprunings werden nun die Gewichte der Klasse 2 Schritt für Schritt gelöscht. Die Anpassung an das Rauschen in den Daten wird dadurch immer schwieriger, und je weniger Parameter zur Verfügung stehen, desto mehr muß der Algorithmus die noch vorhandenen benutzen, um sich auf die wichtige Information zu konzentrieren. Die Abhängigkeit der erzielten Ergebnisse von der verbliebenen Anzahl der Gewichte ist in Abbildung 4.2 zu sehen. Das verwendete Netz hat 200 versteckte Knoten, die Pruningrate betrug 2 Prozent[1].

[1] In den folgenden Abbildungen und Tabellen treten die folgenden Angaben auf: Die Anzahl der versteckten Knoten wird hinter einem `h` angegeben, die Pruningrate hinter einem `p`, und die Verwendung des Weight Decay wird mit `wd` markiert. `h200 p5 wd` bedeutet also, daß ein Netz mit 200 versteckten Knoten, eine Pruningrate von 5 Prozent und Weight Decay verwendet wurden.

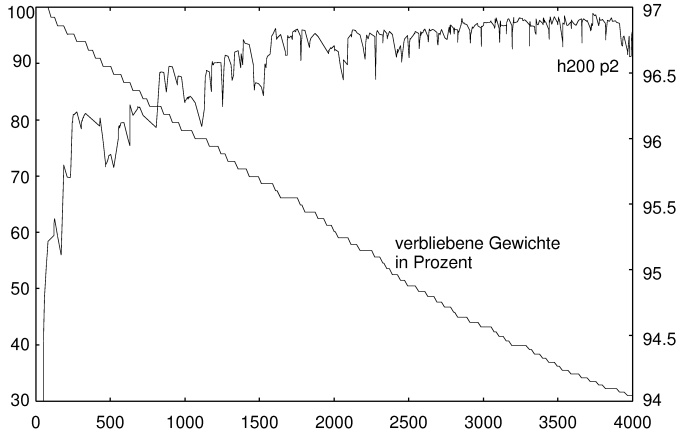

Abbildung 4.2: Abhängigkeit der erzielten Ergebnisse von der Anzahl der verbliebenen Gewichte beim Gewichtspruning (am Beispiel `h200 p2` für Englisch). Die linke y-Achse zeigt die verbliebenen Gewichte in Prozent an (für die von links oben nach rechts unten fallende Linie), die rechte y-Achse die korrekt betonten Wörter (ebenfalls in Prozent). Auf der x-Achse sind die Iterationen des Trainings aufgetragen.

In dem Bild ist gut zu sehen, daß sich die erzielten Ergebnisse mit sinkender Anzahl der Gewichte verbessern. Das Netz hat nun immer weniger freie Parameter zur Lösung der Aufgabe zur Verfügung. Durch das Löschen der unnötigen Verbindungen kristallisiert sich eine Struktur innerhalb des Netzwerkes aus. Nur die tatsächlich benötigten Gewichte überleben. Es wird immer schwerer für die Verbindungen, das Rauschen in den Daten zu repräsentieren. Durch diesen Konzentrationsprozeß wird die Überanpassung an die Trainingsdaten, also das Auswendiglernen, stark unterdrückt. Erst wenn die Anzahl der Gewichte zu stark reduziert wird, ist das Netz nicht mehr in der Lage, die Aufgabe zu lösen. In diesem Beispiel liegt diese Grenze bei ca. 30 Prozent der Gewichte.

Ungefähr ein Drittel der Gewichte sind also in dem Netz mit dem besten Ergebnis noch vorhanden. Nun stellt sich die Frage, ob man nicht einfach mit zu vielen Gewichten angefangen hat zu trainieren. Vielleicht liefert ja ein kleineres Netz mit wenigen Gewichten ebenso gute Ergebnisse. Außerdem kann man das Training vielleicht beschleunigen, indem man mehr Gewichte pro Ausdünnschritt löscht. Dazu wurden weitere Versuche durchgeführt. Zwei verschieden große versteckte Schichten mit 75 bzw. 200 Knoten und zwei Pruningraten von 2 bzw. 5 Prozent wurden ausprobiert. Die Trainingsergebnisse sind in Tabelle 4.10 und Abbildung 4.3 zu sehen.

Das Netz mit 75 versteckten Knoten lernt am Anfang deutlich schneller als das mit 200 Knoten, unabhängig von der Pruningrate. Im Endergebnis ist das mit 200 Knoten aber deutlich besser. Zu beachten ist, daß das Netz mit 75 Knoten am Anfang mehr freie Parameter hat (40500) als das Netz, das bei 200 Knoten das beste Ergebnis erzielt (36167). Trotzdem ist das Ergenis des Netzes mit 75 Knoten auch bei 2 Prozent Pruningrate bei weitem nicht so gut. Es ist also nicht

NN	Pruning-rate	korrekte Wörter (%)	in Iteration	aktive Gewichte gesamt	aktive Gewichte in %
h75	2	96,42	2353	17671	43,6
h75	5	96,50	844	23588	58,2
h200	2	96,95	3732	36167	33,5
h200	5	96,83	1657	14405	13,3

Tabelle 4.10: Die besten Ergebnisse für Englisch mit 75 bzw. 200 versteckten Knoten und 2 bzw. 5 Prozent Pruningrate.

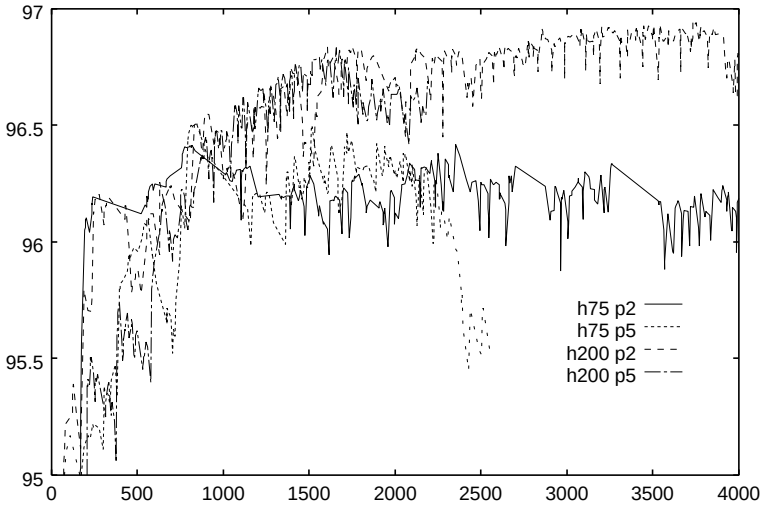

Abbildung 4.3: Korrekt betonte Wörter (in Prozent) für Englisch mit 75 bzw. 200 versteckten Knoten und 2 bzw. 5 Prozent Pruningrate. Die beiden unteren Linien zeigen die Ergebnisse für 75 versteckte Knoten, die beiden oberen für 200 versteckte Knoten.

die Anzahl der Gewichte zu Beginn des Trainings entscheidend für das erzielbare Ergebnis, sondern der Weg, wie man zur optimalen Anzahl der freien Parameter kommt. Dazu bietet es sich an, mit einer relativ großen Anzahl von Gewichten zu beginnen und diese dann langsam auszudünnen. Weiterhin ist zu beobachten, daß eine zu große Pruningrate zu einem zu abrupten Ausdünnen führt. In die Netzwerkstruktur wird zu stark eingegriffen. Deswegen sind die erzielten Ergebnisse bei 5 Prozent Pruningrate nicht so gut wie die bei 2 Prozent.

4.3.2.2 Ergebnisse mit Weight Decay

Um die Ergebnisse noch weiter zu verbessern und das Training zu beschleunigen, wurde das in Abschnitt 2.1.7 vorgestellte Weight Decay implementiert. Die verwendete Architektur ist in Abbildung 3.17 (Seite 74) dargestellt. Weight Decay wird nur auf die Verbindung zwischen der Eingabe und der Zwischenschicht (`wd_input`) angewendet. Die erzielten Ergebnisse sind in Tabelle 4.11 und Abbildung 4.4 denen ohne Weight Decay gegenübergestellt.

Netzwerk	Prozent korrekt	nach Iteration
h75	96,42	2353
h75 wd	97,20	1339
h200	96,95	3732
h200 wd	97,17	1169

Tabelle 4.11: Vergleich der Ergebnisse ohne und mit Weight Decay für Englisch.

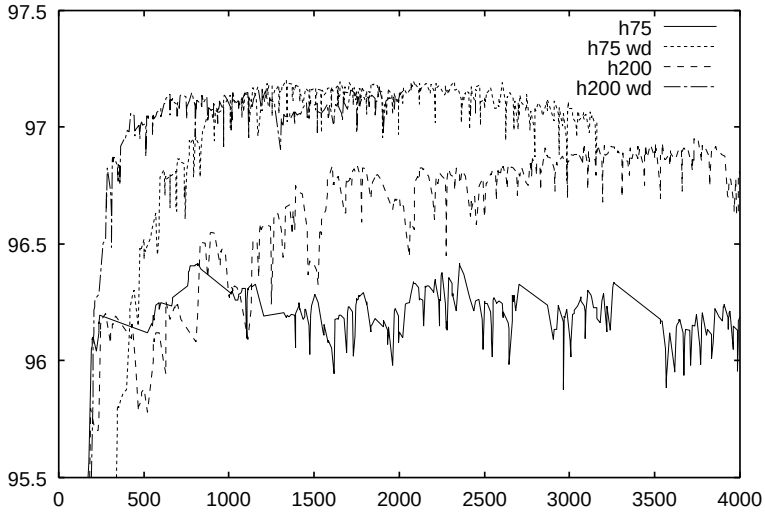

Abbildung 4.4: Richtig betonte Wörter (in Prozent) mit und ohne Weight Decay für Englisch. Die beiden unteren Linien zeigen die Ergebnisse ohne Weight Decay (vgl. Abbildung 4.3), die beiden oberen mit Weight Decay. Die Pruningrate ist in beiden Fällen 2 Prozent.

Es ist gut zu sehen, daß hier beide Vorteile von Weight Decay voll zur Geltung kommen. Zum einen ist das erzielte Ergebnis besser (z. B. bei 75 versteckten Knoten 97,2 statt 96,4 Prozent), und zum anderen werden deutlich weniger Iterationen zum Erreichen des Ergebnisses benötigt. Das zeigt sich auch für Deutsch und Holländisch (siehe Tabelle 4.12 und Abbildung 4.5).

4.3.2.3 Interpretation des Weight Decay als Eingabepruning

Durch die Diagonalmatrix zwischen der Eingabe und der Zwischenschicht wird erreicht, daß jeder Eingabeknoten nur über ein einziges Gewicht mit dem Rest des Netzes verbunden ist. Die Information dieses Knotens fließt also nur über diese eine Verbindung in die Netzwerkentscheidung mit ein. Wird diese Verbindung gekappt, indem das Gewicht dieser Verbindung auf einen Wert nahe 0 gesetzt wird, so spielt der Knoten keine Rolle mehr – er kann als gelöscht betrachtet werden. In den Tabellen B.7, B.8 und B.9 sind die Knoten mit ihren Kontexten angegeben, deren Gewicht der Diagonalmatrix im Bereicht $[0;0,01]$ liegt. Sie sind mit einem *x* gekenn-

Sprache	Netzwerk	Prozent korrekt	nach Iteration
Deutsch	h100	96,95	3697
	h100 wd	97,39	1582
Holländisch	h200	93,47	1529
	h200 wd	95,08	1471

Tabelle 4.12: Vergleich der Ergebnisse ohne und mit Weight Decay für Deutsch und Holländisch.

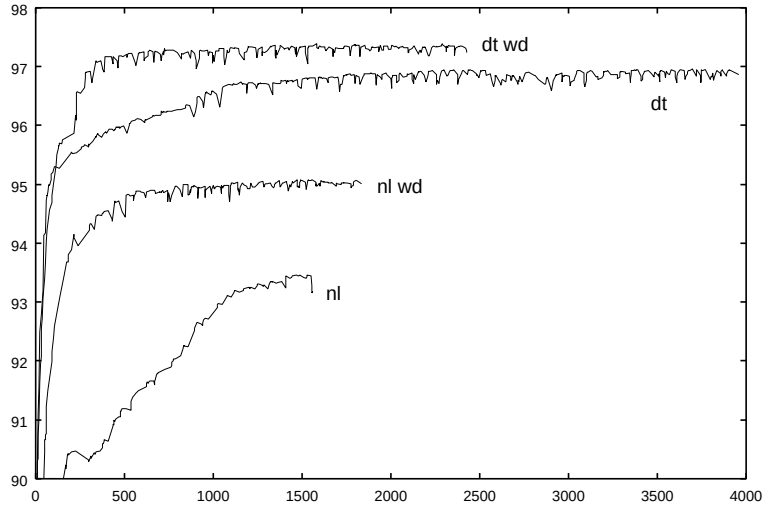

Abbildung 4.5: Richtig betonte Wörter (in Prozent) mit und ohne Weight Decay für Deutsch und Holländisch. Die beiden unteren Linien zeigen die Ergebnisse für Holländisch, die beiden oberen für Deutsch. Die Pruningrate beträgt wiederum 2 Prozent.

zeichnet. Der Wert dieser Knoten wird also mit einem sehr kleinen Gewicht multipliziert und damit ausgeblendet.

Es gibt drei Varianten dieses Ausblendens. Ein Teil der Phoneme wird nur am Wortanfang gelöscht, ein zweiter Teil nur am Wortende und ein dritter im gesamten Kontext. Bei einigen Phonemen wird jegliche Verbindung gekappt. Dies sind z. B. im Deutschen `/Z/`, im Englischen `/D/` und im Holländischen `/dZ/`. Diese Phoneme scheinen also für die Wortbetonung keine Rolle zu spielen, oder anders gesagt, diese Netzwerkarchitektur ist mit den vorhandenen Trainingsmustern nicht in der Lage, eine Information zur Entscheidung über die Wortbetonung aus diesen Phonemen abzuleiten. Ein Grund dafür könnte sein, daß diese Phoneme relativ selten auftreten.

Insgesamt werden 26 bzw. 28 Prozent der Eingabeknoten ausgeblendet (siehe Tabelle 4.13). Ungefähr ein Viertel der Eingabeinformation hat also keinen positiven Einfluß auf die Ergebnisse. Der Einfluß kann im Gegenteil sogar negativ sein, wie man an den schlechteren Ergebnissen

der Architektur ohne Weight Decay sieht. Durch das Ausblenden der uninteressanten bzw. sogar störenden Eingabeknoten ist das Netz also schneller in der Lage, eine Struktur aus den Daten zu lernen, die dann sogar noch bessere Ergebnisse liefert.

	Verbindungen	gelöscht	in Prozent
Englisch	517	145	28
Deutsch	423	110	26
Holländisch	440	118	27

Tabelle 4.13: Anzahl der Verbindungen auf der Diagonalmatrix im Bereich $[0;0,01]$.

Der Verlauf der Weight-Decay-Gewichte während des Trainings ist am Beispiel des ersten Phonemknotens für die englische Wortbetonung in Abbildung 4.6 dargestellt. Die Gewichte werden zu Beginn mit 1 initialisiert. Schon nach einigen Hundert Iterationen kristallisiert sich heraus, ob sie gegen 0 gedrückt werden oder nicht. Gegen Ende des Trainings gibt nur noch geringe Änderungen, die z. B. dadurch zustande kommen können, daß nach einem Pruningschritt ein anderer Knoten weniger Bedeutung hat und das Netz nun wieder auf die Information dieses Knotens angewiesen ist.

4.3.2.4 Training ohne die „gelöschten“ Phoneme

Eine Idee, das Netzwerk weiter zu verkleinern, könnte nun sein, diejenigen Phoneme, die durch das Weight Decay ausgeblendet wurden, in einem neuen Training gar nicht erst anzubieten. Damit spart man sich laut Tabelle 4.13 ungefähr ein Viertel der Eingabeknoten, hat somit auch weniger Trainingsmuster und braucht weniger Gewichte innerhalb des Netzes. Realisiert wurde dieser Versuch, indem die Gewichte auf der Diagonalmatrix, die bei dem Netz mit den besten Ergebnissen kleiner als 0,01 waren, ganz auf 0 gesetzt wurden. Dies bedeutet im SENN, daß das Gewicht gelöscht und somit unveränderbar ist. Das Training sollte nun schneller laufen und zumindest ungefähr die gleichen Ergebnisse liefern.

Aber das Gegenteil ist der Fall. Die Ergebnisse sind deutlich schlechter als bei der Verwendung aller Phoneme. Das ist darin begründet, daß es ein erheblicher Unterschied ist, ob das Netz eine gegebene Information für unwichtig erachtet, oder ob es diese Information gar nicht erst zur Verfügung gestellt bekommt. Der deduktive Weg, also vom Allgemeinen zum Besonderen, ist hier wiederum besser als der induktive, bei dem von wenig Information auf das ganze Wissen geschlossen werden soll.

4.4 Verwendung zusätzlicher Wissensquellen

Die Ergebnisse zeigen, daß durch die Verwendung zusätzlicher Wissensquellen deutliche Verbesserungen zu erreichen sind. Nur mit Hilfe des neuronalen Netzes mit einem Kontext von

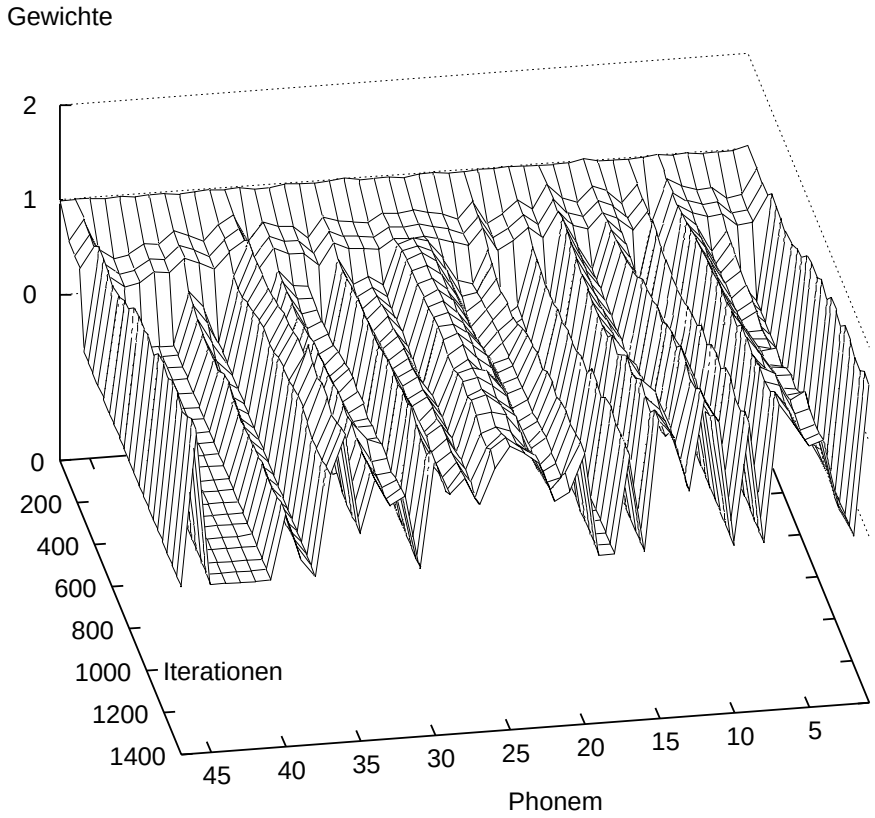

Abbildung 4.6: Verlauf der ersten 47 Gewichte der Diagonalmatrix (erstes Phonem) für die englische Wortbetonung. Die y-Achse stellt die Trainingsiterationen dar, auf der z-Achse sind die Werte der Gewichte aufgetragen.

9 Graphemen konnten 91,2 Prozent der Einträge des deutschen Celex inklusive Silbengrenze und Wortbetonung richtig transkribiert werden, die mindestens 8 Buchstaben lang waren. Bei einem Kontext von 7 Graphemen werden 85,1 Prozent der Wörter richtig transkribiert. Durch die Kombination von Lexikon und neuronalem Netz sowie durch die Auswertung der Graphem-Phonem-Zuordnungen können diese Ergebnisse weiter verbessert werden.

4.4.1 Kombination von Lexikon und neuronalen Netzen

Deutsch

Tabelle 4.14 enthält die Ergebnisse, die für das deutsche CELEX mit den 272.000 Wörtern erzielt wurden, die mindestens 8 Buchstaben lang sind[2], in Abhängigkeit von der Mindestlänge, die ein Eintrag aus dem Lexikon haben mußte, um für die Zerlegung verwendet zu werden, und

[2]Die Ergebnisse für das gesamte Lexikon sind in Tabelle 4.3 aufgeführt.

in Abhängigkeit von der Anzahl der Grapheme, die am Wortanfang übersprungen wurden, wenn der Wortanfang nicht im Lexikon gefunden wurde. Diesen Ergebnissen sind die der neuronalen Netze gegenübergestellt. Aus Gründen der Übersichtlichkeit ist hier nur ein Auszug dargestellt. Die vollständige Tabelle ist in Anhang A.2 zu finden.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		91,2	92,3	94,1	95,3
Länge	Lücke				
3	2	86,0	87,5	89,0	90,7
4	2	91,0	92,2	93,2	94,6
5	2	94,1	95,5	96,0	97,3
6	5	95,0	96,5	96,5	98,0
7	5	**95,3**	**96,6**	**96,7**	**98,1**
8	5	95,2	96,5	96,6	97,9

Tabelle 4.14: Ergebnisse für Deutsch in Abhängigkeit von der Bestandteillänge (vollständige Angaben in Tabelle A.2).

In der ersten Zeile sind die Ergebnisse der neuronalen Netze aufgeführt. In der zweiten Spalte ist zu sehen, daß mit Hilfe der beiden Netze 91,2 Prozent der CELEX-Einträge mit mindestens 8 Graphemen richtig reproduziert werden können. Bei diesen Einträgen sind also alle Phoneme und Silbengrenzen richtig, und auch die Betonung wurde richtig gesetzt. Ohne Beachtung der Silbengrenzen waren 92,3 Prozent der Wörter richtig, ohne Beachtung der Betonung 94,1 Prozent. Bei 95,3 Prozent der Einträge waren zumindest alle Phoneme richtig.

In der Tabelle ist auch gut zu sehen, daß sich die Ergebnisse anfangs mit steigender Mindestlänge verbessern. Das liegt zum einen daran, daß kurze Einträge seltener die „richtige“ Transkription repräsentieren. Außerdem gibt es bei einer geringen Mindestlänge viel mehr Möglichkeiten, von denen die ausgewählte nicht immer die richtige ist.

Bei einer Mindestlänge von 7 Buchstaben und einer Lücke von 5 Buchstaben am Wortanfang werden die besten Ergebnisse erzielt. Dies bedeutet also, daß nur die Wörter des Lexikons für die Zerlegung in Teilwörter verwendet werden, die mindestens 7 Buchstaben lang sind. Wird der Wortanfang nicht im Lexikon gefunden, so werden gleich die ersten 5 Buchstaben übersprungen und erst ab dem 6. Buchstaben weiter nach Lexikoneinträgen gesucht.

Das Ergebnis auf Wortebene ist bei 7/5 um 4 Prozent gegenüber den neuronalen Netzen gestiegen. Außerdem sind bei 98 Prozent der Einträge alle Phoneme richtig. Mit weiter steigender Mindestlänge verschlechtern sich die Ergebnisse wieder. Das liegt daran, daß nun immer weniger Einträge im Lexikon gefunden werden und die Netze mehr und mehr zur Bestimmung der Transkription herangezogen werden müssen, wobei die zu füllenden Lücken auch immer länger werden.

Ein Teil der Betonungsfehler kommt dadurch zustande, daß die Betonung aus dem Lexikon übernommen wird, wenn wenigstens für den Wortanfang ein Eintrag gefunden wurde. Das wirkt

sich dann als Fehler aus, wenn sich die Betonung durch das Anhängen eines Suffixes verschiebt. Ein Beispiel dafür ist *Musik*, bei dem die zweite Silbe betont ist, und *Musiker*, bei dem die Betonung auf der ersten Silbe liegt.

Diese Auswertung wurde bereits während des Trainings der Netze stichprobenweise durchgeführt. Dabei war zu beobachten, daß sich bei immer besser werdenden Netzen die optimale Mindestlänge der Lexikoneinträge ständig erhöht hat. Je weniger Fehler die Netze also machen, desto besser sind sie in der Lage, die Lücken zwischen den Lexikonbestandteilen aufzufüllen. Das erlaubt es, immer längere Bestandteile zu verwenden, was das Risiko verringert, daß die verwendete Phonemfolge nicht die tatsächliche Aussprache repräsentiert.

Weiterhin fällt zumindest bei größeren Mindestlängen auf, daß das beste Ergebnis bei einer Anfangslücke von 5 Buchstaben erzielt wird. Das ist gerade der Kontext, der dem neuronalen Netz am Wortanfang zur Verfügung steht: das Graphem im Zentrum plus die nächsten 4 Grapheme als rechter Kontext. Ein ähnliches, wenn auch nicht so deutliches Bild zeigt sich bei dem Netz mit einem Kontext von 3 linken und rechten Graphemen (siehe Tabelle A.3). Hier wird ebenfalls bei den größeren Kontexten das beste Ergebnis bei einer Anfangslücke von 4 erreicht, was wiederum dem Kontext am Anfang des Wortes entspricht. Allerdings bewegen sich bei diesem Netz die Unterschiede im Promillebereich.

Vergleicht man die absoluten Ergebnisse, die unter Verwendung der beiden Netze erzielt werden, so sieht man die Stärke dieses Verfahrens vor allem für schlechtere Netze. Bei dem Netz mit 7 Graphemen am Eingang werden 85,1 Prozent der Wörter richtig transkribiert, bei dem Netz mit 9 Graphemen 91,2 Prozent, also ein Plus von über 6 Prozent. Bei dem hybriden Ansatz fallen die Unterschiede mit 94,1 bzw. 95,3 Prozent nicht mehr so deutlich aus. Dadurch, daß man die korrekte Transkription aus dem Lexikon verwenden kann und nur evtl. auftretende Lücken füllen muß, wird der Einfluß durch das neuronale Netz stark abgeschwächt. So ist man also auch mit weniger guten Netzen in der Lage, akzeptable Ergenisse zu erreichen.

Englisch und Holländisch

Für diese beiden Sprachen sind die Ergebnisse nicht so gut wie für Deutsch. Englisch ist eine Sprache, bei der es sehr viele Unregelmäßigkeiten gibt, und oft ist es nicht möglich, vom Graphemkontext auf den richtigen Laut zu schließen. Der große Unterschied zu Deutsch ergibt sich aber auch aus der Tatsache, daß in das Training der Netze für Deutsch ein viel größerer Aufwand gesteckt wurde. Während das Training für Englisch und Holländisch wirklich vollautomatisch ohne jeglichen manuellen Eingriff erfolgte, wurden z. B. im deutschen CELEX mehrere Transkriptionsfehler korrigiert und Fremdwörter entfernt.

Durch den hybriden Ansatz können die Ergebnisse für Englisch um 10,7 Prozent und die für Holländisch um 3,8 Prozent verbessert werden. Zu beachten ist, daß diese Ergebnisse wieder für die Einträge im Lexikon gelten, die mindestens 8 Buchstaben lang sind.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		67,0	67,5	68,7	69,3
Länge	Lücke				
3	2	58,5	60,1	61,4	63,4
4	1	65,5	67,0	68,5	70,2
5	5	76,5	77,6	79,3	80,6
6	6	**77,7**	**78,5**	**80,0**	**81,0**
7	7	76,0	76,6	78,0	78,8
8	3	73,9	74,5	75,7	76,4

Tabelle 4.15: Ergebnisse für Englisch in Abhängigkeit von der Bestandteillänge für 37.000 Wörter (vollständige Angaben in Tabelle A.4).

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		84,9	86,4	89,1	90,7
Länge	Lücke				
3	2	63,4	66,4	68,2	72,0
4	2	75,7	79,4	79,8	84,0
5	5	83,3	86,5	87,0	90,4
6	6	86,9	89,7	89,9	92,8
7	7	88,3	91,0	91,2	93,9
8	8	**88,7**	**91,3**	**91,3**	**94,0**

Tabelle 4.16: Ergebnisse für Holländisch in Abhängigkeit von der Bestandteillänge für 268.000 Wörter (vollständige Angaben in Tabelle A.5).

4.4.2 Auswertung der Graphem-Phonem-Zuordnungen

Die Phonemverwechslungen in Abschnitt 4.2.1.3 zeigen, daß in einigen Fällen Phoneme vom neuronalen Netz ausgegeben werden, die keinesfalls aus den eingegebenen Graphemen generiert werden können. So wird z. B. das Wort *Bundhose* vom Netz wie folgt transkribiert:

```
/ b U n C - o: - z @ /
```

Das ausgegebene Phonem für den Graphemkontext *bun-d-hose* ist also das `/C/` mit einer Gruppierung von 2. Nun ist aber das Graphem `<d>` nicht in der Liste der gültigen Grapheme (siehe Abschnitt 3.5.2) enthalten:

```
C, ch g
```

Deswegen wird nach dem nächstbesten Phonemknoten gesucht. Dieser führt zum `/t/` mit Gruppierung 1, und `<d>` ist hier als eines der möglichen Grapheme angegeben.

```
t, t th tt d dt
```

Somit wird die Entscheidung korrigiert und Phonem /t/ gewählt. Dies führt zu der richtigen Transkription

```
/ b U n t - h o: - z @ /
```

Man kann also mit der Liste der Graphem-Phonem-Zuordnungen im Anschluß an die Entscheidung des Netzes überprüfen, ob diese Entscheidung überhaupt möglich ist. Eine Untersuchung der Ergebnisse mit und ohne Verwendung dieser Zuordnungen zeigt, daß bei dem ohnehin schon guten Netz mit linkem und rechten Kontext 4 kaum eine Verbesserung zu erzielen ist, bei dem Netz mit Kontext 3 dagegen der Einfluß deutlicher sichtbar ist.

Netz	ohne GPZ		mit GPZ	
	korr. Wörter	in %	korr. Wörter	in %
Kontext 3+3	269351	87,140	271794	87,930
Kontext 4+4	291407	94,275	291441	94,286

Tabelle 4.17: Einfluß der Korrektur von Phonemfehlern mit Hilfe der Graphem-Phonem-Zuordnungen (GPZ) für Deutsch bei zwei verschiedenen neuronalen Netzen. Der Test wurde auf einem Lexikon mit 309103 Wörtern durchgeführt.

Während bei dem kleineren Netz immerhin 2442 Wörter mehr komplett richtig transkribiert wurden, sind es bei dem größeren Netz nur 34. Das liegt zum einen daran, daß das große Netz weniger dieser offensichtlichen Fehler macht. Zum anderen können nur die groben Fehler damit behoben werden. Eine Verwechslung von ähnlichen Phonemen wie z. B. /a/ und /a:/ kann damit nicht verhindert werden.

Netz	ohne GPZ		mit GPZ	
	korr. Wörter	in %	korr. Wörter	in %
Englisch	45365	73,970	45409	74,042
Holländisch	276222	89,374	276483	89,459

Tabelle 4.18: Einfluß der Korrektur von Phonemfehlern mit Hilfe der Graphem-Phonem-Zuordnungen (GPZ) für Englisch und Holländisch.

Für Englisch und Holländisch zeigt sich ein ähnliches Bild (Tabelle 4.18). Beim englischen Lexikon werden 44 Wörter mehr richtig transkribiert, bei dem holländischen sind es 261. Als Fazit läßt sich daraus sagen, daß die Verwendung der Graphem-Phonem-Zuordnungen nur bei schlechten Netzen wirklich sinnvoll ist.

Kapitel 5

Zusammenfassung, Anwendung und Ausblick

In der vorliegenden Arbeit wurde ein System vorgestellt, mit dessen Hilfe man die Aussprache von Wörtern bestimmen kann, die nicht im phonetischen Lexikon der entsprechenden Sprache enthalten sind. Dieses System ist vollautomatisch und ohne zusätzliches Expertenwissen trainierbar und somit einfach auf eine neue Sprache anzupassen (Voraussetzungen dafür siehe Einleitung). Die Vorteile sollen an dieser Stelle noch einmal kurz vorgestellt und erläutert werden.

Die Grundlage für das Training der neuronalen Netze bilden Aussprachelexika, die vollautomatisch aufbereitet werden. Zusätzliche Wissensquellen sind nicht nötig. Für die Aufbereitung wird ein sprachenunabhängiger Algorithmus verwendet, der schnell und effizient die erforderlichen Zuordnungen der Phoneme zu den sie erzeugenden Graphemen ermittelt. Diese Zuordnungen sind Ausgangspunkt für die Erstellung der Trainingsmuster.

Das Training der neuronalen Netze erfolgt ebenfalls automatisch. Für beide Netze wird die gleiche Prozedur verwendet. Das Anpassen der Lernschrittweite und die Entscheidung, wann und mit welcher Strategie ein Gewichtspruning erfolgt, werden zu Beginn über wenige Parameter vorgegeben. Danach ist ein Eingriff von außen nicht mehr nötig. Der Einsatz des Weight Decay auf die Verbindung zwischen der tatsächlichen und einer Zwischeneingabe beschleunigt das Training und führt zu einer Verbesserung der Ergebnisse. Das Netz wird dadurch in die Lage versetzt, die an den Eingabeknoten angelegte Information während des Trainings zu bewerten und die einzelnen Knoten entsprechend zu gewichten.

Durch den Einsatz zweier Nebenprodukte der Lexikonaufbereitung können die erzielten Ergebnisse nochmals verbessert werden. Die Verwendung des hybriden Ansatzes erlaubt es, ein unbekanntes Wort auch dann in Bestandteile aus dem Lexikon zerlegt werden, wenn diese Zerlegung nicht vollständig möglich ist. Die vorhandenen Lücken werden mit Hilfe der neuronalen Netze aufgefüllt. Durch die Liste der möglichen Graphem-Phonem-Zuordnungen ist man in der Lage, grobe Fehler des Netzes auszuschließen.

Das Verfahren wurde erfolgreich auf die drei Sprachen Deutsch, Englisch und Holländisch angewendet. Die erzielten Ergebnisse sind z. T. deutlich besser als die aus anderen Arbeiten.

Durch den Einsatz neuronaler Netze und die Möglichkeit, zur Verbesserung der Ergebnisse weitere Wissensquellen zu verwenden, ist das Verfahren optimal skalierbar. Bei geringen Resourcen an Rechenzeit und Speicher werden nur die neuronalen Netze verwendet. Bei leistungsfähigeren Systemen kann zusätzlich ein Lexikon verwendet werden, das man ggf. auf die Wörter beschränken kann, die durch die Netze nicht richtig transkribiert werden können. Für Serverlösungen steht schließlich der hybride Ansatz zur Verfügung.

5.1 Anwendung

Eine erste Anwendung haben die neuronalen Netze in einem Demonstrator für ein TTS-System für Mobiltelefone erfahren. Hier gab es starke Restriktionen bezüglich des Speicherplatz- und Rechenbedarfs [HVF03]. Um die Vorgaben einhalten zu können, wurden einige Anpassungen vorgenommen. Um Speicher zu sparen, wurden weniger Gewichte verwendet. Dies wurde durch die Verwendung eines kleineren Kontextes von jeweils 3 Graphemen und einer kleineren versteckten Schicht erreicht. Da die Berechnung auf dem Zielsystem durch einen Mikrocontroller durchgerführt wird, der keine Gleitkommazahlen verarbeiten kann, wurden die Gewichte, die während des Trainings als 32 Bit breite Gleitkommazahlen (`float`) vorliegen, in Festkommazahlen mit einer Breite von 8 Bit konvertiert.

Somit ergab sich für das Netz zur Graphem-Phonem-Konvertierung mit ca. 26000 Gewichten und das Netz für die Wortbetonung mit ca. 11000 Gewichten ein Speicherbedarf von gerade einmal 37 KB. Bei der Anwendung dieser Netze auf das Lexikon mit 270000 Einträgen mit einer Länge von mindestens 8 Buchstaben werden 81 Prozent der Wörter vollständig richtig transkribiert. Die richtig transkribierten Wörter belegen in unkomprimierter Form ungefähr 6 MB und werden durch die Netze verlustfrei auf 37 KB komprimiert. Das entspricht einem Verhältnis von 160:1, wobei man mit den Netzen immer noch in der Lage ist, unbekannte Wörter zu transkribieren, was bei einer einfachen Kompression eines Lexikons nicht der Fall ist. Einfache Kompressionsprogramme wie `gzip` oder `bzip2` erreichen eine Verringerung auf ca. 1,2 MB, also nur ein Verhältnis von 5:1.

Zur Bestimmung der Rechenzeit wurden Tests in einer Emulationsumgebung für die Zielplatform durchgeführt. Diese ergaben, daß bei einem Prozessor (ARM-920T) mit 150 MHz lediglich 5 MHz benötigt werden, um die Transkription in Echtzeit zu generieren. Damit eignen sich die hier vorgestellten neuronalen Netze sehr gut für den Einsatz in Mobiltelefonen oder anderen Geräten mit wenig Rechen- und Speicherkapazität.

5.2 Ausblick

Zukünftige Aufgaben liegen vor allem in der weiteren Verbesserung der Lexikonaufbereitung. Ein Teil der Fehler, die das neuronale Netz für die Graphem-Phonem-Konvertierung macht, kommen durch falsche Zuordnungen im aufbereiteten Lexikon. Aber auch die Ergebnisse der neuronalen Netze müssen weiter verbessert werden, vor allem für Englisch. Ein Ansatz dafür könnte die Anwendung einer anderen Netzwerkarchitektur sein. Die in dieser Arbeit verwendete MLP-Architektur läßt keine zeitlichen Abhängigkeiten der Eingabeinformationen untereinander erkennen. Die Reihenfolge, in der die Grapheme oder Phoneme am Eingang des Netzes angelegt werden, ist beliebig vertauschbar. Das Netz sieht nicht, daß ein Knoten, der rechts von einem anderen Knoten liegt, dessen Nachfolger ist.

Außerdem zeigt die Auswertung, daß die Ergebnisse der neuronalen Netze um so besser sind, je größer der Eingabekontext ist, der ihnen für eine Entscheidung zur Verfügung steht. Das geht natürlich auch nur bis zu einem gewissen Grad, aber ein Netz für die Graphem-Phonem-Konvertierung mit einem Kontext von 4 Graphemen links und rechts liefert bessere als ein Netz mit einem Kontext von jeweils 3 Graphemen (siehe Tabelle 4.17). Je größer aber der Kontext ist, desto größer ist auch die Anzahl der Gewichte. Das Training dauert dann um so länger, und der Speicherbedarf steigt ebenfalls.

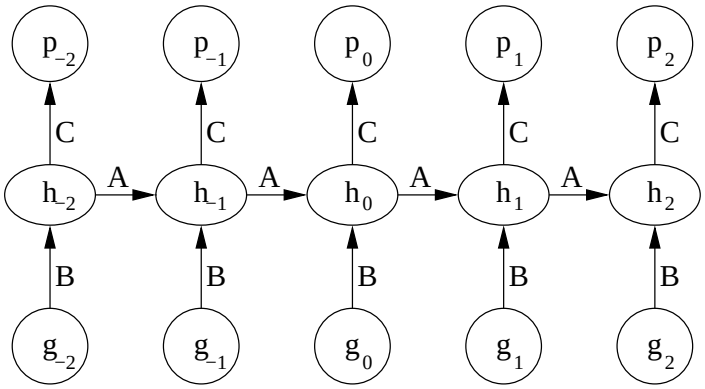

Abbildung 5.1: Beispiel für ein zeitfaltendes Netz für die Graphem-Phonem-Konvertierung. Die Knoten g_{-2} bis g_2 stehen für die Graphemeingabe und p_{-2} bis p_2 für die Phonemausgabe, wobei in der Anwendung nur die Ausgabe von p_0 interessant ist. *A*, *B* und *C* sind die gemeinsam genutzen Gewichtsmatrizen, die bei allen Verbindungen gleichermaßen verwendet werden.

Eine alternative Netzwerkarchitektur, die die gerade beschriebenen Nachteile nicht hat, ist in Abbildung 5.1) dargestellt. Hierbei handelt es sich um ein sogenanntes „zeitfaltendes neuronales Netz“ (engl. finite unfolding in time neural network [RHW86]), wie es im TTS-System „Papageno“ bereits zur Bestimmung der akustischen Prosodie [EZ02b] und zur Generierung der Betonungsmarker in der symbolischen Prosodie eingesetzt wird [Mül03]. In diesem Netz gibt es einen gerichteten Informationsfluß, bei dem der zweite Vorgängerknoten g_{-2} vom Zentrum g_0 „weiter entfernt“ ist als der direkte Vorgänger g_{-1}, und durch die Verwendung von gemein-

sam genutzten Gewichtsmatrizen (engl. shared weights) steigt die Anzahl der Gewichte nicht bei der Vergrößerung des Kontextes.

Es handelt sich hierbei quasi um mehrere MLP-Netze, eines für jeden Kontext, die aber untereinander verbunden sind. Die Strecke $g \rightarrow B \rightarrow h \rightarrow C \rightarrow p$ entspricht dem Netzwerk in Abbildung 3.10 für ein Eingabegraphem. Die „zeitliche“ Abhängigkeit der Kontexte wird durch die Verbindung mit Matrix A erreicht. Die Information von g_0 fließt dirkekt über B in die Entscheidung über p_0 ein, die von g_{-1} über die Strecke $B \rightarrow A$ und g_{-2} über $B \rightarrow A \rightarrow A$. Somit ist der Einfluß von Kontext 0 am größten, der von Kontext -1 etwas schwächer, der von -2 noch schwächer usw., genau so, wie es der Wichtigkeit der einzelnen Kontexte (Transinformation, sie Abbildung 3.13) entspricht.

In Abbildung 5.1 gibt es allerdings keinen Informationsfluß vom rechten Kontext hin zum Zentrum. Dies könnte man zum einen ändern, indem man die Richtung der Verbindung von h_2 nach h_1 und h_1 nach h_0 umkehrt. Dann müßte man aber konsequenterweise auch eine andere Matrix A' verwenden. Zum anderen könnte man analog zu [EZ02a] und [Mül03] eine zweite Strecke in die Gegenrichtung verwenden ($g \rightarrow B' \rightarrow h' \rightarrow C' \rightarrow p$ mit einer Verbindung von h'_2 nach h'_1 über A'), sodaß der komplette Informationsfluß in beide Richtungen parallel verarbeitet wird. Dies wäre in mehreren Versuchen auszuprobieren.

Anhang A

Tabellen zur Graphem-Phonem-Konvertierung

A.1 Netz für Phonemfolge und Silbengrenze

In diesem Abschnitt sind die Tabellen für die Graphem-Phonem-Konvertierung aufgeführt. Tabelle A.1 listet die Ergebnisse des Trainings für das deutsche Netz ohne Gruppierung und mit Weight Decay auf. Angegeben sind jeweils die Muster, die das Phonem in Spalte 1 als Ausgabe haben, wobei noch eine Aufteilung in Trainings- und Testmuster erfolgt. Es gibt also insgesamt 10462 Muster für das Phonem /aI/, von denen 7373 Muster in der Trainings- und 3089 Muster in der Testmenge enthalten sind. Von den Trainingsmustern werden 99,92 % richtig verarbeitet und von den Testmustern 99,90 %.

A.2 Kombination von Lexikon und neuronalen Netzen

Die Tabellen A.2 bis A.5 zeigen die Ergebnisse für das Gesamtsystem, also die Bestimmung der Phonemfolge inklusive Silbengrenze und der Wortbetonung, wobei nur die Wörter im Lexikon beachtet werden, die mindestens 8 Buchstaben haben. In der ersten Zeile sind jeweils die Ergebnisse eingetragen, die nur mit den beiden neuronalen Netzen erreicht wurden. Die folgenden Zeilen geben die Ergebnisse des hybriden Ansatzes an, also die Verbindung von Lexikonsuche und neuronalen Netzen (siehe Abschnitt 4.4.1).

Phonem	Trainings-muster	korrekt in %	Test-muster	korrekt in %	gesamt	korrekt in %
aI	7373	99,92	3089	99,90	10462	99,91
aU	4328	100,00	1865	99,89	6193	99,97
OY	2122	99,39	912	99,01	3034	99,27
a:	12842	98,91	5485	95,00	18327	97,74
E:	4056	98,96	1837	97,44	5893	98,49
e:	8169	97,25	3470	94,03	11639	96,29
i:	16278	99,34	6896	97,82	23174	98,89
o:	9078	99,71	3880	98,32	12958	99,30
2:	2821	99,93	1181	98,98	4002	99,65
u:	7473	99,61	3177	97,99	10650	99,13
y:	3158	99,34	1436	98,19	4594	98,98
a	17030	99,42	7295	96,97	24325	98,68
E	15053	98,59	6634	96,56	21687	97,97
I	17928	99,74	7733	99,13	25661	99,56
O	6014	98,57	2609	97,09	8623	98,12
9	1195	99,75	553	98,01	1748	99,20
U	6707	99,75	3011	97,91	9718	99,18
Y	2693	99,70	1135	98,59	3828	99,37
6	8853	99,67	3822	98,80	12675	99,41
@	41672	99,82	17621	99,27	59293	99,65
b	11365	99,97	4747	99,75	16112	99,91
C	6932	99,99	3024	99,54	9956	99,85
d	9846	99,98	4120	99,81	13966	99,93
f	12164	99,94	5298	99,70	17462	99,87
g	14905	99,99	6197	99,73	21102	99,91
h	4782	100,00	2030	100,00	6812	100,00
j	513	97,47	231	96,97	744	97,31
k	14651	99,64	6275	99,43	20926	99,57
l	30503	99,97	12881	99,95	43384	99,97
m	13106	99,95	5735	99,93	18841	99,95
N	5360	98,30	2272	98,42	7632	98,34
n	29977	99,95	12821	99,81	42798	99,91
p	8971	99,81	3927	99,57	12898	99,74
r	38036	99,97	16429	99,93	54465	99,96
S	8772	99,68	3834	99,27	12606	99,56
s	19472	99,56	8349	98,83	27821	99,34
t	42780	99,91	18377	99,67	61157	99,83
v	6327	99,94	2723	99,45	9050	99,79
x	2857	99,82	1211	99,83	4068	99,83
z	8543	99,96	3504	99,69	12047	99,88
Z	94	58,51	53	41,51	147	52,38
kv	213	100,00	87	100,00	300	100,00
pf	1546	100,00	685	100,00	2231	100,00
ts	11496	99,83	5008	99,34	16504	99,68
tS	171	97,66	79	93,67	250	96,40
ks	1743	99,66	733	98,91	2476	99,43
∑	499968		214271		714239	

Tabelle A.1: Anzahl und Prozentsatz richtig behandelter Muster für die deutsche Graphem-Phonem-Konvertierung.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		91,16	92,25	94,13	95,29
Länge	Lücke				
3	1	85,68	87,13	88,63	90,31
3	2	**86,03**	**87,49**	**89,00**	**90,67**
3	3	85,27	87,33	88,20	90,50
4	1	90,63	91,89	92,89	94,24
4	2	**90,96**	**92,21**	**93,21**	**94,56**
4	3	90,48	92,21	92,74	94,57
4	4	90,06	92,15	92,30	94,50
5	1	93,87	95,22	95,70	97,11
5	2	**94,12**	**95,46**	**95,95**	**97,36**
5	3	93,85	95,56	95,70	97,48
5	4	93,95	95,69	95,81	97,61
5	5	93,94	95,71	95,79	97,63
6	1	94,38	95,88	95,85	97,39
6	2	94,66	96,15	96,13	97,67
6	3	94,66	96,30	96,15	97,83
6	4	94,99	96,42	96,48	97,96
6	5	**95,04**	**96,51**	**96,53**	**98,04**
6	6	95,01	96,49	96,51	98,03
7	1	94,29	95,75	95,65	97,16
7	2	94,87	96,33	96,24	97,74
7	3	94,96	96,47	96,34	97,89
7	4	95,27	96,57	96,65	98,00
7	5	**95,30**	**96,62**	**96,69**	**98,05**
7	6	95,29	96,62	96,68	98,05
7	7	95,28	96,61	96,67	98,04
8	1	94,48	95,87	95,81	97,23
8	2	94,90	96,29	96,23	97,65
8	3	94,98	96,39	96,33	97,76
8	4	95,19	96,44	96,54	97,82
8	5	**95,21**	**96,47**	**96,55**	**97,85**
8	6	95,20	96,46	96,55	97,84
8	7	95,18	96,46	96,53	97,84
8	8	95,19	96,46	96,53	97,84

Tabelle A.2: Ergebnisse für Deutsch in Abhängigkeit von der Bestandteillänge für das Netz mit linkem und rechtem Kontext von 4 Graphemen.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		85,07	87,19	87,58	89,83
Länge	Lücke				
3	1	85,47	86,95	88,40	90,09
3	2	**85,82**	**87,29**	**88,75**	**90,45**
3	3	85,00	87,07	87,89	90,21
4	1	90,30	91,60	92,51	93,91
4	2	**90,63**	**91,92**	**92,84**	**94,23**
4	3	90,09	91,86	92,29	94,17
4	4	89,54	91,69	91,73	93,99
5	1	93,41	94,81	95,18	96,65
5	2	93,66	95,06	**95,43**	96,90
5	3	93,33	95,10	95,10	96,95
5	4	93,29	**95,11**	95,07	**96,96**
5	5	93,21	95,08	94,97	96,93
6	1	93,71	95,25	95,10	96,70
6	2	93,98	95,53	95,38	96,98
6	3	93,90	95,60	95,31	97,06
6	4	**94,05**	95,62	**95,45**	97,09
6	5	94,03	**95,66**	95,44	**97,12**
6	6	93,96	95,61	95,36	97,07
7	1	93,25	94,83	94,54	96,16
7	2	93,82	95,40	95,11	96,74
7	3	93,84	95,47	95,13	96,81
7	4	**93,98**	95,48	**95,27**	96,83
7	5	93,96	**95,49**	95,26	**96,84**
7	6	93,91	95,46	95,21	96,80
7	7	93,86	95,42	95,15	96,76
8	1	92,86	94,48	94,10	95,76
8	2	93,27	94,89	94,51	96,17
8	3	93,30	94,94	94,54	96,22
8	4	**93,40**	**94,94**	**94,64**	**96,23**
8	5	93,38	94,94	94,62	96,23
8	6	93,34	94,91	94,58	96,20
8	7	93,30	94,89	94,54	96,18
8	8	93,29	94,88	94,53	96,17

Tabelle A.3: Ergebnisse für Deutsch in Abhängigkeit von der Bestandteillänge für das Netz mit linkem und rechtem Kontext von 3 Graphemen.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		66,97	67,54	68,70	69,31
Länge	Lücke				
3	1	58,35	60,07	61,25	63,38
3	2	**58,48**	**60,14**	**61,36**	**63,44**
3	3	57,99	59,80	60,77	63,02
4	1	**65,49**	**66,95**	**68,45**	**70,19**
4	2	65,45	66,84	68,41	70,08
4	3	65,10	66,61	68,00	69,81
4	4	65,25	66,69	68,15	69,89
5	1	75,25	76,69	78,03	79,71
5	2	75,46	76,82	78,24	79,84
5	3	75,57	76,99	78,33	80,00
5	4	76,21	77,37	78,96	80,38
5	5	**76,53**	**77,58**	**79,30**	**80,59**
6	1	76,53	77,75	78,82	80,25
6	2	76,74	77,87	79,04	80,38
6	3	77,00	78,14	79,28	80,64
6	4	77,40	78,31	79,68	80,80
6	5	77,57	78,40	79,86	80,90
6	6	**77,69**	**78,46**	**79,98**	**80,95**
7	1	75,28	76,31	77,32	78,52
7	2	75,56	76,48	77,60	78,70
7	3	75,67	76,61	77,68	**78,80**
7	4	75,81	76,59	77,83	78,78
7	5	75,89	76,60	77,90	78,78
7	6	75,95	**76,61**	77,97	78,79
7	7	**76,00**	76,60	**78,01**	78,78
8	1	73,80	74,47	75,58	76,37
8	2	73,87	74,50	75,65	**76,40**
8	3	**73,89**	**74,51**	**75,66**	76,39
8	4	73,85	74,43	75,62	76,31
8	5	73,86	74,43	75,63	76,30
8	6	73,87	74,42	75,63	76,30
8	7	73,87	74,42	75,62	76,29
8	8	73,87	74,42	75,63	76,29

Tabelle A.4: Ergebnisse für Englisch in Abhängigkeit von der Bestandteillänge für das Netz mit linkem und rechtem Kontext von 4 Graphemen.

		gesamt	nur Betonung	nur Silbengrenze	ohne Bet. und SG
Netz		84,89	86,39	89,06	90,65
Länge	Lücke				
3	1	63,23	66,28	68,05	71,95
3	2	**63,35**	66,36	**68,18**	72,04
3	3	62,89	**66,38**	67,67	**72,04**
4	1	75,53	78,76	79,64	83,39
4	2	**75,65**	78,79	**79,77**	83,42
4	3	75,49	78,96	79,59	83,60
4	4	75,39	**79,41**	79,46	**84,04**
5	1	82,01	84,95	85,63	88,82
5	2	82,11	84,96	85,73	88,83
5	3	82,19	85,26	85,82	89,14
5	4	82,92	85,88	86,55	89,76
5	5	**83,34**	**86,52**	**86,97**	**90,41**
6	1	84,94	87,82	87,98	90,97
6	2	85,04	87,82	88,08	90,97
6	3	85,44	88,25	88,48	91,41
6	4	86,13	88,80	89,17	91,96
6	5	86,52	89,29	89,57	92,45
6	6	**86,89**	**89,67**	**89,94**	**92,84**
7	1	86,68	89,49	89,50	92,37
7	2	86,86	89,52	89,68	92,39
7	3	87,18	89,80	90,00	92,69
7	4	87,63	90,19	90,46	93,08
7	5	87,96	90,54	90,79	93,43
7	6	88,19	90,79	91,02	93,69
7	7	**88,33**	**90,96**	**91,17**	**93,86**
8	1	87,73	90,43	90,34	93,10
8	2	87,85	90,44	90,46	93,11
8	3	88,01	90,58	90,63	93,26
8	4	88,23	90,77	90,85	93,45
8	5	88,41	90,96	91,04	93,65
8	6	88,55	91,13	91,18	93,81
8	7	88,64	91,23	91,27	93,92
8	8	**88,68**	**91,29**	**91,31**	**93,97**

Tabelle A.5: Ergebnisse für Holländisch in Abhängigkeit von der Bestandteillänge für das Netz mit linkem und rechtem Kontext von 4 Graphemen.

Anhang B

Tabellen zur Wortbetonung

Hier sind die Tabellen für die Wortbetonung aufgelistet. Abschnitt B.1 enthält die Tabellen, mit deren Hilfe der Kontext für den silbenbasierten Ansatz bestimmt wurde, Abschnitt B.2 enthält die Tabellen zur Bestimmung des Kontextes beim phonembasierten Ansatz, und in Abschnitt B.3 sind die Ergebnisse des Weight Decay dargestellt.

B.1 Silbenbasierter Ansatz

	1	2	3	4	5	6	7	8	∑	%
1	5116	0	0	0	0	0	0	0	5116	1,7
2	40815	7179	0	0	0	0	0	0	47994	15,5
3	69737	27381	4109	0	0	0	0	0	101227	32,7
4	53097	19029	14952	1869	0	0	0	0	88947	28,8
5	23196	8087	7107	6170	785	0	0	0	45345	14,7
6	6271	2556	1962	2161	2012	270	0	0	15232	4,9
7	1317	655	501	431	564	539	47	0	4054	1,3
8	271	104	144	168	155	94	88	7	1031	0,33
9	44	21	26	55	27	26	3	21	223	0,07
10	3	2	0	11	1	0	0	1	18	0,006
11	4	0	0	0	1	0	0	0	5	0,002
∑	199871	65014	28801	10865	3545	929	138	29	309192	
%	64,6	21,0	9,3	3,5	1,1	0,3	0,04	0,01		

Tabelle B.1: Verteilung der Betonungen auf die Silben für Deutsch. In den Zeilen sind die Wörter mit n Silben angegeben, in den Spalten die Häufigkeit, mit der die Silbe an Position m betont wurde. So gibt es z. B. 101227 Wörter mit drei Silben. Bei diesen Wörtern liegt die Betonung 69737 Mal auf der ersten Silbe, 27381 Mal auf der zweiten usw.

	1	2	3	4	5	6	7	∑	%
1	7997	0	0	0	0	0	0	7997	12,98
2	19536	3329	0	0	0	0	0	22865	37,12
3	11293	5762	949	0	0	0	0	18004	29,23
4	2925	3384	2364	139	0	0	0	8812	14,31
5	260	1117	1047	688	18	0	0	3130	5,08
6	14	52	277	182	135	0	0	660	1,07
7	0	0	13	43	48	15	0	119	0,19
8	0	0	0	0	2	5	3	10	0,02
∑	42025	13644	4650	1052	203	20	3	61597	
%	68,23	22,15	7,55	1,71	0,33	0,03	0,005		

Tabelle B.2: Verteilung der Betonungen auf die Silben für Englisch.

	1	2	3	4	5	6	7	8	∑	%
1	6113	0	0	0	0	0	0	0	6113	1,99
2	43851	9689	0	0	0	0	0	0	53540	17,41
3	69430	24145	7653	0	0	0	0	0	101228	32,92
4	44710	18426	15872	5227	0	0	0	0	84235	27,39
5	14340	9067	7233	7994	1969	0	0	0	40603	13,20
6	4435	3193	2763	2082	2527	483	0	0	15483	5,03
7	1187	1021	804	584	514	507	78	0	4695	1,53
8	239	284	233	237	105	71	96	13	1278	0,42
9	57	44	60	50	45	12	17	9	294	0,01
10	10	2	1	7	19	0	1	0	40	0,01
11	0	0	0	1	7	0	0	0	8	0,003
∑	184372	65871	34619	16182	5186	1073	192	22	307517	
%	59,95	21,42	11,26	5,26	1,69	0,35	0,06	0,007		

Tabelle B.3: Verteilung der Betonungen auf die Silben für Holländisch.

B.2 Phonembasierter Ansatz

n	Wörter mit n Phonemen	%	Betonung auf Phonem n	%
1	6	0,002	59227	19,16
2	113	0,04	104924	33,93
3	1144	0,37	36332	11,75
4	4698	1,52	32608	10,55
5	11191	3,62	29320	9,48
6	21102	6,82	19054	6,16
7	32729	10,59	11516	3,72
8	42819	13,85	6852	2,22
9	46105	14,91	4293	1,39
10	42319	13,69	2677	0,87
11	33866	10,95	1480	0,48
12	25966	8,40	526	0,17
13	18409	5,95	236	0,076
14	12331	3,99	95	0,031
15	7006	2,27	36	0,012
16	4019	1,30	14	0,005
17	2323	0,75		
18	1420	0,46	2	0,001
19	801	0,26		
20	439	0,14		
21	214	0,07		
22	98	0,03		
23	49	0,016		
24	11	0,0036		
25	3	0,001		
26	3	0,001		
27	5	0,002		
28	1	0,0003		
29	2	0,0006		

Tabelle B.4: Verteilung der Betonungen auf Phonemebene für Deutsch. In der ersten Spalte ist die Anzahl der Phoneme und in der zweiten die Anzahl der Wörter mit soviel Phonemen angegeben. Es gibt also z. B. 21102 Wörter mit 6 Phonemen. In der dritten Spalte steht die Anzahl der Betonungen auf dem Phonem aus Spalte 1. So liegt die Wortbetonung 104924 Mal auf dem zweiten Phonem.

n	Wörter mit n Phonemen	%	Betonung auf Phonem n	%
1	4	0,006	3176	5,16
2	227	0,37	28242	45,85
3	2396	3,89	11520	18,70
4	5464	8,87	7668	12,45
5	9405	15,27	4617	7,50
6	10529	17,09	2763	4,49
7	10003	16,24	1717	2,79
8	8410	13,65	983	1,60
9	6034	9,80	424	0,69
10	4137	6,72	269	0,44
11	2519	4,09	132	0,21
12	1328	2,16	56	0,09
13	640	1,04	23	0,04
14	297	0,48	5	0,008
15	136	0,22	2	0,003
16	42	0,07		
17	17	0,03		
18	8	0,01		
19	1	0,002		

Tabelle B.5: Verteilung der Betonungen auf Phonemebene für Englisch.

n	Wörter mit n Phonemen	%	Betonung auf Phonem n	%
1	0	0	35783	11,64
2	127	0,04	103844	33,77
3	1767	0,57	44618	14,51
4	5339	1,74	30762	10,00
5	11612	3,78	28498	9,27
6	23238	7,56	20982	6,82
7	35419	11,52	15208	4,95
8	43564	14,17	10880	3,54
9	45796	14,89	7341	2,39
10	40470	13,16	4809	1,56
11	31837	10,35	2571	0,84
12	23589	7,67	1280	0,42
13	16545	5,38	508	0,17
14	10790	3,51	251	0,08
15	6901	2,24	77	0,03
16	4254	1,38	79	0,03
17	2644	0,86	12	0,004
18	1553	0,51	12	0,004
19	968	0,31	2	0,0007
20	503	0,16	1	0,0003
21	314	0,10		
22	138	0,04		
23	83	0,03		
24	39	0,01		
25	18	0,006		
26	9	0,003		
27	1	0,0003		

Tabelle B.6: Verteilung der Betonungen auf Phonemebene für Holländisch.

B.3 Interpretation des Weight Decay als Eingabepruning

Phonem	Kontext										Anzahl
	1	2	3	4	5	6	7	8	9	10	
}	x	-	-	-	-	-	-	-	-	-	1
|:	x	-	-	-	-	-	-	-	-	x	2
y:	x	-	-	-	-	-	-	-	x	x	3
u:	-	-	-	-	-	-	-	x	x	x	3
o:	-	-	-	-	-	-	-	-	-	-	0
i:	x	-	-	-	-	-	-	-	-	-	1
e:	-	-	-	-	-	-	-	-	-	-	0
a:	-	-	-	-	-	-	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	x	1
M	-	-	x	-	x	x	x	x	x	x	7
L	-	-	-	-	-	-	x	x	x	x	4
K	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	-	-	0
E:	x	x	x	-	-	-	-	-	x	x	5
E	-	-	-	-	-	-	-	-	-	x	1
A	-	-	-	-	-	-	-	-	x	-	1
@	x	-	-	-	-	-	-	-	-	-	1
<	x	x	x	x	x	x	x	x	x	x	10
*	x	x	x	x	x	x	x	x	x	x	10
(	x	x	x	x	x	x	x	x	x	x	10
!	x	x	x	x	x	x	x	x	x	x	10
z	-	x	-	-	-	-	-	-	-	x	2
x	-	-	-	-	-	-	-	-	-	-	0
w	-	-	-	-	-	-	-	x	x	x	3
v	-	-	-	-	-	-	-	x	-	-	1
t	-	-	-	-	-	-	-	-	-	-	0
s	-	-	-	-	-	-	-	-	-	-	0
r	-	-	-	-	-	-	-	-	-	-	0
p	-	-	-	-	-	-	-	-	-	-	0
n	-	-	-	-	-	-	-	-	-	-	0
m	-	-	-	-	-	-	-	-	-	-	0
l	-	-	-	-	-	-	-	-	-	-	0
k	-	-	-	-	-	-	-	-	-	-	0
j	-	x	-	-	-	-	-	-	-	-	1
h	-	x	-	-	-	-	-	-	-	-	1
g	x	x	x	x	x	x	x	x	x	x	10
f	-	-	-	-	-	-	-	-	-	-	0
dZ	x	x	x	x	x	x	x	x	x	x	10
d	-	-	-	-	-	-	-	-	-	-	0
b	-	-	-	-	-	-	-	-	-	x	1
Z	x	x	x	x	-	-	x	x	x	x	8
S	x	x	x	-	-	x	x	x	x	x	8
N	x	-	-	-	-	-	-	-	-	-	1
G	x	x	-	-	-	-	-	-	-	-	2

Tabelle B.7: Ausgeblendete Phoneme für Holländisch. Ein *x* steht für den Kontext, in dem die Weight-Decay-Verbindung einen Wert im Bereich $[0;0,01]$ hat.

Phonem	Kontext											Anzahl
	1	2	3	4	5	6	7	8	9	10	11	
{	-	-	-	-	-	-	-	-	-	-	-	0
u:	x	-	-	-	-	-	-	-	-	x	x	3
i:	-	-	-	-	-	-	-	-	x	x	x	3
eI	-	-	-	-	-	-	-	-	-	-	-	0
aU	-	-	-	-	-	x	x	x	x	x	x	6
aI	-	-	-	-	-	-	-	-	-	x	-	1
V	-	-	-	-	-	-	x	x	x	x	x	5
U	x	-	-	-	-	x	-	x	-	x	x	5
Q	-	-	-	-	-	-	-	-	x	x	x	3
OI	x	-	-	x	-	-	x	x	x	x	x	7
O:	-	-	-	-	-	-	-	-	x	x	x	3
I	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	-	-	-	-	-	x	-	1
A:	-	-	-	-	-	-	-	-	x	x	x	3
8	-	-	-	-	-	x	x	x	x	x	x	6
7	x	-	-	-	-	-	-	-	-	-	-	1
5	-	-	-	-	-	-	-	-	-	-	x	1
3:	x	-	-	-	-	-	-	-	x	x	x	4
z	x	x	-	-	-	-	-	-	-	-	-	2
w	-	-	-	-	-	-	-	x	x	x	x	4
v	-	-	-	-	-	-	-	x	-	-	-	1
tS	-	x	-	x	-	x	x	x	x	x	-	7
t	-	-	-	-	-	-	-	-	-	-	-	0
s	-	-	-	-	-	-	-	-	-	-	-	0
r	-	-	-	-	-	-	-	-	-	-	-	0
p	-	-	-	-	-	-	-	-	-	x	-	1
n	-	-	-	-	-	-	-	-	-	-	-	0
m	-	-	-	-	-	-	-	-	-	-	-	0
l	-	-	-	-	-	-	-	-	-	-	-	0
k	-	-	-	-	-	-	-	-	-	-	-	0
j	x	-	-	-	-	-	-	-	x	x	x	4
h	-	x	-	-	x	x	x	x	x	x	x	8
g	-	-	-	-	-	-	x	x	x	x	x	5
f	-	-	-	-	-	-	-	x	-	x	x	3
dZ	-	x	-	-	-	x	x	-	x	x	x	6
d	-	-	-	-	-	-	-	-	-	-	-	0
b	-	-	-	-	-	-	-	-	-	-	-	0
Z	x	x	x	x	x	x	x	x	x	x	x	11
T	-	x	-	x	x	x	x	x	x	x	x	9
S	-	x	-	-	-	-	-	-	-	-	-	1
R	x	x	x	x	x	x	-	-	-	-	-	6
P	x	x	x	-	-	-	-	x	-	-	-	4
N	x	-	-	x	-	-	-	-	-	-	-	2
H	x	x	x	-	-	-	-	-	-	-	-	3
D	x	x	x	x	x	x	x	x	x	x	x	11
@	-	-	-	-	-	-	-	-	-	-	-	0
9	x	-	-	-	-	x	-	x	-	x	x	5

Tabelle B.8: Ausgeblendete Phoneme für Englisch.

Phonem	Kontext									Anzahl
	1	2	3	4	5	6	7	8	9	
aI	-	-	-	-	-	-	-	x	x	2
aU	-	-	-	-	-	-	x	x	-	2
OY	-	-	-	-	-	-	-	x	x	2
a:	-	-	-	-	-	-	-	-	-	0
2:	-	-	-	-	-	-	-	-	x	1
e:	-	-	-	-	-	-	-	-	x	1
i:	x	-	-	-	-	-	-	-	-	1
o:	-	-	-	-	-	-	-	-	-	0
u:	-	-	-	-	-	-	-	x	x	2
y:	-	-	-	-	-	-	-	x	x	2
E:	x	-	-	-	-	-	-	-	-	1
a	-	-	-	-	-	-	-	x	x	2
9	x	-	x	-	-	x	x	x	x	6
E	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	-	0
O	-	-	-	-	-	-	x	x	x	3
U	-	-	-	-	-	-	-	-	-	0
Y	x	-	-	-	-	x	x	x	x	5
@	x	-	-	-	-	-	-	-	-	1
6	x	x	-	-	-	-	-	-	-	2
l	-	-	-	-	-	-	-	-	-	0
m	-	-	-	-	-	-	-	-	-	0
n	-	-	-	-	-	-	-	-	-	0
r	-	-	-	-	-	-	-	-	-	0
N	x	-	-	x	-	-	-	-	x	3
Z	x	x	x	x	-	x	x	x	x	8
j	-	x	x	x	x	x	x	x	x	8
v	-	-	-	-	-	-	-	-	x	1
z	-	x	-	-	-	-	-	-	x	2
C	x	x	-	-	-	-	-	-	-	2
S	-	x	-	-	-	-	-	-	-	1
b	-	-	-	-	-	-	-	x	-	1
d	-	-	-	-	-	-	-	x	-	1
f	-	-	-	-	-	-	-	-	x	1
g	-	-	-	-	-	-	-	-	-	0
h	-	x	-	-	-	x	x	x	x	5
k	-	-	-	-	-	-	-	-	-	0
p	-	-	-	-	-	-	-	-	x	1
s	-	-	-	-	-	-	-	-	-	0
t	-	-	-	-	-	-	-	-	-	0
x	x	-	-	-	x	x	-	-	x	4
pf	-	x	-	-	x	x	x	x	x	6
ts	-	x	-	-	-	-	-	-	-	1
tS	x	x	x	x	x	x	x	x	x	9
dZ	x	x	x	x	x	x	x	x	x	9
ks	x	-	-	-	-	x	x	x	x	5
kv	x	x	x	x	x	x	x	x	x	9

Tabelle B.9: Ausgeblendete Phoneme für Deutsch.

Literaturverzeichnis

[AD94] ANDERSEN, OVE und PAUL DALSGAARD: *A Self-Learning Approach to Transcription of Danish Proper Names*. Proc. ICSLP, Seiten 1627–1630, 1994.

[AZ02] ALTMANN, HANS und UTE ZIEGENHAIN: *Phonetik, Phonologie und Graphemik fürs Examen*. Westdeutscher Verlag, 2002.

[Bags98] BAGSHAW, PAUL C.: *Phonetic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression*. Computer, Speech and Language, 12:119–142, 1998.

[BBvdB99] BERTJAN BUSSER, WALTER DAELEMANS und ANTAL VAN DEN BOSCH: *Machine Learning of Word Pronunciation: The Case Against Abstraction*. Eurospeech, 1999.

[Burn90] BURNAGE, G.: *CELEX: a guide for users*. Technischer Bericht, University of Nijmegen, Center for Lexical Information, 1990.

[Buss83] BUSSMANN, HADUMOD: *Lexikon der Sprachwissenschaft*. Alfred Kröner Verlag Stuttgart, 1983.

[CMR90] CONTENT, A., P. MOUSTY und M. RADEAU: *BRULEX: Une base de données lexicales informatisée pour le français écrit et parlé*. L'Année Psycholoqique, Seiten 551–566, 1990.

[Coil90] COILE, BERT VAN: *Inductive Learning of Grapheme-to-Phoneme Rules*. Proc. ICSLP, 2:19.1.1–19.1.4, 1990.

[Coil91] COILE, BERT VAN: *Inductive Learning of Pronunciation Rules with the DEPES System*. Proc. ICASSP, 2:745–748, 1991.

[Coil93] COILE, BERT VAN: *On the Development of Pronunciation Rules for Text-to-Speech Synthesis*. Eurospeech, 2:1455–1458, 1993.

[Colt78] COLTHEART, M.: *Lexical access in simple reading tasks*. In: UNDERWOOD, G. (Herausgeber): *Strategies of information processing*, Seiten 151–216. Academic, London, 1978.

[DE97] DAMPER, R. I. und J. F. G. EASTMOND: *Pronunciation by Analogy: Impact of Implementational Choices on Performance*. Language and Speech, 40(1):1–23, 1997.

[Domi96] DOMINGOS, P.: *Unifying instance-based and rule-based induction*. Machine Learning, 24:141–168, 1996.

[DvdB93] DAELEMANS, WALTER und ANTAL VAN DEN BOSCH: *TabTalk: Reusability in Data-orientet Grapheme-to-Phoneme Conversion*. Eurospeech, 2:1459–1462, 1993.

[DYB95] DELIGNE, SABINE, FRANÇOIS YVON und FRÉDÉRIC BIMBOT: *Variable-length Sequence Matching for Phonetic Transcription Using Joint Multigrams*. Eurospeech, 3:2243–2246, 1995.

[EZ02a] ERDEM, ÇAĞLAYAN und HANS GEORG ZIMMERMANN: *A Data-driven Method for Input Feature Selection within Neural Prosody Generation*. ICASSP, 1:477–480, 2002.

[EZ02b] ERDEM, ÇAĞLAYAN und HANS GEORG ZIMMERMANN: *Segmental duration control by time delay neural networks with asymmetric causal and retro-causal information flows*. European Symposium on Artificial Neural Networks, 2002.

[EZH01] ERDEM, ÇAĞLAYAN, HANS GEORG ZIMMERMANN und RÜDIGER HOFFMANN: *Datengetriebene Optimierung von Eingangsgrößen der Prosodiegenerierung*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 228–235, 2001.

[Frie95] FRIEDRICHTS, BERND: *Kanalcodierung: Grundlagen und Anwendungen in modernen Kommunikationssystemen*. Springer Verlag, 1995.

[Gall68] GALLAGER, ROBERT G.: *Information Theory and Reliable Communication*. John Wiley ands Sons, 1968.

[Glus79] GLUSHKO, R. J.: *The organization and activation of orthographic knowledge in reading aloud*. Journal of Experimental Psychology: Human Perception and Performance, 5:674–691, 1979.

[Glus81] GLUSHKO, R. J.: *Principles for pronouncing print: The psychology of phonography*. In: LESGOLD, A. M. und C. A. PERFETTI (Herausgeber): *Interactive process in reading*, Seiten 61–84. Lawrence Erlbaum, Hillsdale, NJ, 1981.

[Hain96] HAIN, HORST-UDO: *Baseline-System für die deutsche Sprachsynthese*. Diplomarbeit, Technische Universität Dresden, 1996.

[Hain99a] HAIN, HORST-UDO: *Automation of the Training Procedure for Neural Networks Performing Multi-lingual Grapheme to Phoneme Conversion*. Eurospeech, 5:2087–2090, 1999.

[Hain99b] HAIN, HORST-UDO: *Datengetriebene Vorgehensweise zur Disambiguierung linguistischer Kategorien und zur Satzendemarkierung*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 216–221, 1999.

[Hain00a] HAIN, HORST-UDO: *An automatically trainable multi-lingual system for grapheme-to-phoneme conversion for speech synthesis*. Workshop on Multi-Lingual Speech Communication, Seiten 122–127, 2000. ATR, Kyoto.

[Hain00b] HAIN, HORST-UDO: *Ein hybrider Ansatz zur Graphem-Phonem-Konvertierung unter Verwendung eines Lexikons und eines neuronalen Netzes*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 160–167, 2000.

[Hain00c] HAIN, HORST-UDO: *A hybride approach for grapheme-to-phoneme conversion based on a combination of partial string matching and a neural network*. Proc. ICSLP, III:291–294, 2000.

[Hain01] HAIN, HORST-UDO: *Verfahren zur Aufbereitung einer Datenbank für die automatische Sprachverarbeitung*. Patentschrift DE 199 42 178 C1, 2001.

[Hain03a] HAIN, HORST-UDO: *Graphem-Phonem-Konvertierung*. Patentschrift DE 100 42 944 C2, 2003.

[Hain03b] HAIN, HORST-UDO: *Verfahren zur Sprachsynthese*. Patentschrift DE 100 42 942 C2, 2003.

[Hain03c] HAIN, HORST-UDO: *Zuordnung von Phonemen zu den sie erzeugenden Graphemen*. Patentschrift DE 100 42 943 C2, 2003.

[Hayk99] HAYKIN, SIMON: *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[Hoff98] HOFFMANN, RÜDIGER: *Signalanalyse und -erkennung*. Springer Verlag, 1998.

[Holz00] HOLZAPFEL, MARTIN: *Konkatenative Sprachsynthese mit großen Datenbanken*. Doktorarbeit, Technische Universität Dresden, 2000.

[HS97] HOCHREITER, SEPP und JÜRGEN SCHMIDHUBER: *Flat Minima*. Neural Computation, 9:1–42, 1997.

[HVF03] HAIN, HORST-UDO, THOMAS VOLK und TIM FINGSCHEIDT: *Preprocessing and Prosody Generation for a TTS System with a Very Small Footprint*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 272–279, 2003.

[HZ01a] HAIN, HORST-UDO und HANS GEORG ZIMMERMANN: *A Multi-lingual System for the Determination of Phonetic Word Stress Using Soft Feature Selection by Neural Networks*. 4th ISCA Workshop on Speech Synthesis, 2001.

[HZ01b] HAIN, HORST-UDO und HANS GEORG ZIMMERMANN: *Optimierung der Eingabe eines neuronalen Netzes zur Bestimmung der Wortbetonung mit Hilfe von Weight Decay*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 282–289, 2001.

[Jess99] JESSEN, MICHAEL: *German*. In: HULST, HARRY VAN DER (Herausgeber): *Word Prosodic Systems in the Languages of Europe*, Seiten 515–545. Mouton de Gruyter, 1999.

[KH95] KROGH, ANDERS und JOHN A. HERTZ: *A Simple Weight Decay Can Improve Generalization*. In: MOODY, J. E., S. J. HANSON und R. P. LIPPMANN (Herausgeber): *Advances in Neural Information Processing Systems 4*, Seiten 950–957. Morgan Kauffmann Publishers, San Mateo, CA, 1995.

[Koho84] KOHONEN, TEUVO: *Self-Organization and Associative Memory*. Springer Verlag, 1984.

[Komm91] KOMMENDA, MARKUS: *Automatische Wortstrukturanalyse für die akustische Ausgabe von deutschem Text*. Doktorarbeit, Technische Universtät Wien, 1991.

[LK86] LAWRENCE, S. G. C. und G. KAYE: *Alignment of phonemes with their corresponding orthography*. Computer Speech and Language, 1:153–165, 1986.

[Meng99] MENGEL, ANDREAS: *Deutscher Wortakzent. Symbole, Signale*. Phorm Verlag, München, 1999.

[Mitt92] MITTEN, R.: *Computer-usable version of Oxford Advanced Learner's Dictionary of Current English*. Oxford Text Archive, 1992.

[Mül03] MÜLLER, ACHIM F.: *Generierung prosodischer Marker für ein multilinguales Sprachsynthesesystem*. Doktorarbeit, Technische Universität Dresden, 2003.

[NZ98] NEUNEIER, RALPH und HANS GEORG ZIMMERMANN: *How to Train Neural Networks*. In: ORR, GENEVIEVE B. und KLAUS-ROBERT MÜLLER (Herausgeber): *Neural Networks: Tricks of the Trade*. Springer Verlag, 1998.

[Pehl98] PEHL, ERICH: *Digitale und analoge Nachrichtenübertragung*. Hüthig Verlag Heidelberg, 1998.

[Quin93] QUINLAN, J. R.: *C4.5 Programs for Machine Learning*. CA Morgan Kaufman, San Mateo, 1993.

[RHW86] RUMMELHART, D. E., G. E. HINTON und R. J. WILLIAMS: *Learning Internal Representations by Error Propagation*. In: RUMMELHART, D. E. und J. L. MCCLELLAND (Herausgeber): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Band I, Seiten 318–362. MIT Press/Bradford Books, Cambridge, MA, 1986.

[RJ93] RABINER, LAWRENCE und BIING-HWANG JUANG: *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[RKV02] ROJC, MATEJ, ZDRAVKO KAČIČ und DARINKA VERDONIK: *Design and Implementation of the Slovenian Phonetic and Morphology Lexicons for the Use in Spoken Language Applications*. LREC, 2002.

[Rook87] ROOK, JÖRG: *Linguistisch-Phonetische Transkription auf der Basis von Graphem- und Phonemclustern und Ermittlung einer Wortbetonung für ein Sprachsynthese-System*. Doktorarbeit, Technische Universität Berlin, 1987.

[Rose96] ROSENKE, KATRIN: *Realisierung der linguistisch-phonetischen Transkription für die Sprachsynthese durch neuronale Netze mit Multilayer-Perceptron-Struktur*. Doktorarbeit, Technische Universität Berlin, 1996.

[RPJ01] RIIS, SØREN KAMARIC, MORTEN WITH PEDERSEN und KÅRE JEAN JENSEN: *Multilingual Text-to-Phoneme Mapping*. Eurospeech, Seiten 1441–1444, 2001.

[RSWH+99] ROJC, MATEJ, JANEZ STERGAR, RALPH WILHELM, HORST-UDO HAIN, MARTIN HOLZAPFEL und BOGOMIR HORVAT: *A Multilingual Text Processing Engine for the Papageno Text-to-Speech Synthesis System*. Proc. Eurospeech, 5:2107–2110, 1999.

[RZ94] REHKUGLER, HEINZ und HANS GEORG ZIMMERMANN: *Neuronale Netze in der Ökonomie*. Verlag Franz Vahlen München, 1994.

[Salz91] SALZBERG, S.: *A nearest hyperrectangle learning method*. Machine Learning, 6:277–309, 1991.

[Siem98] SIEMENS BUSINESS SERVICES: *Senn Version 3.0 Benutzerhandbuch*, 1998.

[SR86] SEJNOWSKI, TERRENCE J. und CHARLES R. ROSENBERG: *NETtalk: A Parallel Network that Learns to Read Aloud*. Technischer Bericht, The John Hopkins University Electrical Engineering and Computer Science, 1986.

[SR87] SEJNOWSKI, TERRENCE J. und CHARLES R. ROSENBERG: *Parallel Networks the Learn to Pronounce English Text*. Complex Systems, 1:145–168, 1987.

[Sün02] SÜNDERMANN, DAVID: *Development of a Tagger for the Text-To-Speech System Papageno*. Diplomarbeit, Technische Universität Dresden, Institut für Akustik und Sprachkommunikation, 2002.

[SWHK+00] STÖBER, KARLHEINZ, PETRA WAGNER, JÖRG HELBIG, STEFANIE KÖSTER, DAVID STALL, MATTHIAS THOMAE, JENS BLAUERT, WOLFGANG HESS, RÜDIGER HOFFMANN und HELMUT MANGOLD: *Speech Synthesis Using Multilevel Selection and Concatenation of Units from Large Speech Corpora*. In: WAHLSTER, WOLFGANG (Herausgeber): *Verbmobil: Foundations of Spech-to-Speech Translation*, Seiten 519–534. Springer Verlag, 2000.

[THNP+99] TRABER, CHRISTOF, KARL HUBER, KARIM NEDIR, BEAT PFISTER, ERIC KELLER und BRIGITTE ZELLNER: *From Multilingual to Polyglott Speech Synthesis*. Eurospeech, 2:835–838, 1999.

[Trab95] TRABER, CHRISTOF: *SVOX: The Implementation of a Text-to-Speech System for German*. Doktorarbeit, ETH Zürich, 1995.

[TV97] TRANCOSO, ISABEL und M. CÉU VIANA: *On the Pronunciation Mode of Acronyms in Several European Languages*. Eurospeech, 2:573–576, 1997.

[UZW98] UTE ZIEGENHAIN, STEFFEN HARENGEL, JANEZ KAISER und RALPH WILHELM: *Creating Large Pronunciation Lexica for Speech Applications*. First International Conference on Language Resources and Evaluation, 1998.

[vdB99] BOSCH, ANTAL VAN DEN: *Careful abstraction from instance families in memory-based language learning*. Journal of Experimental and Theoretical Artificial Intelligence, 1999.

[vdBD93] BOSCH, ANTAL VAN DEN und WALTER DAELEMANS: *Data-Oriented Methods for Grapheme-to-Phoneme Conversion*. 6th Conference of the European Chapter of ACL, 1993.

[VPB98] VINCENT PAGEL, KEVIN LENZO und ALAN W. BLACK: *Letter to Sound Rules for Accented Lexicon Compression*. Proc. ICSLP, 5:2015–2018, 1998.

[Wagn01] WAGNER, PETRA: *Systematische Überprüfung deutscher Wortbetonungsregeln*. Konferenz Elektronische Sprachsignalverarbeitung, Seiten 329–338, 2001.

[Weid98] WEIDE, R. L.: *Carnegie Mellon Pronouncing Dictionary releae 0.6*. www.cs.cmu.edu, 1998.

[Woth93] WOTHKE, KLAUS: *Morphologically based automatic phonetic transcription*. IBM SYSTEMS JOURNAL, 32(3):486–511, 1993.

[Zell94] ZELL, ANDREAS: *Simulation Neuronaler Netze*. Addison-Wesley, 1994.