Buchstabiererkennung mit neuronalen Netzen in Auskunftssystemen

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
der Fakultät für Informatik
der Universität Karlsruhe (Technische Hochschule)
genehmigte

Dissertation

von
Hermann Hild
aus Marbach a.N.

Tag der mündlichen Prüfung: **28. Mai 1997**

Erster Gutachter: **Prof. Dr. Alexander Waibel**
Zweiter Gutachter: **Prof. Dr. Rüdiger Dillmann**

Berichte aus der Informatik

Hermann Hild

Buchstabiererkennung mit neuronalen Netzen in Auskunftssystemen

Shaker Verlag
Aachen 1997

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Hild, Hermann:
Buchstabiererkennung mit neuronalen Netzen in Auskunftssystemen /
Hermann Hild. - Als Ms. gedr. -
Aachen : Shaker, 1997
(Berichte aus der Informatik)
Zugl.: Karlsruhe, Univ., Diss., 1997
ISBN 3-8265-3155-8

Als Manuskript gedruckt. Printed in Germany.

ISBN 3-8265-3155-8
ISSN 0945-0807

Shaker Verlag GmbH • Postfach 1290 • 52013 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • eMail: info@shaker.de

Schlagwörter

Spracherkennung, Buchstabiererkennung, Buchstabieren, neuronale Netze, TDNN, MS-TDNN, Sprachmodelle, Suche, Auskunftssysteme, spontane Sprache

Kurzfassung

Die Erkennung buchstabierter Wörter ist eine Teilaufgabe der automatischen Spracherkennung. Sie kommt besonders bei Anwendungen zum Tragen, die eine Erkennung beliebiger Namen oder Adressen erfordern, oder kann in interaktiven Erkennern zur Korrektur falsch erkannter oder zur Eingabe „neuer Wörter" eingesetzt werden. Während heute der Wortschatz von Spracherkennern bereits mehrere 10 000 Wörter umfassen kann, liegt bei der Buchstabiererkennung die Herausforderung nicht im Umfang, sondern in der hochgradigen Verwechselbarkeit der einzelnen Wörter des Vokabulars: Die 26 Buchstaben des deutschen oder englischen Alphabets sind selbst für den menschlichen Hörer bisweilen schwer zu unterscheiden (etwa {T,D,B,P,G ...} oder {N,M}).
Im Rahmen der Promotion wurde ein Buchstabiererkenner entwickelt und evaluiert. Durch Verbesserungen der Architektur und der Trainingsverfahren für die akustische Modellierung sowie durch Sprachmodelle zur Unterstützung der Hypothesensuche konnten die derzeit besten Erkennungsraten erzielt werden. Weiterhin werden erstmals Buchstabiereffekte im Kontext fließender Sprache untersucht. Schwerpunkte der Arbeit sind:

Entwicklung und Evaluierung eines Spracherkenners für Buchstabensequenzen. Der Erkenner basiert auf einer Weiterentwicklung des „Time-Delay Neural Network" (TDNN), dem sogenannten „Multi-State TDNN". Das TDNN liefert Phonemhypothesen für jeden der kurzen Zeitabschnitte, in die das Eingabewort unterteilt wird. Aus den Phonemen werden Modelle für die zu erkennenden Buchstaben gebildet, die wie die einzelnen Phoneme diskriminativ trainiert werden können. Die Buchstaben werden nicht isoliert, sondern als kontinuierlich gesprochene Sequenzen sprecherunabhängig klassifiziert.
Konventionelle statistische **Sprachmodelle** berechnen die Wahrscheinlichkeit für ein zu erkennendes Wort aus dem Kontext der letzten ein oder zwei erkannten Wörter. Für Buchstabieranwendungen wie beispielsweise eine automatische Telefonauskunft ist es darüberhinaus sinnvoll, die Erkennung auf eine vorgegebene Wort-/Namensliste mit einigen hunderttausend Einträgen einzuschränken. Verschiedene Sprachmodellierungs- und Suchverfahren wurden entwickelt und evaluiert, um dieses zusätzliche Wissen für eine möglichst optimale und effiziente Erkennung auszunutzen.
Bei der **Buchstabiererkennung in spontaner Sprache** wird ein Sprecher nicht ausschließlich auf Buchstaben festgelegt. Vielmehr geschieht das Buchstabieren spontan, freisprachlich. Es werden Situationen untersucht, in denen ein Name nicht nur buchstabiert, sondern auch fließend gesprochen wird, Buchstabensequenzen in umfangreichere Äußerungen eingebettet sind („`Die Nummer von Hild bitte, H I L D ...`") oder komplexere Formen annehmen, wie „`D wie Dora`" oder „`Alpha Bravo Charlie`".

Danksagung

Diese Arbeit entstand am Institut für Logik, Komplexität und Deduktionssysteme in der *Interactive Systems Laboratory* Gruppe. Deren Leiter Prof. Dr. Alex Waibel danke ich herzlich für die Betreuung dieser Arbeit. Durch sein unermüdliches, weltweites Engagement eröffnete er mir die Gelegenheit, in einer mit internationalen Kontakten und Geräten erstklassig ausgestatteten Forschungsumgebung in Karlsruhe und an der Carnegie Mellon University (CMU) in Pittsburgh arbeiten zu können. Prof. Waibel war eine beständige Quelle guter Ideen und ehrgeiziger Ziele; er verstand es immer wieder, uns aus nah und fern mit viel guter Laune und Humor zu neuen Leistungen anzuspornen. Ebenfalls zu Dank verpflichtet bin ich Herrn Prof. Dr. Dillmann, der das Korreferat zu dieser Arbeit übernommen hat.

Teile dieser Arbeit wurden von der Siemens AG und im Rahmen des BMBF-Projektes VERBMOBIL gefördert. Ich danke unseren Kooperationspartnern für ihre finanzielle und freundliche fachliche Unterstützung.

Erfolgreiches Arbeiten im komplexen, rechen- und datenintensiven Feld der Spracherkennung erfordert vielfältige Interaktionen aller Beteiligten. Für eine gute und freundliche Zusammenarbeit möchte ich mich bei meinen jetzigen und ehemaligen Kollegen in Karlsruhe und an der CMU herzlich bedanken, darunter: Markus Baur, Uli Bodenhausen, Christoph Bregler, Finn-Dag Buø, Noah Coccaro, Matthias Denecke, Paul Duchnowski, Michael Finke, Jürgen Fritsch, Petra Geutner, Ricky Houghton, Susanne Kaufmann, Thomas Kemp, Detlef Koll, Stefan Manke, Arthur McNair, Uwe Meier, Wolfgang Minker, Klaus Ries, Ivica Rogina, Deb Roy, Tanja Schultz, Tilo Sloboda, Rainer Stiefelhagen, Bernhard Suhm, Joe Tebelskis, Minh Tue Vo, Ye-Yi Wang, Martin Westphal, Christoph Windheuser, Cindy Wood, Monika Woszczyna, Klaus Zechner und Torsten Zeppenfeld. Besonders gewinnbringend war mir die in Wort und Tat erfahrene fachliche und persönliche Unterstützung von Joe Tebelskis, Thomas Kemp, Monika Woszczyna, Ivica Rogina, Michael Finke und meinem langjährigen Zimmerkollegen Martin Westphal.

Auch die fleißigen Hände zahlreicher wissenschaftlicher Hilfskräfte trugen zum Gelingen dieser Arbeit bei. Ein herzliches Dankeschön an Christiane Reihl, Thomas Schaaf, Peter Scheytt, Gudrun Socher und Henrike Tries für ihr Engagement beim Sammeln von mehreren zehntausend gesprochenen Buchstaben, sowie an Martin Betz und Michael Meyer, deren Diplomarbeiten ebenfalls wichtige Beiträge geleistet haben.

Das seit der Gründung des Lehrstuhls 1991 stetig wachsende Netz aus inzwischen über 40 Rechnern und 100 Gigabyte Daten wurde und wird durch die intensive Betreuung von Stefan Manke, Markus Baur und nun Frank Dreilich und Martin Klein am Leben gehalten. Für einen reibungslosen organisatorischen Ablauf garantierten im Sekretariat immer freundlich und hilfsbereit Silke Dannenmaier, Ingrid Gemen, Evelyn Kimmich und Sonja Seitz.

Eine wertvolle Erfahrung waren mir die Aufenthalte an der CMU in Pittsburgh, wo ich überall sofort sehr freundlich aufgenommen wurde. Besondere Gastfreundschaft habe ich

bei Torsten und Ivica genossen, die mir mehrfach für längere Zeit Unterkunft gewährten, sowie bei Joe, der mir vieles aus der kulturellen Vielfalt Amerikas näherbrachte.

Mein besonderer Dank gilt Ivica Rogina, Tanja Schultz, Martin Westphal, Matthias Back und Jutta Klein, die die Fertigstellung dieser Ausarbeitung mit emsigen Rotstiften und konstruktiven Anregungen begleiteten.

Eine große Hilfe war mir die liebevolle Unterstützung von Jutta, die mir stets fürsorglich, fröhlich und hilfsbereit mit Rat und Tat zur Seite stand und mir geduldig die vielen langen Abende und Wochenenden nachsah, die ich nicht ihr, sondern dieser Arbeit zugute kommen ließ.

Danken möchte ich auch meiner Familie und meinen Verwandten, die ebenfalls mit viel Anteilnahme und Einsatz für mein Wohl sorgten. Ich widme diese Arbeit meinen längsten Lebensbegleitern, meinen Eltern Maja und Walter Hild, deren warmherzige, verantwortungsvolle und fröhliche Menschlichkeit mir von Anfang an einen guten Weg wies.

Karlsruhe, im April 1997 Hermann Hild

Inhaltsverzeichnis

Kapitel 1

Einführung

In einer rasanten technologischen Entwicklung haben sich Rechen- und Speicherleistung von Computern im Laufe der letzten zehn Jahre vertausendfacht. Aus einfachen Text-Bildschirmen wurden graphische Benutzeroberflächen, die im beginnenden Multi-Media-Zeitalter mit Ausgabemöglichkeiten für HiFi-Stereo, Videosequenzen und Sprachsynthese ergänzt werden. Auf der Seite der Eingabe hat sich die Kommunikationsbandbreite, außer mit der Einführung der Maus, nicht erhöht. Erkennung von Sprache, Interpretation von Bildern und Gesten – kognitive Fähigkeiten, über die der Mensch scheinbar mühelos verfügt – haben sich als ein anhaltend schwieriges Problem herausgestellt. Viele der vormaligen Prophezeiungen über den Fortschritt der künstlichen Intelligenz haben sich bis jetzt als zu optimistisch erwiesen.
Die Geschichte der Spracherkennung beginnt in den fünfziger Jahren mit der Erkennung isolierter Ziffern. Die weitere Entwicklung hat große Fortschritte erbracht, insbesondere markiert durch die Einführung der „Hidden Markov Models" (ab 1975) und die Modellierung kontextabhängiger Wortuntereinheiten. Inzwischen können kontinuierlich gesprochene Texte mit einem Vokabular von über 20 000 Wörtern sprecherunabhängig erkannt werden. Dennoch muß eine der menschlichen Leistung vergleichbare „universelle Spracherkennung", also die Erkennung beliebiger Texte auch unter erschwerten Bedingungen, weiterhin als offenes Forschungsproblem angesehen werden.

Heute wird an vielen Universitäten und von den meisten großen, in der Informationstechnologie tätigen Firmen im Bereich der Spracherkennung geforscht. Allein in Deutschland wird die Sprachtechnologie im Rahmen des BMBF[1]-Projektes VERBMOBIL mit etwa 100 Millionen Mark über acht Jahre gefördert. Potentielle Anwendungen beinhalten automatische Auskunftssysteme (Zug-, Flugzeugauskunft, Telefonauskunft, ...), Diktiersysteme sowie Steuerungsaufgaben („hands-busy" applications). Demonstrationssysteme dazu werden in Forschungslabors in aller Welt präsentiert, jedoch sind bisher nur wenige davon in die Praxis gelangt, da dort wesentlich höhere Anforderungen an die Robustheit eines Systems gestellt werden müssen.

[1] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie

Der Schwerpunkt dieser Arbeit liegt auf der Erkennung buchstabierter Wörter, einer Teilaufgabe der automatischen Spracherkennung. Buchstabiert wird ein Wort dann, wenn dieses in „normaler Sprache" schwierig zu kommunizieren ist: Sowohl zwischen Menschen, insbesondere aber im Mensch-Maschine-Dialog kann nicht immer von der Aussprache eines Wortes auf seine Schreibweise oder gar Identität geschlossen werden. Probleme dieser Art ergeben sich besonders bei Anwendungen, die eine Erkennung beliebiger Namen oder Adressen erfordern. Heutige Spracherkenner können große Vokabulare mit einem Umfang von mehreren zehntausend Wörtern erkennen. Erkennungsvokabulare von Millionen von Eigennamen (Vor-, Nach-, Städte-, Straßen-, Hotelnamen) sind jedoch derzeit nicht realisierbar, da so viele Namen zum einen zu ressourcenintensiv und zum anderen nicht mehr ausreichend zu diskriminieren sind. In solchen Situationen bietet sich das Buchstabieren an.

Mit nur 26 Buchstaben liegt bei der Buchstabiererkennung die Herausforderung nicht im Umfang, sondern in der hochgradigen Verwechselbarkeit der einzelnen Wörter des Vokabulars. Dies gilt insbesondere für Buchstabengruppen wie {T,D,B,P,G,...}, {M,N} oder bei Rauschen (Telefon, Auto) auch {F,S}, die selbst für den menschlichen Hörer bisweilen schwer zu unterscheiden sind.

Neben Eigennamen sind *Klärungsdialoge* ein weiteres wichtiges, aber bisher wenig erforschtes Anwendungsfeld; denn im Unterschied zum Labor müssen in der Praxis die unvermeidlichen Erkennungsfehler nicht nur gemessen, sondern auch korrigiert werden. Hier bietet sich ebenfalls Buchstabieren an, um eine robuste Korrektur falsch erkannter Wörter zu ermöglichen oder dem System bisher unbekannte Wörter einzugeben.

Diese Arbeit befaßt sich mit mehreren Aspekten der Buchstabiererkennung. Zunächst wird ein konnektionistischer Erkenner beschrieben, der Buchstabensequenzen sprecherunabhängig erkennen kann. Dabei kann fließend, also ohne künstliche Pausen buchstabiert werden. Mit Buchstabenerkennungsraten von bis zu 99% auf sprecherabhängigen und 90% auf sprecherunabhängigen Daten wurden erstklassige Ergebnisse erzielt und alle Vergleichssysteme übertroffen. Dennoch wird auch bei einer 90% korrekten Buchstabenerkennung mit einer Wahrscheinlichkeit von etwa 50% ein Fehler in einem 5 bis 6 Buchstaben langen Nachnamen auftreten. Erlaubt man nur die Erkennung „legaler" Namen, können durch geschicktes Ausnutzen dieses Wissens über das Telefon buchstabierte Namen selbst dann noch in 90% der Fälle korrekt erkannt werden, wenn Telefonbücher mit einer Million Einträgen zugrunde gelegt werden.
Ein zu buchstabierendes Wort wird nicht immer Buchstabe für Buchstabe gesprochen. In spontaner Sprache kann erschwerend hinzukommen, daß einzelne oder alle Buchstaben eingeleitet, korrigiert, kommentiert, umschrieben oder spezifiziert werden. Einige dieser Phänomene sind im letzten Abschnitt der Arbeit untersucht.

1.1 Anwendungen der Buchstabiererkennung

Die Buchstabiererkennung bietet sich an für Eigennamen (Nach-, Vor-, Straßen-, Orts-, Hotelnamen), „normale" lexikalische Wörter oder beliebige Buchstabensequenzen, wie aus den folgenden Beispielen ersichtlich wird:

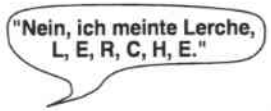
Telefonauskunft. Die Deutsche Telekom nutzt bereits Sprachsynthese zur Ausgabe erfragter Telefonnummern. Für eine vollautomatische Auskunft muß auch der Name des gewünschten Anschlusses maschinell erkannt werden. Die potentiellen Einsparmöglichkeiten sind gewaltig. Die Telefongesellschaften in den USA wickeln jedes Jahr über 6 Milliarden Auskünfte ab [KSS95], bei einem Kostenaufwand von über 1.5 Milliarden Dollar [LBM95]. Bei einer durchschnittlichen Gesprächslänge von etwa 25 Sekunden kann der Wert jeder eingesparten Sekunde auf ca. 60 Millionen Dollar pro Jahr geschätzt werden [LBM95].

Hilbrath Robert Wesumark-12	49 38 52
Hilbrecht Edwin Ried-32	89 06 11
Hild Alain	81 28 83
-Claus W. Dr. Numismatik Rintheimer-2	69 84 76
-Dieter Schneidemühler-27a	67 01 94
-Edeltraud Rudolf-24	69 63 92
-Hermann Passagehof 22	2 65 90
-Hugh Reuthener-10	67 98 42

Adresseingabe. Marktreife Fahrzeugnavigationssysteme können bereits heute Fahrtrouten zu einem gewünschten Ziel vorschlagen. Zur Eingabe der Zieladresse muß bisher mühsam Buchstabe für Buchstabe mit Cursortasten angesteuert und eingegeben werden. Eine Spracheingabe, beispielsweise durch Buchstabieren der Adresse, bietet hier eine schnellere und komfortablere Lösung an.

Auto- oder Bestellnummern erfordern die Erkennung **beliebiger Sequenzen**.

Klärungsdialoge erlauben einen benutzerfreundlichen Umgang mit Erkennungsfehlern. Durch Buchstabieren lassen sich falsch erkannte Wörter ganz ohne Maus und Tastatur korrigieren oder bisher unbekannte Wörter definieren.



Ganz allgemein ist Buchstabieren sinnvoll für alle Anwendungen, bei denen **beliebige Namen oder Adressen** vorkommen.

1.2 Warum ist Buchstabiererkennung schwierig?

Verglichen mit anderen Spracherkennungsaufgaben stellt das Alphabet einen sehr kleinen Wortschatz dar. Dafür sind die einzelnen Buchstaben sehr kurz und hochgradig verwechselbar. Im wesentlichen repräsentiert jeder Buchstabe ein Phonem, nämlich seinen Laut im gesprochenen Wort. Die Vokale unter den Buchstaben bestehen nur aus diesem Phonem, bei den „Kon-Sonanten" klingt der besseren Aussprechbarkeit wegen ein zusätzlicher Vokallaut mit. Beispielsweise sind die Buchstaben {B,C,D,G,P,T,W} und {F,S,L,M,N} um den Vokal E erweiterte Phoneme. Die daraus entstehende akustische Verwechselbarkeit ist auch optisch erkennbar: In Abbildung 1.1 sind zwei klangähnliche Buchstabengruppen einem Paar „normaler" Wörter gegenübergestellt. Auch der Mensch hat bisweilen Schwierigkeiten mit der Unterscheidung mancher Buchstaben. Als Abhilfe werden oft Aussprachen mit höherer Redundanz benutzt, wie etwa das Funkeralphabet oder Konstrukte wie „`A wie Auto`". Menschliche Buchstabier-Erkennungsleistungen werden in [Dal87] mit 98.4% (sprecherunabhängig, kontinuierlich buchstabiert, in hoher Qualität im Tonstudio aufgenommen) und mit 93% bei Telefonqualität (einzeln präsentierte Buchstaben, aufgenommen von 100 Sprechern über verschiedene Telefonleitungen) [FCR92] angegeben.

Quantisierung der phonetischen Verwechselbarkeit

Natürlich gibt es auch in einem „normalen" Vokabular Wörter, die sich, wie die meisten der Buchstaben, nur in einem Phonem unterscheiden, etwa „Hut", „gut" und „Mut". Um diese Art der Verwechselbarkeit besser zu quantifizieren, kann man folgendes Experiment durchführen. Zu einem gegebenen Vokabular $V = \{w_1, w_2, \ldots, w_N\}$ bestimmt man für jedes Wort w_i die Anzahl k_i derjenigen Wörter $w_j, j \neq i$, die sich von w_i nur in einem Phonem unterscheiden[2]. Neben der *durchschnittlichen* 1-Phonem-Verwechselbarkeit C_D ist es sinnvoll, eine *zu erwartende* Verwechselbarkeit C_E zu bestimmen, die den unterschiedlichen Auftrittshäufigkeiten $p(w_i)$ der Wörter Rechnung trägt.

$$C_D := \frac{1}{N}\sum_{i=1}^{N} k_i \qquad \text{(Durchschnittliche 1-Phonem-Verwechselbarkeit)}$$

$$C_E := \sum_{i=1}^{N} p(w_i) * k_i \qquad \text{(Erwartungswert für 1-Phonem-Verwechselbarkeit)}$$

Natürlich sind die hier gemessenen orthographischen 1-Phonem-Abstände nur eine grobe Maßzahl für die Verwechselbarkeit des Vokabulars. Ein genaueres Maß müßte auch noch die unterschiedliche Verwechselbarkeit der jeweils verglichenen Phonempaare berücksichtigen.

Als Beispiel für einen großen Wortschatz wurden zwei Wörterbücher mit 5 000 und 20 000 Einträgen aus der „Wall Street Journal" (WSJ)-Task herangezogen. Zusätzlich

[2] Gemessen nach der *Editierdistanz*, siehe Abschnitt 3.5.2.

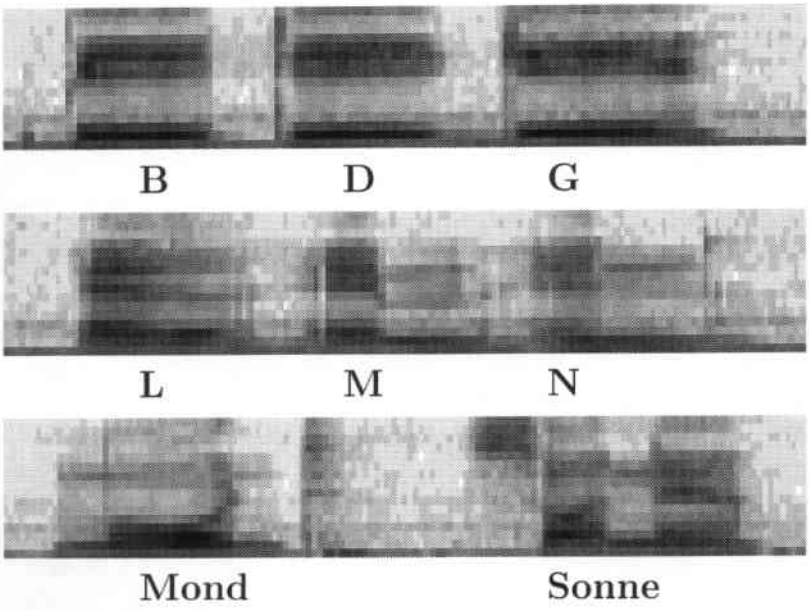


Abbildung 1.1: Die Spektrogramme der Buchstaben B,D,G oder L,M,N sind sich sehr ähnlich, im Gegensatz zu zwei zufälligen Wörtern „Mond" und „Sonne" im unteren Spektrogramm

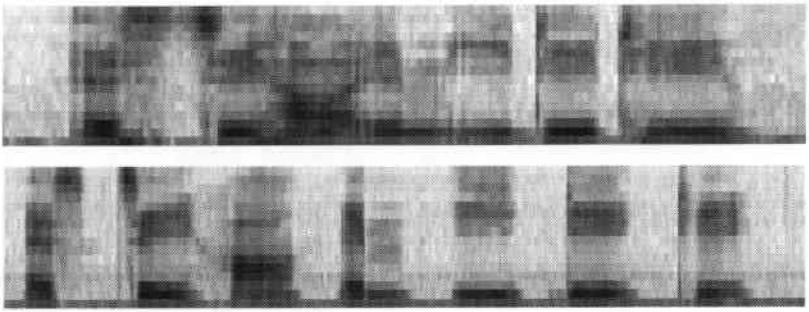


Abbildung 1.2: „S C H M I D T" kontinuierlich (oben) und mit Pausen (unten) buchstabiert

wurden mit den häufigsten 1000, 500 und 100 Wörtern drei weitere Vokabulare definiert. Die Ergebnisse der 1-Phonem-Verwechselbarkeit sind in Abbildung 1.3 als Tabelle und graphisch zusammengefaßt. Im Alphabet gibt es zu jedem Buchstaben durchschnittlich 4 weitere, die sich in nur einem Phonem unterscheiden – erst bei einer Größe von 5 000 bzw. 20 000 Wörtern wird im WSJ-Vokabular diese erwartete oder durchschnittliche Verwechselbarkeit erreicht! Andererseits ersieht man aus der Tabelle, daß Ziffernerkennung eine relativ einfache Aufgabe ist.

Vokabular	C_D	C_E
Deutsches Alphabet, ohne Umlaute und ß	4.4	5.8
Englisches Alphabet	4.5	4.9
Ziffern 0-9	0.0	0.0
20 000 Wörter WSJ	4.6	14.0
5 000	2.3	6.1
1 000	2.1	3.8
500	2.2	3.3
100	1.4	1.7

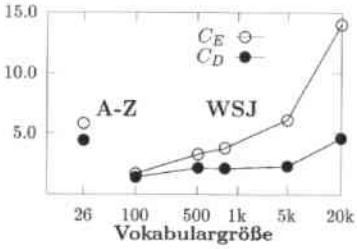


Abbildung 1.3: Verwechselbarkeit von Vokabularen gemessen an der Anzahl von Wörtern, die sich von anderen in nur einem Phonem unterscheiden

Fließende Buchstabierungen

Aus orthographischer Sicht ist ein buchstabiertes Wort gewissermaßen die „diskontinuierliche" Version des gesprochenen Wortes. Daher wird häufig fälschlicherweise vermutet, Buchstabiererkennung sei keine kontinuierliche Erkennung. Zwischen zwei gesprochenen Buchstaben ist jedoch nicht notwendigerweise eine Pause, sie können genauso fließend ineinander übergehen wie „normale" Sprache, wodurch Koartikulationseffekte und gelegentlich sogar Mehrdeutigkeiten („H A" versus „A J" im Englischen) auftreten können. Im Gegensatz zu einigen anderen Buchstabiererkennern ist in unserem System die Erkennung kontinuierlicher Buchstabierungen explizit vorgesehen. Abbildung 1.2 zeigt den Unterschied von Buchstabieren mit und ohne Pausen am Beispiel des Namens „S C H M I D T".

1.3 Eine Taxonomie von Buchstabiereffekten

Was wird buchstabiert, und **warum**? Wie genau wird dabei ein Buchstabe **spezifiziert**, in welchem **Umfang** und **Kontext** wird ein Wort buchstabiert? Wir versuchen, typische Buchstabierphänomene anhand dieser Fragen zu kategorisieren:

WAS wird buchstabiert?

- **Wörter.** Es kann unterschieden werden zwischen: **Eigennamen**, wie Vor- und Nachnamen oder Adressen (Straßen, Städte, Hotelnamen), und **„regulären" lexikalischen Wörtern**.
- **Akronyme** kürzen längere Ausdrücke ab: IBM, HP, ZDF, ICASSP, USA. Manche (USA, ZDF, CD) sind weit verbreitet, andere nur in Fachkreisen bekannt (REM, DTW). Die gängigeren Akronyme werden vermutlich gar nicht mehr als Buchstabensequenzen, sondern als eigenständige Wörter wahrgenommen, und deshalb nicht bewußt, also weniger artikuliert buchstabiert. Manche Akronyme werden gar nicht buchstabiert, sondern wie ein reguläres Wort fließend ausgesprochen, etwa „AGFA".
- Buchstaben kommen bei **Aufzählungen** zum Einsatz („`Appendix B`", „`Halle C`") und dienen zur **Identifikation** von Nummernschildern (`KA - HX 762`) oder Bestellnummern.

WARUM wird buchstabiert? Akronyme verkürzen oder haben sich zu Wörtern verselbständigt. Wörter werden buchstabiert, um **Homophone**, also gleichklingende Wörter zu disambiguieren. Viele lexikalische Wörter („`Fälle, Felle`") und insbesondere Eigennamen wie „`Maier, Mayr, Meier, ...`" oder „`Schmid, Schmitt, Schmidt, ...`" können nur durch explizite Benennung der Schreibweise unterschieden werden. Oft ist die **Orthographie unbekannt** und kann nicht aus dem Klang erschlossen werden. Das kann für lexikalische Wörter der eigenen Muttersprache gelten („`Wie schreibt man Rhythmus?`"), ist aber besonders bei Eigennamen aus weniger vertrauten Kulturkreisen ein Problem.

SPEZIFIKATION. Wie werden Buchstaben(gruppen) spezifiziert? Zusätzliche Erläuterungen erhöhen die Redundanz und damit die Robustheit bei der Übermittlung der leicht verwechselbaren Buchstaben. In sicherheitskritischen Situationen wie im Luftverkehr werden einzelne Buchstaben durch ganze Wörter aus dem Funkeralphabet ersetzt.

UMFANG. Welche Teile eines Wortes werden spezifiziert? Nicht immer wird jeder Buchstabe genau einmal genannt: Wiederholungen oder Beschränkungen auf kritische Segmente sind typisch („`Maier mit A I`").

In welchem **KONTEXT** wird buchstabiert? Wird ein Benutzer in einem starren Dialog aufgefordert, einen Namen zu buchstabieren, kann man erwarten, daß er nur Buchstaben benutzt. In einer weniger zielgerichteten Situation muß man damit rechnen, daß natürlichsprachliche Konstrukte verwandt werden, die entweder gar nichts mit der Buchstabierung zu tun haben oder diese einleiten, kommentieren oder korrigieren.

Einige Beispiele der vielfältigen Spielarten des Buchstabierens sind in Tabelle 1.1 zusammengefaßt.

Spezifikation	– pures Buchstabieren	*"M A I E R"*
	– Funkeralphabet	*"Anton Berta Emil Nordpol ... "*
	– Vergleich	*"A wie [in] Apfel "*
	– Assoziation	*"Möhre wie Karotte"*
Umfang	– vollständig, jeder Buchstabe	*"M A I E R"*
	– nur der fließende Name	*"Maier"*
	– unvollständig buchstabiert	*"Maier mit A I, nicht E I"*
		"Schmidt mit D T"
	– Mehrfachspezifikationen	*"Q U E L L, mit L L am Schluß ..."*
Kontext	– Nur Buchstabenspezifikationen	*"H I L D und D wie in ... "*
	– Eingebettete Buchstabierungen	*"Hier spricht Mai, M A I, wie April"*
	– Füllwörter	*" ... und dann L"*

Tabelle 1.1: Beispiele für Buchstabierungen

1.4 Entstehungsgeschichte des Alphabets

Wie sind die – für die Spracherkennung ungünstigerweise so ähnlich klingenden – Buchstaben entstanden? Die Essenz der Schriftgeschichte ist eine stufenweise Annäherung der Schriftzeichen an die Lautstruktur des gesprochenen Wortes [Haa90](S. 17). Die ersten Bilderzählungen in Höhlenmalereien fixieren Konzepte oder Gedankensequenzen, ohne daß es dazu ein direktes sprachliches Korrelat gibt. Die *Logographie* („Gedanken schreiben") orientiert sich am Inhalt der wiederzugebenden Worte. Diese Schreibweise repräsentiert die älteste Evolutionsphase der Schriftgeschichte und hat sich bis heute in der chinesischen Sprachgemeinschaft konserviert. Aus der inhaltsbezogenen Schreibweise hat sich die *Phonographie* („Ton, Laut" + „schreiben") herausgebildet, wobei nacheinander zuerst Lautsegmente, dann Silben- und schließlich Buchstabenschriften entstanden.

Die Vorgeschichte der Schrift beginnt vor etwa 50 000 Jahren in der jüngeren Altsteinzeit mit den Höhlenmalereien etwa in Lascaux (Frankreich) oder Altamira (Spanien). Es gibt Hinweise darauf, daß die ersten menschlichen Fingerritzungen den Kratzspuren von Bären nachgeahmt sind. Man nimmt an, daß die zum Teil erstaunlich kunstvollen Abbildungen und Malereien nicht der Fixierung von Information, sondern kultischen Handlungen dienten.

Nach neueren Erkenntnissen [Haa90] sind die ältesten bekannten Schriftsysteme nicht, wie lange Zeit angenommen, im Kulturkreis der Sumerer in Mesopotamien, sondern bereits etwa 2000 Jahre früher in „Alteuropa" anzusiedeln. Die Anfänge dieser *Vinča*-Zivilisation[3] reichen bis in das 7. Jahrtausend v. Chr. zurück. Man geht davon aus, daß es sich bei den etwa 200 Symbolen der Schrift um eine religiös motivierte Sakralschrift handelt.
Im Gegensatz dazu ist die Schriftentwicklung bei den Sumerern durch verwaltungstechnische Notwendigkeiten begründet. Im Laufe von etwa 2000 Jahren werden die altsu-

[3]Nach einem Fundort in der Nähe von Belgrad benannt

merischen Bildzeichen immer weiter stilisiert, bis schließlich die Keilschrift entsteht. Gleichzeitig reduziert sich die Anzahl der verwendeten Schriftzeichen, und es tritt eine Phonetisierung der Schrift ein, d.h. die Schriftzeichen geben Grundelemente der numerischen Lautstruktur wieder. Die Idee des Schreibens wird als Kulturgut von Mesopotamien nach Ägypten importiert. In den ägyptischen Hieroglyphen bleibt die Bildstruktur der Schriftsymbole wesentlich expliziter, jedoch erfolgt eine Phonetisierung der hieroglyphischen Symbole früher als bei den Sumerern. Diese beginnt mit der Doppelbelegung bestimmter Symbole, wenn es zum dargestellten Wort eine gleichlautende zweite Bedeutung gibt. Zur Unterscheidung werden die Symbole mit Deutezeichen (*Determinative*) markiert. Später übernehmen die Bildsymbole die Funktion einzelner Silben und sogar Phoneme.

Die chinesische Schrift ist eine eigenständige Entwicklung und ist auch heute noch eine reine Begriffsschrift mit etwa 40 000 verschiedenen Zeichen. Die Japaner haben im 4. und 5. Jh. n. Chr. die chinesischen Symbole übernommen und benutzen heute knapp zweitausend chinesische Wortzeichen. Ein Teil davon wurde zu einem selbständigen System, bestehend aus etwa fünfzig sogenannten *Kana*-Silbenzeichen (*Hiragana* und *Katakana*), vereinfacht und phonetisiert.

Die erste reine Buchstabenschrift geht auf die Phönizier zurück, die frühsten Schriftzeugnisse datieren etwa in das 17. Jh. v. Chr. Über ihren genauen Ursprung wird spekuliert. Sie stammen wohl aus eigenen Quellen und sind lediglich in Anlehnung, aber nicht in Abhängigkeit von den ägyptischen Hieroglyphen entstanden. Einige ägyptische Hieroglyphen sind in Abbildung 1.4 der nordsemitisch-phönizischen und der jüngeren, nach ihrem Fundort benannten „Sinai"-Schrift gegenübergestellt.

Die frühen *semitischen Buchstabensysteme* bestehen nur aus Konsonanten und dienen als Vorbild für alle späteren Alphabetentwicklungen. Im Osten entstehen die indischen Schriften, im Norden das griechische Alphabet, dessen Buchstaben Herodot nach ihrer Herkunft „Phoinikéia grámmata" nennt. Die Griechen passen das Schriftsystem einer fremden, semitischen Sprache den Bedürfnissen ihrer eignen, indogermanischen Sprache an. Die lautlich stark unterschiedliche phönizische Sprache enthält dem Griechischen unbekannte Halbkonsonanten, die von den Griechen mit Vokallauten besetzt werden; beispielsweise wird aus dem semitischen *Aleph* das griechische *Alpha*. Damit entsteht etwa im 9. Jh. v. Chr. das erste vollständige Alphabet der Welt, in dem konsequent alle Konsonanten und Vokale wiedergegeben werden. In der ersten Schriftreform auf europäischem Boden werden im Jahre 403 v. Chr. die Schreibweisen vereinheitlicht und damit die 24 Zeichen des klassischen griechischen Alphabets festgelegt.

Das archaische lateinische Alphabet wird den Römern nicht direkt von den Griechen, sondern über die Etrusker vermittelt. Drei Zeichen (Θ, χ, Φ für die Laute th, kh, ph) werden von vornherein nicht übernommen, der Laut dz (Z) steht jahrhundertelang ungenutzt zwischen F und H, bis er durch das neue Zeichen G ersetzt wird. Bis dahin werden die Laute k und g mit demselben Buchstaben (C) geschrieben – ein freigelassener Sklave und Lehrer soll dem C einen Strich hinzugefügt und damit das uns bekannte

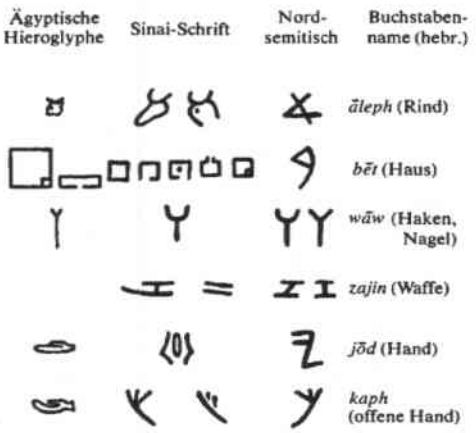


Abbildung 1.4: Vergleichende Tabelle einiger Zeichen der „Sinai"-Schrift mit den entsprechenden ägyptischen Hieroglyphen und den nordsemitischen Konsonanten, aus [FP87]

Zeichen G geschaffen haben [Haa90](S. 296). Das griechische Y wird von den Etruskern für U und dann von den Römern für U und V eingesetzt. Die militärische Angliederung des griechischen Mutterlandes im Jahre 146 v. Chr. an das Römische Reich weitet den griechischen Kultureinfluß aus. Damit verbunden ist die Übernahme vieler griechischer Lehnwörter. Um diese adäquat schreiben zu können, werden später Y und Z direkt aus dem Griechischen entlehnt und hinter den bisher letzten Buchstaben X eingereiht. Bis ins Mittelalter werden V und U gleichwertig benutzt. Ab dem 10. Jh. gebraucht man V als Großbuchstaben und U als Kleinbuchstaben, erst später wird V zum Konsonanten und U zum Vokal. Ähnlich ist J zuerst nur ein großes I und wird dann eigenständiger Konsonant. Das Zeichen W ersetzt das bis dahin gebräuchliche doppelte U für den Laut w (englisch auch heute noch *double-U*) [JMMR85].

Die moderne Aussprache unseres Alphabets (Ah, Beh, Ceh, Deh, Eh,...) orientiert sich an der Lautstruktur unserer Sprache: Die Buchstaben A,E,I,O,U werden direkt als die entsprechenden Vokale gesprochen, in den „Kon-Sonanten" klingt ein zusätzlicher Vokal (Beh, Ceh, Deh, Kah, Hah ...) mit. Lediglich in den Buchstaben „Jot", „Ypsilon", „Zet" und „Iks = xi" hat sich die griechische Aussprache bewahrt.

1.5 Gliederung

Die folgenden beiden Kapitel beschreiben Grundlagen und Terminologie, auf denen diese Arbeit aufbaut. *Kapitel 2* gibt eine kurze Einführung in die Problematik und die klas-

sischen Lösungsansätze der **Mustererkennung**. Darauf aufbauend stellt *Kapitel 3* die unterschiedlichen Ansätze für das spezielle Mustererkennungsproblem **Spracherkennung** vor. Hier werden die zentralen Themenbereiche Signalvorverarbeitung, akustische Modellierung und Sprachmodelle beschrieben, insbesondere die klassischen Markovmodelle (HMM). *Kapitel 4* dokumentiert die zur eigenen Arbeit verwandten Ansätze der Spracherkennung mit neuronalen Netzen.

Kapitel 5 beschreibt die acht umfangreichen Sprachdatenbanken, auf denen der Buchstabiererkenner trainiert und getestet wurde. Die Schlüsselbeiträge dieser Arbeit sind in den folgenden drei Kapiteln organisiert: In *Kapitel 6* wird die Entwicklung eines konnektionistischen Klassifikators vom Phonemerkenner zum **kontinuierlichen, sprecherunabhängigen Buchstabiererkenner** beschrieben. In zahlreichen Experimenten werden verschiedene Trainings- und Modellierungsansätze dokumentiert, mit denen die Erkennung signifikant verbessert werden konnte. In *Kapitel 7* kommen zu der akustischen Modellierung **Sprachmodelle für Buchstabensequenzen** als weitere Wissensquelle. Es werden Strategien untersucht, um die beispielsweise durch eine Liste von einer Million Nachnamen gegebenen Einschränkungen möglichst effektiv und effizient zu nutzen. In *Kapitel 8* werden **Buchstabiereffekte in spontaner Sprache** untersucht. Um in fließende Sprache eingebettete Buchstabensequenzen robuster erkennen zu können, wird das Sprachmodell eines spontansprachlichen Erkenners erweitert. Gefundene Buchstabensegmente können mit dem Buchstabiererkenner reklassifiziert werden. Andere untersuchte Phänomene sind spezifizierte Buchstabierungen („`A wie Auto`“) oder Namen, die sowohl fließend gesprochen als auch buchstabiert werden.

Schließlich beleuchtet *Kapitel 9* einige der praktischen Aspekte bei der Implementierung der Systeme. Die Ergebnisse der Arbeit sind in *Kapitel 10* zusammengefaßt. Im *Anhang* finden sich einige mathematische und tabellarische Ergänzungen.

Kapitel 2

Mustererkennung

„ Paradoxically, we are all experts at perception, but none of us knows much about it"
– Duda & Hart, aus [DH73], Seite 1

Dieses Kapitel gibt eine kurze Einführung in das Gebiet der Mustererkennung und orientiert sich dabei an der ausführlicheren Darstellung aus dem klassischen Buch von Duda und Hart [DH73]. Dabei sollen die Terminologie und diejenigen grundlegenden Verfahren aus der Mustererkennung vorgestellt werden, auf denen auch die in Kapitel 3 beschriebenen Methoden der Spracherkennung aufbauen.

2.1 Problemstellung

Ziel der Mustererkennung oder Klassifikation ist es, Merkmalsvektoren $\boldsymbol{x}$ automatisch zu kategorisieren, d.h. einer bestimmten Klasse ω_i zuzuordnen. Die Merkmalsvektoren werden als Meßdaten aus der physikalisch wahrnehmbaren Welt gewonnen. Wir beschränken uns auf die Betrachtung numerischer[1] Werte, es sei $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$. Grauwertbilder oder Audiosignale sind Beispiele für Merkmalsvektoren in Bild- oder Spracherkennung. Meist unterliegen die gemessenen Merkmale weiteren Vorverarbeitungsschritten (z.B. Kantendetektion oder Spektralanalyse), bevor sie klassifiziert werden. Die Zuordnungsvorschrift von Merkmalsvektoren auf Klassen ist nicht von vornherein bekannt, sondern soll anhand möglichst repräsentativer Lernstichproben erlernt werden. Andererseits ist ein „Auswendiglernen" der Stichprobe nicht erwünscht: Ein Klassifikator soll generalisieren können, d.h. auch bisher ungesehene Merkmalsvektoren möglichst oft der richtigen Klasse zuordnen. In diesem Kapitel betrachten wir Merkmalsvektoren konstanter Dimension d. Die im nächsten Kapitel vorgestellten Ansätze in der Spracherkennung

[1] Im Gegensatz zu nominalen („rot, blau") oder ordinalen („langsam, schnell") Werten

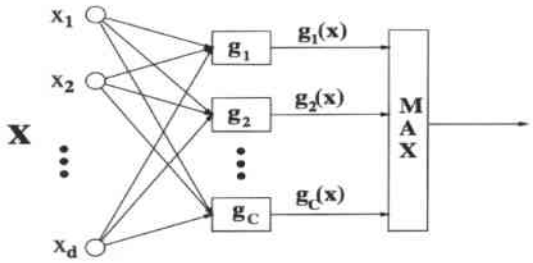


Abbildung 2.1: Entscheidungsfunktionen

unterscheiden sich im wesentlichen gerade dadurch, daß sie auch Muster variabler Länge handhaben können.

Man spricht von **überwachtem Lernen** oder **Lernen mit Lehrer**, wenn die Klassenzugehörigkeit der Muster aus der Lernstichprobe bekannt ist. Jedem Merkmalsvektor ist eine von C Klassen $\{\omega_1, \ldots, \omega_C\}$ zugeordnet. Aus den K Stützstellen $\{(\boldsymbol{x}_k, c_k)\}, c_k \in \{\omega_1, \ldots, \omega_C\}$, einer Lernstichprobe – auch als Trainingsmenge bezeichnet – wollen wir einen Klassifikator gewinnen, der jedes $\boldsymbol{x}$ auf seine korrekte Klasse abbildet.

Bei **unüberwachtem Lernen** ist als Lernstichprobe lediglich eine Menge von Mustern gegeben, deren Klassenzugehörigkeit unbekannt ist. Man nimmt an, daß die Daten nach gewissen Gesetzmäßigkeiten verteilt sind, und versucht, diese Struktur mit einer geeigneten Klasseneinteilung widerzuspiegeln. Dies läuft auf die Suche von Ballungsgebieten hinaus, die möglichst kompakt und von anderen Ballungen gut abgrenzbar sind. Im Gegensatz zum überwachten Lernen ist die Anzahl der Klassen nicht notwendigerweise a priori festgelegt. Ein Beispiel für unüberwachtes Lernen ist die in Abschnitt 3.2.3 beschriebene Vektorquantisierung aus der Spracherkennung. Dabei soll der akustische Merkmalsraum so in Klassengebiete eingeteilt werden, daß man bei möglichst geringem Informationsverlust[2] statt der vieldimensionalen akustischen Vektoren auch deren Klassenindizes als Eingabe nutzen kann.

2.2 Diskriminanzfunktionen

Ein Klassifikator kann in Form einer Menge von Entscheidungsfunktionen (Abbildung 2.1) dargestellt werden. Für jede Klasse ω_j wird eine Entscheidungsfunktion $g_j(\boldsymbol{x})$, $j = 1, \ldots, C$ (auch *Prüfgröße* oder *Diskriminanzfunktion* genannt) bestimmt, und man entscheidet sich für Klasse ω_i, falls

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i \tag{2.1}$$

[2]Hervorgerufen durch Quantisierungsfehler

Ziel des Lernens ist es, einen Klassifikator, d.h. die C Entscheidungsfunktionen $g_i(\boldsymbol{x})$ so zu finden, daß die Trainingsmenge und auch bisher ungesehene Merkmalsvektoren möglichst gut klassifiziert werden. Nach der Bayesschen Entscheidungstheorie macht man den kleinsten Fehler, wenn man sich bei gegebenem $\boldsymbol{x}$ für Klasse ω_i entscheidet, falls

$$P(\omega_i|\boldsymbol{x}) \;>\; P(\omega_j|\boldsymbol{x}) \quad \forall j \neq i \tag{2.2}$$

Somit ist mit $g_i(\boldsymbol{x}) = P(\omega_i|\boldsymbol{x})$ ein optimaler Klassifikator gegeben. Die optimalen Diskriminanzfunktionen $g_i(\boldsymbol{x})$ sind aber keineswegs eindeutig oder notwendigerweise als Wahrscheinlichkeiten interpretierbar – offensichtlich kann man sie gleichwertig, ohne Auswirkungen auf die Entscheidung, durch $a * g_i(\boldsymbol{x}) + b\,, a > 0$ oder allgemeiner durch $f(g_i(\boldsymbol{x}))$ ersetzen, falls f eine streng monoton wachsende Funktion ist.

Klassifikatoren unterscheiden sich im wesentlichen in der Realisierung ihrer Entscheidungsfunktionen:

- **Lernen von Verteilungsfunktionen:** Nach der Bayesschen Formel

$$P(\omega_i|\boldsymbol{x}) \;=\; \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})} \tag{2.3}$$

  können wir die gesuchten a posteriori Wahrscheinlichkeiten $P(\omega_i|\boldsymbol{x})$ indirekt mit Hilfe der sogenannten klassenbedingten Wahrscheinlichkeiten $p(\boldsymbol{x}|\omega_i)$ sowie der a priori Wahrscheinlichkeiten $P(\omega_i)$ und $p(\boldsymbol{x})$ berechnen. Da $p(\boldsymbol{x})$ unabhängig von den Klassen ω_i ist, kann aus Gl. (2.3) die Entscheidungsfunktion

$$g_i(\boldsymbol{x}) \;=\; p(\boldsymbol{x}|\omega_i)P(\omega_i) \tag{2.4}$$

  oder

$$g_i(\boldsymbol{x}) \;=\; \log p(\boldsymbol{x}|\omega_i) + \log P(\omega_i) \tag{2.5}$$

  abgeleitet werden. Die unbekannte Dichte $p(\boldsymbol{x}|\omega_i)$ wird häufig durch eine parametrisierte Exponentialfunktion[3] $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ mit dem **Maximum-Likelihood-Verfahren** approximiert. Dazu sucht man denjenigen Parametervektor $\boldsymbol{\theta}_i^*$, der $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ maximiert, der also gewissermaßen die Trainingsdaten am besten „erklärt".

- **Verteilungsfreie Klassifikatoren:** Beim Lernen von Verteilungsfunktionen werden die Klassengebiete mit klassenbedingten Verteilungen $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ modelliert. Verteilungsfreie Klassifikatoren versuchen, direkt eine parametrisierte Entscheidungsfunktion $g_i(\boldsymbol{x}|\boldsymbol{\theta})$ zu schätzen.
  Zwei Entscheidungsfunktionen g_i und g_j separieren die Klassengebiete von ω_i und

[3] Dann vereinfacht sich $p(\boldsymbol{x}|\omega_i)$ durch Logarithmieren, siehe dazu auch Gl. (2.9).

ω_j durch eine Trennfläche $\{\boldsymbol{x}|g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})\}$ – das direkte Lernen von Entscheidungsfunktionen wird daher auch als **diskriminatives Lernen** bezeichnet. Klassische Vertreter dieser Technik sind lineare oder nichtlineare Trennfunktionen, die durch künstliche neuronale Netze realisiert werden können. Häufig versucht man, die Entscheidungsfunktionen direkt in Form der a posteriori Wahrscheinlichkeiten zu approximieren:

$$g_i(\boldsymbol{x}|\boldsymbol{\theta}) = P(\omega_i|\boldsymbol{x}) \tag{2.6}$$

Wird während des Lernvorgangs $g_i(\boldsymbol{x}|\boldsymbol{\theta})$ erhöht, muß gleichzeitig $\sum_{j\neq i} g_j(\boldsymbol{x}|\boldsymbol{\theta})$ erniedrigt werden, um der Randbedingung

$$\sum_{j=1}^{C} P(\omega_j|\boldsymbol{x}) \;=\; 1 \tag{2.7}$$

zu genügen – auch hierin manifestiert sich der diskriminative Charakter dieser Vorgehensweise.

2.3 Schätzen von Verteilungsdichten

Sind die klassenbedingten Wahrscheinlichkeiten $p(\boldsymbol{x}|\omega_i)$ und die a priori Wahrscheinlichkeiten $P(\omega_i)$ bekannt, dann kann daraus gemäß (2.3) oder (2.4) ein optimaler Klassifikator bestimmt werden. Die Lernstichprobe sollte groß genug sein, um aus den Auftrittshäufigkeiten der einzelnen Klassen ω_i eine ausreichend gute Schätzung von $P(\omega_i)$ zu erlauben. Gegenstand dieses Abschnittes ist das schwierigere Problem, die klassenbedingten Wahrscheinlichkeiten $p(\boldsymbol{x}|\omega_i)$ zu schätzen.

2.3.1 Parametrische Dichtefunktionen

Wir gehen von parametrisierten, klassenbedingten Dichten $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ aus, deren Parametervektoren $\boldsymbol{\theta}_k$ durch die Lernstichprobe geschätzt werden sollen. In manchen Fällen erlaubt das Problemwissen, sinnvolle Verteilungsannahmen zu treffen – meist sind wir dazu allein durch die Größe der Merkmalsräume[4] gezwungen, denn eine vernünftige Schätzung der Dichte für jeden Punkt $p(\boldsymbol{x})$ ist ohne einschränkende Annahmen auch für große Trainingsmengen nur schwer zu bestimmen.

[4]Die Größe eines Merkmalsraums $M \subset \mathbb{R}^d$ wächst exponentiell mit seiner Dimension d. Die Fläche $M = [-1,1]^2$ wird mit 1000 Datenpunkten gut abgedeckt, im 20-dimensionalen Hyperwürfel $M = [-1,1]^{20}$ kann damit nur jeder tausendste Eckpunkt besetzt werden!

Meist wird angenommen, daß $p(\boldsymbol{x}|\omega_i)$ aus der Familie der Normalverteilungen $N(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ stammt – zu schätzen sind also der Mittelwertsvektor $\boldsymbol{\mu}_i$ und die Kovarianzmatrix $\boldsymbol{\Sigma}_i$ der *multivariaten Gaußdichte*

$$p(\boldsymbol{x}|\omega_i) = N(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_i|}} \cdot e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)} \tag{2.8}$$

Zusammen mit der a priori Wahrscheinlichkeit $P(\omega_i)$ ergeben sich nach Logarithmierung gemäß (2.5) (und Multiplikation mit -2) die Entscheidungsfunktionen

$$g_i(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i) + \log((2\pi)^d|\boldsymbol{\Sigma}_i|) - 2\log(P(\omega_i)) \tag{2.9}$$

Hinsichtlich der Komponenten von $\boldsymbol{x}$ hat (2.9) quadratische Gestalt. Die Trennfläche $\{\boldsymbol{x}|g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})\}$ zwischen dem Klassengebiet von ω_i und ω_j hat daher die Form allgemeiner Kegelschnitte (Geraden, Hyperbeln, Ellipsen).

Die Normalverteilungsannahme schränkt das statistische Modell stark ein: Die Gaußverteilung ist unimodal und symmetrisch bezüglich der Mahalanobisdistanz $(\boldsymbol{x} - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)$. Durch die gewichtete Kombination einer hinreichend großen Anzahl von Gaußverteilungen können beliebige Verteilungsdichten approximiert werden:

$$p(\boldsymbol{x}|\boldsymbol{c}_i, \{\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_{N_i}, \boldsymbol{\Sigma}_1 \ldots \boldsymbol{\Sigma}_{N_i}\}) = \sum_{j=1}^{N_i} c_{ij} N(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\ , \quad \sum_{j=1}^{N_i} c_{ij} = 1 \tag{2.10}$$

Man nennt (2.10) *Gaußsche Mischverteilungsdichte*. Abbildung 2.2 zeigt ein Beispiel einer eindimensionalen Mischverteilung.

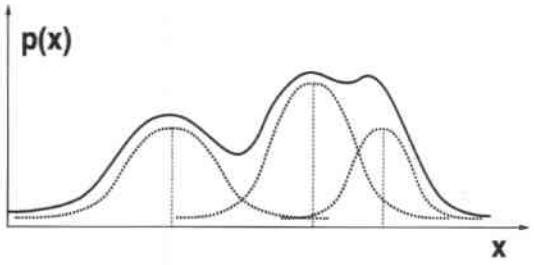


Abbildung 2.2: Eindimensionale Gaußsche Mischverteilung

2.3.2 Maximum-Likelihood-Schätzung

Gegeben sei eine Lernstichprobe $\boldsymbol{X}_1 \uplus \cdots \uplus \boldsymbol{X}_C$, wobei (wie zuvor) die Elemente von $\boldsymbol{X}_i$ zur Klasse ω_i gehören und gemäß der uns unbekannten Dichte $p(\boldsymbol{x}|\omega_i)$ unabhängig

voneinander gewonnen wurden. Die tatsächliche Verteilung der Daten soll durch parametrisierte, klassenbedingte Dichten $p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta}_i)$ approximiert werden. Ziel ist, die Parametervektoren $\boldsymbol{\theta}_i$ aus den Trainingsdaten $\boldsymbol{X}_i, i = 1, \ldots, C$, zu schätzen.

Wir betrachten die Parametervektoren vereinfachend als *funktional unabhängig*, d.h. die Stichprobe $\boldsymbol{X}_i$ enthält keine Information über $\boldsymbol{\theta}_j$ für $i \neq j$. Dann kann die Parameterschätzung für jede der C Klassen isoliert betrachtet werden: Aus der Stichprobe $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ einer beliebigen, aber festen Klasse soll der Parametervektor $\boldsymbol{\theta}$ der Dichte $p(\boldsymbol{x}|\boldsymbol{\theta})$ bestimmt werden. Da die Elemente $\boldsymbol{x}_k \in \boldsymbol{X}$ unabhängig voneinander gezogen wurden, gilt

$$p(\boldsymbol{X}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\boldsymbol{x}_k|\boldsymbol{\theta}) \tag{2.11}$$

Als Funktion von $\boldsymbol{\theta}$ betrachtet, heißt $p(\boldsymbol{X}|\boldsymbol{\theta})$ die *Likelihoodfunktion* bezüglich der Stichprobe $\boldsymbol{X}$. Die *Maximum-Likelihood-Schätzung* von $\boldsymbol{\theta}$ ist derjenige Parametervektor $\boldsymbol{\theta}^*$, der sozusagen die Trainingsdaten am besten erklärt:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{X}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{k=1}^{n} p(\boldsymbol{x}_k|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{k=1}^{n} \log p(\boldsymbol{x}_k|\boldsymbol{\theta}) \tag{2.12}$$

Für einfache Dichtefunktionen, wie beispielsweise die Normalverteilung, kann die Maximum-Likelihood-Schätzung (2.12) analytisch berechnet werden. Besitzt die Likelihoodfunktion verborgene Variablen, können diese mit dem iterativen *Expectation-Maximization (EM)-Algorithmus* geschätzt werden.

2.4 Nichtparametrische Verfahren

Im nichtparametrischen Fall liegt keine Verteilungsannahme zugrunde – jede Dichte $p(\boldsymbol{x}|\omega_k)$ muß direkt aus den Klasse ω_k zugeordneten Mustern $\boldsymbol{X}_k = \{\boldsymbol{x}_{k_1}, \boldsymbol{x}_{k_2}, \ldots\}$ der Lernstichprobe $\boldsymbol{X}_1 \uplus \cdots \uplus \boldsymbol{X}_C$ geschätzt werden.

2.4.1 Parzenfenster zur Schätzung von Dichtefunktionen

Gegeben seien n Muster $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, die unabhängig nach der uns unbekannten Dichte $p(\boldsymbol{x})$ gezogen wurden. $p(\boldsymbol{x})$ kann geschätzt werden, indem die relative Häufigkeit von Datenpunkten in einer kleinen Umgebung von $\boldsymbol{x}$ gezählt wird. Die Wahrscheinlichkeit, daß $\boldsymbol{x}$ in einen Bereich B fällt, ist

$$P = \int_B p(\boldsymbol{x}')d\boldsymbol{x}' \tag{2.13}$$

Der Erwartungswert dafür, daß von den n Mustern k in B liegen, ist $E(k) = nP$ und damit $P \approx k/n$. Wenn wir annehmen, daß die Dichte $p(\boldsymbol{x})$ in einem kleinen Bereich B mit Volumen V konstant ist, ergibt sich

$$P = \int_B p(\boldsymbol{x}')d\boldsymbol{x}' = p(\boldsymbol{x})V \approx \frac{k}{n} \tag{2.14}$$

und damit die Schätzung

$$p(\boldsymbol{x}) = \frac{k/n}{V} \tag{2.15}$$

Sei ϕ eine Fensterfunktion, die alle Punkte innerhalb des um $\mathbf{0}$ zentrierten d-dimensionalen Einheitshyperwürfels ausschneidet:

$$\phi(u) = \begin{cases} 1 & |u_i| \leq 1/2,\ i = 1, \ldots, d \\ 0 & \text{sonst} \end{cases} \tag{2.16}$$

Dann kann die Anzahl k der in $B(\boldsymbol{x})$ liegenden Muster bestimmt werden zu

$$k = \sum_{i=1}^{n} \phi(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}), \tag{2.17}$$

wobei h die Kantenlänge des Hyperwürfels bestimmt, indem die Muster gezählt werden. Die Fensterfunktion ϕ kann zu einer Wahrscheinlichkeitsdichte verallgemeinert werden, man spricht dann von einem *Parzenfenster*. Dann werden die Muster nicht innerhalb eines Hyperwürfels gezählt, sondern beispielsweise mit einer multivariaten Normalverteilung gewichtet.

In ein großes Volumen V fallen ausreichend viele Datenpunkte, und die Wahrscheinlichkeit $P \approx k/n$ wird gut approximiert. Dafür ist das geschätzte $p(\boldsymbol{x})$ aber nur eine unscharfe, weil gemittelte Annäherung an die echte Verteilung. Ist umgekehrt V zu klein, fallen nur wenige oder gar keine Muster in B, und wir erhalten die nutzlose Schätzung $p(\boldsymbol{x}) \approx 0$. Je mehr Daten vorliegen, desto kleiner kann V gewählt werden. Man wird also $V = V_n$ in Abhängigkeit der Anzahl n der Muster wählen, beispielsweise $V_n = 1/\sqrt{n}$ [DH73]. Statt in einem konstanten Volumen die Anzahl k zu bestimmen, kann man auch die Anzahl der Muster k vorgeben, und V solange erhöhen, bis k erreicht ist. Diese *k-Nächste-Nachbarn-Schätzung* hat den Vorteil, daß sich die Schärfe der Auflösung an die Gegebenheiten der Daten anpaßt.

2.4.2 Nächster-Nachbar-Klassifikation

Mit dem Parzenfenster und der k-Nächste-Nachbarn-Schätzung konnten die klassenbedingten Wahrscheinlichkeiten $p(\boldsymbol{x}|\omega_i)$ approximiert werden. Sei k die Anzahl der Muster

aller Klassen eines Bereiches B um $\boldsymbol{x}$ und k_i die Anzahl der Muster aus Klasse ω_i. Dann bestimmt sich die Verbundwahrscheinlichkeit $p(\boldsymbol{x}, \omega_i)$ zu

$$p(\boldsymbol{x}, \omega_i) = \frac{k_i/n}{V} \tag{2.18}$$

Die a posteriori Wahrscheinlichkeit ergibt sich zu

$$p(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \omega_i)}{p(\boldsymbol{x})} \approx \frac{\frac{k_i/n}{V}}{\frac{k/n}{V}} = \frac{k_i}{k} \tag{2.19}$$

Die Klasse mit der höchsten a posteriori Wahrscheinlichkeit wird gefunden durch die *k-Nächste-Nachbarn-Regel*: Wähle die Klasse ω_i, die unter den k nächsten Nachbarn des zu klassifizierenden Merkmalsvektors $\boldsymbol{x}$ am häufigsten vertreten ist. Wird im Spezialfall k=1 nur ein einziger Nachbar berücksichtigt, spricht man von der *Nächster-Nachbar-Regel*: Wähle die Klasse ω_i des am nächsten bei $\boldsymbol{x}$ liegenden Musters.

2.5 Neuronale Netze

Künstliche neuronale Netze (kNN) bestehen aus einer oft großen Anzahl miteinander verbundener, einfacher Recheneinheiten (Neuronen, Knoten), deren Funktionsweise an die biologischer Nervensysteme angelehnt ist. Ein Neuron hat eine bestimmte Ausgabe, die über gewichtete Verbindungen an andere Neuronen weitergeleitet wird. Alle in einem Neuron ankommenden Signale werden zu einer „Eingangserregung" aufsummiert[5], mittels einer Transferfunktion verarbeitet und dann an alle ausgabeseitig verbundenen Neuronen weitergeleitet. Als lediglich grobe Annäherung an die biologischen Vorbilder bezeichnet man kNN auch neutraler als *konnektionistische Systeme*.

Es gibt eine Vielzahl von Netztypen, die sich in ihrer Verbindungstopologie (rekurrent oder nicht-rekurrent, vollständige oder modulare Verbindungsstruktur), ihren Transferfunktionen (Sigmoid, Radial-Basis-Funktionen), ihren Berechnungsschemata (synchron, asynchron) und Lernalgorithmen (überwacht, unüberwacht, halb-überwacht (reinforcement learning)) unterscheiden. Relevant für diese Arbeit und daher im folgenden beschrieben ist das Mehrschichtperzeptron (Multi-Layer Perceptron), das mit einem Gradientenabstiegsverfahren (Fehlerrückführung, Error Backpropagation) trainiert wird.

2.5.1 Geschichtlicher Rückblick

Die Erforschung der biologischen Nervensysteme beginnt bereits Ende des letzten Jahrhunderts. McCulloch und Pitts haben 1943 mit den binären Schwellwertneuronen ein erstes mathematisches neuronales Modell vorgeschlagen. Rosenblatt entwickelt 1962 ein

[5] Bzw. Distanzberechnung bei Radial-Basis-Funktionen

Lernverfahren für das einschichtige Perzeptron (siehe unten). Das Perzeptron kann nur linear separierbare Probleme klassifizieren. Diese Schwäche haben Minsky und Papert 1969 [MP69] anhand der einfachen, aber nicht linear separierbaren booleschen X-OR-Funktion (exklusives Oder) illustriert und so negativ bewertet, daß die Forschungsaktivitäten in diese Richtung für die nächsten 15 Jahre stark reduziert wurden.

Anfang der achtziger Jahre entstehen eine ganze Reihe verschiedener Netztypen. Hopfield bringt erstmals die Notation einer zu minimierenden Energiefunktion ins Spiel. Kohonen-Netze lernen unüberwacht und werden auch als selbstorganisierte Merkmalskarten bezeichnet. Am weitesten verbreitet sind die sogenannten Mehrschicht-Perzeptronen (Multi-Layer Perceptron, MLP), die mit einem von Rummelhard und McClelland popularisierten Gradientenabstiegsverfahren [RM86] (Error Backpropagation) eingelernt werden. Die neuronale Komponente des später vorgestellten MS-TDNN basiert auf einem MLP. „Vorgänger" des MLP ist das einschichtige Perzeptron, das zuerst kurz beschrieben wird.

2.5.2 Lineare Klassifikatoren

Eine einfache, aber für viele Anwendungen bereits ausreichende Entscheidungsfunktion kann durch

$$g(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + w_0$$

realisiert werden, auch *linearer Klassifikator* oder *Perzeptron* genannt. Der Gewichtsvektor $\boldsymbol{w}$ berechnet eine Linearkombination der Komponenten von $\boldsymbol{x}$; w_0 wird als Schwellwert bezeichnet. Für ein Zweiklassenproblem ist eine einzelne Diskriminanzfunktion ausreichend. Man entscheidet sich für ω_1, falls $g(\boldsymbol{x}) > 0$, und für ω_2 sonst. $g(\boldsymbol{x}) = 0$ definiert eine *Trennhyperebene*, die den Merkmalsraum in eine „positive" ($g(\boldsymbol{x}) \geq 0$) und eine „negative" ($g(\boldsymbol{x}) < 0$) Hälfte teilt.

In der Lernphase soll eine Trennhyperebene so bestimmt werden, daß alle Muster einer Klasse auf der positiven und die restlichen Muster auf der negativen Seite der Trennhyperebene zu liegen kommen. Das *Perzeptron-Lernverfahren* versucht mittels eines iterativen Gradientenabstiegsverfahrens eine Ebene zu finden, die alle Muster richtig klassifiziert. Ein Problem heißt *linear separierbar*, falls es eine solche Ebene gibt: Dann kann die Konvergenz des Verfahrens bewiesen werden.

Ein „weicheres" Optimierungskriterium versucht die Trennebene so zu legen, daß alle Muster in einem bestimmten Abstand auf der richtigen Seite zu liegen kommen. Dazu muß der mittlere quadratische Abstand aller Muster von ihrer Sollposition minimiert werden. Für die Minimierung der Summe der quadrierten Abstände (d.h. Fehler) gibt es eine Reihe klassischer Verfahren, z.B. kann mit der sogenannten Pseudoinversen eine analytische Lösung in einer geschlossenen Formel gefunden werden.

2.5.3 Mehrschichtige Netze

Durch die Verschaltung mehrerer einschichtiger Perzeptronen können komplexere, nichtlineare Trennebenen modelliert werden. Eine typische Architektur für ein solches *mehrschichtiges Perzeptron* (*Multi-Layer Perceptron, MLP*) ist in Abbildung 2.3 illustriert. Ein MLP besitzt eine Eingabeschicht, eine oder mehrere verborgene Schichten sowie eine Ausgabeschicht, die jetzt um eine Nichtlinearität erweitert ist. Aufeinanderfolgende Schichten sind vollständig miteinander verbunden. Die gewichteten Eingaben $w_{ij}y_j$ werden zusammen mit einem Schwellwert b_i zur Eingangserregung x_i aufsummiert. x_i passiert eine Nichtlinearität f und wird als $y_i = f(x_i)$ zur nächsten Schicht weitergeleitet. Im weiteren bezeichnen wir x_i als die *Aktivierung*, y_i als die *Ausgabe* und f als die *Transferfunktion*[6] eines Neurons i. Der Einfachheit halber stellen wir uns b_i als ein weiteres Gewicht zu einem virtuellen Eingabeknoten mit konstantem Wert 1 vor, so daß wir alle Parameter des Netzes in einer Gewichtsmatrix $\boldsymbol{W}$ repräsentieren können.

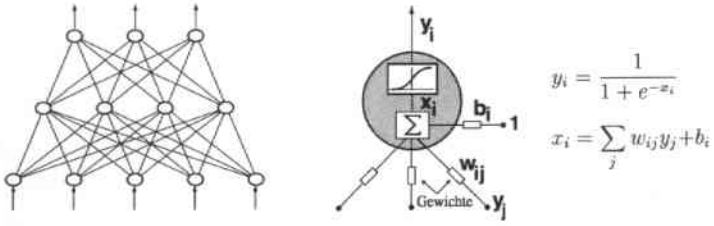


Abbildung 2.3: Ein *Multi-Layer Perceptron* mit Modell eines Neurons

Ein MLP mit D Knoten in der Eingabeschicht, einer Sigmoidfunktion in C Knoten der Ausgabeschicht sowie Gewichten $\boldsymbol{W}$ realisiert eine parametrisierte Funktion $f : \mathbb{R}^D \to [0,1]^C$. Mit den Knoten der Ausgabeschicht als Entscheidungsfunktionen $g_i(\boldsymbol{x})$ erhalten wir einen verteilungsfreien[7] Klassifikator für ein C-Klassenproblem auf einem D-dimensionalen Merkmalsraum.

Die Lernaufgabe besteht darin, die Gewichte $\boldsymbol{W}$ so einzustellen, daß die Muster der Trainingsmenge $\{(\boldsymbol{x}_k, c_k)\}, \boldsymbol{x}_k \in \mathbb{R}^D, c_k \in \{\omega_1, \ldots, \omega_C\}$, möglichst gut klassifiziert werden. Bei der gemeinhin verwendeten *1-aus-N-Klassenkodierung* repräsentiert jedes Ausgabeneuron i eine Klasse bzw. eine Entscheidungsfunktion g_i. Gemäß (2.2) bestimmt die höchste Ausgabe die erkannte Klasse. Angestrebt wird also eine möglichst hohe Ausgabe für das Neuron der korrekten Klasse bei gleichzeitig kleinen Ausgaben der restlichen Neuronen.

[6] In der Literatur gelegentlich auch als Ausgabefunktion oder Aktivierungsfunktion bezeichnet

[7] Es wird keine Annahme über die Verteilung $p(\boldsymbol{x}|\omega)$ der Daten gemacht.

Nach Anlegen eines Musters $\boldsymbol{x}$ der Klasse ω_i wird die Ausgabe $\mathrm{MLP}(\boldsymbol{x}) = \boldsymbol{y}$ des Netzes der „idealen Ausgabe" $\boldsymbol{d}$ gegenübergestellt:

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_D \end{pmatrix} \longleftrightarrow \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} d_1 \\ \vdots \\ d_i \\ \vdots \\ d_D \end{pmatrix} = \boldsymbol{d}$$

Der Sollwert $d_j = \delta_{i,j}$ ist 1, falls $\boldsymbol{x}$ zur Klasse ω_i gehört, und 0 sonst. Die Abweichung der Ausgabe $\boldsymbol{y}$ von der Sollvorgabe $\boldsymbol{d}$ wird mit einer Fehlerfunktion $E(\boldsymbol{y}, \boldsymbol{d})$ bewertet. Dann werden mittels der Ableitung $\partial E / \partial w_{ij}$ die Gewichte so verändert, daß der Fehler E kleiner wird.

Statt der 1-aus-N-Kodierung kann eine verteilte Repräsentation der Ausgabe sinnvoll sein. Im bekannten NETTALK-Experiment [SR86] wurden Phoneme durch Bitvektoren repräsentiert, die die An- oder Abwesenheit bestimmter artikulatorischer Merkmale identifizierten. Ähnliche Phoneme besitzen dadurch ähnliche Eigenschaften. Dietterich nutzt in [DB95] fehlerkorrigierende Kodierungen, um robustere Ausgaberepräsentationen und bessere Klassifikationsraten zu erhalten. Windheuser [PBW95] versucht, unter zufällig verteilten Repräsentationen experimentell die beste zu finden.

2.5.4 Fehlerfunktionen

Das beim Lernen zu minimierende Kriterium ist der Gesamtfehler

$$\sum_{k=1}^{K} E(\boldsymbol{y}_k, \boldsymbol{d}_k) \tag{2.20}$$

auf der Trainingsmenge $\{(\boldsymbol{x}_k, c_k)\}, k = 1, \ldots, K$. Es sind verschiedene, der Einfachheit halber hier nur an einem Paar $(\boldsymbol{y}, \boldsymbol{d})$ illustrierte Fehlerfunktionen bekannt. Das klassische Maß ist die *Summe der quadratischen Fehler*, typischerweise MSE (mean squared error) abgekürzt:

$$E_{MSE} = \frac{1}{2}||\boldsymbol{y} - \boldsymbol{d}||^2 = \frac{1}{2}\sum_j (y_j - d_j)^2$$

Ein anderes, aus der Informationstheorie abgeleitetes Maß ist die *Cross Entropy (CE)*, die von binären Sollwerten $d_j \in \{0, 1\}$ ausgeht:

$$E_{CE} = -\sum_j [d_j \log(y_j) + (1 - d_j)\log(1 - y_j)] = \sum_j \begin{cases} -\log(y_j) & d_j = 1 \\ -\log(1 - y_j) & d_j = 0 \end{cases}$$

Die CE kann als Spezialfall der (in Gl. (2.29) eingeführten) relativen Entropie interpretiert werden, wenn die Ausgabe y_j und der Sollwert d_j als Wahrscheinlichkeiten betrachtet werden.

Auf J. McClelland geht das nach seinem Namen als MCL abgekürzte Fehlermaß zurück:

$$E_{MCL} = -\sum_j \log(1 - (d_j - y_j)^2)$$

Im Gegensatz zur MSE-Fehlerfunktion bestrafen die CE- und die MCL-Fehlerfunktion eine Abweichung vom Sollwert überproportional stark, im Extremfall $|d_j - y_j| \to 1$ sogar mit einem gegen unendlich strebenden Fehler.

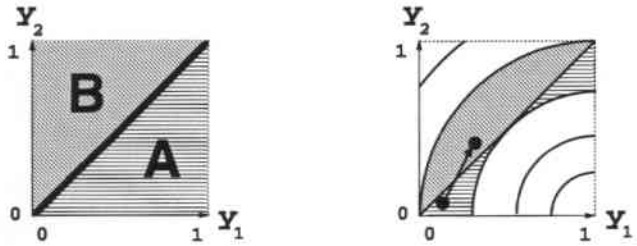


Abbildung 2.4: MSE-Fehlerbeitrag für ein Beispiel mit zwei Klassen (A,B) mit Sollausgabe $(y_1, y_2) = (1, 0)$. Links: Ausgaben unterhalb der Diagonalen werden als Klasse A, oberhalb als B klassifiziert. Rechts: Die Ausgaben im quer schraffierten Bereich sind richtig klassifiziert, aber zu jedem dieser Punkte gibt es einen im grau unterlegten Gebiet, der trotz eines kleineren MSE-Fehlerbeitrags falsch klassifiziert wird.

Classification Figure of Merit

Die MSE-, CE- und MCL-Fehlerfunktionen bestrafen die Abweichung der Netzausgabe von der idealen Ausgabefunktion. Deren Sollwerte (0 für die falsche, 1 für die richtige Klasse) werden jedoch nicht notwendigerweise erreicht, wenn das Lernverfahren kein globales Optimum garantiert, es Klassenüberlappungen im Merkmalsraum gibt oder diese Funktion vom Netz gar nicht approximiert werden kann. Für die MSE-, CE- oder MCL-Fehlerfunktion ist der Ausgaberaum nicht monoton in dem Sinne, daß eine Reduzierung des Fehlermaßes automatisch mit weniger Klassifikationsfehlern einhergeht. Diese erstaunliche Tatsache wurde 1990 von John Hampshire in [HIW90c] beschrieben und ist in Abbildung 2.4 für ein Zweiklassenproblem illustriert. Ein Muster aus Klasse A hat die Sollausgabe (1,0). Für Netzausgaben (y_1, y_2) ist der MSE-Fehlerbeitrag

$E_{MSE} = \frac{1}{2}((1-y_1)^2 + y_2^2)$ als Konturplot aufgetragen. Da wir uns für die Klasse mit der höheren Ausgabe entscheiden, werden alle Ausgaben unterhalb der Diagonalen Klasse A, die restlichen Klasse B zugeschlagen. Der grau unterlegte Bereich markiert diejenigen falsch klassifizierten Ausgaben, zu denen es Ausgaben mit höherem MSE gibt, die jedoch richtig klassifiziert werden, d.h. rechts der Diagonalen zu liegen kommen.

Die von Hampshire [HIW89a, HIW90c] vorgeschlagene, neue Fehler- oder besser Kriterienfunktion *Classification Figure of Merit* bewertet nicht die Abweichung von einer idealen Sollausgabe, sondern orientiert sich direkt an einer Minimierung des Klassifikationsfehlers. Dazu wird der Abstand $d_j = y_k - y_j$ zwischen der Ausgabe y_k der korrekten Klasse ω_k und den $C-1$ restlichen Ausgaben y_j bestimmt.

$$E_{CFM} = \frac{1}{C-1} \sum_{j=1, j \neq c}^{C} s(y_k - y_j) \tag{2.21}$$

Die Abstände $d_j = y_k - y_j$ werden erst nach Anwendung einer Sigmoidfunktion

$$s(d_j) = \frac{1}{1 + e^{-\beta * d_j + \gamma}} \tag{2.22}$$

aufsummiert, deren Steilheit und Verschiebung durch die Parameter β und γ bestimmt werden. Die Sigmoidfunktion mißt gewissermaßen die Klassifikationsgenauigkeit: Ist die korrekte Ausgabe y_k ausreichend größer als ein konkurrierendes y_j, erreicht $s(y_k - y_j) = s(d_j)$ die Sättigung auf einem Wert von 1. Umgekehrt strebt $s(d_j)$ für hinreichend große negative Differenzen d_j gegen Null.

Generalized Probabilistic Descent (GPD)

Auf derselben Idee basiert das wenig später von Juang und Katagiri veröffentlichte [Hua92] *Generalized Probabilistic Descent/Minimum Error Classification (GPD/MCE)*-Verfahren. Hierbei wird zwischen der Entscheidungsfunktion $g_k(\boldsymbol{x})$ der korrekten Klasse k und den restlichen g_j ein „Mißklassifikationsmaß" $d(\boldsymbol{x})$ bestimmt:

$$d(\boldsymbol{x}) = -g_k(\boldsymbol{x}) + [\frac{1}{C-1} \sum_{j, j \neq k} g_j(\boldsymbol{x})^\eta]^{\frac{1}{\eta}} \tag{2.23}$$

Die „Kosten" l für $d(\boldsymbol{x})$ werden wie in (2.21) und (2.22) als Sigmoidfunktion

$$l(d) = \frac{1}{1 + e^{-\beta * d + \gamma}} \tag{2.24}$$

bestimmt, die hier aber auf die Summe der Differenzen angewandt wird. Der Exponent realisiert eine L_η-Norm und erlaubt, die einzelnen Entscheidungsfunktionen abhängig von ihrem Wert unterschiedlich zu gewichten. Für $\eta = 1$ ergibt sich wie in (2.21) eine

Gleichbewertung aller g_j. Mit wachsendem η werden größere Werte stärker berücksichtigt, im Grenzfall $\eta \to \infty$ ergibt sich:

$$d(\boldsymbol{x}) = -g_k(\boldsymbol{x}) + g_i(\boldsymbol{x}), \tag{2.25}$$

wobei g_i, $i = \text{argmax}_{j \neq k}\, g_j(\boldsymbol{x})$, die Entscheidungsfunktion mit dem höchsten Wert (für eine falsche Klasse) ist. Zu (2.25) gibt es einige Varianten, am populärsten ist

$$d(\boldsymbol{x}) = -g_k(\boldsymbol{x}) + \log[\frac{1}{C-1} \sum_{j,j \neq k} e^{g_j(\boldsymbol{x})\eta}]^{\frac{1}{\eta}} \tag{2.26}$$

Maximum Mutual Information (MMI)

Um das MMI-Kriterium zu erklären, führen wir zuerst einige Begriffe aus der Informationstheorie [CT91] ein. Gegeben sei eine Informationsquelle, repräsentiert durch eine diskrete Zufallsvariable X, die n unterschiedliche Symbole $x_1, \ldots, x_n$ mit der Wahrscheinlichkeitsverteilung $p(x_1), \ldots, p(x_n)$ ausgeben kann. Dann ist der *Informationsgehalt* $I(x_i)$ definiert als

$$I(x_i) = \log \frac{1}{p(x_i)} \tag{2.27}$$

und besitzt die Einheit *bit*, falls der Logarithmus zur Basis 2 gebildet wird. Ein seltenes Zeichen hat einen hohen Informationsgehalt, während das sichere Ereignis $p(x_i) = 1$ keine Information beinhaltet. Der Erwartungswert für den Informationsgehalt des Zufallsprozesses X wird als *Entropie* $H(X)$ bezeichnet:

$$H(X) = \sum_i p(x_i) \log \frac{1}{p(x_i)} = -\sum_i p(x_i) \log p(x_i) \tag{2.28}$$

Die Entropie sagt aus, wieviele Bits im Durchschnitt benötigt werden, um die Zeichen der Quelle X zu kodieren. Die höchste Entropie, nämlich $H(X) = \log_2 n$, hat eine gleichverteilte Zufallsvariable $p(x_i) = 1/n, \forall i$.

Mit der *relativen Entropie* oder *Kullback-Leibler-Distanz* $D(p||q)$ mißt man den (nicht symmetrischen) Abstand zwischen zwei Wahrscheinlichkeitsverteilungen p und q:

$$D(p||q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{2.29}$$

Die *Transinformation* (*Mutual Information*) $I(X;Y)$ mißt den Informationsgehalt, den eine Zufallsvariable über eine andere enthält. Seien X und Y Zufallsvariablen mit der Verbundwahrscheinlichkeit $p(x_i, y_j)$ sowie marginalen Wahrscheinlichkeiten $p(x_i), p(y_j)$.

Dann ist die Transinformation die relative Entropie zwischen $p(x_i, y_j)$ und $p(x_i)p(y_j)$:

$$\begin{aligned} I(X;Y) &= D(p(x_i,y_j)||p(x_i)p(y_j)) = \sum_i \sum_j p(x_i,y_j) \log \frac{p(x_i,y_j)}{p(x_i)p(y_j)} \\ &= E_{p(x,y)} \left(\log \frac{p(X,Y)}{p(X)p(Y)} \right) \end{aligned} \tag{2.30}$$

Das MMI-Kriterium versucht nun, die Parameter $\boldsymbol{\theta}$ eines Klassifikators so einzustellen, daß die Transinformation zwischen den Eingaben $\mathbf{X}$ und ihren Klassen c_i maximiert wird. Für ein konkretes Paar $(\mathbf{X}_k, c_k)$ muß dazu der Ausdruck

$$\log \frac{p(\mathbf{X}_k, c_k|\theta)}{p(\mathbf{X}_k|\theta)p(c_k)} = \log \frac{p(\mathbf{X}_k|c_k,\theta)}{p(\mathbf{X}_k|\theta)} = \log p(\mathbf{X}_k|c_k,\theta) - \log \sum_i p(\mathbf{X}_k|c_i,\theta)p(c_i) \tag{2.31}$$

maximiert werden. Der erste Term in (2.31) entspricht gerade der Maximum-Likelihood-Zielfunktion, die hier aber mit den Beiträgen aller Klassen konkurrieren muß. Damit ist (2.31) ein diskriminatives Kriterium, das mit einem positiven Beitrag für die korrekte Klasse und negativen Beiträgen für die anderen Klassen eine interessante Ähnlichkeit zur GPD- und CFM-Fehlerfunktion aufweist.

2.5.5 Gradientenabstieg

Nachdem nun die wichtigsten Fehlerfunktionen vorgestellt wurden, betrachten wir jetzt Verfahren, um diese zu minimieren. Für eine gegebene Lernstichprobe ist der Gesamtfehler (2.20) eine Funktion des Parametervektors $\mathbf{W}$. Gesucht sind Gewichte $\mathbf{W}^*$, die den Gesamtfehler $E(\mathbf{W})$ über alle Trainingsbeispiele minimieren:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\, E(\mathbf{W}) = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_k E(\mathbf{W}, \boldsymbol{x}^k, c^k)$$

Eine geschlossene Formel ist nur in sehr einfachen Fällen, nicht jedoch für nichtlineare MLPs verfügbar. Deshalb wird nach $\mathbf{W}^*$ mittels Gradientenabstiegs gesucht. Dazu beginnt man bei einem zufällig gewählten Startwert $\mathbf{W}(0)$ und versucht, ein $\mathbf{W}(t+1)$ mit $E(\mathbf{W}(t+1)) < E(\mathbf{W}(t))$ zu finden, indem man im „Fehlergebirge" von $\mathbf{W}(t)$ ausgehend entlang dem Gradienten $-\nabla E(\mathbf{W})$ ein kleines Stück ϵ abwärts läuft

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \epsilon \nabla E(\mathbf{W}(t))$$

Die Schrittweite ϵ – auch als *Lernrate* bezeichnet – ist kritisch. Wird ϵ zu groß gewählt, schießt man über das Ziel hinaus, bei einem zu kleinem Wert wird die Suche unnötig langsam. Als Heuristik kann der Suche mit einem *momentum term* α eine gewisse Trägheit gegeben werden, indem zum aktuellen Gradienten ein Anteil des davor gemachten Schritts hinzugenommen wird:

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \epsilon \nabla E(\mathbf{W}(t)) + \alpha(\mathbf{W}(t) - \mathbf{W}(t-1))$$

Der Gradientenabstieg findet nicht notwendigerweise ein *globales* Minimum.

Um den Gradienten $\nabla E(\mathbf{W}(t)) = \sum_k \nabla E(\boldsymbol{y}^k, c^k)$ zu bestimmen, müssen die Fehlerbeiträge aller Trainingsmuster aufaddiert werden, erst dann können neue Gewichte $\mathbf{W}(t+1)$ bestimmt werden. Von diesem als *batch learning* oder *learning by epoch* bezeichneten Verfahren weicht man in der Praxis oft ab zum sogenannten *online learning* oder *learning by pattern*, bei dem bereits nach jedem einzelnen Muster neue Gewichte bestimmt werden. Bei großen Trainingsmengen wird dadurch der Lernvorgang erheblich beschleunigt. Das beim *learning by pattern* durch die mathematisch nicht ganz korrekte Vorgehensweise entstehende „Rauschen" kann sich sogar vorteilhaft auswirken, indem es die Wahrscheinlichkeit erhöht, aus lokalen Minima zu entkommen.

2.5.6 Eigenschaften

Ein Mehrschicht-Perzeptron (MLP) besitzt zwei interessante Eigenschaften: Die Ausgaben eines trainierten MLP entsprechen gerade den a posteriori Klassenwahrscheinlichkeiten der Trainingsmuster, und mit einem MLP können stetige Funktionen beliebig genau approximiert werden.

Approximation beliebiger Funktionen

Jede stetige Funktion von n Argumenten kann durch eine endliche Verschachtelung von eindimensionalen Funktionen und deren Addition dargestellt werden. Diese Aussage ergibt sich aus einem Satz des russischen Mathematikers Kolmogorov, der nach [Roj93](S. 206) in einer modernen Variante wie folgt formuliert werden kann:

Satz von Kolmogorov: Gegeben sei eine n-dimensionale stetige Funktion $f : [0,1]^n \to [0,1]$. Es existieren eindimensionale stetige Funktionen g und ϕ_q, $q = 1, \ldots, 2n+1$ sowie Konstanten $\lambda_p, p = 1, \ldots, n$, so daß

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n} \lambda_p \phi_q(x_p)\right) \tag{2.32}$$

Man kann sich (2.32) als Netzwerk realisiert vorstellen: In der ersten verborgenen Schicht passieren die Eingaben x_p die Transferfunktionen ϕ_q. In den nachfolgenden Verbindungen werden sie mit λ_p gewichtet, um dann in den $2n+1$ Neuronen der zweiten verborgenen Schicht aufsummiert und durch die Transferfunktion g geleitet zu werden. Schließlich ergibt die Summierung dieser Werte gerade $f(x_1, x_2, \ldots, x_n)$.

Der Satz von Kolmogorov sagt aus, daß die Funktion f *exakt* darstellbar ist. Allerdings ist der Beweis nicht konstruktiv, d.h. er gibt keine Methode an, um die Funktionen ϕ_q und g zu bestimmen. Verzichtet man auf eine exakte Darstellung, kommt man sogar mit einem Netzwerk aus, das nur eine verborgene Schicht mit Sigmoidfunktion als Transferfunktion

besitzt, wobei allerdings die Anzahl der verborgenen Neuronen beliebig groß werden kann. Im Buch von Hecht-Nielsen [HN90] wird dazu der folgende Satz aufgeführt:

„Backpropagation Approximation"-Satz: Für jedes $\epsilon > 0$ und jede (stückweise) stetige Funktion[8] $f : [0,1]^n \rightarrow \mathbb{R}^m$ existiert ein dreischichtiges MLP, das f mit einem mittleren quadratischen Fehler $\leq \epsilon$ approximieren kann.

Lernen von a posteriori Wahrscheinlichkeiten

Wird ein hinreichend mächtiges Netzwerk mit einer 1-aus-N-Ausgaberepräsentation und dem MSE-Fehlermaß trainiert, approximieren die Netzausgaben die a posteriori Klassenwahrscheinlichkeiten. Der folgende Beweis geht von einem Zweiklassenproblem mit Klassen A und $\bar{A}$ aus und ist in vereinfachter Form aus [Gis90] entnommen. Das Netzwerk realisiert eine Funktion $F(\boldsymbol{x}, \mathbf{W})$, wobei $\boldsymbol{x}$ die Netzeingabe und $\mathbf{W}$ die Netzparameter sind. Die Sollausgabe[9] für $F(\boldsymbol{x}, \mathbf{W})$ sei 1, falls $\boldsymbol{x} \in A$, und 0, falls $\boldsymbol{x} \in \bar{A}$. Dann bestimmt sich der mittlere quadratische Fehler zu:

$$E = \frac{1}{N}\left[\sum_{\boldsymbol{x}\in A}[F(\boldsymbol{x},\mathbf{W})-1]^2 + \sum_{\boldsymbol{x}\in\bar{A}} F^2(\boldsymbol{x},\mathbf{W})\right] \tag{2.33}$$

Ist die Anzahl N der Trainingsmuster groß genug und sind die Muster aus A und $\bar{A}$ proportional zu den a priori Klassenzugehörigkeiten $P(A)$ und $P(\bar{A})$ vertreten, dann kann die Summe in (2.33) durch ein Integral approximiert werden:

$$\begin{aligned}
E &= \int [F(\boldsymbol{x},\mathbf{W})-1]^2 p(\boldsymbol{x},A)d\boldsymbol{x} + \int F^2(\boldsymbol{x},\mathbf{W})p(\boldsymbol{x},\bar{A})d\boldsymbol{x} \\
&= \int F^2(\boldsymbol{x},\mathbf{W})[p(\boldsymbol{x},A)+p(\boldsymbol{x},\bar{A})] - 2F(\boldsymbol{x},\mathbf{W})p(\boldsymbol{x},A)d\boldsymbol{x} + \int p(\boldsymbol{x},A)d\boldsymbol{x} \\
&= \int F^2(\boldsymbol{x},\mathbf{W})p(\boldsymbol{x}) - 2F(\boldsymbol{x},\mathbf{W})P(A|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} + P(A) \\
&= \int [F(\boldsymbol{x},\mathbf{W}) - P(A|\boldsymbol{x})]^2 p(\boldsymbol{x})d\boldsymbol{x} - \int p^2(A|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} + P(A)
\end{aligned} \tag{2.34}$$

In (2.34) ist lediglich der erste Term von den Parametern $\mathbf{W}$ abhängig. Der Gesamtausdruck wird also genau dann minimiert, wenn der quadratische Fehler $[F(\boldsymbol{x},\mathbf{W}) - P(A|\boldsymbol{x})]^2$ minimiert wird. Ist das Netzwerk komplex genug, um im Training zu einem hinreichend kleinen Fehler zu konvergieren, und liegen ausreichend viele und proportional zur a priori Klassenzugehörigkeit verteilte Trainingsmuster vor, dann approximiert die Netzausgabe $F(\boldsymbol{x},\mathbf{W})$ also gerade die a posteriori Wahrscheinlichkeit $P(A|\boldsymbol{x})$.

Diese Eigenschaft gilt auch für mehr als zwei Klassen, wenn diese mit einer 1-aus-N-Kodierung repräsentiert sind [BW90], und auch für die Cross-Entropy-Fehlerfunktion [RL91] oder andere Standard-Fehlerfunktionen [HIP90].

[8] In [HN90] ist der Satz für die allgemeinere Klasse der L_2-Funktionen formuliert. f ist eine L_2-Funktion, wenn jede ihrer Koordinatenfunktionen auf dem Einheitswürfel quadratisch integrierbar ist.

[9] Zur Vereinfachung der Beweisführung wird für den Zweiklassenfall nur eine Ausgabe angenommen.

2.6 Zusammenfassung

Die Mustererkennung, eine Problemstellung aus dem Bereich des maschinellen Lernens, hat zum Ziel, Merkmalsvektoren automatisch zu kategorisieren, d.h. einer bestimmten Klasse zuzuordnen. Die wichtigsten statistisch basierten Techniken zur numerischen Klassifikation wurden, soweit sie für diese Arbeit relevant sind, in den vorangegangenen Abschnitten erläutert. Ein Klassifikator wird realisiert, indem man für ein Eingabemuster $\boldsymbol{x}$ klassenbedingte Prüfgrößen $g_\omega(\boldsymbol{x})$ berechnet und sich für die Klasse $\omega^* = \text{argmax}_\omega\, g_\omega(\boldsymbol{x})$ mit der höchsten entscheidet.

Alle vorgestellten Verfahren „lernen" in dem Sinne, daß anhand einer vorgegebenen Trainingsmenge ein Klassifikator konfiguriert werden soll, der sowohl die Trainingsbeispiele als auch neue Daten möglichst gut klassifizieren kann. Solche Verfahren werden als empirisch oder induktiv bezeichnet, im Gegensatz zu den (hier nicht behandelten) deduktiven, symbolischen Lernverfahren, bei denen Lösungen anhand von Regeln aus formalisiertem Problemwissen abgeleitet werden.

Man spricht von *unüberwachtem* Lernen, wenn den als Merkmalsvektoren vorliegenden Mustern keine Klassen zugeordnet sind, sondern die Muster möglichst aussagekräftig strukturiert werden sollen (Ballungsanalyse). Bei *überwachtem* Lernen ist jedem Merkmalsvektor $\boldsymbol{x}^k$ der Trainingsmenge eine Klasse $x^k \in \{\omega_1, \omega_2, \ldots\}$ zugewiesen. Aus diesen Daten soll eine allgemeine Zuordungsvorschrift gefunden werden, die beliebige Muster auf die korrekte Klasse abbildet. Dazu gibt es zwei grundlegende Lösungsansätze:

Mit *parametrischen* und *nichtparametrischen* statistischen Verfahren werden aus den Trainingsdaten die klassenbedingten Dichten $p(\boldsymbol{x}|\omega)$ geschätzt und zusammen mit den a priori Wahrscheinlichkeiten $P(\omega)$ zu den gewünschten a posteriori Wahrscheinlichkeiten $P(\omega|\boldsymbol{x})$ konvertiert. Im nichtparametrischen Fall wird $p(\boldsymbol{x}|\omega)$ direkt aus den lokalen Häufungseigenschaften der Daten geschätzt. Meist jedoch geht man davon aus, daß die Verteilung der Daten ausreichend genau durch eine Familie von parametrischen Dichtefunktionen, oft in Form von Normalverteilungen[10], approximiert werden kann. Mit dem *Maximum-Likelihood*-Verfahren werden dann diejenigen Parameter bestimmt, die die Daten der Trainingsmenge am besten „erklären".

Dagegen werden neuronale Netze auch als *verteilungsfreie Klassifikatoren* bezeichnet[11], da sie keine Annahmen über die Form der Dichten $p(\boldsymbol{x}|\omega)$ machen, sondern direkt versuchen, Diskriminanzfunktionen $g_\omega(\boldsymbol{x})$ oder sogar a posteriori Klassenwahrscheinlichkeiten $P(\omega|\boldsymbol{x})$ zu bestimmen. Im Gegensatz zu obigen statistischen Verfahren werden also nicht die Klassengebiete, sondern deren Trennflächen modelliert. Man spricht daher auch von *diskriminativen* Verfahren. Während des Lernens wird mittels eines Gradientenabstiegsverfahrens (*error backpropagation*) versucht, die Parameter (*Gewichte*) der neuronalen Netze so einzustellen, daß eine geeignete Fehlerfunktion auf den Mustern der Lernstichprobe minimiert wird.

[10] Oder Linearkombinationen von Normalverteilungen

[11] Falls sie (wie meist und in dieser Arbeit) als Klassifikatoren eingesetzt werden

Kapitel 3

Spracherkennung

Das Spracherkennungsproblem besteht darin, eine gesprochene Äußerung auf ihre korrekte textuelle Darstellung[1] abzubilden. Dies erfordert einen komplexeren Lösungsansatz als die im letzten Kapitel vorgestellten Verfahren der Mustererkennung: Während bisher von Mustern konstanter Größe ausgegangen wurde, bedingen Sprechgeschwindigkeit und Umfang der gesprochenen Äußerung Signale variabler Länge. Weiterhin ist Spracherkennung ein mehrstufiges Problem, in welchem Phoneme zu Wörtern und Wörter zu Sätzen zusammengefaßt werden.

Die Erkennungsaufgabe besteht darin, zu einer akustischen Eingabe $\mathbf{X}$ die wahrscheinlichste Wortsequenz W^* zu finden, die $\mathbf{X}$ erzeugt:

$$\begin{aligned} W^* &= \underset{W}{\operatorname{argmax}}\, P(W|\mathbf{X}) = \underset{W}{\operatorname{argmax}}\, \frac{P(\mathbf{X}|W) \cdot P(W)}{P(\mathbf{X})} \\ &= \underset{W}{\operatorname{argmax}}\, P(\mathbf{X}|W) \cdot P(W) \end{aligned} \tag{3.1}$$

Gleichung (3.1) wird oft als „Fundamentalgleichung der Spracherkennung" bezeichnet; aus ihr können wir die zentralen Themenbereiche der Spracherkennung ablesen:

- **Merkmalsextraktion oder Signalvorverarbeitung:** Wie werden geeignete Merkmalsvektoren $\mathbf{X}$ aus dem akustischen Signal gewonnen?
- **Akustische Modellierung:** Wie modellieren wir Phoneme, Wörter, Sätze, um $P(\mathbf{X}|W)$ zu berechnen?
- **Sprachmodelle:** Wie berechnet man die a priori Wahrscheinlichkeit $P(W)$?
- **Suche**, auch **Dekodierung** genannt: Wie findet man auf effiziente Weise die Wortsequenz W^*, die $P(\mathbf{X}|W) \cdot P(W)$ maximiert?

[1] Man spricht von *Sprachverstehen*, falls nicht eine möglichst genaue orthographische Transkription, sondern die Intention der Aussage erfaßt werden soll.

3.1 Geschichte

In den fünfziger Jahren wurden erste Systeme entwickelt, die mit Hilfe einfacher Zeitbereichsmerkmale oder Filterbänke versuchten, Phoneme, isolierte Ziffern oder einsilbige Wörter zu erkennen. Angeregt durch die menschliche Fähigkeit des Spektrogrammlesens basierten viele Ansätze auf einer Untersuchung der lokalen Spektralformen. In den siebziger Jahren wurde der Vergleich zeitlich verzerrter Muster nach dem Prinzip der dynamischen Programmierung eingeführt, womit sich bereits relativ leistungsfähige sprecherabhängige und -unabhängige Einzelworterkenner bauen ließen.

In einem von 1971 - 76 geförderten Projekt der US-Regierung wurden an Expertensystemen oder künstlicher Intelligenz orientierte Ansätze (Hearsay, HWIM) verfolgt, in denen zahlreiche Wissensbasen und Analysemethoden von einer übergeordneten Instanz koordiniert wurden.

Der Übergang vom dynamischen Mustervergleich mit akustischen Wortprototypen zur statistischen Modellierung mit Hilfe der *Hidden Markov Models (HMM)* markiert einen entscheidenden Paradigmenwechsel. Pionierarbeit in der Entwicklung der HMMs [Bak75, Jel76] wurde bereits ab 1975 von IBM geleistet. HMMs erlauben eine sehr flexible und mathematisch fundierte Modellierung der akustischen Realisierung von Sprache. Statt akustische Prototypen abzuspeichern werden in HMMs mit Hilfe der Trainingsdaten die Parameter von statistischen Modellen automatisch erlernt. Ein weiterer Durchbruch gelang mit der Technik der kontextabhängigen Phoneme [Lee89], die der Koartikulation in der Sprache Rechnung tragen. Auch heute noch sind HMMs die am weitesten verbreitete Technik in der Spracherkennung. Mit der Renaissance der Technik der künstlichen neuronalen Netze (kNN) Ende der achtziger Jahre gewannen die kNN auch in der Spracherkennung an Bedeutung. Für kleinere Aufgaben gibt es einige sehr erfolgreiche, ausschließlich konnektionistische Systeme, während sich für große Vokabulare hybride kNN-HMM-Techniken etabliert haben.

Wie schwierig ist ein Spracherkennungsproblem? Hier spielen viele Faktoren eine Rolle, die wichtigsten sind: Größe und Verwechselbarkeit des Vokabulars, isoliert oder kontinuierlich gesprochene Sprache, sprecherabhängige oder -unabhängige Erkennung, Einschränkungen des Sprachmodells, gelesene oder spontane Sprache, die Kooperationsbereitschaft und Sprechgeschwindigkeit der Sprecher und die Aufnahmebedingungen (Nahbesprechungsmikrofon oder Telefonaufnahme, Hintergrundgeräusche etc.).

Der Fortschritt der Spracherkennungsforschung läßt sich anhand der seit 1986 in den USA jährlich stattfindenden, offiziellen Evaluationen dokumentieren, an denen viele bedeutende Spracherkennungsgruppen teilnehmen. Die erste Aufgabe war *Resource Management*, bei der zum ersten Mal auf einem relativ großen Vokabular von 1000 Wörtern sprecherunabhängig und kontinuierlich gesprochene Sätze erkannt werden mußten, die unter guten Aufnahmebedingungen nach einer relativ eingeschränkten Grammatik (Perplexität 60) vorgelesen wurden. Die besten erzielten Erkennungsergebnisse lagen 1991 bei etwa 97% Wortakkuratheit. 1992 folgte die *Wall Street Journal Task* mit sehr großen

Mengen gelesener Zeitungstexte, was in komplexeren Grammatiken und einem praktisch unbegrenzten Vokabular resultierte. Mit Erkennungsvokabularen von 20 000 oder 60 000 Wörtern konnten bei dieser Aufgabe Erkennungsraten von etwa 93% erreicht werden. In der seit 1995 evaluierten *Switchboard Task* wurden zwei neue Schwierigkeitsgrade eingeführt: Statt Texte zu lesen, unterhalten sich zwei Sprecher in spontaner Sprache über ein Thema ihrer Wahl, und zwar per Telefon. Bei dieser wesentlich schwierigeren Aufgabe werden gegenwärtig etwa 70% Wortakkuratheit erzielt. Die neueste, parallel zu Switchboard verfolgte Aufgabenstellung heißt *Marketplace* nach einer gleichnamigen Radiosendung. Die Herausforderung bei Marketplace besteht darin, gleichzeitig mit einem breiten Spektrum von Sprachqualitäten fertig zu werden, die vom geschulten Radiosprecher im ruhigen Studio oder mit Musikhintergrund bis zu Telefongesprächen und Interviews auf der Straße reichen. In Deutschland wird seit 1992 vom BMBF das insgesamt achtjährige Verbundprojekt VERBMOBIL gefördert, in welchem mehrere Universitäten und Industriepartner an der Erkennung und Übersetzung von spontansprachlichen Terminabsprachen arbeiten. Auch der in dieser Arbeit beschriebene Buchstabiererkenner ist in das VERBMOBIL-Projekt integriert.

3.2 Signalvorverarbeitung

Ziel der Signalvorverarbeitung ist es, die akustische Spracheingabe in eine für die Spracherkennung geeignete digitale Repräsentation zu transformieren. Nach der Messung und Diskretisierung des Sprachschalls sollen für die Erkennung nützliche Merkmale extrahiert, und – auch aufgrund der hohen Datenraten – irrelevante Merkmale unterdrückt werden.

3.2.1 Diskretisierung

Ein Mikrofon wandelt die mit einer Schallwelle verbundenen Druckänderungen der Luft in ein kontinuierliches elektrisches Signal $s(t)$. Die Messung von $s(t)$ in äquidistanten Zeitabständen T heißt *Abtastung* mit der Abtastfrequenz $1/T$ und resultiert in einer Folge diskreter Stützstellen $s(kT)$. Die Umwandlung des reellwertigen Signals mit einem Analog-Digital-Wandler (AD-Wandler) ist von beschränkter Genauigkeit, und bewirkt eine *Quantisierung* der Abtastwerte. Dabei wird das Signal üblicherweise durch 2^B Quantisierungsstufen repräsentiert, wobei B je nach Qualität des AD-Wandlers zwischen 8 und 16 liegt – man spricht dann etwa von einem 8- oder 16-bit AD-Wandler.

Bei der Wahl der Abtastfrequenz ist das Abtasttheorem zu beachten: Eine bandbegrenzte Funktion f mit oberer Grenzfrequenz f_G kann aus ihren Abtastwerten $f(kT)$ vollständig rekonstruiert werden, falls die Abtastfrequenz $1/T$ größer ist als die doppelte Grenzfrequenz, d.h. $1/T > 2f_G$.

Eine Bandbreite von 8 kHz und damit eine Abtastfrequenz von 16 kHz ist zur Spracherkennung ausreichend. Für Aufnahmen über das öffentliche Telefonnetz mit einer Frequenzbandbreite von (in Deutschland) etwa 300 bis 3400 Hz ist eine Abtastrate von 8 kHz ausreichend.

3.2.2 Kurzzeitanalyse

Das abgetastete und quantisierte Sprachsignal der Schallwelle liegt als eine Folge $s(kT) = s_1, s_2, \ldots$ von Amplitudenwerten vor. Diese „rohen" Sprachdaten enthalten viele Informationen, die für die Spracherkennung als wenig relevant betrachtet werden, beispielsweise die Phaseninformation. Weiterhin möchte man die hohen Datenraten reduzieren – bei einer Abtastfrequenz von 16 kHz fallen mit einem 16-Bit AD-Wandler jede Sekunde 32 kByte an Daten an –, soweit dies ohne Erkennungsverluste möglich ist.

Für die *Kurzzeitanalyse* nimmt man an, daß das Signal über kurze Zeiträume von etwa 5 bis 30 Millisekunden näherungsweise stationär ist. Aufeinanderfolgende Zeitfenster dieser Länge werden mit einer geeigneten Fensterfunktion (Hamming-, Hanning-, oder Gauß-Fenster) aus dem Signal ausgeblendet und analysiert.

Merkmale können direkt im Zeitbereich (aus der Wellenform) oder im Spektralbereich gewonnen werden. Zeitbereichsmerkmale (z.B. Energie, Anzahl der Nulldurchgänge, Peak-to-Peak, Autokorrelation, siehe [SR75]) spielen in heutigen Erkennern eine untergeordnete Rolle und werden nur noch als zusätzliche Merkmale genutzt. Die gegenwärtig verbreitetsten Verfahren basieren auf:

- **Melscale-Spektral-Koeffizienten.** Die mit einer Fouriertransformation gewonnenen Spektralkoeffizienten werden zu gehörphysiologisch motivierten Energiebändern zusammengefaßt.
- **Cepstral-Koeffizienten** machen Gebrauch von der Modellannahme, daß das Sprachsignal $s(t) = e(t) \otimes h(t)$ aus einer Faltung der Stimmanregung $e(t)$ mit der Impulsantwort des Vokaltraktes $h(t)$ entsteht. Mit Hilfe einer *Homomorphen Analyse* versucht man, diese Komponenten wieder zu trennen: Nach einer Fouriertransformation $\mathcal{F}$ gilt im Frequenzraum $\mathcal{F}(s(t)) = \mathcal{F}(e(t)) \cdot \mathcal{F}(h(t))$, und nach Logarithmierung und einer inversen Fouriertransformation erhält man die als *Cepstrum* bezeichnete, additive Zerlegung der Komponenten e und h

$$\mathcal{F}^{-1}(\log \mathcal{F}(s)) = \mathcal{F}^{-1}(\log \mathcal{F}(e)) + \mathcal{F}^{-1}(\log \mathcal{F}(h)) \tag{3.2}$$

  Für stimmhafte Laute ist der Verlauf des Spektrums $\mathcal{F}(s(t))$ geprägt durch periodische Spitzen, die von der Anregungsgrundfrequenz und ihren Harmonischen herrühren und die Einhüllende des Spektrums überlagern. Die inverse Fouriertransformation in Gl. (3.2) leistet eine „Spektralanalyse des Spektrums" [ST95], in der sich die Grundfrequenz von e in den hochfrequenten, und die durch h charakterisierte Einhüllende in den niedrigen Koeffizienten widerspiegelt.

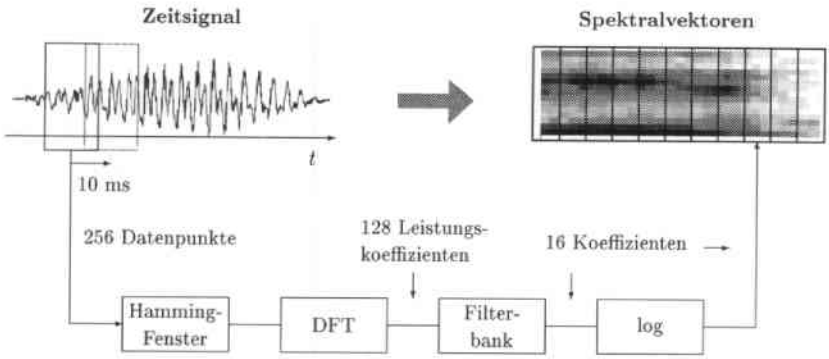


Abbildung 3.1: Vorverarbeitung: Kurzzeit-Spektralanalyse des Sprachsignals

- Die **Lineare Prädiktion** beruht darauf, die Abtastwerte $s(t)$ in einem als stationär betrachteten Kurzzeitfenster durch Vorhersagekoeffizienten α_j zu beschreiben:

$$s(t) = \sum_{j=1}^{P} \alpha_j s(t-j)$$

Die oben vorgestellten Koeffizientenvektoren sind stationäre Momentaufnahmen und werden meist durch dynamische Merkmale ergänzt, die den zeitlichen Verlauf charakterisieren. Dazu wird im einfachsten Fall ein Merkmalsvektor um seine zeitlichen Nachbarn erweitert, oder es werden als Annäherung an die erste Ableitung *Delta-* oder *Differenzen-Koeffizienten* von benachbarten Merkmalsvektoren gebildet.

Abbildung 3.1 illustriert die Berechnung von Melscale-FFT-Koeffizienten, wie sie in dieser Arbeit eingesetzt wird. In zeitlichen Abständen von 10 ms wird aus den rohen Sprachdaten mit einem Hamming-Fenster ein 16 ms oder N = 256 Datenpunkte breites[2] Analysefenster ausgeblendet, aus dem mit einer diskreten Fouriertransformation 256 komplexwertige Fourierkoeffizienten $(re(\omega_i), im(\omega_i))$ gewonnen werden. Diese werden unter Verlust der Phaseninformation in ein symmetrisches, reellwertiges Leistungsspektrum $|\omega_i| = \sqrt{re(\omega_i)^2 + im(\omega_i)^2}$ überführt. Die relevanten 128 Koeffizienten des Leistungsspektrums werden nach der gehörorientierten *Mel-Skala* zu 16 Frequenzbändern zusammengefaßt. Dabei wird in den tieferen Frequenzen feiner aufgelöst, mit höheren Frequenzen wachsen der Mittenabstand und die Breite der Bandpaßfilter. Schließlich werden die Energien dieser 16 Kanäle mittels Logarithmierung in ein pegelähnliches Maß transformiert.

[2]Bei einer Abtastfrequenz von 16 kHz. N = 128 für 8 kHz Abtastrate.

3.2.3 Merkmalstransformationen

Zwischen die Merkmalsextraktion durch die Kurzzeitanalyse und die Spracherkennung können weitere Verarbeitungsschritte geschaltet sein mit einem oder mehreren der folgenden Ziele.

Dimensionalitätsreduktion

Ein *Vektorquantisierer* ist eine Abbildung $\mathbb{R}^D \rightarrow Z = \{z_1, \ldots, z_K\}$, die jedem Merkmalsvektor $\boldsymbol{x} \in \mathbb{R}^D$ einen *Prototypenvektoren* $q(\boldsymbol{x}) = z_k$ zuordnet. Ziel ist es, die Prototypenvektoren des sogenanntes *Codebuchs* Z und eine Abbildung q bzw. die durch q induzierte Partitionierung des Merkmalsraums so zu bestimmen, daß $\boldsymbol{x}$ durch $q(\boldsymbol{x})$ im Mittel möglichst genau approximiert wird. Offensichtlich wird ein optimaler Vektorquantisierer einen Vektor $\boldsymbol{x}$ immer auf den nächstliegenden Prototypenvektor abbilden, wodurch die Abbildung q durch die Wahl der Codebuchvektoren bestimmt ist. Mit iterativen Ballungsverfahren wie dem LBG-Algorithmus (nach den Autoren Linde, Buzo und Gray) kann ein lokal optimales Codebuch gefunden werden.

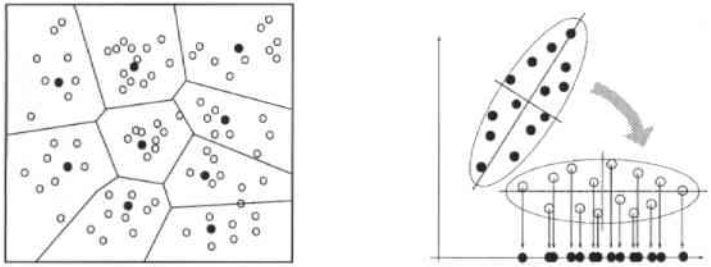


Abbildung 3.2: Dimensionalitätsreduktion durch Vektorquantisierung (links) oder Hauptachsentransformation (rechts)

Eine weniger drastische Dimensionalitäts- und Informationsreduzierung wird durch die *Karhunen-Loéve (KL)-* oder *Hauptachsentransformation* erzielt. Dabei werden die Merkmalsvektoren $\boldsymbol{x}$ so transformiert, daß ihre Koeffizienten x_i unkorreliert sind und die Merkmalsvarianz für x_1 am größten ist und sich mit zunehmendem i verringert. Das Prinzip der Vektorquantisierung und der KL-Transformation ist in Abbildung 3.2 veranschaulicht.

In der KL-Transformation werden die Klassenzugehörigkeiten der Merkmalsvektoren nicht berücksichtigt, entsprechend wird durch die resultierende Rotation nicht notwendigerweise die Trennschärfe zwischen den Klassen verbessert. Statt der KL-Transformation wird daher meist eine *Lineare Diskriminanzanalyse (LDA)* durchgeführt. Dazu wird

nach einer Rotation bzw. nach einem reduzierten Basissystem gesucht, das die Klassengebiete kompakt und gleichzeitig ihre Zentren voneinander entfernt hält. Es wird also simultan die durchschnittliche Varianz innerhalb der Klassengebiete minimiert, bei gleichzeitiger Aufspreizung der Klassenzentren.

Adaption

Bevor das Sprachsignal digitisiert werden kann, durchläuft es eine Reihe von Kanälen, in denen es modifiziert wird, etwa durch die Mikrofoncharakteristik, die Raumakustik oder eine Telefonleitung. Gemeinhin werden die Kanaleigenschaften als lineare Filterung (Faltung, $\otimes$) mit der Impulsantwort $h(t)$ des Kanals sowie einer additiven Rauschkomponente $n(t)$ modelliert:

$$s(t) = x(t) \otimes h(t) + n(t)$$

Dann versucht man, die störenden Komponenten $h(t)$ und $n(t)$ durch Normierungs- oder Adaptionsverfahren weitestmöglich zu unterdrücken. Auch der Sprecher selbst kann als Übertragungskanal interpretiert und in die Adaption einbezogen werden, beispielsweise durch eine Vokaltraktnormalisierung [ZW97].

3.3 Akustische Prototypen

Aus den Anfängen der Spracherkennung stammt die Klassifikation mit der „Nächster-Nachbar-Regel". Für jedes zu erkennende Wort w_i wird ein akustischer Prototyp[3] als Referenzmuster $\mathbf{Y}_i$ abgespeichert. Als erkanntes Wort w_{k^*} entscheidet man sich für das ähnlichste Referenzmuster $\mathbf{Y}_{k^*}$ mit

$$k^* = \operatorname*{argmin}_k D(\mathbf{X}, \mathbf{Y}_k)$$

Der Abstand $D(\mathbf{X}, \mathbf{Y})$ zwischen der Eingabe $\mathbf{X} = (\boldsymbol{x}_1 \dots \boldsymbol{x}_N)$ der Länge N und einem Referenzmuster $\mathbf{Y} = (\boldsymbol{y}_1 \dots \boldsymbol{y}_M)$ der Länge M wird bestimmt mittels *Dynamischer Zeitverzerrung*, in der Literatur mit **DTW** für *dynamic time warping* abgekürzt. Als lokales Abstandsmaß $d(i, j)$ zwischen zwei Sprachvektoren $\boldsymbol{x}_i, \boldsymbol{y}_j \in \mathbb{R}^d$ wird meistens die euklidische Distanz benutzt.

Der Gesamtabstand $D(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{N} d(i, \pi(i))$ zwischen Eingabe- und Referenzmuster wird als Summe lokaler Distanzen $d(i, j)$ bestimmt. Für Sprachmuster unterschiedlicher Länge ($M \neq N$) muß $\mathbf{X}$ in seiner ganzen Länge so auf $\mathbf{Y}$ abgebildet werden, daß die entsprechende Summe der lokalen Distanzen minimiert wird, wie in Abbildung 3.3 schematisiert. Die zeitliche Abbildung $\pi : \{1 \dots N\} \to \{1 \dots M\}$, die jedem Vektor der

[3] Oder mehrere Prototypen von verschiedenen Sprechern

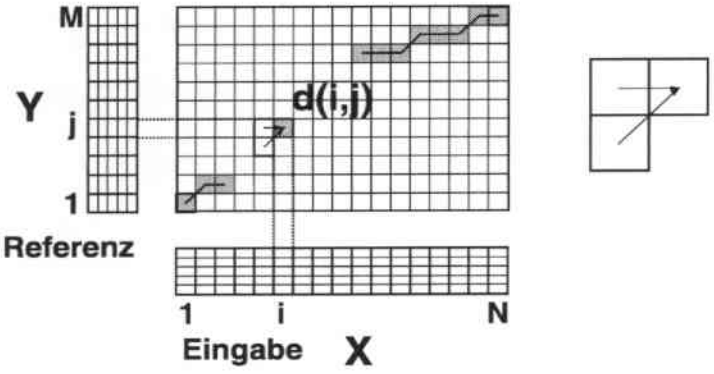


Abbildung 3.3: Dynamische Zeitverzerrung zwischen Eingabesignal und Referenzmuster

Eingabe einen Vektor der Referenz zuweist, nennen wir einen Pfad $\boldsymbol{\pi} = \pi(1)\ldots\pi(N)$. Aus der Natur des Problems folgen als Einschränkungen für die Pfadfunktion, daß sie monoton wachsend und von beschränkter Steigung sein muß. Die Bestimmung des optimalen Pfades $\boldsymbol{\pi}^*$

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \sum_{i=1}^{N} d(i, \pi(i))$$

ist ein Optimierungsproblem, das trotz einer mit M exponentiell wachsenden Anzahl möglicher Pfade mit einem Aufwand von $O(N * M)$ gelöst werden kann. Dazu wird nach dem Prinzip der dynamischen Programmierung [Bel57] eine optimale partielle Distanz $D(i,j)$ rekursiv wie folgt berechnet:

$$D(i,j) = \begin{cases} d(1,1) & i = j = 1 \\ \infty & i = 1, j > 1 \\ d(i,j) + \min\{D(i-1,j), D(i-1,j-1)\} & \text{sonst} \end{cases} \quad (3.3)$$

Die gesuchte Gesamtdistanz $D(\mathbf{X}, \mathbf{Y})$ ist somit $D(N, M)$. Der Minimumausdruck in (3.3) bestimmt die lokalen Transitionen des Pfades, die hier mit nur zwei möglichen Schritten (Abbildung 3.3 rechts) sehr einfach gewählt sind. In [SC78] werden verschiedene Transitionstypen ausführlich diskutiert.

Der Mustervergleich mit akustischen Prototypen eignet sich nur bedingt für eine sprecherunabhängige Erkennung, da in diesem Fall für jeden oder zumindest für viele unterschiedliche Sprecher ein eigener Prototyp benötigt wird. Außerdem muß für jedes neu ins Erkennervokabular aufgenommene Wort ein neuer Prototyp zur Verfügung gestellt werden. Diese Nachteile werden mit den statistischen, im nächsten Abschnitt vorgestellten Markov-Modellen oder mit konnektionistischen Ansätzen (Kapitel 4, 6) überwunden.

3.4 Hidden Markov Models (HMMs)

Markov Modelle, weithin als HMM für *Hidden Markov Model* bekannt, wurden Mitte der achtziger Jahre erstmals zur Spracherkennung eingesetzt [Bak75, Jel76]. Als die inzwischen am weitesten verbreitete Technik zur Spracherkennung sind HMMs in der Standardliteratur [WL90, ST95] ausführlich beschrieben. Die folgende Einführung lehnt sich an die Notation in [Rab89, ST95] an.

Ähnlich wie beim Abstandsklassifikator wird auch in einem HMM jedes zu erkennende Wort w_i durch ein Modell repräsentiert, das jetzt nicht mehr ein abgespeicherter akustischer Prototyp $\mathbf{Y}_i$, sondern ein abstrakteres statistisches Modell $\boldsymbol{\lambda}_i$ ist, dessen Parameter automatisch erlernt werden können.

Der Einfachheit halber beschränken wir uns vorerst auf Einzelworterkennung. Gesucht ist das Wort w_i aus dem Erkennungswortschatz $W = \{w_1, w_2, \ldots\}$, das für eine gegebene akustische Spracheingabe $\mathbf{X}$ die höchste a posteriori Wahrscheinlichkeit besitzt:

$$P(w_i|\mathbf{X}) = \frac{p(\mathbf{X}|w_i)P(w_i)}{p(\mathbf{X})}$$

Der Nenner $p(\mathbf{X})$ ist von w unabhängig, leistet also keinen Beitrag zur Entscheidung. $P(w_i)$ wird durch das Sprachmodell bestimmt. Die Aufgabe des HMMs $\boldsymbol{\lambda}_i$ ist es, die Verteilungsdichte $p(\mathbf{X}|w_i)$ für die akustische Realisierung $\mathbf{X}$ des Wortes w_i durch $p(\mathbf{X}|\boldsymbol{\lambda}_i)$ zu modellieren. $p(\mathbf{X}|\boldsymbol{\lambda}_i)$ wird in einem zweistufigen Zufallsprozeß berechnet: In jedem Zeitschritt wird zuerst ein interner Zustand des HMMs bestimmt und davon abhängig die Wahrscheinlichkeit für die Ausgabe eines Sprachvektors.

3.4.1 Definition des HMMs

Ein HMM $\boldsymbol{\lambda}$ wird beschrieben durch eine Menge S von N Zuständen

$$S := \{S_1, S_2, \ldots, S_N\} \tag{3.4}$$

Diese werden in einem diskreten stochastischen Prozeß durchlaufen, wobei eine Zustandsfolge

$$\boldsymbol{q} = q_1 q_2 \ldots q_T\,, \quad q_t \in S \tag{3.5}$$

entsteht. Die Wahl des ersten Zustandes q_1 wird durch $\boldsymbol{\pi} = \{\pi_i\}$, der Wechsel von einem Zustand in den nächsten durch eine $N \times N$-Matrix $\mathbf{A} = (a_{ij})$ von *Übergangs-* oder *Transitionswahrscheinlichkeiten* bestimmt:

$$\pi_i = P(q_1 = S_i)\,, \quad i = 1 \ldots N \tag{3.6}$$

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i)\,, \quad i, j = 1 \ldots N \tag{3.7}$$

Dieser durch $\mathbf{A}$ und $\boldsymbol{\pi}$ charakterisierte Prozeß heißt *Markovkette*. In der zweiten Stufe des Zufallsprozesses gibt das HMM in jedem Zeitschritt eines von K Symbolen des Observationsalphabets V aus,

$$V := \{v_1, v_2, \ldots, v_K\}. \tag{3.8}$$

Die Wahrscheinlichkeit, in einer beobachteten Folge $O = o_1 o_2 \ldots o_T$ die Ausgabe $o_t = v_k$ zu beobachten, ist nur vom aktuellen Zustand q_t abhängig und wird durch eine Matrix $\mathbf{B} = (b_j(k))$ von *Beobachtungs-* oder *Emissionswahrscheinlichkeiten* beschrieben:

$$b_j(k) = P(o_t = v_k | q_t = S_j)\,, \quad j = 1 \ldots N,\ k = 1 \ldots K \tag{3.9}$$

Die Wahrscheinlichkeitsverteilungen $\mathbf{A}$, $\mathbf{B}$ und $\boldsymbol{\pi}$ müssen den folgenden Bedingungen genügen:

$$\begin{aligned} a_{ij}, \pi_i, b_i(k) &\geq 0\,, && i, j = 1 \ldots N\,,\ k = 1 \ldots K \\ \sum_{j=1}^{N} a_{ij} = \sum_{j=1}^{N} \pi_j = \sum_{k=1}^{K} b_i(k) &= 1\,, && i = 1 \ldots N \end{aligned}$$

Der Zufallsprozeß ist stationär, denn der absolute Zeitpunkt t spielt keine Rolle. Außerdem wird vereinfachend angenommen, daß für die nächste Entscheidung nur der unmittelbar vorangehende Zustand von Bedeutung ist (HMM erster Ordnung):

$$P(q_t | q_1 \ldots q_{t-1}, o_1 \ldots o_{t-1}) = P(q_t | q_{t-1}) \tag{3.10}$$

$$P(o_t | q_1 \ldots q_t, o_1 \ldots o_{t-1}) = P(o_t | q_t) \tag{3.11}$$

Durch die Menge der Zustände S, das Ausgabealphabet V sowie die Parameter $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ ist ein HMM vollständig definiert. Um nun eine Observationssequenz $O = o_1 o_2 \ldots o_T$ zu generieren, führt man die folgenden Schritte für $t = 1 \ldots T$ aus:

1. Falls $t = 1$ (Startzeitpunkt), wähle einen initialen Startzustand $q_1 = S_i$ gemäß der Wahrscheinlichkeitsverteilung $\boldsymbol{\pi}$. Ansonsten gehe vom aktuellen Zustand q_t in einen neuen Zustand q_{t+1} über, der gemäß den in $\mathbf{A}$ gegebenen Übergangswahrscheinlichkeiten $a_{q_t, q_{t+1}}$ gewählt wird.

2. Wähle ein Symbol $o_t = v_k$ gemäß den Emissionswahrscheinlichkeiten $b_i(v_k)$ des aktuellen Zustands S_i.

In Abbildung 3.4 ist ein HMM anhand eines Urnenbeispiels veranschaulicht. Das abgebildete HMM besitzt zwei Zustände S_1 und S_2, zwischen denen mit Transitionswahrscheinlichkeiten a_{ij} umgeschaltet wird. Das Observationsalphabet besteht aus 3 Symbolen R,G,B, nämlich roten, grünen und blauen Kugeln. Nachdem ein neuer Zustand

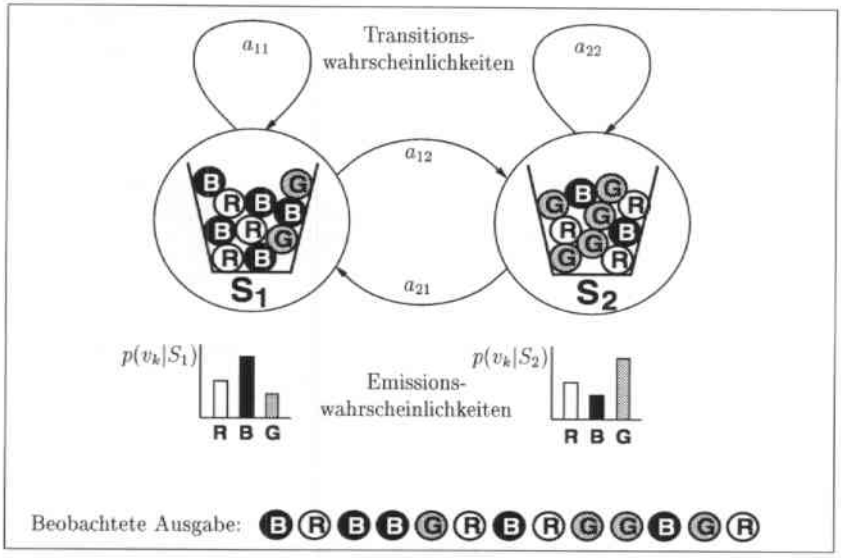


Abbildung 3.4: HMM am Beispiel eines Urnenmodells. Ein Beobachter sieht nur die Folge der ausgegebenen Kugeln, die je nach internem Modellzustand (S_1, S_2) aus unterschiedlichen Urnen gezogen werden.

eingenommen wurde, wird aus der entsprechenden Urne zufällig (mit Zurücklegen) eine Kugel gezogen. Die vom Zustand S_i abhängige Wahrscheinlichkeit, eine bestimmte Farbe v_k zu ziehen, entspricht den Emissionswahrscheinlichkeiten $b_i(k) = P(o_t = v_k|q_t = S_i)$. Ein externer Beobachter sieht nur die Folge der ausgegebenen Kugeln $O = o_1 o_2 \ldots o_T$, nicht aber, in welchem Zustand sie gezogen wurden. Aus dieser Tatsache rührt die Bezeichnung „Hidden" im englischen Namen des HMM.

3.4.2 Der „Vorwärts-Algorithmus"

Wie groß ist die Wahrscheinlichkeit, daß eine bestimmte Beobachtungsfolge $O = o_1 o_2 \ldots o_T$ von einem gegebenen Modell $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ erzeugt wurde? Dabei kann dieselbe Sequenz O beim Durchlaufen unterschiedlicher Zustandsfolgen generiert werden.

Unter der zusätzlichen Bedingung einer konkreten Zustandsfolge $\boldsymbol{q} = q_1 q_2 \ldots q_T$ kann $P(O|\boldsymbol{\lambda}, \boldsymbol{q})$ durch Aufmultiplizieren der entsprechenden Übergangs- und Emissionswahr-

scheinlichkeiten entlang von q berechnet werden:

$$P(O|\boldsymbol{\lambda}, \boldsymbol{q}) = \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(o_t) \tag{3.12}$$

Die Wahrscheinlichkeit, daß O auf irgendeinem aller möglichen Pfade $\boldsymbol{q} \in S^T$ erzeugt wurde, ist

$$P(O|\boldsymbol{\lambda}) = \sum_{\boldsymbol{q} \in S^T} P(O|\boldsymbol{\lambda}, \boldsymbol{q}) = \sum_{\boldsymbol{q} \in S^T} \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(o_t) \tag{3.13}$$

Die Anzahl möglicher Zustandsfolgen und damit der Aufwand $O(TN^T)$ zur Berechnung von (3.13) wächst exponentiell mit der Länge T. Wiederum kann mit dem Prinzip der dynamischen Programmierung ein wesentlich effizienterer Algorithmus formuliert werden. Dazu werden *Vorwärtswahrscheinlichkeiten* α als Hilfsgrößen definiert

$$\alpha_t(j) = P(o_1 \ldots o_t,\, q_t = S_j|\boldsymbol{\lambda})\,, \quad t = 1 \ldots T\,, \quad j = 1 \ldots N\,, \tag{3.14}$$

die rekursiv mit Aufwand $O(T)$ berechnet werden können:

$$\alpha_t(j) = \begin{cases} \pi_j b_j(o_1) & \text{falls } t = 1 \\ \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) & \text{falls } t > 1 \end{cases} \tag{3.15}$$

Dann gilt

$$P(O|\boldsymbol{\lambda}) = \sum_{j=1}^{N} \alpha_T(j) \tag{3.16}$$

3.4.3 Der Viterbi-Algorithmus

In der kontinuierlichen Spracherkennung wollen wir aus den in Abschnitt 3.5 angeführten Gründen nicht die Gesamtwahrscheinlichkeit $P(O|\boldsymbol{\lambda})$, sondern denjenigen Pfad $\boldsymbol{q}^*$ bestimmen, der O am wahrscheinlichsten verursachte:

$$\boldsymbol{q}^* = \operatorname*{argmax}_{\boldsymbol{q} \in S^T} P(\boldsymbol{q}|O, \boldsymbol{\lambda}) = \operatorname*{argmax}_{\boldsymbol{q} \in S^T} \frac{P(O, \boldsymbol{q}|\boldsymbol{\lambda})}{P(O|\boldsymbol{\lambda})} = \operatorname*{argmax}_{\boldsymbol{q} \in S^T} P(O, \boldsymbol{q}|\boldsymbol{\lambda}) \tag{3.17}$$

Letztere Umformung ist möglich, da $P(O|\boldsymbol{\lambda})$ für den Vergleich verschiedener Pfade keine Rolle spielt. $\boldsymbol{q}^*$ wird berechnet mit einer Variante des Vorwärtsalgorithmus, dem *Viterbi-Algorithmus*. Statt der partiellen Gesamtwahrscheinlichkeit $\alpha_j(t)$ wird

$$\theta_t(j) = \max_{\boldsymbol{q} \in S^t} P(o_1 o_2 \ldots o_t,\, q_1 q_2 \ldots q_{t-1},\, q_t = S_j|\boldsymbol{\lambda}) \tag{3.18}$$

berechnet, und in der Rekursion (3.15) wird die Summe durch eine Maximumbildung ersetzt:

$$\theta_t(j) = \begin{cases} \pi_j b_j(o_1) & \text{falls } t = 1 \\ \max_{i=1}^{N} \theta_{t-1}(i) a_{ij} b_j(o_t) & \text{falls } t > 1 \end{cases} \tag{3.19}$$

und es ist

$$P^*(O|\boldsymbol{\lambda}) = \max_{j=1}^{N} \theta_T(j) \tag{3.20}$$

$P^*(O|\boldsymbol{\lambda}) := P(O, \boldsymbol{q}^*|\boldsymbol{\lambda})$ ist stark korreliert mit und meist eine ausreichende Näherung für $P(O|\boldsymbol{\lambda})$.

Numerische Probleme bei der Multiplikation vieler sehr kleiner Wahrscheinlichkeiten können durch eine Logarithmierung umgangen werden. Damit erhält die Rekursion in (3.19) die Form

$$\max_{i=1}^{N} \{\theta'_{t-1}(i) + a'_{ij}\} + b'_j(o_t) \tag{3.21}$$

und ist, von den Transitionswahrscheinlichkeiten abgesehen, mit der Berechnung des DTW (Gl. (3.3) in Abschnitt 3.3) praktisch identisch.

3.4.4 Der Baum-Welch-Algorithmus

HMMs werden nach dem Maximum-Likelihood-Verfahren trainiert, d.h. man versucht, die Parameter des Modells so einzustellen, daß sie die Lernstichprobe bestmöglich erklären. Es sei $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ ein Modell für ein Wort w und O ein Trainingsmuster, d.h. eine akustische Realisierung von w. Gesucht sind die Parameter $\boldsymbol{\lambda}^*$, die die Produktionswahrscheinlichkeit von O maximieren:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\, P(O|\boldsymbol{\lambda}) \tag{3.22}$$

Für einfache Verteilungsfunktionen, beispielsweise eine Gaußfunktion $N(\mu, \sigma)$, kann ein optimaler Parametersatz (μ^*, σ^*) zur Optimierung von $P(o_t|\mu, \sigma)$ in einer geschlossenen Form bestimmt werden. Aufgrund des zweistufigen Zufallsprozesses besitzt ein HMM jedoch verborgene Variablen, daher verschließt sich die Berechnung der optimalen Parameter $\boldsymbol{\lambda}^*$ einer analytischen Lösung.

Unter dem Namen *Baum-Welch-* oder *forward-backward-*Algorithmus ist ein Verfahren zur iterativen Schätzung bekannt, mit dem zumindest ein lokales Optimum berechnet werden kann. Anhand initialer Parameter $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ können aus den Trainingsdaten neue Parameter $\boldsymbol{\lambda}' = (\mathbf{A}', \mathbf{B}', \boldsymbol{\pi}')$ geschätzt werden, von denen man zeigen kann, daß entweder $\boldsymbol{\lambda}' = \boldsymbol{\lambda}$ oder $P(O|\boldsymbol{\lambda}') > P(O|\boldsymbol{\lambda})$. Um $\boldsymbol{\lambda}'$ zu bestimmen, werden die im Training

unter Annahme von $\boldsymbol{\lambda}$ durchlaufenen Zustände und Zustandsübergänge gezählt. Mit den in [Rab89, ST95] aufgeführten Schätzformeln können daraus neue Werte für $\mathbf{A}', \mathbf{B}'$ und $\boldsymbol{\pi}'$ berechnet werden.

Mit dem in Abschnitt 2.5.4 eingeführten *Maximum-Mutual-Information (MMI)*-Kriterium ist es möglich, einen HMM-Erkenner auch diskriminativ zu trainieren. Allerdings ist ein MMI-Training sehr aufwendig, denn es müssen nicht nur die Wahrscheinlichkeiten für das korrekte, sondern für alle konkurrierenden Modelle berechnet werden. Zur Maximierung des Kriteriums sind keine analytischen Lösungen oder Schätzformeln bekannt. Es muß daher ein komplexes Gradientenabstiegsverfahren implementiert werden, dessen Konvergenz nicht garantiert werden kann. In der Praxis wird deshalb MMI-Training nur für Erkenner mit relativ kleinen Wortschätzen und oft nur als zusätzlicher Verfeinerungsschritt nach dem regulären ML-Training eingesetzt.

3.4.5 Kontinuierliche und semikontinuierliche HMMs

Diskrete HMMs arbeiten auf Symbolfolgen $O = o_1 o_2 \ldots o_T$, die durch eine Vektorquantisierung aus einer Folge mehrdimensionaler, reellwertiger Merkmalsvektoren $\mathbf{X} = \boldsymbol{x}_1 \boldsymbol{x}_2 \ldots \boldsymbol{x}_T$, $\boldsymbol{x}_i \in \mathbb{R}^d$ gewonnen werden. Die in Abschnitt 3.2.3 beschriebene Quantisierung, eine Transformation $\mathbb{R}^d \rightarrow \{1, \ldots, K\}$, reduziert die zu verarbeitenden Datenmengen erheblich, ist aber mit einem Informationsverlust[4] verbunden. **Kontinuierliche Markovmodelle** modellieren die Ausgabeverteilungsdichten direkt über dem akustischen Merkmalsraum $\mathbb{R}^d$. Statt einer Wahrscheinlichkeitstabelle $b_j(k)$ über den diskreten Symbolen des Ausgabealphabets $V = \{v_1, \ldots, v_K\}$ besitzt ein Zustand S_j nun eine Wahrscheinlichkeitsdichte $b_j(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^d$. Typischerweise wird $b_j(\boldsymbol{x})$ durch die in Abschnitt 2.3.1 vorgestellte und in Abbildung 2.2 illustrierte Gaußsche Mischverteilung modelliert, wobei L_j die Anzahl der in Zustand S_j eingesetzten Normalverteilungen ist:

$$b_j(\boldsymbol{x}) = \sum_{l=1}^{L_j} c_{jl} N(\boldsymbol{x} | \boldsymbol{\mu}_{jl}, \boldsymbol{\Sigma}_{jl}) \quad , \quad \sum_{l=1}^{L_j} c_{jl} = 1 \tag{3.23}$$

Die gewichtete Summe hinreichend vieler Normalverteilungen erlaubt die Approximation beliebiger Dichtefunktionen; in der Praxis benutzt man Mischverteilungen aus größenordnungsmäßig 5 bis 100 Normalverteilungen.

Semikontinuierliche Markovmodelle verbinden die Vorteile kontinuierlicher Dichten mit dem wesentlich ökonomischeren Parametereinsatz eines diskreten HMMs. Wie zuvor besitzt jeder Zustand individuelle Mischgewichte c_{jl}, die Normalverteilungen werden jedoch global von allen Zuständen gemeinsam genutzt. Die Berechnung von $b_j(\boldsymbol{x})$ unterscheidet sich von Gl. (3.23) nur in der Indizierung der L Normalverteilungen, die

[4] Typische Größenordnungen sind etwa die Reduzierung eines $d = 16$-dimensionalen kontinuierlichen Raumes auf $K = 256$ diskrete Werte, wodurch große Quantisierungsfehler entstehen.

nun nicht mehr von den Zuständen S_j abhängig sind:

$$b_j(\boldsymbol{x}) = \sum_{l=1}^{L} c_{jl} N(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) \quad , \quad \sum_{l=1}^{L} c_{jl} = 1 \tag{3.24}$$

3.5 Kontinuierliche Spracherkennung

In der kontinuierlichen Spracherkennung will man statt Einzelwörtern w_i ganze Sätze, also Wortsequenzen $(w_{i_1} w_{i_2} \ldots)$ erkennen. Eine Erkennung ganzer Sätze nach dem Prinzip der Einzelworterkennung ist impraktikabel – es müßten dazu für alle denkbaren Wortsequenzen $(w_{i_1} w_{i_2} w_{i_3} \ldots)$ die Produktionswahrscheinlichkeiten $P(O|(\boldsymbol{\lambda}_{i_1} \boldsymbol{\lambda}_{i_2} \boldsymbol{\lambda}_{i_3} \ldots))$ von entsprechend verketteten Markov-Modellen berechnet werden.

3.5.1 Viterbi-Suche

Statt ganzer Satzmodelle konstruiert man deshalb mit einer rückgekoppelten Parallelschaltung aller Wortmodelle $\boldsymbol{\lambda}_i$ ein HMM $\boldsymbol{\lambda}$, das beliebige Wortsequenzen modellieren kann, wie in Abbildung 3.5(links) am Beispiel von drei Wörtern A,B,C illustriert. Eine Berechnung von $P(O|\boldsymbol{\lambda})$ läßt jedoch keinen Rückschluß auf die tatsächliche Zustands- bzw. Wortfolge zu, mit der O erzeugt wurde. Deshalb bestimmt man mit dem Viterbi-Pfad die wahrscheinlichste Zustandsfolge und betrachtet die dadurch definierte Wortfolge als Erkennungsergebnis. Wie zuvor kann der Viterbi-Pfad in $O(T \cdot N)$ Schritten berechnet werden, wobei N nun die Gesamtzahl der Zustände aller Wörter im Erkennungsvokabular ist. Der Suchprozeß ist in Abbildung 3.5(rechts) illustriert: Innerhalb eines Wortmodells werden die Viterbi-Pfade wie bei der Einzelworterkennung berechnet, wobei nun zusätzlich aus dem letzten Zustand des Wortes in den Anfangszustand eines neuen Wortes gesprungen werden kann. Prinzipiell kann zu jedem Zeitpunkt ein Wort neu beginnen. Als Vorgängerzustand wählt man das Modell $\boldsymbol{\lambda}_i$ mit der höchsten Bewertung im letzten Zustand.

Meist werden in der Suche statistische Sprachmodelle eingesetzt. Dadurch wird die Wahrscheinlichkeit für das nächste zu durchlaufende Wort vom vorausgehenden Text beeinflußt, und zwar direkt im Suchprozeß beim Sprung in ein neues Modell.

3.5.2 Messung der Fehlerraten

Neben Verwechslungen treten in der kontinuierlichen Spracherkennung Einfügungen und Auslassungen von Wörtern als neue Fehlerquelle hinzu. Um diese zu messen, stellt man den Referenzsatz und die erkannte Hypothese mit Hilfe dynamischer Programmierung einander so gegenüber, daß die Gesamtzahl von Verwechslungen (N_{sub}) sowie irrtümlichen Einfügungen (N_{ins}) und Auslassungen (N_{del}) minimiert wird. Der so

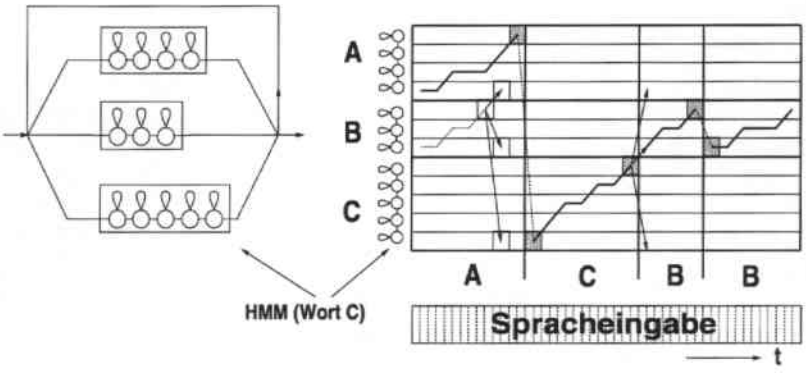


Abbildung 3.5: Verkettetes HMM zur Erkennung von Wortsequenzen (links) und Viterbi-Pfade durch die entsprechende Suchmatrix (rechts)

ermittelte „Abstand" $N_{sub} + N_{ins} + N_{del}$ zwischen Referenz und Hypothese wird anschaulich auch als *Editierdistanz* bezeichnet. Damit erhält man als Gütekriterium die *Wortakkuratheit (WA)*

$$WA = \frac{N - N_{sub} - N_{ins} - N_{del}}{N}, \tag{3.25}$$

wobei N die Anzahl der Wörter im Referenzsatz ist. Bei Buchstabensequenzen sprechen wir entsprechend von *Buchstabenakkuratheit (BA)*. Das Beispiel

```
S T U T T G A R T
| | |  \ \ \  | |
S T U U T T G R D
```

enthält je eine Einfügung (`U`), Auslassung (`A`) und Verwechslung (`T↔D`), es ergibt sich eine BA von $(9 - 1 - 1 - 1)/9 = 66.7\%$.

In Abgrenzung von der Akkuratheit spricht man von „% korrekt", wenn Einfügungs- und Auslassungsfehler unberücksichtigt bleiben, etwa weil bei bekannten Buchstabengrenzen keine auftreten können. Die im Verlauf der Arbeit immer wieder vorkommenden Maße und ihre Abkürzungen seien hier noch einmal zusammengefaßt:

% BA Buchstabenakkuratheit (inkl. Einfügungen und Auslassungen)
% BK Buchstaben korrekt (keine Einf. und Ausl., da Grenzen bekannt)
% NK Namen korrekt (vollständig korrekt erkannte Buchstabensequenzen)

3.6 Sprachmodelle

In der Suche wird diejenige Wortsequenz gesucht, die den Ausdruck $P(\mathbf{X}|W) \cdot P(W)$ in Gl. (3.1) maximiert. Die Aufgabe der Sprachmodellierung ist, die a priori Wahrscheinlichkeiten $P(W)$ für Wortsequenzen $W = w_1 w_2 w_3 \ldots$ bereitzustellen. $P(W)$ ist unabhängig von $\mathbf{X}$ und kann daher losgelöst von der akustischen Modellierung betrachtet werden. Für einen Einzelworterkenner braucht nur eine a priori Wahrscheinlichkeit $P(w_i)$ für jedes Wort des Erkennungsvokabulars bestimmt zu werden. Problematischer wird es, wenn wir im Falle kontinuierlicher Erkennung die Wahrscheinlichkeiten für ganze Sätze schätzen müssen. Die Verfahren hierzu lassen sich in zwei Hauptrichtungen einteilen. Die **linguistischen Ansätze** versuchen, mit Hilfe etwa von Grammatiken syntaktisches und semantisches Wissen über die Struktur der Sprache explizit zu kodieren. Ihnen gegenüber stehen die **statistischen Sprachmodelle**, die aus möglichst großen Textkorpora Wahrscheinlichkeiten für Wortübergänge bestimmen. Mit dem Einsatz stochastischer Grammatiken in den linguistischen sowie weitreichenden und klassenbasierten Sprachmodellen in den statistischen Ansätzen scheinen sich die beiden Lager aufeinander zuzubewegen.

Bisher haben sich die statistischen Ansätze gegenüber den linguistischen durchgesetzt. Zwar können statistische Ansätze die syntaktischen Einschränkungen, die durch die Grammatik einer Sprache gegeben sind, nur sehr eingeschränkt modellieren. Ihr Vorteil ist jedoch, daß sich semantische Einschränkungen in der Statistik widerspiegeln, die linguistisch nur schwierig zu erfassen sind.

N-Gramm-Modelle

Die statistisch basierten N-Gramme sind die am weitesten verbreitete Technik in der Sprachmodellierung: Praktisch alle Spracherkenner arbeiten mit sogenannten Bi- oder Trigrammen, die Wahrscheinlichkeiten $P(w_i|w_{i-1})$ bzw. $P(w_i|w_{i-2}w_{i-1})$ bestimmen. Zur Sprachmodellierung von Buchstabensequenzen werden wir andere, wesentlich stärkere Einschränkungen nutzen. Für die allgemeine Spracherkennung sind N-Gramme jedoch von grundlegender Bedeutung.

Die zu bestimmende Wahrscheinlichkeit einer Wortsequenz $P(W) = P(w_1 w_2 w_3 \ldots)$ kann man umschreiben als

$$P(w_1 w_2 \ldots w_k) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1 w_2) \cdots P(w_k|w_1 w_2 \ldots w_{k-1}) \qquad (3.26)$$

Die einzelnen Faktoren enthalten zu viele Variablen, um auf verfügbaren Textkorpora geschätzt werden zu können. Nimmt man an, daß in $P(w_k|w_1 w_2 \ldots w_{k-1})$ der Einfluß eines Wortes mit seinem Abstand zu w_k sinkt, kann man Äquivalenzklassen bilden, indem Kontexte bereits als identisch gelten, wenn die letzten $N-1$ Wörter identisch sind:

$$P(w_i|w_1 w_2 \ldots w_{i-1}) \equiv P(w_i|w_{i-N+1} \ldots w_{i-1})$$

Man nennt $P(w_i|w_{i-N+1}\dots w_{i-1})$ ein N-Gramm, und es ergibt sich

$$P(w_1w_2\dots w_k) = P(w_1)\cdot P(w_2|w_1)\cdots P(w_{N-1}|w_1\dots w_{N-2})\prod_{i=N}^{k} P(w_i|w_{i-N+1}\dots w_{i-1}) \quad (3.27)$$

$P(w_i|w_{i-N+1}\dots w_{i-1})$ gibt also die Wahrscheinlichkeit für das Wort w_i an, gegeben daß direkt zuvor die $N-1$ Wörter $w_{i-N+1}\dots w_{i-1}$ beobachtet worden sind. In der Praxis werden meist nur 1-Gramme (*Monogramme*) $P(w_i)$ sowie 2-Gramme (*Bigramme*) $P(w_i|w_{i-1})$ direkt in der Suche eingesetzt. In größeren Vokabularen benötigt man bereits bei Bigrammen Heuristiken wie Strahlsuche, um nicht die volle Zahl der Übergänge aus jedem in jedes Wort berücksichtigen zu müssen. Entsprechend werden 3-Gramme (*Trigramme*) $P(w_i|w_{i-2}, w_{i-i})$ nur in eingeschränktem Umfang in der Suche direkt eingesetzt. Eine bewährte Technik ist jedoch, sie zur Neubewertung bereits gefundener Hypothesen (*Rescoring*) einzusetzen.

N-Gramme werden aus möglichst großen Textkorpora geschätzt, indem die Auftrittshäufigkeiten der entsprechenden Worttupel gezählt werden:

$$P(w_i|w_{i-N+1}\dots w_{i-1}) = \frac{\text{freq}(w_{i-N+1}\dots w_{i-1}w_i)}{\text{freq}(w_{i-N+1}\dots w_{i-1})}$$

Selbstverständlich können aus einem endlichen Trainingstext nur Näherungen der echten Wahrscheinlichkeiten bestimmt werden. Das Problem ist sogar noch gravierender: Man kann leicht ausrechnen, daß bei einem Vokabular von beispielsweise 5000 Wörtern von den $5000^3 = 125$ Milliarden potentiellen Trigrammen nur ein verschwindend kleiner Teil tatsächlich beobachtet wird. Das Dilemma kann gelöst werden, indem ein Teil der Wahrscheinlichkeitsmasse für die ungesehenen Trigramme reserviert wird. Zur Schätzung dieser Trigramme greift man dann auf unspezifischere, dafür aber robustere Wahrscheinlichkeiten, nämlich Bigramme und Monogramme, zurück.

Konnte also für ein Worttripel $w_1w_2w_3$ kein Trigramm $P(w_3|w_1w_2)$ berechnet werden, begnügt man sich mit dem Bigramm $P(w_3|w_2)$ und multipliziert dieses mit einem Korrekturfaktor, üblicherweise als *Backoff* bezeichnet. Entsprechendes gilt für unbeobachtete Bigramme, so daß sich folgendes Schema ergibt:

$$\tilde{P}(w_2|w_1) := \begin{cases} P(w_2|w_1) & \text{falls genügend } w_1w_2 \text{ beobachtet} \\ b(w_1)\cdot P(w_2) & \text{sonst} \end{cases}$$

$$\tilde{P}(w_3|w_1w_2) := \begin{cases} P(w_3|w_1w_2) & \text{falls genügend } w_1w_2w_3 \text{ beobachtet} \\ b(w_1,w_2)\cdot \tilde{P}(w_3|w_2) & \text{falls genügend } w_1w_2 \text{ beobachtet} \\ \tilde{P}(w_3|w_2) & \text{sonst} \end{cases}$$

Die Berechnung der Backoff-Faktoren $b(.)$ ist beispielsweise in [Jel90] beschrieben.

Sprachmodelle in der Suche

Das Ziel der Suche ist die Maximierung von $P(\mathbf{X}|W)P(W)$. Dazu wird bei der Berechnung der Viterbi-Wahrscheinlichkeit $P^*(\boldsymbol{x}_1 \ldots \boldsymbol{x}_T|w_1 w_2 \ldots w_M)$ in jedem Wortübergang von w_m nach w_{m+1} des Viterbi-Pfades die entsprechende Bigrammwahrscheinlichkeit $P(w_{m+1}|w_m)$ berücksichtigt. Die Viterbi-Wahrscheinlichkeiten werden im logarithmischen Bereich berechnet. Es sei $d_t = \log P^*(\boldsymbol{x}_1 \ldots \boldsymbol{x}_t|w_1 \ldots w_m)$ die Bewertung des Viterbi-Pfades im letzten Zustand s_i des Wortes w_m. Ein Übergang zum Zeitpunkt $t+1$ in ein neues Wort w_{m+1} ergibt einen partiellen Viterbi-Pfad

$$d_{t+1} = d_t + \log b_j(\boldsymbol{x}_{t+1}) + \log P(w_{m+1}|w_m) \tag{3.28}$$

$b_j(\boldsymbol{x}_{t+1})$ ist die Emissionswahrscheinlichkeit für Vektor $\boldsymbol{x}_{t+1}$ im ersten Zustand s_j des neuen Wortes w_{m+1}. An die Stelle einer wortinternen Transitionswahrscheinlichkeit $a_{ij} = P(q_{t+1} = s_j|q_t = s_i)$ tritt $P(w_{m+1}|w_m)$ für den Wechsel in das neue Wort w_{m+1}. In der Praxis kommt man nicht ohne zwei weitere Parameter α und β aus:

$$d_{t+1} = d_t + \log b_j(\boldsymbol{x}_{t+1}) + \alpha * \log P(w_{m+1}|w_m) + \beta \tag{3.29}$$

α gewichtet das Sprachmodell relativ zum akustischen Modell, und die „Worteingangsstrafe" β erschwert oder erleichtert den Übergang in ein neues Wort. Beide Parameter werden üblicherweise manuell auf einer Kreuzvalidierungsmenge eingestellt.

Perplexität

Eine Maßzahl für den Schwierigkeitsgrad eines Sprachmodells ist die *Perplexität*, die den Informationsgehalt eines Textes mißt und auf dem in Gl. (2.28) eingeführten Begriff der Entropie beruht. Gegeben sei ein Text mit einem Vokabular aus K Wörtern $\{v_1, v_2, \ldots, v_K\}$. Eine Quelle erzeuge nach einem Zufallsprozeß einen Text $W = w_1 w_2 \ldots w_n = v_{t_1} v_{t_2} \ldots v_{t_n}$ der Länge n. Ist die Quelle unabhängig und „gutartig" (ergotisch), so daß aus ausreichend langen Sequenzen ihre statistischen Eigenschaften bestimmt werden können, dann kann die Entropie H des Textes bestimmt werden als [Jel90]

$$\begin{aligned} H &= -\frac{1}{n}\log_2 P(W) = -\frac{1}{n}\log_2 P(v_{t_1} v_{t_2} \ldots v_{t_n}) \overset{(*)}{=} -\frac{1}{n}\log_2 \prod_{i=1}^{n} P(v_{t_i}) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log_2 P(v_{t_i}) \overset{(**)}{=} -\sum_{j=1}^{K}\frac{n_j}{n}\log_2 P(v_j) \simeq -\sum_{j=1}^{K} P(v_j)\log_2 P(v_j) \end{aligned} \tag{3.30}$$

In Schritt $(**)$ wurden jeweils alle n_j identischen Wörter v_j zusammengefaßt, es kann dann $P(v_j)$ als n_j/n approximiert werden.

Handelt es sich bei W um Sätze einer Sprache, ist die W erzeugende Quelle nicht unabhängig und damit Schritt $(*)$ in Gl. (3.30) nicht zulässig. Ebensowenig kann man

die Wahrscheinlichkeit $P(W)$ eines Textes W direkt bestimmen, deshalb wird H bzw. $\log_2 P(W)$ wie in (3.27) durch eine Faktorisierung mit N-Grammen (hier Trigramme) approximiert:

$$LP := -\frac{1}{n}\sum_{i=1}^{n}\log_2 P(w_i|w_{i-2}w_{i-1}) \approx -\frac{1}{n}\log_2 P(w_1 w_2 \ldots w_n) \qquad (3.31)$$

Die *Perplexität* PP ist definiert als

$$PP := 2^{LP},$$

und gibt anschaulich gesprochen den „mittleren Verzweigungsgrad" des Sprachmodells an. Sind alle Wörter eines Vokabulars V gleich wahrscheinlich und unabhängig voneinander produziert, entspricht die Perplexität gerade der Anzahl der Wörter in V, also $PP = |V|$, oder anders ausgedrückt: Hinsichtlich des Sprachmodells ist ein Text der Perplexität PP genauso schwierig zu erkennen wie ein Text aus PP gleichwahrscheinlichen und unabhängig voneinander produzierten Wörtern.

3.7 Zusammenfassung

Die Aufgabe der automatischen Spracherkennung besteht darin, ein akustisches Signal in diejenige Wortfolge zurückzutransformieren, die zuvor als Sprache artikuliert wurde. Das Problem zerfällt dabei in folgende Teilbereiche:

Mit der **Signalvorverarbeitung** werden aus dem Zeitsignal zur Erkennung geeignete Merkmale gewonnen. Nach einer Diskretisierung des Signals wird dieses innerhalb kurzer Zeitfenster (5 bis 30 Millisekunden) analysiert, wobei die als näherungsweise stationär angenommenen Zeitfenster gewöhnlich im Abstand von 10 Millisekunden fortgeschaltet werden. Die aus jedem Zeitfenster gewonnenen Merkmale können direkt aus dem Zeitsignal extrahiert werden, überwiegend werden jedoch spektrale Merkmale betrachtet. Eine typische Vorgehensweise besteht darin, mit Bandpaßfiltern die Energie innerhalb kritischer, gehörorientierter Frequenzbänder zusammenzufassen. In diesem Fall erhält man beispielsweise aus den 16 000 Abtastwerten einer Sekunde Sprache eine Folge von 100 Bandspektren mit je 16 Energiekoeffizienten. Zusätzliche Merkmalstransformationen dienen der Dimensionalitätsreduktion oder der Adaption an unterschiedliche Übertragungskanäle.

Die **akustische Modellierung** bewertet die gesprochene Äußerung mit Hilfe interner Wortmodelle. In früheren Ansätzen wurden dazu für die zu erkennenden Wörter akustische Prototypen gespeichert und mit den jeweiligen Eingabemustern verglichen. Um die unterschiedliche Länge und Sprechgeschwindigkeit von Prototyp und Eingabe zu berücksichtigen, werden die beiden Muster mit Hilfe der dynamischen Zeitverzerrung (*dynamic time warping, DTW*) zeitelastisch so aneinander angepaßt, daß die Summe der lokalen Distanzen ihrer Merkmalsvektoren minimiert wird.

Die akustischen Prototypen wurden seit Mitte der achtziger Jahre durch statistische Methoden abgelöst. Die akustische Modellierung mit neuronalen Netzen wird in den Kapiteln 4 und 6 vorgestellt; in diesem Kapitel wurde die gebräuchlichste, inzwischen schon klassische Technik der *Hidden Markov Models (HMMs)* beschrieben. Ein HMM $\boldsymbol{\lambda}_w$ ist ein statistisches Modell für ein Wort w, mit dem die Wahrscheinlichkeit dafür bestimmt wird, daß ein Eingabemuster $\mathbf{X}$ durch $\boldsymbol{\lambda}_w$ erzeugt wurde. Das HMM $\boldsymbol{\lambda}_w$ besteht aus Zuständen, die phonetische oder subphonetische Abschnitte des Wortes w repräsentieren und gemäß der Modelltopologie miteinander verbunden sind. Jeder Zustand besitzt eine eigene statistische Ausgabefunktion, von der diskrete Symbole oder kontinuierliche Merkmalsvektoren erzeugt werden. Eine Folge von Symbolen wird in einem zweistufigen Zufallsprozeß produziert: Zuerst wird ein neuer interner Zustand des HMM und in Abhängigkeit von diesem die Ausgabe eines Symbols gewählt. Dabei wird das Verhalten des HMM durch seine Parameter, die Transitions- und Emissionswahrscheinlichkeiten charakterisiert. Erstere bestimmen die Zustandsübergänge, letztere definieren die zustandsspezifischen, statistischen Ausgabefunktionen. Somit wird in einem HMM die zeitliche Variation der Sprache durch die Folge durchlaufener Zustände, und die spektrale Variation durch die jedem Zustand zugeordneten Ausgabewahrscheinlichkeiten modelliert.
Das HMM $\boldsymbol{\lambda}_w$ mit der höchsten Wahrscheinlichkeit $p(\mathbf{X}|\boldsymbol{\lambda}_w)$ bestimmt die Identität der zu klassifizierenden Eingabe $\mathbf{X}$. Umgekehrt werden zum Training der HMMs diejenigen Parameter $\boldsymbol{\lambda}_w$ mit Hilfe eines Maximum-Likelihood-Verfahrens gesucht, die die Wahrscheinlichkeit der gegebenen Trainingsdaten maximieren.

Neben der Akustik wird in der Spracherkennung mit **Sprachmodellen** die Plausibilität einer hypothetisierten Wortfolge $W = w_1 w_2 \ldots w_k$ bewertet. Praktisch alle Erkenner arbeiten mit N-Gramm-Modellen. Dabei wird die zu schätzende Wahrscheinlichkeit $P(w_1 \ldots w_k)$ in ein Produkt von *Bigrammen* $P(w_i|w_{i-1})$ oder *Trigrammen* $P(w_i|w_{i-2}w_{i-1})$ faktorisiert.

Kontinuierliche Sprache wird erkannt, indem die einzelnen Wortmodelle $\boldsymbol{\lambda}_w$ parallelgeschaltet und rückgekoppelt werden, so daß beliebige Wortfolgen durchlaufen werden können. Die Aufgabe der **Suche** ist es, zu einer Eingabe $\mathbf{X}$ diejenige Wortfolge $W = w_1 \ldots w_k$ zu finden, mit der die gemeinsame Wahrscheinlichkeit $P(\mathbf{X}|W) \cdot P(W)$ des akustischen und des Sprachmodells maximiert wird. Dazu wird innerhalb eines Wortes ein optimaler Pfad bezüglich der akustischen Modelle gesucht, und an den Wortübergängen werden die Tri- oder Bigrammwahrscheinlichkeiten berücksichtigt. Dieses kombinatorische Optimierungsproblem wird auch *Dekodierung* genannt und kann mit einer Variante des DTW-Algorithmus effizient gelöst werden.

Kapitel 4

Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über verwandte Arbeiten aus den Themenbereichen der konnektionistischen Spracherkennung sowie der Buchstabiererkennung. Weitere verwandte Arbeiten zur erst in Kapitel 7 eingeführten Problematik der Erkennung von Nachnamen auf der Grundlage großer Namenslisten sind dort aufgeführt.

4.1 Konnektionistische Spracherkennung

Zu der bereits in Abschnitt 3.1 skizzierten Entwicklung der Spracherkennung gesellen sich seit Ende der achtziger Jahre eine wachsende Zahl konnektionistischer Ansätze, aus denen auch die in Kapitel 6 vorgestellten eigenen Beiträge dieser Arbeit hervorgegangen sind.

Die im folgenden beschriebenen konnektionistischen Ansätze sind gegliedert nach statischen, dynamischen und hybriden Ansätzen, wobei diese Reihenfolge auch in etwa die historische Entwicklung widerspiegelt. **Statische Netze** haben keine Mechanismen, um die zeitvariablen Anteile des Sprachsignals zu modellieren, und erfordern daher eine längennormierte und nach Möglichkeit zentrierte Eingabe. **Dynamische Netze** können Eingaben variabler Länge positionsinvariant erkennen. **Hybride Ansätze** verbinden konnektionistische Ansätze mit Techniken zur nichtlinearen Zeitanpassung (DTW oder, spezieller, HMMs).
Seltenere, hier nicht weiter vertiefte Anwendungsfelder von neuronalen Netzen in der Spracherkennung liegen in der Signalvorverarbeitung (etwa zur Rauschunterdrückung [TW88]) oder in der Nachverarbeitung bereits erkannter Hypothesen [BH93].

4.1.1 Statische Netze

In den ersten Ansätzen Ende der achtziger Jahre werden statische Netzwerke zur Klassifikation von Phonemen oder kleinen Wortschätzen eingesetzt. Die fest vorgegebene

Dimension der Eingabeschicht erfordert eine Eingaberepräsentation, die in der Länge normiert und auf eine bestimmte Position zentriert ist.

Ein sehr anschauliches Beispiel sind die 1988 von Huang und Lippmann [HL88, Lip89] klassifizierten Vokaldaten von Peterson und Barney. Jedes der 338 Trainings- und 333 Testmuster entspricht einem im Kontext der Buchstaben „/hVd/" gesprochenen Vokal V und ist durch die Frequenzen der ersten beiden Formanten (F1 und F2) repräsentiert. Die von einem 3-schichtigen MLP mit 50 verborgenen Knoten gelernten Trennhyperebenen sind in Abbildung 4.1 dargestellt. Die Vokalerkennungsrate lag mit 80.2% etwas unter der mit einem k-Nächste-Nachbarn-Ansatz erzielten Rate von 82.0%. Die Frequenzen der beiden Formanten sind zwar wichtige Merkmale, stellen aber eine zu starke Informationsreduzierung der akustischen Eingabe dar.

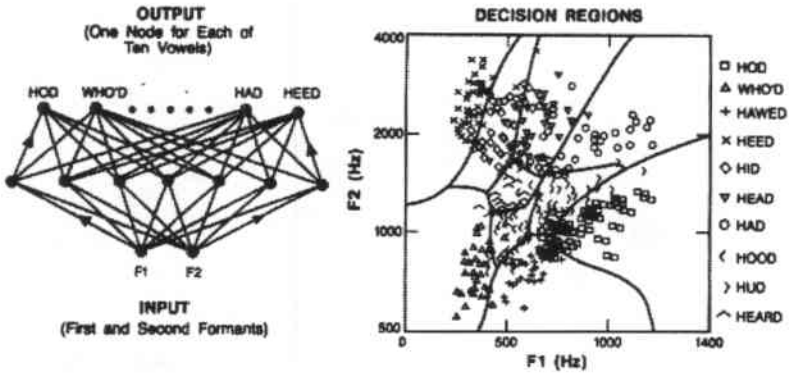


Abbildung 4.1: Ein MLP lernt die Peterson-Barney Vokal-Daten, aus [Lip89]

Eine ähnliche Netzarchitektur, allerdings mit einer geeigneteren, höherdimensionaleren Eingabe von 20 Vektoren aus je 16 spektralen Filterbankkoeffizienten wurde von Elman und Zipser [EZ87] untersucht. Das Sprachmaterial bestand aus 505 Konsonant-Vokal-Silben, kombiniert aus /b,d,g/ und /i,a,u/ von einem Sprecher, und wurde von Hand für die Netzeingabe zentriert. Mit einer verborgenen Schicht von 6 Knoten konnten etwa 95% der Konsonanten, und 99.5% der Vokale korrekt klassifiziert werden.

Eine Reihe weiterer, ähnlicher Netztypen zur Erkennung von Einzelziffern oder anderen kleinen Wortschätzen sind im Übersichtsartikel von Lippmann [Lip89] aufgeführt. Auch zur Buchstabiererkennung gibt es einige Ansätze mit statischen Netzen. Drei Beispiele, die Arbeiten von Burr, Reynolds und Tarassenko, sowie Cole et al. sind in Abschnitt 4.2 beschrieben.

4.1.2 Dynamische Netze

Im Gegensatz zu den statischen Netzen erlauben dynamische Netze Eingaben $\mathbf{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_T$ von variabler Länge T, indem sie diese Schritt für Schritt verarbeiten und zeitlich integrieren, entweder durch rekurrente Netze oder durch Netze mit verschiebungsinvarianten Gewichten.

Rekurrente Netze

In *rekurrenten Netzen* werden aus der neuen Eingabe x_{t+1} und dem bisherigen Zustand s_t eine neue Bewertung y_{t+1} und ein neuer Zustand errechnet:

$$(s_{t+1}, y_{t+1}) = f(x_{t+1}, s_t)$$

Der Zustand $s(t)$ ist in Neuronen repräsentiert, die ihre Ausgaben über rekurrente Verbindungen im nächsten Zeitschritt zur Verfügung stellen. Typische Vertreter rekurrenter Netze sind *Jordan-* und *Elman*-Netze (siehe Abbildung 4.2), in welchen die Aktivierungen der Ausgabeschicht bzw. der verborgenen Schicht in eine Kontextschicht kopiert werden, die parallel zur Eingabeschicht vollständig mit der verborgenen Schicht verbunden ist und gewissermaßen als Zustandsgedächtnis fungiert.

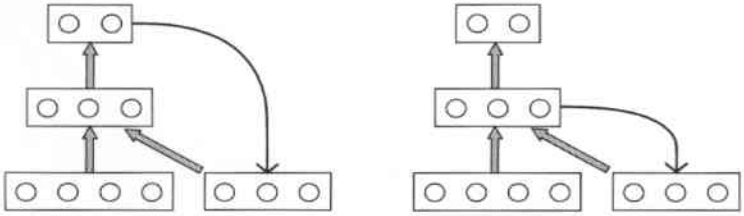


Abbildung 4.2: Rekurrente Netze: Schema eines Jordan- (links) bzw. Elman-Netzes (rechts)

Im Vergleich zu MLPs sind rekurrente Netze aufwendiger in Entwicklung und Training und haben sich daher nur in wenigen Fällen (z.B. im ABBOT-System, s.u.) längerfristig behauptet. Einige frühe Beispiele für rekurrente Netze in der Spracherkennung sind in [Lip89] beschrieben, etwa ein von Robinson und Fallside (1988) zur Erkennung von 27 Phonemen eingesetztes Jordan-Netz oder die von Prager, Harrison und Fallside trainierten Boltzmann-Maschinen zur Klassifikation von 11 Vokalen.

Time-Delay-Netze

Während die oben vorgestellten rekurrenten Netze den zeitlichen Verlauf des Eingabesignals als Folge sich verändernder interner Zustände kodieren, berechnet die Familie der

Time-Delay-Netze unabhängig zu jedem Zeitpunkt Bewertungen für die lokale Eingabe, die dann zu einer Gesamtbewertung integriert werden. Dynamische Merkmale werden durch ein Analysefenster berücksichtigt, das einen gewissen zeitlichen Kontext umfaßt.

In Abbildung 4.3 ist das erstmals von Waibel et al. [WHH+87, WHH+89] vorgestellte *Time-Delay Neural Net (TDNN)* abgebildet. Die zeitverzögerten Verbindungen (*Time-Delays*) bilden in der Eingabeschicht drei, in der verborgenen Schicht fünf Vektoren (die Aktivitäten von drei bzw. fünf benachbarten Zeitschritten) auf die jeweils nächste Schicht ab. Diese Fenster werden über die Eingabe hinweggeschoben. Die Gewichte sind also zu jedem Zeitpunkt gleich, wodurch ein verschiebungs- bzw. zeitinvarianter Merkmalsdetektor realisiert wird. Für jedes der drei zu erkennenden Phoneme b,d,g gibt es ein Neuron in der Ausgabeschicht, das mit allen Neuronen des entsprechenden Phonems aus der zweiten verborgenen Schicht verbunden ist.

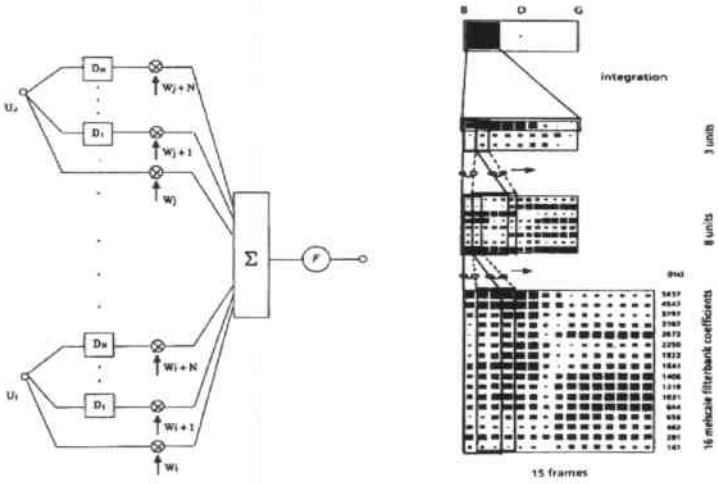


Abbildung 4.3: Die TDNN-Architektur: Links ein einzelnes Neuron mit zeitverzögerten Verbindungen ($D_1, \ldots, D_N$), rechts das Gesamtsystem (aus [WHH+89])

Für jeden von drei Sprechern wurde das TDNN sprecherabhängig mit jeweils etwa 600 Phonemen[1] trainiert und weiteren 600 getestet. Mit einer durchschnittlichen Phonemerkennungsrate von 98.5% konnten mit dem TDNN exzellente Ergebnisse erzielt werden, wohingegen ein auf der gleichen Aufgabenstellung trainiertes HMM 93.7% erreichte.

[1] Die Phoneme wurden manuell aus isoliert gesprochenen Wörtern ausgeschnitten

In [Wai89, WSS89a, WSS89b] erweitert Waibel das „bdg"-TDNN auf die Erkennung aller Konsonanten, indem er mehrere, jeweils einzeln auf verwechselbaren Konsonantengruppen trainierte TDNNs in einem modularen System zusammenfaßt. Die 18 Phoneme in der japanischen Datenbank wurden mit 95.9% korrekt erkannt.

Basierend auf obigen Experimenten haben McDermott et al. [MIKT90] dieselbe Aufgabenstellung mit dem Ansatz der *Learned Vector Quantization (LVQ)* bearbeitet. LVQ erlernt in einem iterativen Prozeß eine optimale Position von Referenzvektoren, die dann für eine Nächster-Nachbar-Klassifikation eingesetzt werden können. Wie im TDNN wird ein insgesamt 7 Sprachvektoren umfassendes Eingabefenster über die Eingabe geschoben und pro Zeitschritt eine Bewertung berechnet. An die Stelle der vom TDNN berechneten Phonembewertungen tritt jedoch ein Distanzmaß, das den Abstand vom jeweils nächsten Referenzvektor (in der Trainingsphase mit LVQ bestimmt) der entsprechenden Klasse angibt. Wie im TDNN werden die pro Zeitschritt anfallenden Distanzen (nach einer Normierung) aufsummiert, und die geringste Gesamtdistanz entscheidet die erkannte Klasse. Dabei werden mit dem LVQ-Ansatz 97.1%, mit einem TDNN 96.7% korrekte Phoneme erzielt.

Während in obigen Experimenten alle Eingaben eine konstante Länge von 15 Sprachvektoren hatten, nutzen Lang, Waibel und Hinton in [LWH90] das TDNN zur Erkennung variabel lang gesprochener Buchstaben B,D,E und V, wobei eine sprecherunabhängige Erkennungsrate von 90.9% erzielt wird, die Lang mit einer Erkennungsrate von 89% für das von Brown in [Bro87] entwickelte HMM vergleicht.

Ein interessanter, in paralleler Hardware ausgeführter Ansatz ist das *Time Concentration Network* von Tank und Hopfield [TH87]. Während das Sprachsignal durch den Erkenner fließt, werden Stimuli für die verschiedenen Phoneme ausgelöst. Diese werden für jedes Wortmodell unterschiedlich um diejenige Zeitspanne verzögert, die dem Abstand dieses Phonems zum Wortende entspricht. Gleichzeitig wird der Stimulus um so mehr verschliffen, je weiter das entsprechende Phonem vom Wortende entfernt ist, um die Unsicherheit seiner genauen Position zu modellieren. Im Idealfall werden also im korrekten Wortmodell die Stimuli der einzelnen Phoneme gerade so verzögert, daß sie alle auf das Wortende fallen und sich dort zu einer hohen Aktivierung aufsummieren. Das Time Concentration Network wurde zur sprecherabhängigen Ziffernerkennung eingesetzt mit einer Erkennungsrate von 99.3%.

4.1.3 Hybride Systeme

Unter hybriden Systemen verstehen wir Erkenner, die neben rein konnektionistischen auf weitere Techniken zurückgreifen, die meist einer zeitlichen Integration der variabel langen Spracheingaben dienen. Dazu berechnet die neuronale Komponente pro Eingabevektor (typischerweise alle 5 - 20 msec) phonetische Bewertungen, die mit Hilfe einer nichtlinearen Zeitanpassung (DTW) oder direkt mit HMMs zu einer Gesamtbewertung für jedes der zugrundeliegenden konkurrierenden Wortmodelle zusammengefaßt werden.

Zuerst werden mit dem Viterbi- und dem Alpha-Netz zwei Ansätze vorgestellt, die direkt ein HMM als konnektionistisches System modellieren. Die danach beschriebenen Ansätze des DNN, LPNN und MS-TDNN sind konnektionistische Klassifikatoren, in die eine DTW-Komponente integriert ist. Die hybriden NN-HMM Systeme schließlich berechnen die Emissionswahrscheinlichkeiten im HMM nicht mit Gaußschen Mischverteilungen, sondern mit neuronalen Netzen.

Viterbi- und Alpha-Netz

Die erste Schicht des von Lippmann vorgestellten Viterbi-Netzes [LG87] bildet einen Gauß-Klassifikator nach und berechnet zur Eingabe $\boldsymbol{x}_t$ für jeden Zustand S_i eines Wortmodells eine approximierte (logarithmierte) Emissionswahrscheinlichkeit. Mit Hilfe von als Netzwerken kodierten Maximum-Operatoren und Verzögerungsgliedern wird in einer zweiten Schicht die Rekursion des Viterbi-Algorithmus von Gl. (3.19) simuliert. Da keine Rückwärtsverzeigerung gespeichert wird, eignet sich das Viterbi-Netz nicht zur Erkennung kontinuierlicher Sprache. Nicht unerwartet erzielen das Viterbi-Netz und ein vergleichbares HMM praktisch identische Ergebnisse von 99.5% korrekt auf einer sprecherabhängigen Erkennungsaufgabe mit einem Vokabular von 35 Wörtern. Auf ähnliche Weise berechnet das 1990 von Bridle vorgeschlagene *Alpha-Netz* [Bri90] nicht den besten Pfad, sondern die beste Gesamtwahrscheinlichkeit für ein Wortmodell, indem die Rekursion des Vorwärtsalgorithmus aus Gl. (3.15) implementiert wird.

DNN

Das *Dynamic Programming Network (DNN)*, einer der ersten Ansätze, der ein MLP mit dynamischer Programmierung verbindet, ist in [SI87, SIY+89] von Sakoe et al. beschrieben. Die Eingabeschicht des DNN besteht aus J Blöcken $\boldsymbol{x}_1 \ldots \boldsymbol{x}_J$. Der Block $\boldsymbol{x}_t$ nimmt einen Kontext von τ Sprachvektoren $\boldsymbol{a}_{t-\tau-1} \ldots \boldsymbol{a}_t$ auf.

Jeder der Blöcke $\boldsymbol{x}_t$ ist vollständig mit einer Anzahl Neuronen $\boldsymbol{y}_t$ in der verborgenen Schicht verbunden, der (zusammen mit $\boldsymbol{x}_t$) als Phonemdetektor oder Zustand in einem Wortmodell interpretiert wird. Über weitere Gewichte werden alle $\boldsymbol{y}_t$ zu einer „Wortausgabe" zusammengefaßt, die im beschriebenen Zweiklassenfall [SIY+89] nur aus einem Knoten besteht und die Klassenzugehörigkeit der Eingabe anzeigt. Die dynamische Zeitanpassung findet zwischen der Spracheingabe $\mathbf{A} = \boldsymbol{a}_1 \ldots \boldsymbol{a}_I$ und den Blöcken $\boldsymbol{x}_1 \ldots \boldsymbol{x}_J$ statt. Ein zu erkennendes Muster wird immer so angepaßt, daß es die größtmögliche Ausgabe erzeugt. Für das Training wurde entweder eine aus einem Referenzmuster gewonnene konstante Zeitzuordnung vorgeschrieben oder diese nach jeder Iteration mit dem DNN neu adaptiert, aus Stabilitätsgründen allerdings nur für die Muster einer der beiden Klassen. Auf japanischen, isolierten Ziffern konnten mit 99.3% sehr gute, sprecherunabhängige Erkennungsraten erzielt werden, womit ein ausschließlich auf DTW basierter Erkenner (98.9%) übertroffen wurde.

MS-TDNN

Das *Multi-State Time-Delay Neural Network (MS-TDNN)* ist eine zum ersten Mal von Haffner in [HFW91] beschriebene direkte Erweiterung des TDNN. Statt einzelner Phoneme werden im MS-TDNN Phonemsequenzen zeitlich integriert. Jedes zu erkennende Wort wird durch solch eine Phonemsequenz modelliert, für die mittels dynamischer Programmierung (DTW) ein optimaler Pfad, also die zeitliche Segmentierung mit der höchstmöglichen Gesamtbewertung gefunden wird. Die Gesamtbewertungen für alle Wörter, also die Summen aller Phonembewertungen entlang der entsprechenden Pfade, stellen die Ausgabeschicht des MS-TDNN dar, wie in Abbildung 4.4 ersichtlich. Das MS-TDNN kann auf Wortebene diskriminativ trainiert werden: Das im Vergleich mit einer Sollausgabe errechnete Fehlersignal wird mittels Backpropagation entlang der gefundenen Pfade bis zur Eingabe zurückpropagiert.

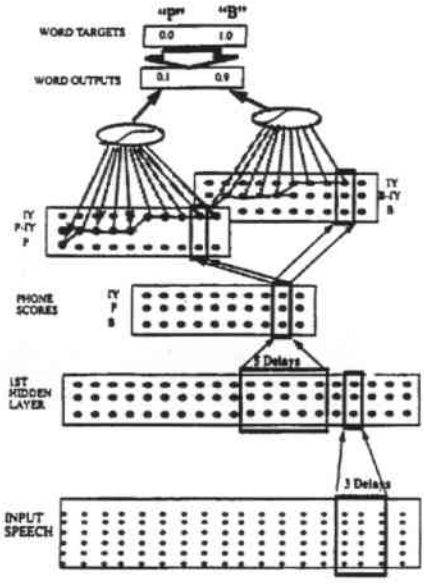


Abbildung 4.4: Die MS-TDNN-Architektur (aus [HFW91])

Das MS-TDNN wurde von Haffner, Waibel et. al. in [HFW91, HW91, HW92] für die Buchstabiererkennung eingesetzt (Ergebnisse siehe Abschnitt 6.9). In späteren Arbeiten wechselt Haffner auf Ziffern-Erkennung und beschäftigt sich mit näher an HMMs angelehnten Trainingsalgorithmen, beispielsweise dem diskriminativen Maximum Mutual

Information (MMI) Training. Sein „α-β-TDNN" berücksichtigt statt eines einzelnen Viterbi-Pfads die Wahrscheinlichkeiten aller Pfade [Haf92, Haf93a, Haf93b, Haf94].

In der vorliegenden Arbeit wird (neben anderen, in Kapitel 7 und 8 beschriebenen Aspekten der Buchstabiererkennung) die Entwicklung der ursprünglichen MS-TDNN-Architektur weitergeführt und durch neue Trainingstechniken verbessert, die in Kapitel 6 im Detail beschrieben sind. Auch andere Forscher haben den MS-TDNN-Ansatz weiterverfolgt:

Bregler, [BMHW93b, BMHW93a] Duchnowski, [DMW94] und Meier [MHD96] haben den in dieser Arbeit entwickelten MS-TDNN-Erkenner um eine optische Komponente erweitert, so daß die Spracherkennung durch „Lippenlesen" unterstützt werden kann.

Tebelskis beschäftigt sich in [TW93, Teb93, Teb95] mit konnektionistischer Spracherkennung für große Vokabulare und setzt dazu LPNNs (siehe nächster Abschnitt) und MS-TDNNs ein. Auf der Resource Management Task mit einem 1000-Wort-Vokabular bei einer Perplexität von 60 (die offizielle Feb. und Okt. '89 Testmenge der DARPA-Evaluationen) erzielt er mit einem hybriden NN-HMM-Ansatz (ähnlich dem des ICSI-Systems, s.u.) eine Wortakkuratheit von 89.2%, die durch Training auf Wortebene mit dem MS-TDNN auf 90.5% verbessert werden konnte. Dieser steht eine Wortakkuratheit von 87.2% des ICSI-Systems gegenüber.
Die Arbeiten von Tebelskis sind in seiner Dissertation [Teb95] zusammengefaßt. Dabei liegt der Schwerpunkt auf der Optimierung des „Trainings auf Phonemebene", also dem Lernen der richtigen a posteriori Wahrscheinlichkeiten der Phoneme mit der TDNN-Komponente des MS-TDNNs.

Zeppenfeld betreibt mit dem MS-TDNN Schlüsselworterkennung (*Keyword Spotting*) [ZHW93]. Dabei muß in einem mehr oder weniger beliebigen Text eine kleine Gruppe bestimmter Schlüsselwörter erkannt werden, aus denen beispielsweise erschlossen werden soll, über welches Thema sich die Gesprächspartner unterhalten. Natürlich will man möglichst viele der Schlüsselwörter erkennen, umgekehrt sollen jedoch die restlichen Wörter nicht fälschlicherweise als Schlüsselwörter identifiziert werden.

Außerhalb der Spracherkennung findet das MS-TDNN in den Arbeiten von Manke [MFW95] in der kursiven *On-Line*-Handschrifterkennung Anwendung. On-Line bedeutet hier, daß die Daten nicht als fertiges Schriftbild, sondern bereits während des Schreibens aufgenommen werden. Damit besteht ein geschriebenes Wort aus einer zeitlichen, zweidimensionalen Koordinatensequenz (sowie aus daraus abgeleiteten Merkmalen wie Schreibgeschwindigkeit, Winkel etc.), ganz ähnlich wie ein gesprochenes Wort als eine zeitliche Sequenz von Frequenzkoeffizienten repräsentiert wird.

Bodenhausen [BW91, Bod94, BH95b] entwickelt am Beispiel des MS-TDNN Algorithmen, die eine automatische Strukturierung der Netzwerkarchitektur erlauben. Dabei werden die Anzahl der Neuronen in der verborgenen Schicht, die Breite der Zeitverzögerungen und die Anzahl der Zustände pro Wortmodell automatisch bestimmt.

LPNN

Eine weitere Klasse hybrider Netze sind die *Linked Predictive Neural Networks (LPNN)*. Wie zuvor werden für jeden Zeitpunkt t ein oder mehrere Sprachvektoren als Eingabe betrachtet. Statt nun eine skalare Bewertung zu berechnen, versucht jedes Phonemmodell, den nächsten Sprachvektor möglichst genau vorherzusagen, und die Voraussage mit der geringsten Abweichung vom tatsächlichen Sprachvektor wird als das plausibelste Modell gewertet.

LPNNs wurden erfolgreich als Einzelziffernerkenner [Lev90, IW90] oder als sprecherabhängige Einzelworterkenner [IW91] eingesetzt, aber auch als kontinuierliche Erkenner mit großen Wortschätzen [PF93, MGR93, IW90, TW90, TWPS91a, TWPS91b, Teb95]. In [Teb95] vergleicht Tebelskis die prädiktiven LPNNs mit klassifikationsbasierten Ansätzen (MS-TDNN, HMM). Die LPNNs schneiden dabei konsistent schlechter ab, was auf das nicht-diskriminative Training sowie die Inkonsistenz zwischen Optimierungskriterium (möglichst gute Vorhersage von Sprachvektoren) und Testbedingungen (Erkennung von Wörtern) zurückgeführt wird.

Hybride NN-HMM-Systeme

MS-TDNN und LPNN betten Wortmodelle und die Algorithmen zur dynamischen Zeitverzerrung in konnektionistische Strukturen und die entsprechenden Trainingsmethoden ein. Stärker aus der Perspektive der Hidden Markov Models entwickelt sind die *hybriden NN-HMM*-Ansätze. Hier werden in einem HMM die Observationswahrscheinlichkeiten nicht konventionell mit Baum-Welch-Training nach dem Maximum-Likelihood-Prinzip, sondern mit Hilfe neuronaler Netze erlernt. In diskreten HMMs können neuronale Netze als Vektorquantisierer die Sprachvektoren auf diskrete Codebuch-Indizes abbilden [Rig93, RNR95, LCVC93]. Verbreiteter ist jedoch, für kontinuierliche HMMs statt mit Gaußschen Mischverteilungen die Emissionswahrscheinlichkeiten $p(\boldsymbol{x}|q_k)$ mit neuronalen Netzen indirekt über die a posteriori Wahrscheinlichkeiten $P(q_k|\boldsymbol{x})$ zu approximieren. Die beiden bekanntesten Vertreter dieser Richtung sind im folgenden beschrieben.

ICSI

Eine lange Tradition hat die Modellierung hybrider NN-HMM-Systeme am International Computer Science Institute (ICSI), vertreten durch die Arbeiten von N.Morgan, H.Bourlard und Mitarbeiter [BW90, RMCF92, BM94, BBDS93]. Eine gute Zusammenfassung dieser Arbeiten findet sich in [BM94] und [MB95]. Unter anderem dort wird auch ein Ansatz beschrieben, bei dem ein sogenanntes *Diskriminatives HMM* direkt mit von einem MLP ausgegebener a posteriori Wahrscheinlichkeit $p(q_k^t|\boldsymbol{x}_t, q_l^{t-1})$ trainiert werden kann. Allerdings erwies sich dieser Ansatz nach eigenen Aussagen der Autoren als nicht erfolgreich, so daß sie in ihrem Erkennungssystem den „konventionellen", hybriden NN-HMM-Ansatz verfolgen. Die aus [MB95] übernommene Abbildung 4.5 zeigt ein

entsprechendes Netz. Eingabe sind der aktuelle Sprachvektor $\boldsymbol{x}_t$ sowie ein beidseitiger Kontext von typischerweise $c = 2 \ldots 4$ Vektoren. Aus dem entsprechenden Sprachsegment $\boldsymbol{x}_{t-c} \ldots \boldsymbol{x}_{t+c} := \mathbf{X}_{t-c}^{t+c}$ werden für alle Zustände q_k des HMMs die a posteriori Wahrscheinlichkeiten $P(q_k|X_{t-c}^{t+c})$ berechnet. Indem man diese durch die a priori Wahrscheinlichkeiten der Zustände $P(q_k)$ teilt[2], erhält man skalierte Emissionswahrscheinlichkeiten

$$p(X_{t-c}^{t+c}|q_k) = z \cdot \frac{P(q_k|X_{t-c}^{t+c})}{P(q_k)}$$

Der Faktor $z = 1/p(X_{t-c}^{t+c})$ ist für eine gegebene Eingabe $\mathbf{X}$ konstant und spielt daher in der Erkennung bei der Bestimmung des Viterbi-Pfades keine Rolle.

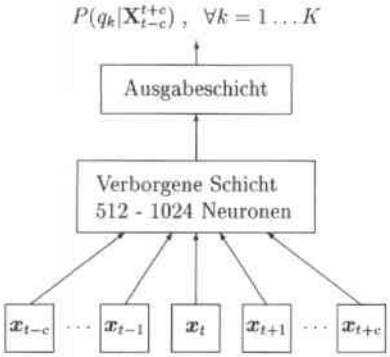


Abbildung 4.5: Architektur des im ICSI-System eingesetzten MLPs, nach [MB95]

Der Netzwerktyp aus Abbildung 4.5 besitzt eine sehr einfache Struktur, die mit einer bemerkenswert großen Anzahl von bis zu 4000 Neuronen in der verborgenen Schicht und über einer Million Gewichten trainiert wurde. Mit einem Schuß Selbstironie bezeichnen die Autoren ihr Netz daher auch als *BDNN* – „Big Dumb Neural Network".

In einem in [MB95] beschriebenen Experiment übernimmt der *DECIPHER*-Spracherkenner des Stanford Research Institute (SRI) die Rolle der HMM-Komponente. Werden im kontextunabhängig trainierten DECIPHER-System die HMM-Emissionswahrscheinlichkeiten durch die mit dem MLP berechneten ersetzt, sinkt die Fehlerrate von 11% auf 6% auf der Resource Management Task (gelesene Sprache bei einer Vokabulargröße von 1000 Wörtern, Perplexität 60). Ein kontextabhängiges DECIPHER-System kann eine Fehlerrate von 4% erzielen, die durch eine Kombination aus den HMM-

[2] $P(q_k)$ wird bestimmt, indem einfach gezählt wird, wie häufig jede Klasse q_k in den etikettierten Trainingsdaten vorkommt.

und MLP-Emissionswahrscheinlichkeiten auf 3.3% verbessert werden konnte. In einer DARPA-Evaluation auf der RM-Testmenge vom Februar 1991 konnte mit einer solchen Kombination sogar das beste Gesamtergebnis erzielt werden [BM94](S. 199). Das MLP besaß etwa 300 000 Parameter, die mit 1.3 Millionen Mustern trainiert wurden.

Weiterhin nahm das ICSI-System an der offiziellen Wall Street Journal (WSJ0) Evaluation teil, in der gelesene Texte mit einem 5000-Wort-Vokabular erkannt werden. Das Trainingsmaterial bestand aus 7 Millionen Mustern (jeweils neun Sprachvektoren mit je 13 PLP-Koeffizienten sowie die ersten und zweiten zeitlichen Ableitungen), mit denen ein sehr großes Netz mit 1.6 Millionen Parametern auf paralleler Spezialhardware trainiert wurde. Dabei erzielte das ICSI-System eine Fehlerrate von 16%, während die wesentlich komplexeren, kontextabhängigen HMM-Erkenner etwa die Hälfte dieser Fehlerrate erreichen konnten [MB95].

Um das MLP zu trainieren, benötigt man eine Etikettierung jedes einzelnen Sprachvektors. Ist das System erst trainiert, kann bei bekannter Transkription für jeden Satz der Trainingsmenge ein Viterbi-Pfad und damit eine neue Etikettierung bestimmt werden, die dann als Grundlage für ein erneutes Training dient. Durch diesen iterativen Prozeß kann die Etikettierung schrittweise verbessert werden. Eine initiale Etikettierung der neuen Daten kann gewonnen werden, indem das System auf einer anderen, bereits etikettierten Datenbank trainiert wird.

In [BMWR92, CHM+93] wird eine **kontextabhängige Modellierung** des MLP beschrieben, in der die Emissionswahrscheinlichkeit $p(\boldsymbol{x}_t|q_k, c_j)$ eines Sprachvektors $\boldsymbol{x}_t$ nicht nur vom HMM-Zustand q_k, sondern nun auch von dessen Kontext c_j abhängt. Durch geschicktes Faktorisieren kann diese Größe so umgeformt werden, daß nur a priori und mit einem MLP modellierbare a posteriori Wahrscheinlichkeiten berechnet werden müssen. Zweimaliges Anwenden der Bayes-Regel ergibt:

$$p(\boldsymbol{x}_t|q_k, c_j) = \frac{P(q_k|\boldsymbol{x}_t, c_j)p(\boldsymbol{x}_t|c_j)}{P(q_k|c_j)} = \frac{P(q_k|\boldsymbol{x}_t, c_j)P(c_j|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{P(q_k|c_j)P(c_j)}$$

Wie zuvor bleibt der konstante Faktor $p(\boldsymbol{x}_t)$ unberücksichtigt, und die a priori Wahrscheinlichkeiten $P(q_k|c_j)$ und $P(c_j)$ können durch Abzählen auf der Trainingsmenge bestimmt werden. $P(q_k|\boldsymbol{x}_t, c_j)$ wird mit einem Netz ähnlich wie in Abbildung 4.5 bestimmt, das für jede Kontextklasse c_j eine eigene Ausgabeschicht besitzt. Zur Berechnung von $P(c_j|\boldsymbol{x}_t)$ benötigt man ein weiteres Netz. Gegenüber dem kontextunabhängigen Vergleichssystem konnte mit dem kontextabhängigen MLP eine Fehlerreduzierung um 15% bis 30% erreicht werden [CHM+93].

Ein im ICSI verfolgter, innovativer Ansatz aus neuerer Zeit modelliert Sprache nicht wie üblich als eine Sequenz von stationären Zuständen, sondern versucht, möglichst informationsreiche und für das Hörempfinden dominante Abschnitte zu modellieren. Diese sind meist Wechsel zwischen stationären Abschnitten und werden von den Autoren als *Avents* („Auditory Events") bezeichnet. Erste Experimente deuten darauf hin, daß sich eine solche Modellierung zumindest in der Gegenwart von Hintergrundgeräuschen als vorteilhaft erweisen kann [MBG+95].

CU-Conn

Das an der Universität Cambridge (CU) von Hochberg, Kershaw, Renals, Robinson und anderen entwickelte System ABBOT [HRRC95, NAH+95, KRR96, RM93, Rob94, RAB+93] nutzt ein einschichtiges rekurrentes neuronales Netz, um a posteriori Wahrscheinlichkeiten von Phonemen zu berechnen. Diesem vorgeschaltet ist ein lineares Netzwerk, das eine lineare Transformation des akustischen Merkmalsraums realisiert und eine überwachte oder unüberwachte Sprecheradaption ermöglicht.

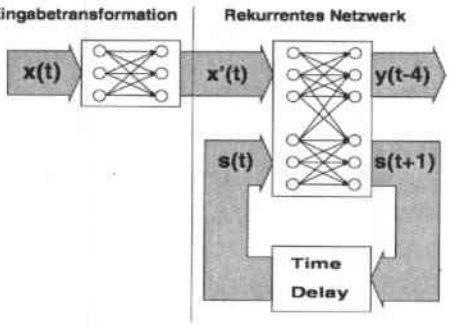


Abbildung 4.6: Das hybride NN-HMM-System der Universität Cambridge

In Abbildung 4.6 ist die Architektur illustriert. Das rekurrente neuronale Netz (RNN) in der rechten Bildhälfte erhält als Eingabe den Sprachvektor $\boldsymbol{x}'_t$ sowie den momentanen Zustand s_t und berechnet daraus die Ausgabe $\boldsymbol{y}_{t-4}$ sowie den nächsten Zustandsvektor s_{t+1}. Die Ausgabe $\boldsymbol{y}_t$ repräsentiert mit einem Knoten pro Phonem eine Schätzung der Phonemwahrscheinlichkeiten

$$y_i(t) = P(q_i(t)|\mathbf{X}_1^{t+4}) \tag{4.1}$$

$\mathbf{X}_1^{t+4}$ ist die Eingabe vom Zeitpunkt 1 bis $t+4$. Durch die um vier Zeitschritte verzögerte Ausgabe wird zur Berechnung von $\boldsymbol{y}_t$ auch zeitlich vorausliegender akustischer Kontext berücksichtigt. Wie zuvor werden die a posteriori Wahrscheinlichkeiten $P(q_i(t)|\mathbf{X}_1^{t+4})$ durch Division mit $P(q_i)$ in skalierte Emissionswahrscheinlichkeiten transformiert, die dann in einer HMM-Viterbi-Suche eingesetzt werden können.

Wenn das RNN trainiert ist, wird ihm ein einschichtiges lineares Netzwerk (LIN) vorgeschaltet. Mit diesem kann der akustische Eingaberaum transformiert und damit eine Sprecheradaption realisiert werden. Dazu werden die Gewichte des bereits trainierten RNN fixiert, und die Gewichte des LIN werden „neutral" als Einheitsmatrix initialisiert. Für eine bekannte oder unbekannte Eingabe wird mit dem momentanen Modell und mittels einer Viterbi-Suche jedem Sprachvektor eine Phonemklasse zugeordnet. Im

nachfolgenden Training werden die Gewichte des LIN, d.h. eine Transformation des Eingaberaumes so eingestellt, daß die Ausgaben möglichst hohe a posteriori Wahrscheinlichkeiten für die gegebenen Phonemklassen erzielen. Weitere Verbesserungen konnten durch kontextabhängige Modelle und durch Kombination mehrerer rekurrenter Netze erreicht werden.

Das ABBOT-System hat in den Jahren 1993-1996 an den offiziellen DARPA-Evaluationen teilgenommen und dabei mit den HMM-Systemen anderer Gruppen konkurriert. Die offizielle Testmenge bestand 1995 aus gelesenen Zeitungstexten aus der Wall Street Journal (WSJ) Datenbank bei unbeschränktem Vokabular, wobei die Erkenner mit Vokabulargrößen von 60 000 Wörtern arbeiten. Der sehr kompakte und schnelle CU-Conn-Erkenner benötigt auch bei kontextabhängiger Modellierung nur etwa ein Zehntel der Parameter eines konventionellen HMM-Systems. Mit Wortfehlerraten von 19.8% und 12.5% auf den offiziellen H3:P0 und H3:C0 Testmengen erzielte er einen guten Platz im Mittelfeld.

4.2 Buchstabiererkennung

Wie die Ziffernerkennung wurde die Buchstabiererkennung, bedingt durch ihr kleines Vokabular, schon frühzeitig zur Entwicklung und zum Test von Spracherkennern herangezogen. Im nachfolgenden Abschnitt sind alle dem Autor bekannten wesentlichen Systeme zur Buchstabiererkennung beschrieben. Konkrete Vergleiche der Ergebnisse sind, soweit möglich, in Abschnitt 6.9 aufgelistet.

Template Matching

Die ersten Ansätze der Buchstabiererkennung basierten auf den damals allgemein praktizierten Verfahren des dynamischen Mustervergleichs mit akustischen Prototypen („Template Matching", s.a. Abschnitt 3.3). R. Cole faßt in [CSM86] etwa 10 verschiedene, im Zeitraum zwischen 1975 und 1982 veröffentlichte Systeme zusammen (etwa [WY83, RLRW79]), die nach diesem Prinzip arbeiten. Auf einem alphanumerischen Erkennungsvokabular[3] wurden je nach Schwierigkeitsgrad (Frequenzbandbreite der Daten, sprecherab- oder sprecherunabhängig) Erkennungsraten zwischen 70 und 93% bei Einzelworterkennung erzielt. Der akustische Mustervergleich gewichtet alle Segmente der Muster gleich, was insbesondere für die leicht verwechselbaren Buchstaben problematisch sein kann. Beispielsweise sind zur Unterscheidung der Buchstaben „B,D,P,..." nur deren kurze Anfänge wichtig, jedoch können irrelevante Variationen auf dem gemeinsamen und zeitlich längeren Vokal übergewichtig sein und dadurch eine korrekte Unterscheidung im ausschlaggebenden Anfangsteil überschreiben.

[3] Die 26 Buchstaben des Alphabets sowie die Ziffern 0 bis 9

Wissensbasierte Ansätze (FEATURE)

Motiviert durch diese Schwierigkeiten und durch die Tatsache, daß es für einen menschlichen Experten möglich ist, anhand eines Spektrogramms sprecherunabhängig Merkmalsmuster für phonetische Segmente visuell zu identifizieren, haben Cole et. al. [CSM86] einen wissensbasierten Ansatz namens FEATURE entwickelt. Dabei sollen automatisch diejenigen Merkmale extrahiert werden, die für eine Klassifikation der Phoneme entscheidend sind.
Dazu werden zuerst aus dem Spektrogramm und anderen zeitlichen Merkmalen (Tonhöhe (*pitch*), Nulldurchgänge, Energie in tiefen, mittleren und hohen Frequenzbändern) für jeden Buchstaben vier „Ankerpunkte" bestimmt, und zwar Beginn und Ende der Aufnahme sowie des Vokalsegmentes des Buchstaben. Mit deren Hilfe werden dann ungefähr 50 verschiedene Merkmale berechnet, wie etwa die Frequenzen und Trajektoren der ersten drei Formanten im Vokalteil, die Dauer aperiodischer Energie und die Frequenzverhältnisse vor dem Anfang oder nach dem Ende des Vokals. Um nun eine Aufnahme zu klassifizieren, wird anhand eines Entscheidungsbaumes die Auswahl möglicher Buchstaben stufenweise eingeengt, bis man sich schließlich an einem Blatt des Baumes für einen einzelnen Buchstaben entschieden hat. Je nach Position im Baum, d.h. je nach den zu unterscheidenden Buchstabengruppen kommen unterschiedliche Merkmale (oder Kombinationen von Merkmalen) zum Tragen. Eine Entscheidung wird anhand eines Bayes-Klassifikators getroffen, die entsprechenden Wahrscheinlichkeiten werden anhand der relativen Häufigkeiten der Merkmale in den Trainingsdaten geschätzt. Mit dem FEATURE-System konnte eine sprecherunabhängige Erkennungsrate von etwa 89% und damit eine deutliche Verbesserung gegenüber den damals üblichen, auf Mustervergleich basierenden Verfahren erzielt werden.

Neuronale Netze

Burr [Bur88a] setzt ein **statisches MLP** mit 160 Eingabe-, 20 verborgenen und 26 Ausgabeknoten zur Erkennung des gesprochenen Alphabets ein. Trainiert und getestet wird auf 104 Buchstaben von einem Sprecher nach der *leaving-one-out*[4]-Technik. Ein Buchstabe wird durch 20 Vektoren mit je 8 Spektralkoeffizienten repräsentiert, Beginn und Ende werden automatisch detektiert. Das trainierte Netz erzielt eine Erkennungsrate von 85%. In einem weiteren Experiment [Bur88b] beschränkt sich Burr auf ein einschichtiges Perzeptron (keine verborgenen Knoten) und auf die 9 Buchstaben der „E-Menge" {B,C,D,E,G,P,T,V,Z}. Mit einer Eingabe von 20 Vektoren zu je 64 Spektralkoeffizienten wurden 91.4% erzielt. Durch eine Fokussierung auf die kritischen ersten 40% der Eingabe konnte die Erkennungsrate auf 98.2% erhöht werden.

Reynolds und Tarassenko [RT92] vergleichen **mehrere Netztypen** auf einer sprecherunabhängigen Buchstabierdatenbank, in der das Alphabet von 104 Sprechern

[4]In mehreren Durchläufen wird jeweils ein anderer, kleiner Teil der Daten als Testmenge genutzt.

je dreimal gesprochen wird. Je 52 Sprecher oder etwa 4000 Buchstaben wurden als Trainings- und Testmenge definiert. Die isoliert gesprochenen Buchstaben wurden auf eine statische Länge von 15 Vektoren mit je 8 Melscale Cepstral-Koeffizienten normiert. Mit einem Perzeptron wurde eine Buchstabenerkennungsrate von 64% erzielt, ein k-Nächste-Nachbarn-Klassifikator (k=10) erreichte 83.2% und ein dreischichtiges MLP mit 75 verborgenen Knoten 88.3%. Am ausführlichsten wurden *Radial Basis Funktionen (RBF)* untersucht. Nach zahlreichen Optimierungsschritten bezüglich Anzahl, Position und Breite der RBFs wurden 89.6% Erkennungsrate erreicht.

Der von der Gruppe um **Ron Cole** und **Mark Fanty** am **Oregon Graduate Institute (OGI)** entwickelte neuronale Buchstabiererkenner [FBC95, CFMG90, CFGJ91, CRF91, JFC91, FCR92, FC90, FC91] setzt ebenfalls ein statisches Netz ein, jedoch wird die Eingabe aus isoliert gesprochenen Buchstaben zuvor automatisch segmentiert. Die ursprünglich regelbasierte Segmentierung wurde durch eine stabilere, suchbasierte Segmentierung ersetzt. Dazu werden die in Abständen von 3 Millisekunden berechneten Sprachvektoren grob phonetisch klassifiziert, so daß Konsonanten, Vokale und Stille unterschieden werden können. Aufgrund der einfachen einsilbigen phonetischen Struktur der Buchstaben (Vokal-Konsonant oder Konsonant-Vokal) kann nun mit Hilfe des Viterbi-Algorithmus die wahrscheinlichste Segmentierung gefunden werden. Ältere Versionen arbeiten mit 5 phonetischen Klassen (SON, GLOT, CLOS, STOP, FRIC), inzwischen werden 22 Kategorien eingesetzt, bei denen die stark verwechselbaren Phoneme zusammengefaßt sind, beispielsweise /p/, /t/, /k/ in einer Klasse /ptk/.
Die eigentliche Klassifikation erfolgt in einem zweiten Schritt durch ein statisches MLP. Ausgehend von der zuvor berechneten Segmentierung wird für jeden Buchstaben eine große Anzahl von etwa 600 Merkmalen extrahiert, ganz ähnlich wie im früher entwickelten FEATURE-System. Gut die Hälfte davon sind spektrale Merkmale, die an bestimmten Positionen berechnet werden. Dazu kommen beinahe zwanzig weitere Merkmalsgruppen, unter anderem Nulldurchgänge (*Zero crossing*), Amplitudenwerte an verschiedenen Positionen sowie diverse Merkmale für die Tonhöhe (*pitch*). Viele dieser Merkmale werden an kritischen Stellen, beispielsweise zu Beginn des stimmhaften Segments, besonders fein aufgelöst. Das MLP besitzt eine verborgene Schicht mit 52 Knoten sowie 26 Ausgabeknoten und erzielt ein hervorragendes Ergebnis von 96% Erkennungsrate [FC91] auf sprecherunabhängigen, einzeln oder isoliert gesprochenen Buchstaben der ISOLET-Datenbank. Weitere Ergebnisse des OGI-Erkenners sind in den Abschnitten 6.9 und 7.5 aufgeführt.

Haffner [HFW91, HW91, HW92] setzt das **MS-TDNN** zur Buchstabiererkennung ein (Abbildung 4.4) und erzielt ausgezeichnete Ergebnisse auf sprecherabhängig, kontinuierlich buchstabierten Sequenzen (Abschnitt 6.9.1) sowie auf kontinuierlichen, aber vorsegmentierten sprecherunabhängigen Daten (Abschnitt 6.9.2).

Ebenfalls bereits in Abschnitt 4.1.3 eingeführt wurde das System von **Bodenhausen**, [BW91, Bod94, BH95b] der am Beispiel des MS-TDNN Buchstabiererkenners Algorithmen zur automatischen Netzwerkstrukturierung entwickelt. **Windheuser** [WB93, PBW95] testet verschiedene Ausgabedarstellungen an einem **hybriden TDNN-**

HMM-System zur Buchstabiererkennung. Nicht ganz so erfolgreich ist das von **Iso** [Iso92] zur Buchstabiererkennung eingesetzte **LPNN**.

HMMs

Das **HMM** von **Jouvet et al.** [JLMG93, JLM93] benutzt Ganzwortmodelle (d.h. ein Modell für jeden Buchstaben) mit diagonalen Kovarianzmatrizen, die je nach Buchstaben zwischen 17 und 41 Zustände besitzen. **Junqua** [JVFM95, Jun97] setzt eine modifizierte Version des HTK-HMM-Erkenners zur Buchstabiererkennung ein. Diese beiden Systeme wurden hauptsächlich zur Erkennung von buchstabierten Sequenzen aus fest vorgegebenen Namenslisten eingesetzt und sind deshalb in Abschnitt 7.5 beschrieben. Auch die HMM-Erkenner der CMU, **SPHINX** (siehe Abschnitt 6.9.2), und der Universität Karlsruhe, **JANUS** (siehe Abschnitt 6.9.4), wurden schon als Buchstabiererkenner eingesetzt.

Ein jüngerer, **spezialisierter HMM-Buchstabiererkenner** ist ausführlich von **Loizou** und **Spanias** [LS96] beschrieben, der in vielfältiger Weise modifiziert wurde, um den Besonderheiten der Buchstabiererkennung Rechnung zu tragen. Mit speziell trainierten HMM-Zuständen konnte die Unterscheidbarkeit innerhalb verwechselbarer Buchstabengruppen verbessert werden. Beispielsweise wurde der Beginn des Buchstabens E mit zusätzlichen HMM-Zuständen versehen, um den glottalen Stop bzw. das E-Anfangssegment genauer zu modellieren, wodurch sich die Unterscheidbarkeit zwischen E,B und D erhöhte. Analog wurde A um einen glottalen Stop erweitert, und zur besseren Unterscheidbarkeit von V und B wurde ein spezieller Übergangszustand im Buchstaben V trainiert, ebenso wurden weitere Zustände buchstabenspezifisch trainiert. Als beste Modelltopologie stellten sich nicht Ganzwortmodelle, sondern eine Kombination von kontextabhängigen und unabhängigen Zuständen heraus. Um die hohen Frequenzen der Stopkonsonanten in B,D,P,T,V besser aufzulösen, wurden in einem zweiten Erkennungslauf statt der konventionellen Melscale eine lineare Skalierung der Frequenzbänder eingesetzt. Mit einer linearen Merkmalstransformation konnte die Diskriminierung von N und M verbessert werden. Insgesamt konnte mit Hilfe all dieser Maßnahmen eine Erkennungsleistung von 97.3% BK auf der OGI-ISOLET-Datenbank sowie eine Buchstabenakkuratheit von 90.6% (91.7% korrekte Buchstaben + 1.1% Einfügungsfehler) auf den OGI Telefondaten erzielt werden.

4.3 Zusammenfassung

Seit der Renaissance der neuronalen Netze Ende der achtziger Jahre werden diese auch in der Spracherkennung eingesetzt. Dabei entstand eine Vielzahl unterschiedlicher Netzwerktypen, deren wichtigste Vertreter in den vorangehenden Abschnitten dokumentiert wurden.

Die frühen Ansätze nutzen **statische Netze**, die aber die zeitvariable Natur der Sprachsignale nur bedingt modellieren können. Sie wurden daher nur zur Erkennung von Phonemen oder zentrierten und längennormierten Einzelwörtern eingesetzt und spielen heute keine bedeutende Rolle mehr, wenn sie nicht in Verbindung mit anderen Techniken genutzt werden.

Dynamische Netze erlauben Eingaben variabler Länge, die im neuronalen Netz entweder intern (rekurrente Netze) oder explizit (Time-Delay Neural Network, TDNN) zeitlich integriert werden.
Rekurrente Netze sind aufwendig zu trainieren und werden für sich allein meist zur Phonem- oder Einzelworterkennung, in Verbindung mit hybriden HMM-Systemen aber auch sehr erfolgreich zur Erkennung großer Wortschätze eingesetzt, wie beispielsweise im ABBOT-System der Cambridge University.
Mit seinen gleitenden, zeitverzögernden Eingabefenstern berücksichtigt das TDNN zeitlichen Kontext sowohl in der Eingabe- als auch in der verborgenen Schicht. Die TDNN-Architektur ist verschiebungsinvariant: Die zur Klassifikation von Phonemen wichtigen Merkmale werden unabhängig von ihrer Position bewertet und anschließend in der Ausgabeschicht zusammengefaßt.

In **hybriden Netzwerktypen** werden explizite akustische Wortmodelle benutzt, die die phonetische Struktur der zu erkennenden Wörter definieren und eine zeitvariable Anpassung an die Eingabe erlauben. Die zwei prominentesten hybriden Netzwerktypen sind das Multi-State-TDNN (MS-TDNN), mit dem das im TDNN entwickelte diskriminative Prinzip der Phonemerkennung auf Wortebene fortgeführt wird, sowie die hybriden NN-HMM-Systeme. In letzteren wird der Mechanismus der Wortmodellierung und Suche direkt aus der HMM-Architektur übernommen, während die akustische Modellierung der Phoneme, also die Berechnung der Emissionswahrscheinlichkeiten, durch neuronale Netze ersetzt wird.

Die **Buchstabiererkennung** reicht zurück in die siebziger Jahre, als noch mit dem damals üblichen dynamischen Vergleich von akustischen Prototypen („Template Matching") gearbeitet wurde. Verbesserte Erkennungsraten konnten mit dem wissensbasierten FEATURE-System erzielt werden. Mit der Renaissance der neuronalen Netze Ende der achtziger Jahre wurden diese auch in der Buchstabiererkennung verstärkt eingesetzt. Aufgrund ihres kleinen Vokabulars eignet sich die Buchstabiererkennung in besonderem Maße für die bei neuronalen Netzen üblichen diskriminativen Trainingsverfahren.
Ein Beispiel für ein heute noch erfolgreich angewandtes statisches Netz ist der OGI-Buchstabiererkenner, der anhand einer großen Anzahl sorgfältig ausgewählter Merkmale Einzelbuchstaben klassifiziert, deren zeitliche Grenzen aber erst mit einer vorgeschalteten Segmentierung bestimmt werden müssen. Dagegen enthält der konnektionistische MS-TDNN-Erkenner eine dynamische Komponente, die eine automatische Segmentierung der Eingabe ermöglicht. Auch für den Einsatz von HMMs gibt es mehrere Beispiele, die zum Teil auf die Besonderheiten der Buchstabiererkennung spezialisiert wurden.

Kapitel 5

Sprachdatenbanken

„There is no data like more data"

Die Qualität eines Spracherkenners steht und fällt mit der Verfügbarkeit ausreichender Mengen an Sprachdaten. Ein Spracherkenner lernt, indem seine Modellparameter anhand gegebener Trainingsdaten geschätzt werden. Nur wenn dieses Sprachmaterial hinreichend umfangreich ist, kann gewährleistet werden, daß das breite Spektrum der Variabilitäten bezüglich Sprecher, Sprechstil und anderer Einflüsse ausreichend abgedeckt wird, um auch in neuen Situationen noch robust erkennen zu können.

Auch in dieser Arbeit wurden umfangreiche Sprachdatenbanken eingesetzt, die aus öffentlich verfügbaren Datenbanken, von Projektpartnern oder aus eigener Sammlung stammen. Insgesamt wurden Spracherkennungsexperimente auf 8 Sprachdatenbanken durchgeführt. Die Übersicht in Tabelle 5.1 schlüsselt ihre wichtigsten Merkmale unter der im Text verwendeten Kurzbezeichnung auf. Die Aufnahmen enthalten entweder ausschließlich Buchstabierungen (Typ „B"), fließend gesprochene und buchstabierte Namen (Typ „F+B") oder Buchstabierungen in spontaner Sprache (Typ „B+Spont"). Als

Name	Typ	Sprache	Umgebung, Abtastrate	Sprecher	Anz. Aufn.	Anz. Buchst.
CMU–Alph	B	E	Büro, 16 kHz	16	6535	33546
RM–Spell	B	E	Büro, 16 kHz	120	1786	10652
KA–Alph	B	D	Büro, 16 kHz	104	11665	79443
BUCNAM	B	D	Telefon, 8 kHz	591	591	3594
OGI–Spell	B	E	Telefon, 8 kHz	> 3000	7753	69973
VODIS-Alph	B	D	Auto, 11 kHz	100	3641	7061
KA–FliBu	F + B	D	Büro, 16 kHz	60	2780	18741
VM–Spell	B+Spont	D	Büro, 16 kHz			

Tabelle 5.1: Übersicht Buchstabier-Sprachdatenbanken

Sprachen liegen Deutsch (D) und Englisch (E) vor. Der Großteil der Daten wurde stationär mit Nahbesprechungsmikrofonen (Abtastrate 16 kHz) in Büroräumen aufgenommen, wobei auf eine ruhige Umgebung (keine Gespräche, keine Telefonanrufe) geachtet wurde. Die über Telefon aufgenommenen Daten haben eine eingeschränkte Bandbreite (Abtastrate 8 kHz) von etwa 4 kHz und sind mit den in Telefonkanälen auftretenden Störgeräuschen und Verzerrungen behaftet. Die stärksten Umgebungsgeräusche treten bei den im fahrenden Auto aufgenommenen VODIS-Daten auf. Weiterhin ist in der Tabelle die Gesamtzahl von Sprechern, Aufnahmen und Buchstaben aufgelistet.

5.1 Die CMU–Alph-Daten

Die CMU–Alph-Datenbank wurde 1990 an der Carnegie Mellon University gesammelt und enthält buchstabierte Wörter und Namen. Außerdem wurden zu etwa 30% zufällige Buchstabensequenzen („F X W P Q") beigemischt, um auch die weniger häufigen Buchstaben in ausreichender Anzahl zu repräsentieren. Alle Aufnahmen wurden handtranskribiert, also von Testhörern phonetisch etikettiert[1]. Tabelle 5.2 listet die Anzahl der buchstabierten Äußerungen von allen 16 Sprechern auf.

Sprecher	Aufnahmen	Buchst.	Sprecher	Aufnahmen	Buchst.
mbds	1110	5688	falz	50	251
mjmt	1000	5113	fsem	50	271
fcaw	1000	5113	fmld	50	252
fee	200	1042	mkjs	50	251
maem	1000	5113	fcon	50	268
flgt	1500	7721	mpwa	50	253
fsgt	50	258	fjmt	225	1170
mrgs	50	262	fsma	100	520
			Summe	6536	33546

Tabelle 5.2: Die CMU–Alph-Buchstabierdaten

5.2 Die Karlsruher Buchstabierdaten

1993–94 wurde an der Universität Karlsruhe unter meiner Anleitung eine umfangreiche deutschsprachige Buchstabierdatenbank gesammelt und transkribiert, die über 11 000 Aufnahmen und etwa 80 000 Buchstaben von insgesamt 104 Sprechern umfaßt. Die Sprecher waren aufgefordert, eine Liste vorgegebener Buchstabensequenzen zu sprechen, die aus 110 Nachnamen, 22 Städtenamen sowie 33 zufälligen Sequenzen bestanden[2].

[1] Ausnahme: Die Sprecherin fsma ist gar nicht, flgt und fjmt sind nur teilweise transkribiert.

[2] Wiederum sorgen die Zufallssequenzen für eine ausgewogenere Verteilung der einzelnen Buchstaben. Durch diese Maßnahme wurde beispielsweise das Verhältnis von Q:E von 3:1000 auf 75:1000 verbessert.

Neben den 26 Buchstaben waren die Umlaute ä, ö, ü, für ß die Aussprachen[3] „Eszett", „Scharf-S" und „Scharfes-S", für „-" die Aussprachen „Strich" und „Bindestrich" sowie „doppel" für sich wiederholende Buchstaben (z.B. „`S-C-H-M-I-doppel-T`") zugelassen. Insgesamt ergibt sich ein erweitertes Buchstabiervokabular von 35 Wörtern:

{A, B, ..., Z, Ä, Ö, Ü, SZ, Scharf-S, Scharfes-S, doppel, Strich, Bindestrich}

Vor einer Sprachspende wurden die Sprecher über diese Aussprachemöglichkeiten instruiert und darauf hingewiesen, daß keine Pausen zwischen einzelnen Buchstaben eingelegt werden müssen, sondern eine fließende Buchstabierung erwünscht ist. Die Daten wurden, wie in Tabelle 5.3 aufgelistet, im Verhältnis 70/10/20 in eine Trainings- (Train), Kreuzvalidierungs- (Cross) und Testmenge (Test) partitioniert. Die Auswahl erfolgte zufällig, jedoch wurde auf ein ausgeglichenes Verhältnis weiblicher und männlicher Sprecher geachtet.

Die Aufnahme erfolgte mit einem Sennheiser-Nahbesprechungsmikrofon bei einer Abtastrate von 16 kHz. Der mit mehreren Rechnern und einem weiteren Arbeitsplatz ausgestattete Aufnahmeraum war nicht weiter schallisoliert, jedoch wurden Gespräche oder sonstige Hintergrundgeräusche soweit wie möglich vermieden.

KA–Alph	Sprecher (m/f)	Aufnahmen	Buchstaben
Train	74 (54/20)	8488	57870
Cross	10 (7/3)	979	6675
Test	20 (15/5)	2198	14898
Summe	104 (76/28)	11665	79443

Tabelle 5.3: Karlsruher Buchstabierdaten

5.3 Die „Resource Management Spell"-Daten

Die RM–Spell-Daten sind ein Teil der englischsprachigen „Resource Management Task". In diesem militärisch geprägten Szenario soll per Sprache der Zustand (Position, Ausrüstung etc.) einer Marine-Flotte angefragt und gesteuert werden. Dabei wird auch aus einer Liste von 600 Begriffen buchstabiert, bei denen es sich überwiegend um Namen von Schiffen, Städten, Ländern oder Wochentagen handelt.

Die RM–Spell-Daten umfassen jeweils 15 (in 10 Fällen nur 14, einmal nur 13) Buchstabierungen von 120 Sprechern, die in ruhiger Umgebung mit 16 kHz Abtastrate aufgenommen wurden. Die Testmenge besteht aus 11 Sprechern. Von den 109 Trainingssprechern wurden alle Aufnahmen von 6 Sprechern sowie jeweils eine Aufnahme der verbleibenden 103 Sprecher als Kreuzvalidierungsmenge abgespalten, wodurch sich die in Tabelle 5.4 dokumentierte Aufteilung ergibt.

[3] Die bevorzugt von Schwaben verwendete Aussprache „Dreierles - Es" für ß wurde anfangs ebenfalls berücksichtigt, dann aber wieder fallengelassen.

RM–Spell	Sprecher (m/f)	Aufnahmen	Buchstaben
Train	103 (74/29)	1432	8568
Cross	109 (78/31)	192	1161
Test	11 (7/4)	162	923
Summe	120 (85/35)	1786	10652

Tabelle 5.4: Resource-Management-Buchstabierdaten

5.4 Die Siemens-BUCNAM-Daten

Die BUCNAM-Datenbank enthält deutschsprachige, über das Telefon aufgenommene Buchstabierungen von 722 Vornamen, die der Universität Karlsruhe im Rahmen einer Kooperation von der Siemens AG zur Verfügung gestellt wurden. In den 722 Aufnahmen wurde in 591 oder 82% der Fälle ausschließlich buchstabiert. In den restlichen Aufnahmen wurde zum größten Teil nach dem Funkeralphabet buchstabiert, also etwa „`romeo alpha lima foxtrott`". In seltenen Fällen wurde der Name fließend statt buchstabiert gesprochen, oder es wurden Füllwörter benutzt, z.B. „`h e i n z neues Wort j ö r g`" oder „`D O R doppel E und N`".

Von den 591 Aufnahmen (3594 Buchstaben) wurden 100 als Test- und 491 als Trainingsmenge zufällig ausgewählt. Wegen der relativ geringen Anzahl der Daten wurde auf eine Kreuzvalidierungsmenge verzichtet.

5.5 Die OGI-Buchstabierdaten

Der „Oregon Graduate Institute (OGI) Spelled and Spoken Word Telephone Corpus" [CRF92] enthält etwa 4000 über das öffentliche Telefonnetz aufgenommene englischsprachige Datensätze. Neben Fragen wie „`Von welcher Stadt aus rufen Sie an?`" oder „`Wie ist Ihr Nachname?`" werden die Sprecher aufgefordert, die folgenden Daten zu buchstabieren:

- Nachnamen (spelled last names, SLN)
- Nachnamen mit kurzen Pausen zwischen einzelnen Buchstaben (SLP)
- Vornamen mit Pausen (SFP)
- Die 26 Buchstaben des Alphabets (ALP)

Alle Aufnahmen sind transkribiert inklusive Geräuschen. Anhand der Transkriptionen wurden die Daten in die folgenden Qualitätsstufen kategorisiert (Tabelle 5.5): 0 = enthält ausschließlich Buchstaben, 1 = enthält zusätzlich „harmlose" Geräusche, mit denen man immer rechnen muß[4], 2 = enthält auffälligere Geräusche[5], 3 = Wortabbrüche, 4 =

[4]Dazu wurden gerechnet die mit (ls, br, ln, bn) transkribierten Geräusche für „Lip-smack", „breath", „line-noise" und „background noise".

[5]Husten, „äh" („cough, uh")

enthält gesprochene Wörter, die keine Buchstaben sind. Letzterer Fall (Kategorie 4, Nicht-Buchstaben) kommt in etwa 8% aller Aufnahmen vor.

Kategorie	0	1	2	3	4	Summe
SLN	2279	990	15	5	308	3597
SLP	1158	482	11	22	73	1746
SFP	1253	392	7	35	51	1738
ALP	766	397	5	23	180	1371
Summe	5456	2261	38	85	612	8452

Tabelle 5.5: Kategorisierung der Qualität der OGI-Aufnahmen

OGI–Spell	SLN	SLP	SFP	ALP	Summe
Train	1858	763	775	736	4132
Cross	740	583	572	168	2063
Test	685	305	305	263	1558
Summe Wörter	3283	1651	1652	1167	7753
Sum. Buchstaben	21020	10260	8513	30180	69973

Tabelle 5.6: Buchstabieraufnahmen aus dem „Oregon Graduate Institute Spelled and Spoken Word Telephone Corpus"

Von den 3617 auf der CD-ROM verfügbaren aufgenommenen Anrufen sind 2095 als Trainings-, 792 als Kreuzvalidierungs- und 767 als Testmenge gekennzeichnet. Ohne „Nicht-Buchstaben" und Abbrüche (Kategorien 4 und 3) ergeben sich die in Tabelle 5.6 aufgelisteten Mengen, wie sie für die Experimente in dieser Arbeit eingesetzt wurden.

5.6 Die VODIS-Autodaten

Im Rahmen des EU-Projektes VODIS (Voice Operated Driving Information System) wurden von der Universität Karlsruhe in Zusammenarbeit mit der Robert-Bosch GmbH im fahrenden Auto Sprachdaten gesammelt. Neben Kommandowörtern zur Steuerung des Navigationssystems und des Radios (z.B. „`Zieleingabe`, `Telefonbuch`, `Lautstärke`") wurden von jedem der 100 Sprecher einmal das Alphabet mit Umlauten und Sonderzeichen sowie 5 Namen buchstabiert. In jedem der vier unterschiedlichen Fahrzeuge (BMW, VW, Audi, Renault) wurden die Sprachdaten über mehrere Kanäle mit Mikrofonen aufgenommen, die an verschiedenen Stellen im Auto positioniert waren (Nahbesprechungsmikrofon am Hals, Gurtmikrofon, Dach links, Mitte und rechts). Für die Experimente in dieser Arbeit wurden das Nahbesprechungsmikrofon und das Mikrofon am Dach links genutzt.

Die Aufteilung in Trainings-, Kreuzvalidierungs- und Testmenge (Tabelle 5.7) ist so angelegt, daß sowohl die Fahrzeugtypen als auch weibliche und männliche Sprecher

VODIS-alph	Sprecher					Aufnahmen		Summe
	BMW	VW	Audi	Ren.	Summe (m/f)	Alpha.	Namen	Buchst.
Train	26	23	14	7	**70 (47/23)**	2201	348	5003
Test	7	7	4	2	**20 (14/6)**	319	49	713
Cross	3	4	2	1	**10 (7/3)**	632	92	1345
Summe	36	34	20	10	**100 (68/32)**	3152	489	7061

Tabelle 5.7: Die VODIS-Alph-Auto-Buchstabierdaten: Im linken Teil die Verteilung der Sprecher auf die unterschiedlichen Fahrzeuge, rechts die Anzahl der als Alphabet buchstabierten Einzelbuchstaben, der buchstabierten Namen und die Summe aller gesprochenen Buchstaben

möglichst proportional zwischen den drei Mengen aufgeteilt sind. Allerdings sind in der Testmenge keine der 15 (von insgesamt 100) Aufnahmen vertreten, bei denen bei offenem Fenster aufgenommen wurde.

5.7 Fließend gesprochene und buchstabierte Namen

In der KA–FliBu-Datenbank werden von jedem Sprecher bis zu 50 Nachnamen fließend gesprochen und dann buchstabiert, also zum Beispiel „`Kemp K E M P`". Die Namen wurden zufällig ausgewählt aus der Liste der 110 000 Karlsruher Nachnamen. Die Aufnahmebedingungen sind die gleichen wie bei den KA–Alph-Daten (Abschnitt 5.2). Insgesamt wurden Daten von 60 Sprechern gesammelt. Drei Sprecher wurden ausgeschlossen: mrg3 wegen eines technischen Aufnahmefehlers, und fhh1 und mal3 wegen ihres Akzents (ausländische Sprecher).

In den KA–FliBu-Daten sind keine Geräusche transkribiert. In Tabelle 5.8 sind die Daten nach folgenden Kategorien aufgelistet: 0 = Name wurde korrekt buchstabiert, 1 = Fehler in buchstabiertem Teil der Aufnahme, 2 = enthält Wörter, die weder Buchstaben noch fließend gesprochene Nachnamen sind. Die zweite Zeile enthält die Anzahl der Daten, zu denen phonetische Transkriptionen aus dem ONOMASTICA-Aussprachewörterbuch verfügbar sind.

Neben schlichten Konzentrationsfehlern sind interessante Fehler der Kategorie (2) vergessene oder klanglich verwechselte Buchstaben. Einige Beispiele dazu werden in Abschnitt 8.8 aufgeführt.

KA–FliBu	Sprecher (m/f)	0	1	2	Summe
alle KA–FliBu-Daten	57 (48/9)	2650	94	36	2780/18741
Transkriptionen verfügbar	57 (48/9)	1301	36	9	1346/8719

Tabelle 5.8: Die KA–FliBu-Daten. Kategorie 0/1 steht für korrekt/falsch buchstabierte Namen, in Kategorie 2 wurden vokabularfremde Wörter benutzt.

5.8 Der VERBMOBIL-Korpus

VERBMOBIL ist ein vom BMBF gefördertes Verbund-Forschungsprojekt mit dem Ziel, einen spontansprachlichen, automatischen, mobilen Übersetzer für Terminabsprachen zu errichten. Im Rahmen des Projektes wurde eine große Anzahl von Dialogen aufgenommen, bei denen die Dialogpartner versuchen, anhand ihrer vorgegebenen Terminkalender einen geeigneten Zeitpunkt für ein gemeinsames Treffen zu arrangieren (Abbildung 5.1). Zu Beginn des Gespräches stellen sich die Teilnehmer gegenseitig vor, dabei wird oft der eigene (gestellte) Nachname buchstabiert. Es treten hier also nicht isolierte, pure Buchstabierungen auf, sondern Buchstabierungen in spontaner Sprache. Einige Beispiele:

```
Hier ist Ableitner A B L E I T N E R

ja grüß Gott mein Name ist von Juliß von V O N J U L I und scharf S ja und ...

Tag mein Name ist von Juliß soll ich Ihnen den buchstabieren V O N und dann
Juliß wie Juli und hinten ein scharfes S und wie ist Ihr Name

guten Tag mein Name ist Schlör-Quell ich buchstabiere Schule Ludwig Ö Richard
Strich Q Ulrich Emil zweimal Ludwig ich möchte gern ...
```

Abbildung 5.1: Das VERBMOBIL-Szenario

Die VERBMOBIL-Datenbank ist in ständigem Wachstum begriffen. Als die in Abschnitt 8.6 beschriebenen Experimente durchgeführt wurden, lagen 4 CD-ROMs mit mehreren tausend Sätzen vor, davon 255 Buchstabiersätze in der Trainingsmenge (CD 1-3) und 115 Sätze in der Testmenge (CD 4).

Kapitel 6

Das MS-TDNN als Buchstabiererkenner

Der in dieser Arbeit vorgestellte Buchstabiererkenner [HW93a, HW93b, HW93c] basiert auf einer konnektionistischen Architektur, dem *Multi-State Time-Delay Neural Network (MS-TDNN)*, einer Weiterentwicklung des *Time-Delay Neural Network (TDNN)*. Dieses Kapitel beginnt mit einer Beschreibung der TDNN- und MS-TDNN-Architekturen. Wichtiger noch als architektonische Details sind die Trainingsverfahren, mit denen das MS-TDNN die Buchstabiererkennung lernt. Dabei wächst das MS-TDNN gewissermaßen in seine Aufgaben hinein, indem zuerst einzelne Phoneme, dann Buchstaben und schließlich ganze Buchstabensequenzen diskriminativ trainiert werden. Dieser schrittweise Aufbau des Systems und die dabei auftretenden Fragestellungen werden ausführlich beschrieben und systematisch mit Experimenten auf mehreren Datenbanken evaluiert. Außerdem werden der Einfluß diverser Architekturparameter sowie Möglichkeiten zur Modularisierung der Netze untersucht. Vor der abschließenden Zusammenfassung sind in Abschnitt 6.9 die Erkennungsergebnisse des MS-TDNN denen anderer Erkenner gegenübergestellt, zusammen mit der Beschreibung einiger Experimente auf weiteren Datenbanken mit über Telefon aufgenommenen Daten.

In diesem Kapitel geht es ausschließlich um die Optimierung der *akustischen* Komponente des Erkenners. Das MS-TDNN soll so trainiert werden, daß kontinuierlich gesprochene Buchstaben sprecherunabhängig möglichst gut klassifiziert werden, ganz unabhängig davon, in welcher Reihenfolge sie präsentiert werden. Wenn nicht zufällige Sequenzen, sondern Wörter oder Eigennamen buchstabiert werden, so treten verschiedene Buchstabenfolgen mit unterschiedlichen Wahrscheinlichkeiten auf. In Kapitel 7 ist beschrieben, wie dieses Wissen in Form von *Sprachmodellen* zu einer deutlichen Verbesserung der Erkennungsleistung eingesetzt werden kann.

6.1 Das Time-Delay Neural Network (TDNN)

6.1.1 Problemstellung und Entwurfskriterien

Das Time-Delay Neural Network (TDNN) wurde 1987 von Waibel und Lang [WHH+87, WHH+89, Lan89, LWH90] zur Klassifikation von Phonemen entwickelt. Eingabe ist ein kurzes Stück Sprache, das eines der drei Phoneme /b/, /d/ oder /g/ repräsentiert. Das TDNN berechnet zu jedem Abschnitt der Eingabe Phonembewertungen, die in der Ausgabeschicht in einem Neuron pro Phonem zusammengefaßt werden, wie in Abbildung 6.1 dargestellt. Die wesentlichen Kriterien bei der Entwicklung des TDNN waren:

- Die **Fähigkeit, zeitliche Zusammenhänge zu erlernen.** Dies wird erzielt, indem sowohl in der Eingabe- als auch in der verborgenen Schicht ein zeitlicher Kontext in die Zukunft und die Vergangenheit (*Time-Delay*) berücksichtigt wird.
- **Verschiebungsinvarianz:** Die charakteristischen Merkmale eines Sprachsignals sollen unabhängig von ihrer Position in der Eingabe erkannt werden. Dazu sind die Gewichte des TDNN verschiebungsinvariant: Ein Eingabefenster mit Zeitverzögerungen (Time-Delays) wird über die Schichten hinweggeschoben und produziert zu jedem Zeitpunkt mit dem gleichen Satz von Gewichten eine Ausgabe in der Phonemschicht. Diese Bewertungen werden dann über die Zeit integriert, d.h. aufsummiert, und in das entsprechende Ausgabeneuron weitergeleitet.
- Im Vergleich zu vollständig verbundenen Schichten benötigen die gleitenden Verbindungen des TDNN nur eine relativ **kleine Anzahl von Parametern.**

Durch die zeitliche Integration der Bewertungen der Phonemschicht ist das TDNN in der Lage, Sprachmuster unterschiedlicher Länge zu erkennen[1].

6.1.2 Architektur

In Abbildung 6.1 sind die vier Schichten des TDNN zu erkennen. An die Eingabeschicht wird das Sprachsignal gelegt, das zunächst in die verborgene, dann in die Phonemschicht propagiert wird. Mit einem gleitenden Fenster von Verbindungen wird aus jeweils mehreren Sprachvektoren der Eingabeschicht ein Vektor der verborgenen Schicht berechnet. Entsprechend erhält man für die Phonemschicht zu jedem Zeitpunkt eine Bewertung der zu klassifizierenden Phoneme. Die variabel lange Phonemschicht wird in der Ausgabeschicht in ein einziges Neuron pro Phonem zusammengeführt. Diese Zusammenhänge sind im folgenden exakt formuliert.

[1] Wenngleich in den original TDNN-Experimenten die zu erkennenden Phoneme alle die gleiche Länge hatten

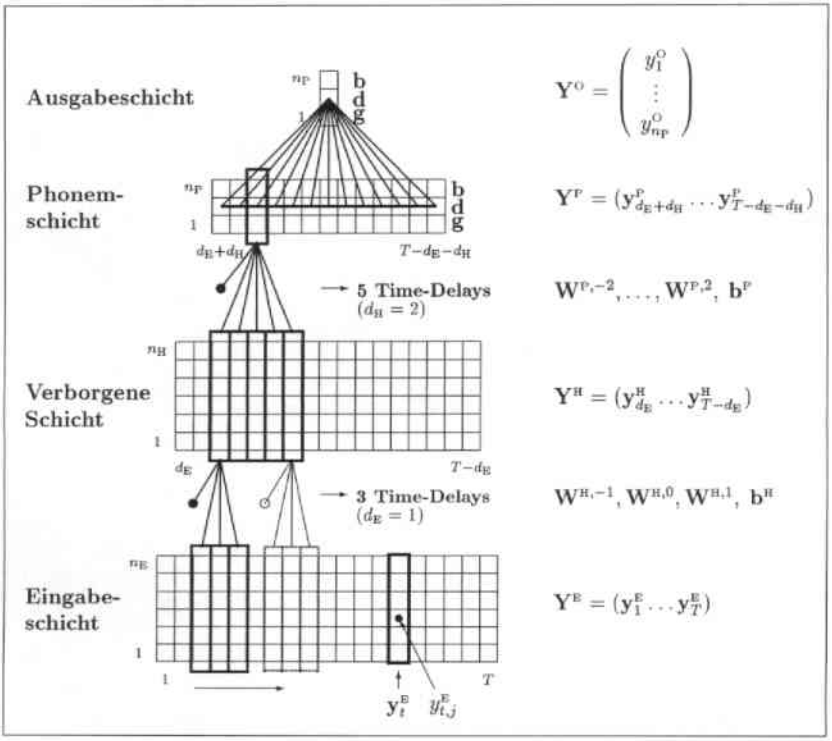


Abbildung 6.1: Architektur und Namensbezeichnungen des TDNN. Die Ausgaben der Netzschichten sind mit $\mathbf{Y}$, die Gewichte mit $\mathbf{W}$ und $\boldsymbol{b}$ bezeichnet.

An die **Eingabeschicht** $\mathbf{Y}^E$ wird das zu klassifizierende Sprachsignal $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_T)$ gelegt, das aus T Sprachvektoren[2] $\mathbf{x}_t$, $t = 1, \ldots, T$ der Dimension n_E vorliegt. In der Eingabeschicht findet keine Verarbeitung statt, das Sprachsignal wird direkt übernommen:

$$\mathbf{Y}^E = \left(\mathbf{y}_1^E \quad \cdots \quad \mathbf{y}_T^E \right) = \left(\begin{pmatrix} y_{1,1}^E \\ \vdots \\ y_{1,n_E}^E \end{pmatrix} \quad \cdots \quad \begin{pmatrix} y_{T,1}^E \\ \vdots \\ y_{T,n_E}^E \end{pmatrix} \right) := \left(\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_T \right)$$

In der **verborgenen Schicht** $\mathbf{Y}^H$ (H = „Hidden Layer") existiert zu jedem Vektor $\mathbf{y}_t^E$ der Eingabeschicht (mit Ausnahme der Ränder) ein entsprechender Vektor $\mathbf{y}_t^H$. Dieser

[2] In der englischen Literatur mit *Frames* („Rahmen") bezeichnet

ist über Gewichte $\mathbf{W}$ und Schwellwerte $\mathbf{b}$ (im Englischen *Bias*) mit $\mathbf{y}_t^{\mathrm{E}}$ sowie mit d_{E} Vektoren zur Linken und zur Rechten vollständig[3] verbunden. $\mathbf{W}$ und $\mathbf{b}$ haben die Form:

$$\mathbf{W} = (w_{i,j}) = \begin{pmatrix} w_{1,1} & \cdots & w_{1,n_{\mathrm{E}}} \\ \vdots & & \vdots \\ w_{n_{\mathrm{H}},1} & \cdots & w_{n_{\mathrm{H}},n_{\mathrm{E}}} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_{\mathrm{H}}} \end{pmatrix}$$

Zwei Superskripten definieren, zu welcher Verbindung eine Gewichtsmatrix $\mathbf{W}^{S,d}$ gehört: S gibt die Zielschicht, d die Zeitverzögerung an. Zur Berechnung der n_{H}-dimensionalen Vektoren $\mathbf{y}_t^{\mathrm{H}}$ benötigt man (für $d_{\mathrm{E}} = 1$) drei $n_{\mathrm{H}} \times n_{\mathrm{E}}$-Gewichtsmatrizen $\mathbf{W}^{\mathrm{H},d}, d = -1, 0, 1$, sowie einen $1 \times n_{\mathrm{H}}$-Schwellwertvektor $\mathbf{b}^{\mathrm{H}}$. Zuerst wird die Aktivierung $\mathbf{x}_t^{\mathrm{H}}$ berechnet, auf die dann komponentenweise die Sigmoidfunktion σ angewandt wird:

$$\mathbf{x}_t^{\mathrm{H}} = \mathbf{b}^{\mathrm{H}} + \sum_{d=-d_{\mathrm{E}}}^{d_{\mathrm{E}}} \mathbf{W}^{\mathrm{H},d} \cdot \mathbf{y}_{t+d}^{\mathrm{E}} \tag{6.1}$$

$$\mathbf{y}_t^{\mathrm{H}} = \sigma(\mathbf{x}_t^{\mathrm{H}}) \tag{6.2}$$

Die Vektoren $\mathbf{y}_t^{\mathrm{P}}$ der **Phonemschicht** $\mathbf{Y}^{\mathrm{P}}$ liefern für jeden Zeitpunkt t eine Bewertung $y_{t,j}^{\mathrm{P}}$ für das j-te Phonem. Mit $d_{\mathrm{H}} = 2$ wird in der verborgenen Schicht ein zeitlicher Kontext von insgesamt fünf Vektoren betrachtet. Aufgrund der Zeitverzögerungen werden in der Phonemschicht nur die Vektoren im Zeitraum $\mathbf{Y}^{\mathrm{P}} = (\mathbf{y}_{d_{\mathrm{E}}+d_{\mathrm{H}}}^{\mathrm{P}} \cdots \mathbf{y}_{T-d_{\mathrm{E}}-d_{\mathrm{H}}}^{\mathrm{P}})$ berechnet, analog zu (6.1) mit Hilfe der $n_{\mathrm{P}} \times n_{\mathrm{H}}$-Gewichtsmatrizen $\mathbf{W}^{\mathrm{P},d}, d = -2, -1, 0, 1, 2$, und $\mathbf{b}^{\mathrm{P}}$:

$$\mathbf{x}_t^{\mathrm{P}} = \mathbf{b}^{\mathrm{P}} + \sum_{d=-d_{\mathrm{H}}}^{d_{\mathrm{H}}} \mathbf{W}^{\mathrm{P},d} \cdot \mathbf{y}_{t+d}^{\mathrm{H}} \tag{6.3}$$

$$\mathbf{y}_t^{\mathrm{P}} = \sigma(\mathbf{x}_t^{\mathrm{P}}) \tag{6.4}$$

Die Phonembewertungen $y_{t,i}^{\mathrm{P}}$ werden über die Zeit integriert und in die **Ausgabeschicht** $\mathbf{y}^{\mathrm{O}}$ propagiert, die für jedes der $n_{\mathrm{P}} = 3$ Phoneme genau ein Neuron enthält:

$$\mathbf{y}^{\mathrm{O}} = (y_i^{\mathrm{O}}) = \begin{pmatrix} y_1^{\mathrm{O}} \\ \vdots \\ y_{n_{\mathrm{P}}}^{\mathrm{O}} \end{pmatrix}$$

Die Ausgaben y_i^{O} sind Diskriminanzfunktionen, für deren Berechnung unterschiedliche Heuristiken vorgeschlagen wurden. In der Originalarbeit von Waibel [WHH+89] werden die Phonembewertungen für jedes Phonem aufsummiert und zusammen mit einem Schwellwert b_i^{O} durch eine weitere Sigmoidfunktion propagiert:

$$y_i^{\mathrm{O}} = \sigma(b_i^{\mathrm{O}} + \sum_{t=d_{\mathrm{E}}+d_{\mathrm{H}}}^{T-d_{\mathrm{E}}-d_{\mathrm{H}}} y_{t,i}^{\mathrm{P}}) \tag{6.5}$$

[3] Jedes Neuron aus dem Zielvektor ist mit jedem Neuron aus dem Ursprungsvektor verbunden.

In [LWH90] wird auf die Sigmoidfunktion verzichtet, dafür werden die Phonemausgaben quadriert, um der besten Bewertung ein höheres Gewicht zu verleihen:

$$y_i^{\mathrm{O}} = \sum_{t=d_{\mathrm{E}}+d_{\mathrm{H}}}^{T-d_{\mathrm{E}}-d_{\mathrm{H}}} (y_{t,i}^{\mathrm{P}})^2 \tag{6.6}$$

Noch einfacher ist der Ansatz, der später in erweiterter Form im MS-TDNN zum Einsatz kommt. Um variabel lange Eingaben zu berücksichtigen, werden dabei die aufsummierten Bewertungen über die Länge $F := T - 2 \cdot (d_{\mathrm{E}} + d_{\mathrm{H}})$ der Phonemschicht normiert:

$$y_i^{\mathrm{O}} = \frac{1}{F} \sum_{t=d_{\mathrm{E}}+d_{\mathrm{H}}}^{T-d_{\mathrm{E}}-d_{\mathrm{H}}} y_{t,i}^{\mathrm{P}} \tag{6.7}$$

Die Werte für die Breiten 3 und 5 der Fenster des betrachteten Kontextes, wie sie in Abbildung 6.1 abgebildet und im weiteren eingesetzt werden, sind empirisch bestimmt. Für diese Netzkonfiguration bestehen die lernbaren Parameter insgesamt aus den acht Gewichtsmatrizen $\mathbf{W}^{\mathrm{H},\{-1,0,1\}}, \mathbf{W}^{\mathrm{P},\{-2,-1,0,1,2\}}$ sowie den Schwellwertvektoren $\mathbf{b}^{\mathrm{H}}, \mathbf{b}^{\mathrm{P}}$ und eventuell $\mathbf{b}^{\mathrm{O}}$. Mit $n_{\mathrm{E}} = 16$ Eingabeneuronen und $n_{\mathrm{H}} = 15$ verborgenen Neuronen pro Vektor besitzt das in [WHH+89] beschriebene Netz die relativ geringe Anzahl von $(15 + 3 * 16 * 15) + (3 + 5 * 15 * 3) + 3 = 966$ Parametern.

6.1.3 Training des TDNN

Lernziel ist die korrekte Klassifikation der Spracheingabe als eines von drei Phonemen b, d oder g. Die Trainingsmenge besteht aus K Mustern $\{(\mathbf{X}^k, c^k)\}, k = 1 \ldots K, c^k \in \{\mathrm{b,d,g}\}$. Die Sollausgaben $\mathbf{z}^k = (z_i^k) = (\delta_{i,c^k})$ sind 1-aus-N-Kodierungen der korrekten Klasse c^k, also 1 für den korrekten und 0 für die restlichen Ausgabeknoten. Das TDNN realisiert eine Funktion $\mathbf{y}^{\mathrm{O}} = \mathrm{TDNN}(\mathbf{W}, \mathbf{X})$. Gesucht sind Parameter oder Gewichte $\mathbf{W}$, die (z.B.) das Kriterium des mittleren quadratischen Fehlers auf den Ausgaben $\mathbf{y}^{\mathrm{O},k}$ aller K Trainingsmuster $\mathbf{X}^k$ minimieren:

$$E_{MSE} = \sum_{k=1}^{K} E_{MSE}(\mathbf{y}^{\mathrm{O},k}, \mathbf{z}^k) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{n_{\mathrm{P}}} (y_i^{\mathrm{O},k} - z_i^k)^2$$

Dies geschieht analog zum bereits vorgestellten Backpropagation-Verfahren eines MLP durch Gradientenabstieg. Für ein gegebenes Trainingsmuster $(\mathbf{X}^k, c^k)$ betrachten wir den Fehler als Funktion der Gewichte, $E(\mathbf{W}) = E(\mathrm{TDNN}(\mathbf{W}, \mathbf{X}^k), \mathbf{z}^k)$. Die Berechnung der Ableitungen des Fehlers $\partial E / \partial w_{i,j}$ erfolgt im wesentlichen wie in einem MLP, zusätzlich müssen jedoch die Time-Delays sowie die zeitinvariante Verschiebung der Gewichte berücksichtigt werden. Als gemeinsame „Schnittstelle" für das TDNN und später das MS-TDNN betrachten wir zuerst die Abhängigkeit von den Phonemausgaben $y_{t,i}^{\mathrm{P}}$, die ihrerseits die Ausgabeschicht $\mathbf{y}^{\mathrm{O}}$ beeinflussen. Mit der Kettenregel ergibt sich die Ableitung des Fehlers nach $y_{t,i}^{\mathrm{P}}$:

$$\frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} = \frac{\partial E}{\partial \mathbf{y}^{\mathrm{O}}} \cdot \frac{\partial \mathbf{y}^{\mathrm{O}}}{\partial y^{\mathrm{P}}_{t,i}} = \sum_{k=1}^{n_{\mathrm{P}}} \frac{\partial E}{\partial y^{\mathrm{O}}_{k}} \cdot \frac{\partial y^{\mathrm{O}}_{k}}{\partial y^{\mathrm{P}}_{t,i}} = \frac{\partial E}{\partial y^{\mathrm{O}}_{i}} \cdot \frac{\partial y^{\mathrm{O}}_{i}}{\partial y^{\mathrm{P}}_{t,i}} = \frac{1}{F} \frac{\partial E}{\partial y^{\mathrm{O}}_{i}} \tag{6.8}$$

Die Summe über k entfällt, da die i-te Phonembewertung $y^{\mathrm{P}}_{t,i}$ nur die i-te Ausgabe y^{O}_{i} beeinflußt. Die gesuchte Ableitung nach den Gewichten $w^{\mathrm{P},d}_{i,j}$ ist dann

$$\frac{\partial E}{\partial w^{\mathrm{P},d}_{i,j}} = \frac{\partial E}{\partial \mathbf{Y}^{\mathrm{P}}} \cdot \frac{\partial \mathbf{Y}^{\mathrm{P}}}{\partial w^{\mathrm{P},d}_{i,j}} = \sum_{t} \sum_{k=1}^{n_{\mathrm{P}}} \frac{\partial E}{\partial y^{\mathrm{P}}_{t,k}} \cdot \frac{\partial y^{\mathrm{P}}_{t,k}}{\partial w^{\mathrm{P},d}_{i,j}} \tag{6.9}$$

Nach einigen weiteren, in Anhang A.2 genau hergeleiteten Umformungen erhält man die in Tabelle 6.1 zusammengefaßten Ableitungen.

Anschaulich gesprochen wird zuerst die Ableitung $\partial E/\partial \mathbf{y}^{\mathrm{O}}$ berechnet, die nur von der eingesetzten Fehlerfunktion abhängt. Geht man mit $\partial E/\partial y$ „rückwärts" durch die Transfer-(Sigmoid-)Funktion, erhält man $\partial E/\partial x$. Dieses Fehlersignal wird dann entlang der Gewichte der Time-Delays in die davorliegende Schicht zurückpropagiert, und so weiter. Für die Berechnung aller Phonembewertungen $\mathbf{y}^{\mathrm{P}}_{t}$ gibt es nur einen Satz von Gewichten, entsprechend werden die Fehlersignale durch die Anzahl F der Phonemvektoren gemittelt.

Die aus jedem Trainingsmuster berechneten Gewichtsänderungen $\partial E/\partial w$ werden nicht akkumuliert, sondern sofort zur Aktualisierung der Gewichte genutzt (*learning by pattern*). Ebenfalls beschleunigend auf das Lernverhalten wirkt sich eine um den Nullpunkt im Bereich $[-1, 1]$ normalisierte Eingabe aus.

$$\frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} = \frac{1}{F} \frac{\partial E}{\partial y^{\mathrm{O}}_{i}} \qquad \text{(entsprechend der Fehlerfunktion)} \tag{6.10}$$

$$\frac{\partial E}{\partial w^{\mathrm{P},d}_{i,j}} = \sum_{t} \frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} \sigma'(x^{\mathrm{P}}_{t,i}) \cdot y^{\mathrm{H}}_{t+d,j} \tag{6.11}$$

$$\frac{\partial E}{\partial y^{\mathrm{H}}_{t,i}} = \sum_{d'=-d_{\mathrm{H}}}^{d_{\mathrm{H}}} \sum_{k=1}^{n_{\mathrm{P}}} \frac{\partial E}{\partial y^{\mathrm{P}}_{t-d',k}} \cdot \sigma'(x^{\mathrm{P}}_{t-d',k}) \cdot w^{\mathrm{P},d'}_{k,i} \tag{6.12}$$

$$\frac{\partial E}{\partial w^{\mathrm{H},d}_{i,j}} = \sum_{t} \frac{\partial E}{\partial y^{\mathrm{H}}_{t,i}} \sigma'(x^{\mathrm{H}}_{t,i}) \cdot y^{\mathrm{E}}_{t+d,j} \tag{6.13}$$

Tabelle 6.1: Berechnung der Fehlersignale für die Gewichte des TDNN. Die nicht aufgeführten Ableitungen für $\partial E/\partial b^{\{\mathrm{H},\mathrm{P}\}}_{i}$ entsprechen (6.11) und (6.13), wenn der dort vorkommende Faktor y durch 1 ersetzt wird.

6.2 Das Multi-State Time-Delay Neural Network (MS-TDNN)

Das Multi-State Time-Delay Neural Network (MS-TDNN) ist eine Weiterentwicklung des TDNN und wurde erstmals von Haffner, Franzini und Waibel [HFW91] vorgestellt. In der MS-TDNN-Architektur werden nun nicht einzelne Phoneme, sondern ganze Wörter, also Sequenzen von Phonemen oder Zuständen erkannt, daher der Name „Multi-State". Statt Phonemen (wie im TDNN) müssen im MS-TDNN also ganze Wortmodelle zeitlich integriert werden. In hybriden NN-HMM-Ansätzen (siehe Abschnitt 4.1.3) wird diese Zeitanpassung mit einem externen HMM realisiert, beim MS-TDNN erfolgt dieser Schritt ebenfalls durch eine dynamische Zeitanpassung (DTW), die aber direkt in die Netzwerkarchitektur und damit in die konnektionistischen Trainingsverfahren integriert ist.

6.2.1 Wortmodelle

Im Falle eines Buchstabiererkenners besteht der zu erkennende Wortschatz V_L aus den (gesprochenen) Buchstaben des Alphabets:

$$V_L := \{L_1, L_2, \ldots, L_n\} = \{A, B, \ldots, Z\}$$

Jedes Wort, also jeder Buchstabe, besteht aus einer Sequenz von Phonemen oder phonemähnlichen Zuständen. Die Menge dieser Zustände bezeichnen wir mit

$$\mathcal{S} := \{s_1, s_2, \ldots, s_{n_P}\}$$

Zu jedem der n_P Zustände gibt es eine Zeile von Neuronen in der Phonemschicht $\mathbf{Y}^P$, wobei $y^P_{t,j}$ die Bewertung von Zustand s_j zum Zeitpunkt t berechnet.

Das Modell für Buchstabe L_i bestehe aus n_i Zuständen, wobei $[i,k]$ der Index des k-ten Zustandes in Buchstabe L_i ist:

$$L_i = (s_{[i,1]}, s_{[i,2]}, \ldots s_{[i,n_i]}) \tag{6.14}$$

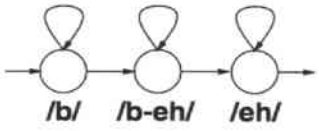


Abbildung 6.2: Akustisches Modell für Buchstabe B

Wir benutzen einfache Links-Rechts-Modelle, wie in Abbildung 6.2 für den Buchstaben B veranschaulicht. Ein Durchlauf durch ein Modell muß stets beim ersten Zustand beginnen und im letzten enden, und von einem Zustand sind nur Übergänge in denselben oder den direkt nachfolgenden erlaubt.

Zu jedem Zeitpunkt t liegt für jeden Zustand s_i eine Bewertung $y^{\mathrm{P}}_{t,i}$ vor. Trägt man diese über der Zeit auf, kann man sich einen Durchlauf durch das Modell L_i als einen Pfad durch die entsprechende Matrix vorstellen. Aufgrund der oben genannten Einschränkungen muß der Pfad in der linken unteren Ecke beginnen, rechts oben enden und entweder in derselben Zeile oder eine Zeile höher weiterlaufen[4]. Ein solcher Pfad ist in Abbildung 6.3 veranschaulicht. Jeder zu erkennende Buchstabe L_i hat sein eigenes Modell und damit eine eigene Bewertungsmatrix, deren Zeilen gerade aus den Phonembewertungen $y^{\mathrm{P}}_{t,[i,1]} \cdots y^{\mathrm{P}}_{t,[i,n_i]}$ der zu Buchstabe L_i gehörenden Phoneme (siehe Gl. (6.14)) bestehen.

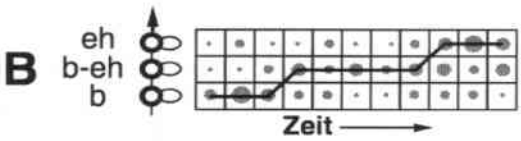


Abbildung 6.3: Pfad durch die Bewertungsmatrix für das akustische Modell von Buchstabe B

Die phonetische Modellierung der englischen Buchstaben wurde im wesentlichen von [HW92] übernommen, sie ist im Anhang in Tabelle B.5 zusammengestellt. Die dort ebenfalls aufgeführten deutschen Buchstabenmodelle sind nach demselben Schema erstellt: Die konventionelle Phonemumschrift, z.B. „/b/ /eh/" für den Buchstaben B, wird um Zustände erweitert, die den Übergang zwischen den beiden Phonemen explizit repräsentieren, in obigem Beispiel etwa „/b/ /b-eh/ /eh/". Dadurch wird eine kontextabhängige Modellierung für den Beginn des Phonems /eh/ erreicht.

Mit dem Wort für Stille ergeben sich im Englischen damit $n_{\mathrm{W}} = 27$ Wortmodelle und $n_{\mathrm{P}} = 59$ Phoneme. Im deutschen Alphabet gibt es mit den Umlauten (Ä, Ö, Ü) und Eszett (ß) insgesamt 30 Buchstaben. Weiterhin wurden „doppel" sowie „Strich" und „Bindestrich" als in Buchstabierungen häufig auftretende Wörter berücksichtigt, so daß sich mit insgesamt drei Aussprachevarianten für Eszett (ß) und dem Wort für Stille insgesamt $n_{\mathrm{W}} = 36$ Wortmodelle ergeben, die mit $n_{\mathrm{P}} = 70$ Zuständen modelliert werden.

6.2.2 Klassifikation auf Wortebene

Im TDNN wurde in der Ausgabeschicht für jedes zu klassifizierende Phonem i mit der i-ten Ausgabe eine Entscheidungsfunktion $g_i(\boldsymbol{x}) = y^{\mathrm{O}}_i$ berechnet. Im MS-TDNN wird

[4] Damit alle konkurrierenden Pfade die gleiche Länge besitzen, sind vertikale Schritte verboten.

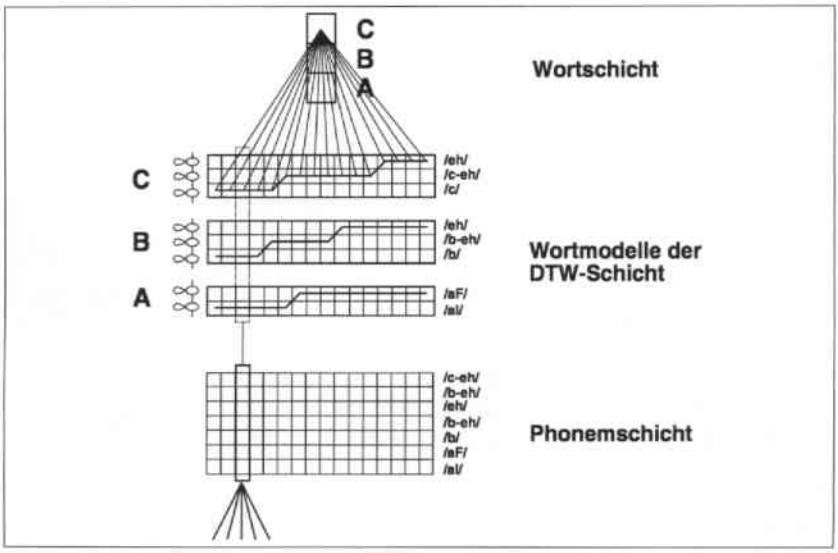


Abbildung 6.4: Struktur des MS-TDNN mit den Wortmodellen der Buchstaben A,B und C

zur Klassifikation von Buchstaben (im allgemeinen Fall von Wörtern) eine Schicht $\mathbf{y}^{\mathrm{w}}$ eingeführt[5] mit einem Knoten y_i^{w} für jeden zu erkennenden Buchstaben L_i:

$$\mathbf{y}^{\mathrm{w}} = \begin{pmatrix} y_1^{\mathrm{w}} \\ \vdots \\ y_{n_{\mathrm{W}}}^{\mathrm{w}} \end{pmatrix}$$

Die Berechnung der y_i^{w} erfolgt mit Hilfe der oben diskutierten Wortmodelle, wie in Abbildung 6.4 schematisch und in Abbildung 6.5 anhand echter Sprachdaten dargestellt.

Entsprechend der in Abschnitt 2.2 dargelegten Unterscheidung zur Berechnung der Entscheidungsfunktionen $g_i(x) = y_i^{\mathrm{w}}$ kann man entweder in konnektionistischer Manier die Entscheidungsfunktionen direkt oder aber mittels eines hybriden NN-HMM-Ansatzes berechnen. Im letzteren Fall werden die Phonembewertungen als a posteriori Wahrscheinlichkeiten $p(s_i|\boldsymbol{x})$ interpretiert, zu klassenbedingten Wahrscheinlichkeiten konvertiert und als Emissionswahrscheinlichkeiten eines HMM interpretiert. Diese beiden Möglichkeiten sind im folgenden genauer beschrieben.

[5] Der Index w steht für Wort und soll andeuten, daß mit dem MS-TDNN statt Buchstaben genausogut andere Wörter wie z.B. Ziffern erkannt werden können.

Konnektionistische Phonemintegration

Wie beim TDNN werden die Phonembewertungen über die Zeit integriert, indem sie aufsummiert werden. Im TDNN mußte dazu jeweils nur ein Phonem zeilenweise summiert werden. Im MS-TDNN gibt es mehrere Phoneme pro Buchstabenmodell, von denen zu jedem Zeitpunkt ebenfalls nur eines berücksichtigt werden soll. Diese Zuordnung geschieht (für jeden Buchstaben L_i) durch den in Abbildung 6.3 visualisierten Pfad $\pi_i(t), t = 1 \ldots F$, der angibt, in welchem Phonem $s_{\pi_i(t)}$ von Buchstabe L_i sich der Pfad zum Zeitpunkt t befindet[6]. Entlang dieses Pfades wird dann aufsummiert:

$$y_i^{\mathrm{W}} = \frac{1}{F} \sum_{t=1}^{F} y^{\mathrm{P}}_{t,\pi_i(t)} \tag{6.15}$$

Mittels dynamischer Programmierung wird ein optimaler Pfad, also diejenige Folge von Indizes $\pi_i(1) \ldots \pi_i(F)$ berechnet, deren Bewertungen die höchste Summe ergeben, unter Berücksichtigung der üblichen Einschränkungen (Beginn im ersten Zustand etc.). Um von den variablen Längen F der Spracheingaben unabhängig zu werden, wird die Summe mit $1/F$ normiert.

Hybrides NN-HMM

Die Phonembewertungen $y^{\mathrm{P}}_{t,j}$ werden als a posteriori Wahrscheinlichkeiten

$$y^{\mathrm{P}}_{t,j} = P(q_t = s_j | \bar{\boldsymbol{x}}_t)$$

interpretiert. Da im TDNN nicht nur ein Vektor, sondern mit den Time-Delays in Eingabe- und verborgener Schicht ein Kontext von insgesamt sieben Vektoren berücksichtigt wird, entspricht $\bar{\boldsymbol{x}}_t$ dem Eingabefenster $(\boldsymbol{x}_{t-3} \ldots \boldsymbol{x}_{t+3})$. Mit der Bayes-Regel können die klassenbedingten Wahrscheinlichkeiten berechnet werden:

$$p(\bar{\boldsymbol{x}}_t | s_j) = \frac{P(s_j | \bar{\boldsymbol{x}}_t) p(\bar{\boldsymbol{x}}_t)}{P(s_j)} \tag{6.16}$$

$p(\bar{\boldsymbol{x}}_t | s_j)$ entspricht der Emissionswahrscheinlichkeit $b_j(\boldsymbol{x})$ eines kontinuierlichen HMM. Mit dem Viterbi-Algorithmus kann zur Eingabe $\mathbf{X}$ der Viterbi-Pfad $\boldsymbol{\pi}^*$ mit der höchsten Observationswahrscheinlichkeit $p^*(\mathbf{X}|\lambda_i) := p(\mathbf{X}, \boldsymbol{\pi}^*|\lambda_i)$ berechnet werden:

$$p^*(\mathbf{X}|\lambda_i) = \max_{\boldsymbol{\pi}} \prod_{t=1}^{F} p(\bar{\boldsymbol{x}}_t | q_t = s_{\pi_i(t)}) \tag{6.17}$$

Transitionswahrscheinlichkeiten sind in (6.17) ignoriert – sie werden auch in den meisten HMM-Systemen nicht berücksichtigt bzw. gleichwahrscheinlich gesetzt.

[6] Nach der Notation aus Abschnitt 6.1.2 ist $F := T - 2 \cdot (d_{\mathrm{E}} + d_{\mathrm{H}})$ die Anzahl der Zeitschritte $t = d_{\mathrm{E}} + d_{\mathrm{H}}, \ldots, T - d_{\mathrm{E}} - d_{\mathrm{H}}$ in der Phonemschicht, die wegen der Time-Delays d_{E} und d_{H} nicht die volle Länge T der Eingabe besitzt. Zur Vereinfachung der Notation werden die Neuronen der Phonemschicht von nun an mit $t = 1, \ldots, F$ indiziert.

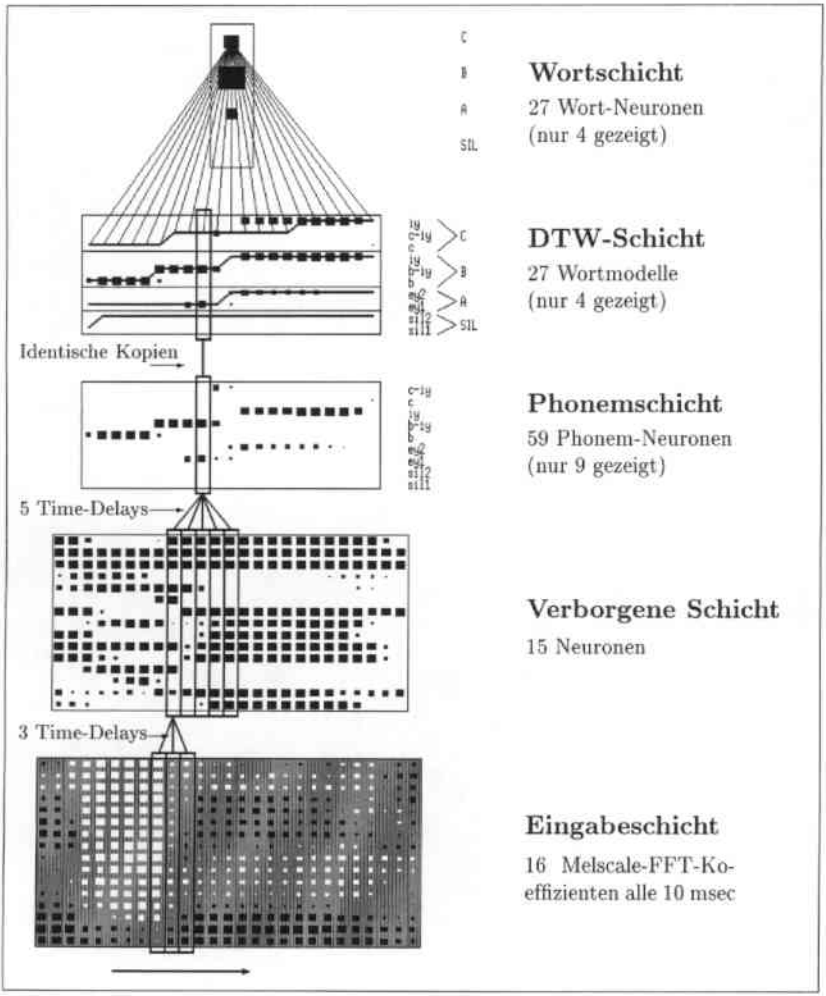


Abbildung 6.5: Das MS-TDNN erkennt den Buchstaben B. In der Phonem-, DTW- und Wortschicht sind nur die Phoneme für Stille (SIL) und die ersten drei (englischen) Buchstaben A,B,C gezeigt.

6.3 Training des MS-TDNN

Das Training des MS-TDNN ist ein mehrstufiger Prozeß, der mit den Phonemen als kleinsten Spracheinheiten beginnt, anschließend Wörter betrachtet und mit ganzen Sätzen endet. Diese in den folgenden Abschnitten beschriebenen drei Trainingsschritte lassen sich wie folgt zusammenfassen:

- Lernziel auf **Phonemebene** ist die Phonemklassifikation. Dazu werden nur die ersten drei Schichten des MS-TDNN benötigt.
- Das Optimierungskriterium auf **Wortebene** ist nicht die Klassifikation von Phonemen, sondern von einzelnen Buchstaben. Die Phonembewertungen werden in der DTW-Schicht in Wortmodellen zusammengefaßt und ergeben so eine Bewertung für jeden Buchstaben. Daraus kann ein Klassifikationsfehler auf Wortebene berechnet werden, der dann durch die gefundenen Pfade in das TDNN zurückpropagiert wird.
- Das Training auf **Satzebene** lernt über Buchstabengrenzen hinweg und versucht, die bei kontinuierlicher Spracherkennung auftretenden Einfügungs- und Auslassungsfehler zu minimieren.

Zu jedem dieser Schritte werden im folgenden verschiedene Trainingstechniken vorgeschlagen und anhand von vier Erkennungsproblemen experimentell evaluiert. Zwei dieser Sprachdatenbanken sind sprecherabhängig und enthalten Buchstabierungen von je einem Sprecher (mjmt und mdbs) aus der CMU–Alph-Datenbank. Ebenfalls englischsprachige Buchstabierungen kommen in der sprecherunabhängigen (insgesamt 120 Sprecher) RM–Spell-Datenbank zum Einsatz. Die umfangreichste, vierte Aufgabe stammt aus der Karlsruher Buchstabierdatenbank (KA–Alph), die sich aus beinahe 80 000 Buchstaben von etwa 100 deutschsprachigen Sprechern zusammensetzt. Die Datenbanken sind in jeweils eine Trainings-, Kreuzvalidierungs- und Testmenge eingeteilt, die in den nachfolgenden Tabellen mit „Train", „Cross" und „Test" abgekürzt werden. Mit der Trainingsmenge werden die Gewichte des Netzes eingelernt. Die Kreuzvalidierungsmenge dient dazu, einen geeigneten Abbruchzeitpunkt für das Training festzulegen oder auch, um weitere Systemeinstellungen zu validieren. Die eigentlichen Erkennungsergebnisse schließlich werden auf der Testmenge gemessen. Die vier Sprachdatenbanken und ihre Aufteilung in Trainings-, Kreuzvalidierungs- und Testdaten sind bereits in Kapitel 5 detailliert beschrieben und in Tabelle 6.2 noch einmal zusammengefaßt.

Alle Sprachaufnahmen sind auf dieselbe, in Abschnitt 3.2 erläuterte Weise zu Melscale-FFT-Koeffizienten vorverarbeitet, d.h. es liegen pro Sekunde 100 16-dimensionale Vektoren vor, deren Koeffizienten die Energie in verschiedenen Frequenzbändern repräsentieren. Die Eingabe wird über ihre gesamte Länge und alle Koeffizienten so normalisiert, daß der kleinste Koeffizient bei -1 und der größte bei +1 zu liegen kommt.

Daten	Train	Cross	Test
mjmt	2561	503	2049
mdbs	2548	514	1953
RM-Spell	8568	1161	923
KA-Alph	57870	6675	14898

Tabelle 6.2: Anzahl der Buchstaben in Trainings-, Kreuzvalidierungs- und Testmenge der vier Sprachdatenbanken

Für den ersten Schritt, das Training auf Phonemebene, müssen die Trainingsdaten phonetisch etikettiert sein, d.h. jeder Sprachvektor $\boldsymbol{x}_t$ eines Eingabemusters $\boldsymbol{X} = (\boldsymbol{x}_1 \dots \boldsymbol{x}_T)$ muß als Repräsentant eines bestimmten Phonems ausgewiesen sein. Die Phonem*sequenz* ist aus der manuell erstellten Transkription der Daten bekannt – ihre initiale Zuordnung zu einzelnen Sprachvektoren, also die Bestimmung der Phonemgrenzen, erfolgt durch menschliche Testhörer oder automatisch durch bereits trainierte Erkenner. Für das Training auf Wortebene ist es ausreichend, die Grenzen zwischen den einzelnen Buchstaben zu kennen, und auf Satzebene benötigt man lediglich die Transkriptionen der Aufnahmen.

6.4 Training auf Phonemebene

Für das Training auf Phonem- und Wortebene betrachten wir einzelne, aus der gesprochenen Buchstabensequenz gemäß der Trainingsetikettierung ausgeschnittene Buchstaben[7].

6.4.1 Zielfunktion

Ein Buchstabe ist durch T Sprachvektoren $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ repräsentiert, und mit $c_1, \dots, c_T$ wird jedem $\boldsymbol{x}_t$ eine Phonemklasse $c_t \in S$ zugeordnet. Zu erlernen ist die ideale Entscheidungsfunktion, also eine 1-aus-N-Repräsentation der Phoneme: Der Sollwert $z_{t,i}$ für den i-ten Ausgabeknoten $y^{\mathrm{p}}_{t,i}$ zum Zeitpunkt t ist 1, wenn es sich um Phonem $c_t = s_i$ handelt, und 0 sonst, d.h. $z_{t,i} = \delta_{i,c_t}$. Aufgrund des zeitlichen Kontextes von 3 und 5 Vektoren wird ein Vektor der Phonemschicht aus insgesamt 7 Vektoren der Eingabeschicht gespeist. Um dennoch jeden Ausgabevektor $\mathbf{y}^{\mathrm{p}}_1, \dots, \mathbf{y}^{\mathrm{p}}_T$ berechnen zu können, wird die Eingabe um den entsprechenden linken und rechten Kontext[8] (also jeweils 3 Vektoren) erweitert. An Phonemgrenzen kommt im Eingabefenster mehr als ein Phonem

[7] Für das Training auf Phonemebene könnte man genausogut die ganze Buchstabensequenz als Eingabe betrachten. Wir beschränken uns auf Buchstaben, um konsistent mit dem nächsten Trainingsschritt zu bleiben.

[8] Bei den mangels Kontext nicht erweiterbaren Phonemen an den Rändern der Aufnahme handelt es sich praktisch immer um Stille.

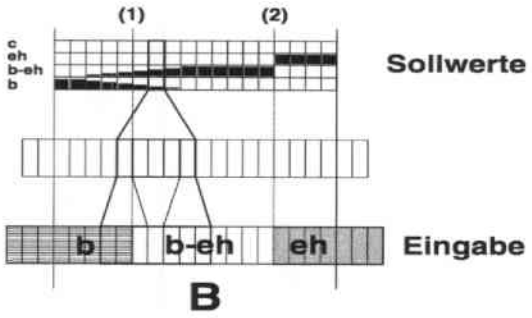


Abbildung 6.6: Sollwerte mit fließenden (1) und harten (2) Übergängen an Phonemgrenzen

zu liegen, daher bietet sich statt einer „harten" Entscheidung $z_{t,i} = \delta_{i,c_t}$ ein fließender Übergang an, der die Sollwerte anteilig zu den im Eingabefenster vertretenen Phonemen verteilt. Diese in Abbildung 6.6 illustrierte Art der Repräsentation ist in den folgenden Experimenten gewählt.

6.4.2 Fehlerfunktionen

Das MS-TDNN-Training auf Phonemebene entspricht im wesentlichen dem Training eines TDNN, lediglich die Aufsummierung der Phonemaktivitäten in eine einzige Ausgabe pro Phonem entfällt. Statt dessen ist das Ziel, für jeden einzelnen Zeitpunkt t die bestmöglichen Phonembewertungen $\mathbf{y}_t^r$ zu erlernen.

Die klassische Fehlerfunktion für das Fehlerrückführungsverfahren (Error Backpropagation) ist der mittlere quadratische Fehler (MSE). Dieser hat sich jedoch für das MS-TDNN-Training als ungünstig erwiesen, denn bei einer 1-aus-N-Repräsentation in einem hochdimensionalen Ausgaberaum (z.B. N = 59 Phoneme) kann ein steigender Fehlerbeitrag eines einzelnen Ausgabeknotens y_i leicht durch kleine Fehlerreduzierungen vieler anderer Knoten kompensiert werden. Eine im Sinne der zu lernenden Aufgabe günstige Ausgabe von 0.8 am korrekten und 0.2 an allen anderen Knoten wird mit einem Fehler von $N * 0.2^2 = 2.36$ bestraft, während die „bedeutungslose" Ausgabe von 0.1 an allen Knoten einen Gesamtfehler von nur $0.9^2 + (N-1) * 0.1^2 = 1.39$ erhält!

Günstiger sind daher Fehlerfunktionen wie „Cross Entropy (CE)" oder der „McClelland Error (MCL)", deren logarithmische Terme „Ausreißer" mit einem höheren Fehler bestrafen, der im Grenzfall $|z_i - y_i| \to 1$ sogar unendlich groß wird.

Neben unterschiedlichen Fehlerfunktionen wird in den unten beschriebenen Experimenten auch der Einfluß der Transferfunktion untersucht. Statt der üblichen Sigmoidfunktion als Transferfunktion der Ausgabeschicht kann die Softmaxfunktion genutzt werden,

die die Bedingung $\sum_i y_{t,i}^{\mathrm{r}} = 1$ erfüllt und damit automatisch eine Interpretation der Ausgaben als Wahrscheinlichkeiten erlaubt. Außerdem wird die Diskriminierung der Ausgaben unterstützt, denn die Erhöhung einer Ausgabe erzwingt automatisch eine Reduzierung der restlichen Ausgaben.

Daten	Beste Kombination Fehler-/ Transferfunkt.	Phoneme (Vektoren) Train	Phoneme (Vektoren) Test	Buchstaben konnektionist. Train	Buchstaben konnektionist. Test	Buchstaben NN-HMM-Hybride Train	Buchstaben NN-HMM-Hybride Test
mjmt	MCL,Sigmoid	85.7	83.0	99.5	99.2	99.6	99.6
mdbs	MCL,Sigmoid	83.2	80.6	98.8	97.1	98.7	97.4
KA–Alph	MCL,Softmax	78.1	76.6	94.7	89.9	95.7	91.9
RM–Spell	MCL,Softmax	78.1	76.6	95.3	89.4	97.0	92.4

Tabelle 6.3: Phonem-Erkennungsrate einzelner Sprachvektoren und Buchstabenerkennungsraten, in % korrekt.

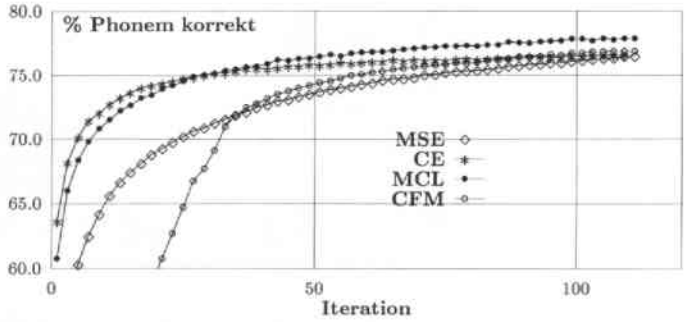


Abbildung 6.7: Konvergenz auf der RM–Spell-Trainingsmenge für verschiedene Fehlerfunktionen

6.4.3 Phonemerkennung

In einer Serie von Experimenten wurden die in Abschnitt 2.5.4 eingeführten MSE-, CE-, MCL- und CFM-Fehlerfunktionen jeweils mit einer Sigmoid- und Softmax-Transferfunktion auf den vier Aufgaben mjmt, mbds, RM–Spell und KA–Alph verglichen. Gemessen wird die Phonemerkennungsrate als der Prozentsatz korrekt erkannter Sprachvektoren[9] $\mathbf{y}_t^{\mathrm{r}}$. Als erkanntes Phonem i zählt das Neuron mit der höchsten Ausgabe $y_{t,i}^{\mathrm{r}}$. Die Ergebnisse der jeweils besten Kombination aus Fehler- und Transferfunktion

[9]Es geht also nicht darum, ein ganzes Sprachsegment als ein bestimmtes Phonem zu erkennen, vielmehr wird jeder Vektor einzeln bewertet.

sind in Tabelle 6.3 (links) zusammengefaßt, die vollständigen Ergebnisse sind im Anhang in Tabelle B.1 aufgelistet. Der McClelland-Fehler erzielt konsistent über alle 4 Aufgaben die besten Ergebnisse. Auch CE schneidet deutlich besser ab (außer bei KA–Alph) als MSE oder CFM. Aus den Trainingskurven in Abbildung 6.7 wird ersichtlich, daß MCL und CE zudem wesentlich schneller konvergieren als MSE und CFM.

6.4.4 Worterkennung: Konnektionistische Phonemintegration versus hybrides NN-HMM

Statt einzelner Sprachvektoren wollen wir nun das gesamte an der Eingabe anliegende Sprachsegment als Buchstabe klassifizieren, indem die Phonembewertungen zur Wortschicht $\mathbf{y}^{\mathrm{W}}$ weiterpropagiert werden. In Abschnitt 6.2.2 wurden zwei Möglichkeiten beschrieben, um aus den Phonembewertungen Wortbewertungen zu berechnen. Bei der direkten konnektionistischen Phonemintegration werden für jeden zu erkennenden Buchstaben L_i die entsprechenden Phonembewertungen über die Zeit entlang eines optimalen Pfades $\boldsymbol{\pi}_i = \pi_i(1) \ldots \pi_i(F)$ aufsummiert, wie bereits in (6.15) dargestellt und hier wiederholt:

$$y_i^{\mathrm{W}} = \frac{1}{F} \sum_{t=1}^{F} y^{\mathrm{P}}_{t,\pi_i(t)}$$

Für den hybriden NN-HMM-Ansatz wird der in (6.17) zu maximierende Ausdruck durch Logarithmieren zu

$$\log p^*(\mathbf{X}|\lambda_i) = \sum_{t=1}^{F} \log p(\vec{\boldsymbol{x}}_t|q_t = s_{\pi_i(t)}) \tag{6.18}$$

Wie in (6.15) ist $\boldsymbol{\pi}_i$ der für jedes Modell L_i zu bestimmende optimale Pfad. Die Emissionswahrscheinlichkeit kann nach Bayes aus den als a posteriori Wahrscheinlichkeiten vorliegenden Phonembewertungen $y^{\mathrm{P}}_{t,i} = p(q_t = s_i|\vec{\boldsymbol{x}}_t)$ bestimmt werden:

$$p(\vec{\boldsymbol{x}}_t|q_t = s_i) = \frac{P(q_t = s_i|\vec{\boldsymbol{x}}_t) \cdot p(\vec{\boldsymbol{x}}_t)}{P(q_t = s_i)} \tag{6.19}$$

Die a priori Wahrscheinlichkeiten $P(q_t = s_i) =: P(s_i)$ werden als vom Zeitpunkt t unabhängig betrachtet und können durch ihre relative Häufigkeit in den Trainingsdaten bestimmt werden. Die Observationswahrscheinlichkeit für die gesamte Sequenz ergibt sich zu

$$\log p(\mathbf{X}|\lambda_i) = \sum \log \frac{P(q_t = s_{\pi_i(t)}|\vec{\boldsymbol{x}}_t) \cdot p(\vec{\boldsymbol{x}}_t)}{P(s_{\pi_i(t)})} \tag{6.20}$$

$p(\vec{x}_t)$ ist nur von der Eingabe und nicht von den konkurrierenden Wortmodellen abhängig und kann daher wegfallen. Mit den in (6.18) unberücksichtigten a priori Wahrscheinlichkeiten $P(\lambda_i)$ der Wörter L_i erhalten wir als Entscheidungsfunktion

$$g_i(\mathbf{X}) = \sum \log \frac{P(q_t = s_{\pi_i(t)}|\vec{x}_t)}{P(s_{\pi_i(t)})} + \alpha \log P(\lambda_i) \tag{6.21}$$

$P(\lambda_i)$ ist ein Monogramm-Sprachmodell, also die a priori Wahrscheinlichkeit eines Buchstabens L_i, die mit α gewichtet wird. Für den Parameter α wird derjenige Wert gewählt, der auf der Kreuzvalidierungsmenge die beste Erkennung erzielt. Der Einfluß von α ist in Abbildung 6.8 am Beispiel der Kombination Sigmoidfunktion/MCL-Fehler ersichtlich. Das Optimum liegt nicht bei $\alpha = 1$, denn bei der Wortausgabe handelt es sich nicht um die Gesamtwahrscheinlichkeit $p(\mathbf{X}|\lambda_i)$, da erstens der konstante Faktor $p(\mathbf{X})$ fehlt und zweitens nur die Wahrscheinlichkeit für den Viterbi-Pfad berechnet wird.

In der zweiten und dritten Doppelspalte von Tabelle 6.3 sind die Erkennungsergebnisse des konnektionistischen und des hybriden TDNN-HMM-Systems auf allen vier Datenbanken aufgelistet, jeweils für die im Durchschnitt am besten abschneidende Kombination aus Fehler- und Transferfunktion. Detaillierte Ergebnisse für alle Kombinationen von Fehler- und Transferfunktion finden sich im Anhang in Tabelle B.2 und B.3. Das hybride System erzielt konsistent die besseren Erkennungsraten, wird aber durch das im nächsten Abschnitt beschriebene Training auf Wortebene wieder vom MS-TDNN überholt.

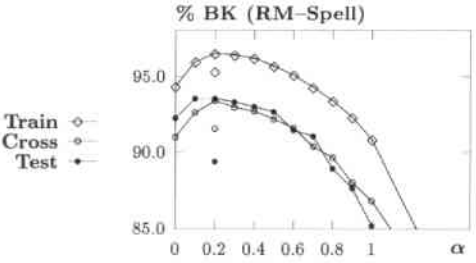


Abbildung 6.8: % Buchstaben korrekt in Abhängigkeit des Sprachmodellgewichts α für den hybriden NN-HMM-Ansatz. Die Ergebnisse der konnektionistischen Phonemintegration (unabhängig von α) sind als einzelne Punkte markiert.

6.5 Training auf Wortebene

Unser Ziel ist die Erkennung von Buchstaben oder fließend gesprochenen Buchstabensequenzen. In der oben beschriebenen, ersten Trainingsstufe des MS-TDNN wurde jedoch

ein anderes Kriterium, nämlich die Phonemerkennung, optimiert. Das Training auf Wortebene geht einen Schritt weiter, indem sich die Optimierung der Parameter direkt an der Worterkennung (in unserem Fall Buchstaben) orientiert. Dadurch wird das Trainings- und Testkriterium bei der Erkennung einzelner Buchstaben konsistent, was sich in deutlich verbesserten Worterkennungsraten bemerkbar macht.
Wie zuvor besteht die Eingabe aus einzelnen, ausgeschnittenen Buchstaben[10], zu denen nun direkt die Wortausgaben berechnet und mit entsprechenden Sollwerten verglichen werden. Dazu sind in den Trainingsdaten nur noch die Buchstabengrenzen notwendig. Die phonetischen Etikettierungen werden nun automatisch vom System bestimmt, denn durch die optimalen Suchpfade $\pi_i(t)$ (siehe Gl. (6.15)) wird jedem Eingabevektor $\boldsymbol{x}_t$ ein Phonem $s_{\pi_i(t)}$ zugeordnet.

Auch auf Wortebene ist die Wahl einer geeigneten Fehlerfunktion sehr wichtig. Außerdem hat sich ein „vorsichtiges" Training bewährt, bei dem sicher erkannte Muster nicht nachtrainiert werden.

6.5.1 Konnektionistische Phonemintegration

Die konnektionistische Phonemintegration erlaubt eine direkte Anwendung des Fehlerrückführungsverfahrens auf Wortebene. Statt in der Phonemschicht wird der Fehler nun direkt in der Wortschicht gemessen, und von dort durch die im vorausgegangenen Testschritt gefundenen, optimalen Pfade durch das gesamte Netz zurückgeführt.

Oberhalb der Phonemschicht gibt es in der MS-TDNN-Architektur keine lernbaren Parameter (Gewichte). Die zusätzliche Aufgabe des Trainings auf Wortebene ist, den Betrag des Fehlersignals abhängig vom erkannten Wort zu bestimmen und ihn mit Hilfe der gefundenen Pfade auf die richtigen Phoneme zu verteilen. Ist das Fehlersignal erst einmal bis auf die Phonemebene zurückpropagiert, ist also $\partial E/\partial y^{\mathrm{P}}_{t,i}$ bekannt, so kann $\partial E/\partial \mathbf{W}$ analog zum Training des TDNN (wie in Anhang A.2 beschrieben) für alle Gewichte des Systems berechnet werden. Mit (6.15) berechnet sich $\partial E/\partial y^{\mathrm{P}}_{t,j}$ zu:

$$\frac{\partial E}{\partial y^{\mathrm{P}}_{t,j}} = \frac{\partial E}{\partial \mathbf{y}^{\mathrm{W}}} \cdot \frac{\partial \mathbf{y}^{\mathrm{W}}}{\partial y^{\mathrm{P}}_{t,j}} = \sum_{i=1}^{n_{\mathrm{W}}} \frac{\partial E}{\partial y^{\mathrm{W}}_i} \cdot \frac{\partial y^{\mathrm{W}}_i}{\partial y^{\mathrm{P}}_{t,j}} \overset{(6.23)}{=} \frac{1}{F} \sum_{i=1}^{n_{\mathrm{W}}} \frac{\partial E}{\partial y^{\mathrm{W}}_i} \cdot \delta_{j,\pi_i(t)} \tag{6.22}$$

$$\frac{\partial y^{\mathrm{W}}_i}{\partial y^{\mathrm{P}}_{t,j}} = \frac{\partial (\frac{1}{F} \sum_{t'=1}^{F} y^{\mathrm{P}}_{t',\pi_i(t')})}{\partial y^{\mathrm{P}}_{t,j}} = \frac{1}{F} \delta_{j,\pi_i(t)} \tag{6.23}$$

[10] Es sei an dieser Stelle angemerkt, daß dies zwar ähnlich, aber nicht vergleichbar mit der Einzelbuchstabenerkennung ist. In letzterer wird jeder Buchstabe für sich isoliert gesprochen und ist von Stille umgeben. In unserem Fall werden die Buchstaben einzeln aus einer kontinuierlichen Buchstabierung ausgeschnitten. Da unmittelbar vorher und nachher andere Buchstaben gesprochen wurden, muß mit Koartikulationseffekten gerechnet werden. Auch ist eine saubere und konsistente Trennung schnell nacheinander gesprochener Buchstaben nicht immer möglich.

Dies besagt gerade, daß der an jedem Wortknoten y_i^W entstehende Fehler über die Länge der Eingabe normiert und entlang des gefundenen wortspezifischen Pfades $\pi_i(1) \ldots \pi_i(F)$ rückpropagiert wird. Der Term $\delta_{j,\pi_i(t)}$ sorgt dafür, daß ein Phonemknoten $y_{t,j}^P$ genau von all den Wörtern L_i ein Fehlersignal erhält, deren Pfade $\pi_i(t)$ zum Zeitpunkt t das entsprechende Phonem s_j durchlaufen.

Prinzipiell ist es auch möglich, mit dem Training direkt auf Wortebene zu beginnen. Da aber am Anfang noch keinerlei Phoneme unterschieden werden können, sind auch die optimalen Pfade und damit die Phonemsegmentierungen zufällig. Auch unter diesen ungünstigeren Umständen konvergiert das Training auf Wortebene zu einem stabilen System, wenn auch mit geringfügig schlechteren Erkennungsraten. Auf RM–Spell wurde ein Verlust von etwa einem, auf KA–Alph von etwa zwei Prozentpunkten beobachtet.

Fehlerfunktionen

Die Wortausgaben y_i^W werden als normierte Summe der Phonemaktivitäten entlang des optimalen Pfades berechnet (siehe Gl. (6.15)). Auf eine Nichtlinearität wurde bewußt verzichtet, um konsistent mit der Satzerkennung zu bleiben, wo ebenfalls direkt die Phonembewertungen verrechnet werden. Eine Transferfunktion auf Wortebene entfällt somit, in der Phonemschicht ist jedoch nach wie vor die Sigmoid- oder Softmaxfunktion wählbar.

Transfer-/ Fehlerfunktion		Training aller Muster		Training nur falsche Muster	
		Train	Test	Train	Test
softmax /	mse	93.5	86.4	97.9	92.3
	ce	92.4	83.8	97.8	90.6
	mcl	95.5	87.8	98.7	92.7
	cfm	97.4	89.7	99.2	94.0
sigmoid /	mse	93.9	87.1	97.9	93.7
	ce	93.4	85.9	98.2	92.4
	mcl	95.7	89.2	98.5	94.4
	cfm	98.0	93.7	99.4	95.4

Tabelle 6.4: % Buchstabenakkuratheit (% BA) nach Training mit verschiedenen Fehler- und Transferfunktionen (RM–Spell)

Die Ergebnisse für verschiedene Fehlerfunktionen (auf Wortebene) und Transferfunktionen (auf Phonemebene) nach einem Training mit 100 Iterationen sind in Tabelle 6.4 aufgelistet. Die Ergebnisse in den linken Spalten wurden erzielt, indem auf konventionelle Art und Weise in jeder Iteration alle Muster (in jeder Iteration in einer neuen zufälligen Reihenfolge) trainiert wurden. Ursprünglich motiviert durch die Erfordernis, die Trainingszeiten zu verringern, wurde das Training so abgeändert, daß nur „nicht sicher erkannte" Muster trainiert werden. Interessanterweise ist dieses Trainingsschema

nicht nur schneller, sondern erreicht auch deutlich bessere Erkennungsraten, wie in den beiden rechten Spalten in Tabelle 6.4 zu sehen ist. Ein Muster gilt als sicher erkannt, wenn die korrekte Ausgabe y_k um einen „Sicherheitsabstand" δ besser als die höchste inkorrekte Ausgabe y_{hi} ist, also $y_k > y_{hi} + \delta$.

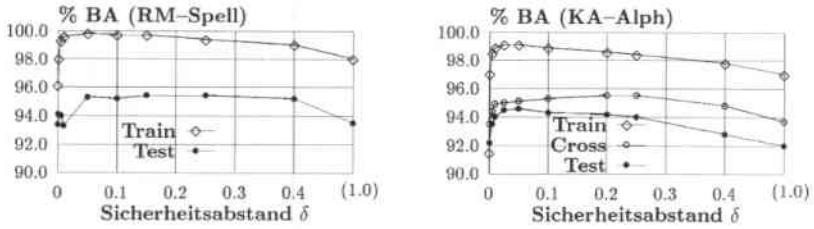


Abbildung 6.9: Trainings- und Testresultate in Abhängigkeit vom „Sicherheitsabstand" δ

Abbildung 6.9 zeigt den Zusammenhang zwischen dem Sicherheitsabstand δ und der Erkennungsrate. Wird δ zu groß gewählt, dann werden zu viele Muster neu trainiert und die Erkennungsleistung sinkt. Im Extremfall $\delta = 1$ werden bedingungslos alle Muster trainiert, wie im Vergleich in den linken Spalten von Tabelle 6.4. Andererseits ist ein gewisser Sicherheitsabstand notwendig, denn für $\delta \to 0$ büßt man ebenfalls Erkennungsleistung ein. Wählt man δ im relativ breiten Bereich zwischen 0.05 und 0.3, gewinnt man absolut etwa 2% BA.

Die CFM-Fehlerfunktion

In allen Trainingsexperimenten auf Wortebene konnte die CFM-Fehlerfunktion gegenüber den MSE-, MCL- und CE-Fehlerfunktionen deutlich bessere Ergebnisse erzielen. Dies kann damit erklärt werden, daß die zu lernenden Klassen, also die einzelnen Buchstaben, nicht disjunkt sind, denn an der Berechnung der Ausgaben y_i^W sind jeweils mehrere Phoneme beteiligt, wobei sich manche Buchstaben gleiche Phoneme teilen, beispielsweise das /eh/ in B,C,D. Es ist daher nicht sinnvoll, etwa für den korrekten Buchstaben B eine Ausgabe von 1 und gleichzeitig für seine Konkurrenten D oder G eine Ausgabe von 0 zu fordern, denn das Phonem /eh/ leistet einen Beitrag zu allen drei Ausgaben. Die CFM-Fehlerfunktion orientiert sich nicht an „idealen" Sollwerten 0 oder 1, sondern lediglich an einem relativen Vergleich zwischen den korrekten und den inkorrekten Ausgaben. Dazu wird ein Abstandsmaß d zwischen der korrekten und den inkorrekten Ausgaben bestimmt. Die CFM-Formulierung von Hampshire [HIW90c] gewichtet zur Berechnung von d alle inkorrekten Ausgaben gleich (Gl. (2.21)). Ähnlich wie bei der Auswahl der Trainingsmuster hat sich auch hier eine „vorsichtigere" Vorgehensweise bewährt, bei der nicht jede, sondern nur die höchste inkorrekte Ausgabe y_{hi} in die Berechnung von d eingeht und damit vom Training betroffen ist.

Diese im folgenden als CFM-hi bezeichnete modifizierte Fehlerfunktion ist in Abbildung 6.10 illustriert. Auf einer Skala ist die zwischen 0 und 1 liegende Bewertung von jedem zu klassifizierenden Buchstaben aufgetragen. Das Mißklassifikationsmaß d berechnet sich einfach als Differenz der Ausgabe der korrekten Klasse k und der Klasse $hi = \text{argmax}_{j \neq k}\, y_j$ mit der höchsten „inkorrekten" Ausgabe:

$$d = y_k - y_{hi} \tag{6.24}$$

Das zu maximierende Kriterium ist analog zu (2.22)

$$E_{CFM} = \frac{1}{1 + e^{-\beta \cdot (-d) + \gamma}} \tag{6.25}$$

Die Ableitung der Sigmoidfunktion geht gegen Null für sehr kleine, aber auch für sehr große Werte von d. Um für große Abweichungen d auch große Ableitungen zu erhalten, wurde in [HW93c]

$$E_{CFM} = (1 - d)^2 \tag{6.26}$$

als Fehlermaß vorgeschlagen. Damit konnten anfänglich bessere Ergebnisse erzielt werden, was jedoch durch eine suboptimale Wahl der Parameter in (6.25) bedingt war. Mit $\beta = 5$ und $\gamma = 0$ ergeben sich für beide Varianten (6.25) und (6.26) vergleichbare Erkennungsergebnisse.

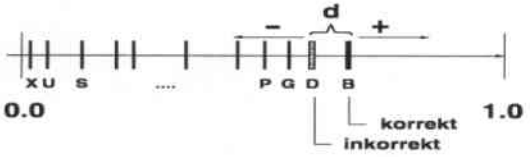


Abbildung 6.10: CFM-Fehlerfunktion: Nur der Abstand d zwischen der korrekten (B) und der höchsten inkorrekten (D) Ausgabe wird im Training berücksichtigt

Bei CFM erhält der korrekte Buchstabe ein positives Fehlersignal, und ein negatives Fehlersignal von gleichem Betrag verteilt sich auf alle inkorrekten Beispiele. Mit CFM-hi konzentriert sich das negative Training auf die höchste inkorrekte Ausgabe. Statt aller inkorrekten Buchstaben wird also nur der schärfste Konkurrent auf der Bewertungsskala nach unten verschoben. Die Auswirkungen dieser Vorgehensweise auf das Training sind anhand eines Beispiels in Abbildung 6.11 dargestellt. Der Buchstabe B ist korrekt und erhält daher positives Training. Der Konkurrent mit der höchsten Ausgabe ist D, das betragsmäßig dieselbe Quantität an negativem Training erhält. Im gemeinsamen /eh/-Teil der beiden Buchstaben heben sich positives und negatives Training gerade auf, während in dem zur Unterscheidung kritischen Anfang diskriminiert wird.

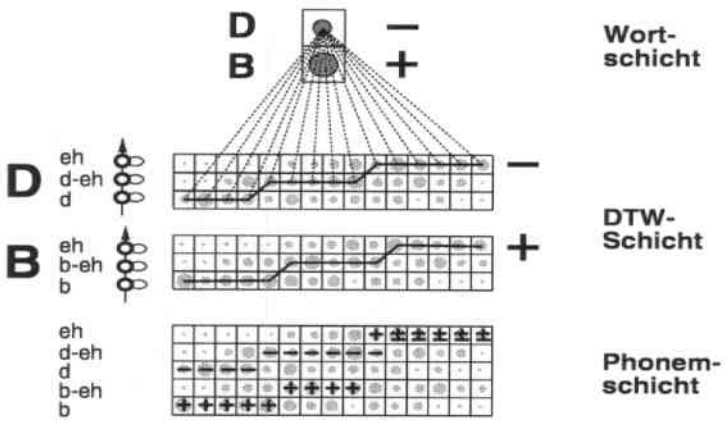


Abbildung 6.11: CFM-Training. Im gemeinsamen Teil /eh/ neutralisieren sich positives und negatives Training, nur die unterschiedlichen Anfänge werden diskriminativ trainiert

In der GPD-Formulierung (Gl. (2.26)) kann die Gewichtung der inkorrekten Ausgaben durch den Parameter η beeinflußt werden. Mit wachsendem η gewinnen die höheren inkorrekten Ausgaben an Gewicht, im Grenzfall $\eta \to \infty$ entspricht GPD gerade CFM-hi. In Tabelle 6.5 sind einige Experimente mit CFM-Fehlerfunktionen zusammengetragen. Mit steigender Konzentration ($\eta = 1, 5, 10, 20$) der GPD-Funktion auf die höchste inkorrekte Ausgabe verbessert sich die Erkennungsleistung und wird für $\eta = 20$ von CFM-hi nicht mehr unterscheidbar.

	RM–Spell		KA–Alph	
Fehlerfunktion	Train	Test	Train	Test
original CFM	97.3	94.4	96.5	92.2
GPD (η=1)	96.3	93.8	94.7	90.3
GPD (η=5)	98.3	95.8	97.7	93.2
GPD (η=10)	99.1	95.5	98.6	94.2
GPD (η=20)	99.3	95.4	98.8	94.1
CFM-hi	99.2	95.3	98.4	94.0

Tabelle 6.5: Verschiedene Mißklassifikationsmaße, in % Buchstaben korrekt

6.5.2 Hybrides NN-HMM-System

Im hybriden NN-HMM-Ansatz werden in den Wortmodellen wie in einem HMM die klassenbedingten Wahrscheinlichkeiten berechnet, die Ausgabe der Wortknoten ist also

die Größe[11] $p^*(\mathbf{X}|\lambda_i)$. Für eine gegebene Eingabe $\mathbf{X}$ soll $p^*(\mathbf{X}|\lambda_i)$ maximiert werden – das bedeutet jedoch nicht notwendigerweise, daß die anderen Wahrscheinlichkeiten $p^*(\mathbf{X}|\lambda_j), j \neq i$ kleiner werden müssen.

Ein diskriminatives Training auf Wortebene ist daher nur bedingt sinnvoll und würde beispielsweise einen MMI-Ansatz (siehe Abschnitt 2.5.4) erfordern. Es ist allerdings möglich, die durch die Berechnung der Viterbi-Pfade gefundenen Phonemgrenzen als neue phonetische Etikettierung einzusetzen. Wenn diese im Lauf des Trainings genauer wird, dann kann auch eine Verbesserung im diskriminativen Training der Phoneme erwartet werden. In den hybriden NN-HMM-Systemen am ICSI und an der Cambridge University wurde diese Vorgehensweise erfolgreich eingesetzt.

Im Rahmen dieser Arbeit wurde ebenfalls mit mehreren Varianten dieses Trainingsschemas experimentiert. Dabei konnte jedoch keine signifikante Verbesserung erzielt werden. Da umgekehrt das konnektionistische Training auf Wortebene deutliche Verbesserungen erzielte, wurde der hybride NN-HMM-Trainingsansatz auf Wortebene nicht weiter verfolgt.

6.6 Training auf Satzebene

Sprache wird hierarchisch modelliert: Sätze bestehen aus Wörtern und Wörter aus Phonemen. In diesem Sinne sind auch Buchstaben einfach nur kurze Wörter, und wir bezeichnen Buchstabensequenzen als (Buchstaben-)Sätze. Bei der Erkennung von Sätzen treten neben Wortverwechslungen weitere Fehler durch irrtümliche Einfügungen und Auslassungen von Wörtern hinzu. Die kurzen und ähnlich klingenden Buchstaben sind für diese Art von Fehlern besonders anfällig. So kann etwa statt „T E“ leicht nur „T“, oder „0 0“ statt „0“ erkannt werden.

Zur Einzelworterkennung wurde zu jedem Buchstabenmodell ein optimaler Pfad bestimmt und unter allen der beste ausgewählt. Die Erweiterung des MS-TDNN zur Satzerkennung geschieht analog zu der in Abschnitt 3.5 beschriebenen, kontinuierlichen Erkennung: Um die Wortsequenz mit der besten Bewertung zu finden, bestimmt man einen optimalen Pfad durch ein verkettetes Buchstabenmodell, bei dem durch eine rückgekoppelte Parallelschaltung der Einzelmodelle beliebige Sequenzen durchlaufen werden können, wie in Abbildung 3.5 illustriert.

Dehnt man die Erkennung nach diesem Schema auf ganze Sätze aus, wird man zunächst durch hohe, hauptsächlich auf irrtümliche Einfügungen zurückzuführende Fehlerraten überrascht: Die Buchstabenakkuratheit[12] auf ganzen Sätzen lag zunächst unter 10%.

[11] Wie zuvor bezeichnet $p^*(.)$, daß es sich nicht um die volle, sondern nur um die Wahrscheinlichkeit des wahrscheinlichsten Pfades handelt.

[12] Siehe Abschnitt 3.5.2 zur Bestimmung der Erkennungsraten für kontinuierliche Sprache, in die neben Wortverwechslungen auch Worteinfügungen und -auslassungen eingehen.

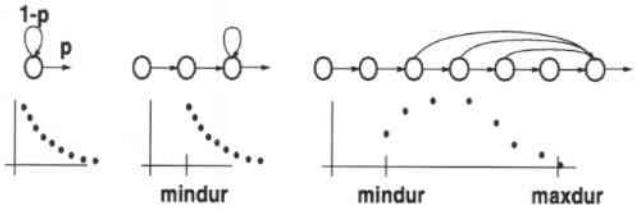


Abbildung 6.12: Zeitdauermodellierung für Phoneme

Die kurzen Buchstabenmodelle, insbesondere die der Vokale, können in der Zeitspanne des tatsächlich gesprochenen Buchstabens ohne weiteres mehrfach durchlaufen werden. Mit einer Zeitdauermodellierung kann eine angemessene Verweildauer innerhalb der Buchstaben erzwungen werden. Die Zeitdauermodellierung kann auf Phonem- und Wortebene stattfinden, und sie kann manuell eingestellt oder automatisch erlernt werden. Sind Einfügungen und Auslassungen einigermaßen ausbalanciert, können die akustischen Modelle auf Satzebene weitertrainiert werden.

6.6.1 Phonemdauer

Die zahlreichen Einfügungsfehler entstehen, weil die kurzen Buchstabenmodelle zu schnell und damit zu oft durchlaufen werden. Um ein längeres Verweilen in einem Phonem zu erzwingen, kann man genau wie in den Markov-Modellen Übergangswahrscheinlichkeiten einführen, so daß der Wechsel in den nächsten Zustand mit einer kleinen Wahrscheinlichkeit p bestraft wird. Allerdings sind die daraus resultierenden Wahrscheinlichkeiten $P(Länge = t) = (1-p)^{t-1}p$ (in Abbildung 6.12(links) skizziert) kein gutes Modell für die tatsächliche Längenverteilung. Mit aufwendigeren Modellen (Abbildung 6.12(Mitte und rechts)) können beliebige Längenverteilungen approximiert werden. Diese erfordern jedoch mehr Speicher, mehr Rechenzeit und stellen uns vor das Problem, viele zusätzliche Parameter schätzen zu müssen.

Phonemübergangsstrafen. In einem HMM werden entlang eines Pfades Observations- und Übergangswahrscheinlichkeiten multipliziert, bzw. im logarithmischen Bereich addiert. Im direkt diskriminativ trainierten MS-TDNN stellen die Phonembewertungen $y^p_{t,i}$ keine Observationswahrscheinlichkeiten dar, sondern können als Evidenz[13] für die auf einem Pfad liegenden Phoneme interpretiert werden. Ein Übergang in ein neues Phonem wird nun erschwert, indem bei jedem Wechsel in ein neues Phonem eine Strafe,

[13] Nach dem Training auf Phonemebene können die Phonemausgaben noch als a posteriori Wahrscheinlichkeiten interpretiert werden, nicht mehr jedoch nach dem Training auf Wortebene, denn die CFM-Fehlerfunktion orientiert sich nur an der Unterscheidbarkeit der einzelnen Klassen, nicht aber an der Höhe der Ausgaben.

also ein negativer Wert, zu den bis dahin akkumulierten Bewertungen addiert wird. In Abbildung 6.14(oben) ist die experimentell ermittelte Abhängigkeit der Erkennungsrate von Phonemübergangsstrafen aufgetragen. Die Strafen variieren zwischen 0 und 3, und müssen in Relation zu den Phonembewertungen betrachtet werden, die immer zwischen 0 und 1 liegen. Auf beiden Aufgaben (RM–Spell und KA–Alph) ist die Performanz auf Trainings-, Kreuzvalidierungs- und Testmenge stark korreliert, so daß ein angemessener Wert für die Übergangsstrafe zuverlässig mit einer Kreuzvalidierungsmenge ermittelt werden kann.

Erzwungene minimale Dauer. Indem in einem Wortmodell n Kopien desselben Phonems in Serie geschaltet werden (wie in Abbildung 6.12(Mitte)), kann eine minimale Verweildauer *mindur* von n Zuständen erzwungen werden. Da ein einzelner Zeitschritt nur 0.01 Sekunden Sprache entspricht, ist es durchaus sinnvoll, eine Mindestdauer von mehreren Zeitschritten zu fordern. In einem zweiten Experiment wurde die Mindestdauer, also die Anzahl der duplizierten Zustände, zwischen 1 und 10 variiert. Die Ergebnisse sind in Abbildung 6.14(Mitte) zu sehen. Mit der Kreuzvalidierungsmenge kann eine optimale Mindestdauer von 5 Zuständen für KA–Alph und von 7 Zuständen für RM–Spell ermittelt werden.

Da die einzelnen Phoneme ganz unterschiedliche Längenverteilungen haben, ist es sinnvoll, statt einer globalen eine **phonemspezifische Mindestdauer** $mindur_i$ zu fordern, die leicht aus entsprechenden Statistiken über die Längen der Phoneme s_i gewonnen werden kann: Es sei h_i ein Histogramm und $h_i(l)$ die Anzahl der in der Trainingsmenge beobachteten Phoneme s_i der Länge l. Als minimale Verweildauer $mindur_i$ kann beispielsweise die kürzeste beobachtete Länge $min_i := \text{argmin}_l\{h_i(l) \neq 0\}$ bestimmt werden. Um auch die phonemspezifische Mindestdauer sinnvoll variieren zu können, definieren wir $mindur_i(x)$ als diejenige Länge l, für die $x \cdot 100\%$ aller Phoneme s_i kürzer sind als l. Für $-1 \leq x \leq 0$ wird der Bereich $0 \ldots min_i$ abgedeckt:

$$mindur_i(x) = \begin{cases} \text{argmin}_l\{\sum_{l'=1}^{l} h_i(l')/N_i > x\} & 0 \leq x \leq 1 \\ (1 - |x|) * min_i & -1 \leq x < 0 \end{cases}$$

wobei $N_i = \sum_{l=1}^{\infty} h_i(l)$. Es ist also $mindur_i(-1) = 0, mindur_i(0) = min_i$ und $mindur_i(1) = max_i$. In Abbildung 6.13 sind zwei Histogramme für ein kurzes und ein langes Phonem sowie ein Beispiel für die Bestimmung von $mindur_i(0.1)$ abgebildet. Die Ergebnisse der entsprechenden Experimente können in Abbildung 6.14(unten) für Werte von $-0.6 \leq x \leq 0.6$ abgelesen werden. Ein bestimmter x-Wert entspricht einer mittleren Mindestdauer $\overline{m} := 1/n_p \sum_i^{n_p} mindur_i(x)$, die parallel zu den x-Werten aufgetragen ist.

Die Ergebnisse obiger Versuche sind in Tabelle 6.6 zusammengefaßt. Die Erkennungsraten (in % Buchstabenakkuratheit) beziehen sich immer auf diejenige Einstellung, mit der auf der Kreuzvalidierungsmenge die besten Ergebnisse erzielt werden konnten. Ganz offensichtlich erhält man ohne Längenmodellierung katastrophale Ergebnisse. Die Ergebnisse zu Experimenten mit Dauermodellierungen aus den vorangehenden drei Abschnitten (Phonemübergangsstrafen sowie globalen oder phonemspezifischen Mindestlängen)

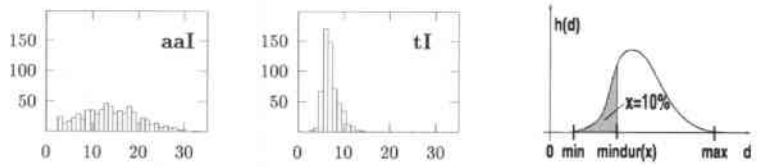


Abbildung 6.13: Links, Mitte: Histogramme der Phonemdauer für das jeweils erste Phonem der englischen Buchstaben R und T. Rechts: Bestimmung von $mindur(10\%)$.

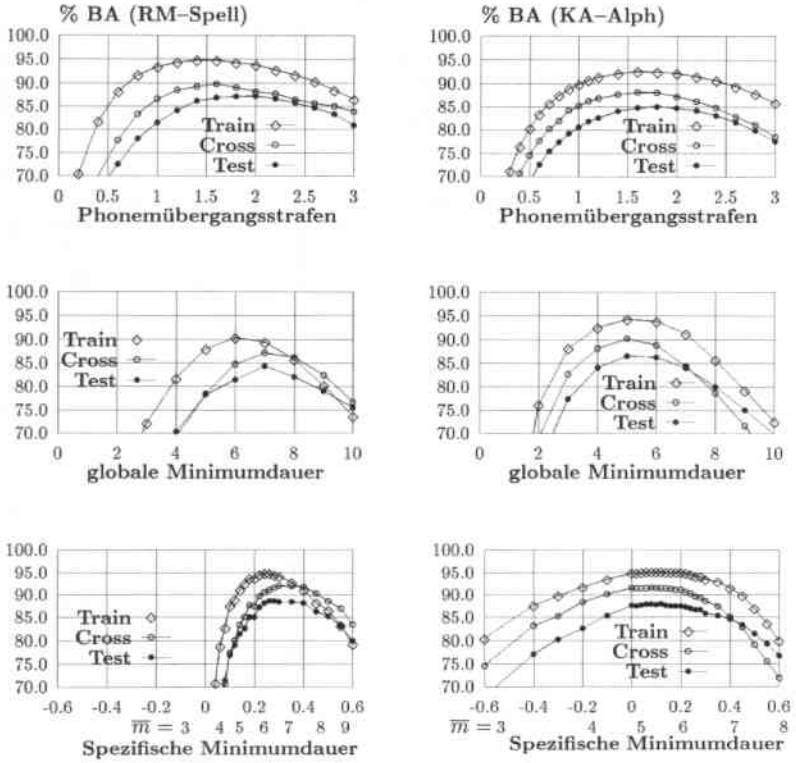


Abbildung 6.14: Buchstabenakkuratheit (BA) in Abhängigkeit der Phonemübergangsstrafen

Längenmodellierung	RM-Spell			KA-Alph		
	Train	Cross	Test	Train	Cross	Test
Ohne Längenmodellierung	28.4	2.0	9.4	38.9	32.8	25.0
(1) Phonemübergangsstrafen	94.7	89.8	86.8	92.6	88.1	84.8
(2) Erzwung. minimale Dauer (global)	89.4	87.1	84.3	94.3	90.1	86.6
(3) minimale Dauer (phonemspezifisch)	94.1	92.0	88.4	95.2	91.5	87.8
(4) Kombination aus (1) und (3)	95.2	92.8	**89.0**	95.4	91.8	**88.4**
Gewichte von Phonemtraining, wie (3)	84.5	84.1	82.6	87.1	85.2	80.8

Tabelle 6.6: Erkennungsergebnisse in % BA für verschiedene Methoden der Längenmodellierung von Phonemen auf der RM-Spell- und KA-Alph-Datenbank

finden sich in den Zeilen (1) - (3) wieder sowie in den Graphen in Abbildung 6.14. Mit einer Kombination aus Phonemübergangsstrafe ($p = 0.2$) und phonemspezifischer Mindestdauer konnte eine weitere Verbesserung erreicht werden (Zeile (4)). Auch für die hier nicht aufgeführten sprecherabhängigen Ergebnisse (mjmt, mdbs) schneidet die phonemspezifische Mindestdauer konsistent am besten ab. In der letzten Zeile wird noch einmal der Wert des Trainings auf Wortebene verdeutlicht, denn betreibt man die kontinuierliche Erkennung (bei einer phonemspezifischen Mindestdauer) direkt nach dem Training auf Phonemebene, ergeben sich über 6% (absolut) schlechtere Ergebnisse auf den Testmengen.

6.6.2 Wortdauer

Globale Worteingangsstrafen

In Abschnitt 3.6 wurde bereits erwähnt, daß in praktisch allen Spracherkennungssystemen in der Suche eine Worteingangsstrafe (der Term β in Gl. (3.29)) eingesetzt wird, um die Anzahl von Einfügungen und Auslassungen auszubalancieren. Die Worteingangsstrafe β gilt global für alle zu erkennenden Wörter, und mit wachsendem β wird ein Übergang in das folgende Wort zunehmend erschwert. Der Einfluß von β ist in den beiden oberen Graphen in Abbildung 6.15(oben) ersichtlich, wo die Erkennungsergebnisse in % Buchstabenakkuratheit für Werte von β zwischen 0 und 10 aufgetragen sind. Auch hier wird eine endgültige Einstellung von β anhand des besten Ergebnisses auf der Kreuzvalidierungsmenge bestimmt.

Lernen von wortspezifischen Eingangsstrafen

Nicht alle Buchstaben werden gleich häufig eingefügt. Dies motiviert zu einem weitergehenden Schritt: Statt einer globalen benutzen wir eine wortspezifische Eingangsstrafe β_i, die automatisch eingestellt werden soll. Während in der Spracherkennung großer Wortschätze Statistiken zur Häufigkeit von Einfügungen und Auslassungen einzelner

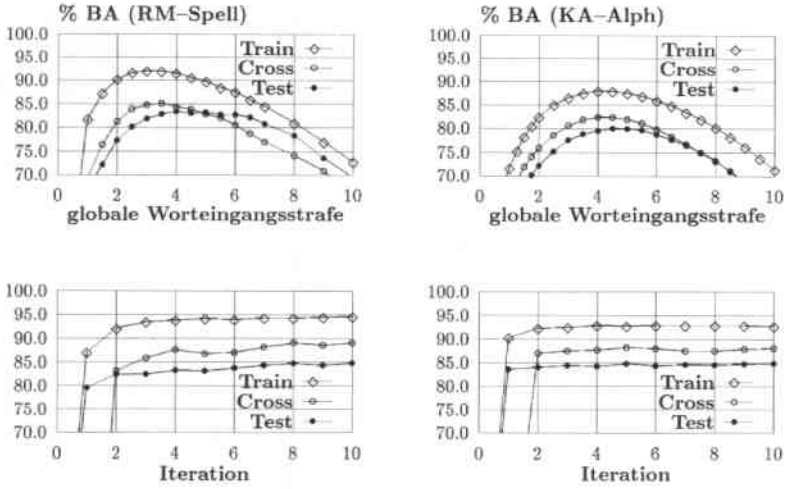


Abbildung 6.15: Buchstabenakkuratheit (BA) in Abhängigkeit der Worteingangsstrafen

Wörter nur ungenau geschätzt werden können, ist dies im Spezialfall der Buchstabiererkennung möglich, da hier alle Buchstaben hinreichend oft wiederholt werden.

Als Fehlermaß betrachten wir (für jeden Buchstaben L_i einzeln) die relativen Häufigkeiten $H_i = (I_i - D_i)/n_i$ des Ungleichgewichts zwischen I_i Einfügungen und D_i Auslassungen bei insgesamt n_i Buchstaben L_i.

Offensichtlich sind H_i und β_i negativ korreliert, denn mit wachsender Eingangsstrafe sinkt die Zahl der Einfügungen. Neben β_i haben viele andere Faktoren Einfluß auf H_i, eine mathematisch exakte Abhängigkeit $H_i(\beta_i)$ ist daher nicht praktikabel formulierbar. Deshalb wird als Heuristik ein linearer Zusammenhang zwischen H_i und β_i angenommen. Dann kann mit einem Gradientenabstiegsverfahren die Eingangsstrafe β_i^t in jeder Iteration t wie folgt angepaßt werden:

$$\beta_i^{t+1} = \beta_i^t + \epsilon \cdot \frac{I_i^t - D_i^t}{n_i} \tag{6.27}$$

Um eine zügige Anpassung zu erreichen, werden I_i^t und D_i^t jeweils nach wenigen Buchstabensätzen neu bestimmt. Mit einer Lernrate $\epsilon = 0.1$ und einem Momentum-Term von 0.5 ergeben sich die in Abbildung 6.15 dargestellten Ergebnisse. Bereits nach wenigen Iterationen ergeben sich Werte, die für KA–Alph um 5 und für RM–Spell um 2 Prozentpunkte besser sind als mit der besten globalen Worteingangsstrafe.

	RM-Spell			KA-Alph		
	Train	Cross	Test	Train	Cross	Test
(1) Globale Worteingangsstrafen	91.9	85.1	82.8	87.9	82.4	79.6
(2) Gelernte Worteingangsstrafen	94.3	89.0	84.7	92.7	88.2	84.8

Tabelle 6.7: Ergebnisse in % BA bei Kontrolle der Wortdauer durch Worteingangsstrafen

Die Ergebnisse der Wortdauermodellierung sind in Tabelle 6.7 zusammengefaßt. Mit den gelernten wortspezifischen Worteingangsstrafen erzielt man etwa um 6 Prozentpunkte bessere Ergebnisse, die allerdings immer noch um etwa 4 Prozentpunkte schlechter sind als bei einer Phonemdauermodellierung. Natürlich können Phonem- und Wortdauermodellierung auch gemeinsam angewandt werden. Wegen der Explosion der Kombinationsmöglichkeiten ist eine systematische Untersuchung der rechenzeitaufwendigen Tests jedoch nur bedingt möglich. Da einige stichprobenartige Versuchsreihen einer kombinierten Längenmodellierung nur insignifikante Verbesserungen gegenüber einer reinen Phonemlängenmodellierung erbrachten, wird hier auf eine Darstellung verzichtet.

6.6.3 Korrektives Training

Ganz ähnlich wie auf Buchstabenebene kann auch auf Satzebene diskriminativ trainiert werden. Dazu müssen die einzelnen Phonembewertungen so verändert werden, daß sich die Gesamtbewertung der korrekten Satzhypothese erhöht und die aller inkorrekten vermindert. Die zeitliche Zuordnung der Phoneme zum erkannten Satz ist aus der Suche bekannt. Ist der Satz inkorrekt, bestimmt man einen zusätzlichen Phonempfad für den korrekten Satz, indem die Suche gezwungen wird, die im Training bekannte Phonemsequenz der korrekten Wortfolge zu durchlaufen (*forced alignment*). Anschließend erhalten alle Phoneme entlang des korrekten Pfades „positives Training" (d.h. ein Fehlersignal, das ihren Ausgabewert erhöht), und die Phoneme entlang des inkorrekten Pfades erhalten „negatives Training". Wie auf Wortebene wird also die CFM-Fehlerfunktion berechnet, und der Betrag des Fehlers bestimmt sich analog zu (6.24) und (6.25).

Abbildung 6.16 veranschaulicht diese Vorgehensweise am Beispiel eines Satzes „`C A B`". Dabei gelten im wesentlichen die gleichen Überlegungen wie bereits im Worttraining im Beispiel aus Abbildung 6.11, nur jetzt auf Satzebene: Der beste von der Suche gefundene Pfad repräsentiert die inkorrekte Hypothese „`C A A B`". Danach wird ein Pfad für die korrekte Sequenz „`C A B`" bestimmt. An den fehlerfreien Stellen des Satzes sind die beiden Pfade weitgehend identisch, so daß sich die Effekte von negativem und positivem Training neutralisieren. Lediglich an den kritischen, fehlerhaften Stellen wird effektiv trainiert, wodurch ein effizientes diskriminatives Training realisiert wird.

Würde die Suche nicht nur die beste, sondern alternative Hypothesen finden, könnte auch ein korrekt erkannter Satz gegen seinen schärfsten Konkurrenten diskriminativ trainiert werden. Ein solcher, sogenannter N-Besten-Suchalgorithmus ist jedoch im MS-TDNN nicht implementiert.

In den Experimenten zum korrektiven Training wurde zur Längenmodellierung jeweils diejenige phonemspezifische Mindestdauer zusammen mit einer Phonemeingangsstrafe benutzt, die in den Experimenten in Abschnitt 6.6.1 die besten Ergebnisse erzielte. Durch das korrektive Training konnte die Buchstabenakkuratheit nochmals um 2 Prozentpunkte gesteigert werden. Die genauen Ergebnisse können aus der ersten Zeile in Tabelle 6.8 abgelesen werden.

Erweiterung über die Wortgrenzen

Im Training auf Wortebene werden die Phonemgrenzen innerhalb eines Wortes nicht mehr benötigt, denn sie werden bei der Berechnung der Viterbi-Pfade automatisch gefunden und optimiert. Nach wie vor fest vorgegeben sind die Grenzen an den Rändern der Buchstaben, die jedoch nicht notwendigerweise richtig oder optimal für den Trainingsprozeß sind. Um auch diese Grenzen vom System selbst optimieren zu lassen, ohne dabei gleich eine völlig freie Erkennung auf Satzebene mit ihren Problemen (Einfügungen und Auslassungen) zu erlauben, wurde als Zwischenschritt ein *Training über Wortgrenzen* eingeführt. Wie in Abbildung 6.17 illustriert, wird dazu die Eingabe um einige[14] Zeitschritte nach links und rechts erweitert, wodurch das letzte Phonem des vorangehenden sowie das erste Phonem des nachfolgenden Wortes in der Eingabe sichtbar werden. Entsprechend müssen auch alle Buchstabenmodelle an ihrem Anfang und Ende um diese beiden Phoneme ergänzt werden. Mit den so erweiterten Buchstaben wird wie auf Wortebene trainiert. Dabei befinden sich die ehemals festen Buchstabengrenzen nun *innerhalb* des trainierten Wortes und können daher wie alle anderen Phonemgrenzen frei gewählt und automatisch optimiert werden.

Auf RM–Spell sowie auf den sprecherabhängigen mjmt- und mdbs-Daten[15] konnte damit ein weiterer Prozentpunkt gewonnen werden. Lediglich auf KA–Alph erwies sich das Training über Wortgrenzen als nicht hilfreich.

Training	RM–Spell			KA–Alph		
	Train	Cross	Test	Train	Cross	Test
korrektives Training	96.3	92.9	**89.6**	97.6	92.9	**90.4**
korrektives Training + Wortgrenzen	95.8	92.2	**91.4**	97.5	92.7	**89.7**

Tabelle 6.8: % Buchstabenakkuratheit (BA) nach korrektivem Training

6.6.4 Zusammenfassende Darstellung der Trainingsphasen

Eine tabellarische Übersicht über die wichtigsten Systemmodellierungen der vorangehenden Abschnitte ist in der Zusammenfassung in Abschnitt 6.10 zu finden. Hier sollen

[14] Typischerweise um einen heuristisch bestimmten Wert von 10 Zeitschritten (= 100 ms)

[15] Siehe Tabelle 6.17 in der Zusammenfassung dieses Kapitels für die Testergebnisse auf den Daten von mjmt und mdbs.

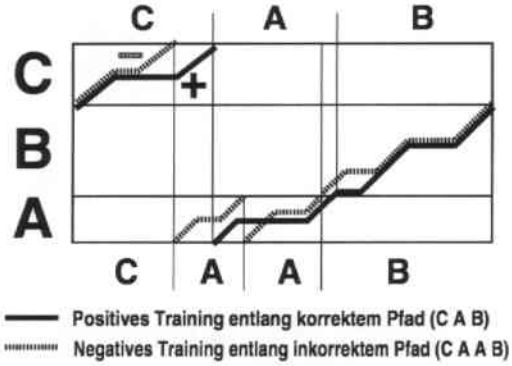


Abbildung 6.16: Korrektives Training auf Satzebene

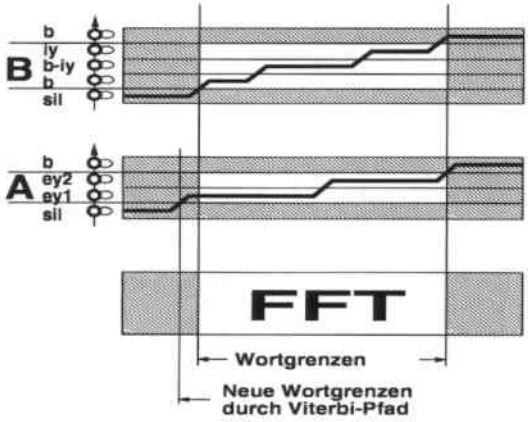


Abbildung 6.17: Training über die Wortgrenzen hinaus

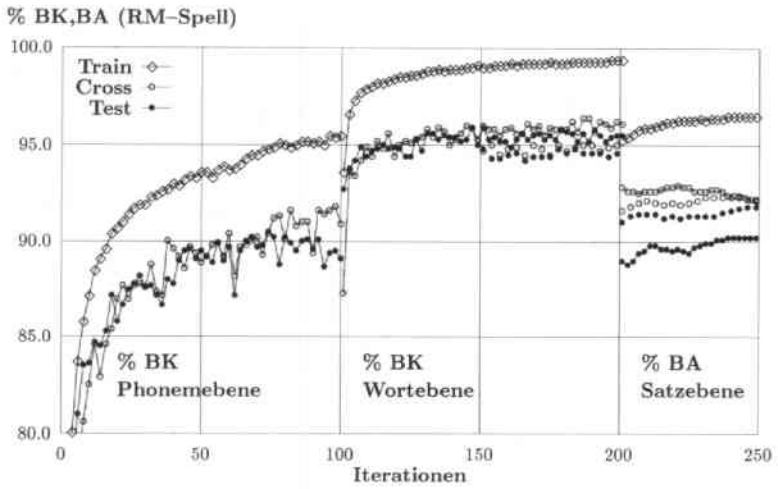


Abbildung 6.18: Lernkurven auf der RM-Spell-Datenbank. Phonemtraining bis Iteration 100, anschließend Worttraining bis Iteration 200 (% Buchstaben korrekt auf ausgeschnittenen Buchstaben). Ab Iteration 201 Satztraining (% Buchstabenakkuratheit auf ganzem Satz). Alternativ wurde ab Iteration 150 Training über Wortgrenzen benutzt (Kurven mit nicht verbundenen Punkten), ebenfalls mit anschließendem Satztraining, das nun insgesamt besser ausfällt.

am Beispiel der RM-Spell-Erkennungsaufgabe die Auswirkungen der unterschiedlichen Trainingsschritte anhand des gesamten Trainingsverlaufes im Graphen in Abbildung 6.18 veranschaulicht werden.

Aufgetragen sind die Erkennungsleistungen auf der Trainings-, Kreuzvalidierungs- und Testmenge für die verschiedenen Trainingsphasen: An das Training auf Phonemebene (Iteration 1 bis 100) schließt sich das Training auf Wortebene (Iteration 101-200) mit einer deutlichen Verbesserung der Erkennungsleistung an. Geht man von der Erkennung ausgeschnittener Buchstaben (gemessen in % Buchstaben korrekt, BK) auf Buchstabensequenzen über (gemessen in % Buchstabenakkuratheit, BA), sinkt die Erkennungsrate wegen der jetzt zusätzlich möglichen Einfügungen und Auslassungen, wird aber durch das Training auf Satzebene (Iteration 200 bis 250) verbessert. Wird nach dem „normalen" Training auf Wortebene noch über Wortgrenzen hinaus trainiert (Kurven mit nicht verbundenen Punkten), fällt das jetzt anschließende Training auf Satzebene besser aus als zuvor.

6.7 Architekturparameter

Der Entwurf eines neuronalen Netzes läßt viele Freiheitsgrade offen, insbesondere wenn es sich dabei wie beim MS-TDNN um ein relativ stark strukturiertes Netz handelt: Wie viele Parameter („Hidden Units") benötigt man? Wie und wo werden Zeitverzögerungen (Time-Delays) eingesetzt? Welche Transfer- und Fehlerfunktionen, welche Trainingsverfahren wählt man? Mit wievielen Zuständen werden die Buchstabenmodelle ausgestattet?

Ein interessanter Lösungsansatz zu diesen Fragen wurde von Bodenhausen verfolgt, der am Beispiel des MS-TDNN Algorithmen untersuchte, die eine automatische Strukturierung der Netzwerkarchitektur erlauben [BW91, Bod94, BH95b]. Dabei wird die Anzahl der Neuronen in der verborgenen Schicht und in den Wortmodellen sowie die Breite der Zeitverzögerungen automatisch während des Trainingsprozesses eingestellt. Ausgehend von einer Minimalkonfiguration wird die Netzwerkarchitektur solange mit zusätzlichen Ressourcen ausgestattet, bis eine überwachende Instanz mit einem ausgeklügelten Kreuzvalidierungsverfahren eine weitere Vergrößerung des Netzes unterbindet. In einzelnen Fällen konnte mit diesem automatischen Verfahren annähernd dieselbe Performanz wie mit manuellen Netzwerkoptimierungen erreicht werden; einige Ergebnisse sind in Abschnitt 6.9.1 aufgeführt.

Die bisherigen Überlegungen in diesem Kapitel galten hauptsächlich dem dynamischen Verhalten des Netzes, insbesondere den Trainingsverfahren und der Zeitdauermodellierung. In diesem Abschnitt soll auf zwei statische Parameter der Netzwerkarchitektur eingegangen werden, nämlich die Anzahl der Parameter sowie die Art und Breite der Zeitverzögerungen. Der Aspekt der Netzwerkmodularisierung wird in Abschnitt 6.8 besprochen.

6.7.1 Anzahl der Parameter

Wie zuvor bezeichnen wir die Anzahl der Neuronen in der Eingabe-, der verborgenen und der Phonemschicht mit n_{E}, n_{H} und n_{P}. Die Zeitverzögerungen in der Eingabe- und der verborgenen Schicht erfassen jeweils d_{E} und d_{H} zusätzliche Zeitschritte in die Vergangenheit und die Zukunft. Daraus errechnet sich eine Gesamtzahl von Parametern (Verbindungen oder Gewichten, inkl. Schwellwertgewichten) von

$$n = ((2d_{\mathrm{E}} + 1) \cdot n_{\mathrm{E}} + 1) \cdot n_{\mathrm{H}} + ((2d_{\mathrm{H}} + 1) \cdot n_{\mathrm{H}} + 1) \cdot n_{\mathrm{P}} \tag{6.28}$$

n_{E} ist durch die Dimension des Merkmalsraumes und n_{P} durch die Anzahl der Zustände (Phoneme) in den Buchstabenmodellen bestimmt. Die Breite der Zeitverzögerungen wird zunächst nicht variiert, somit beeinflußt lediglich n_{H} die Anzahl der Gewichte. Mit $n_{\mathrm{P}} = 70$ für den deutschen oder $n_{\mathrm{P}} = 59$ für den englischen Phonemsatz, sowie mit $d_{\mathrm{E}} = 1, d_{\mathrm{H}} = 2, n_{\mathrm{E}} = 16$ ergibt sich für die bisher untersuchten Netze mit $n_{\mathrm{H}} = 50$ verborgenen Neuronen eine Gesamtzahl von 17 259 bzw. 20 020 Parametern.

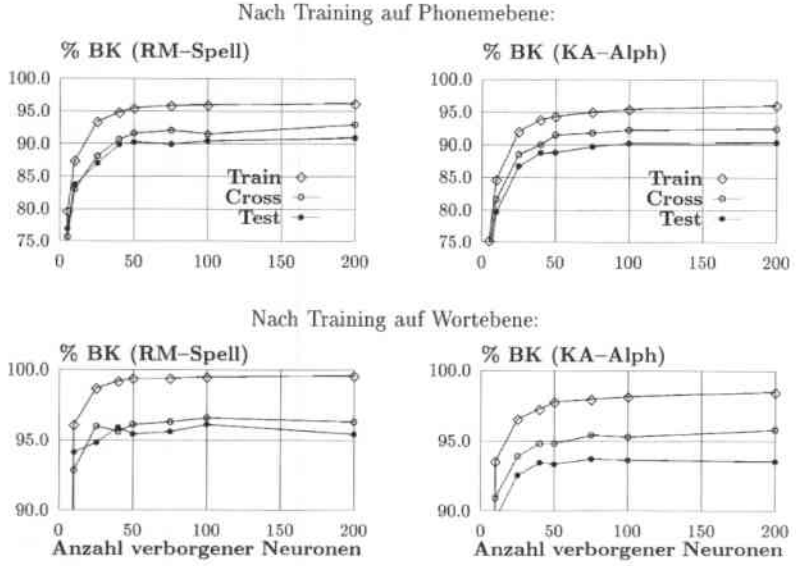


Abbildung 6.19: % Buchstaben korrekt (BK) in Abhängigkeit der Anzahl der Parameter

In einer Serie von Experimenten wurde das MS-TDNN mit einer wachsenden Anzahl von verborgenen Neuronen trainiert. Die Anzahl der Parameter wächst linear mit n_H, jedes zusätzliche Neuron in der verborgenen Schicht resultiert in etwa 400 weiteren Parametern (Gewichten). In Abbildung 6.19 sind die Erkennungsraten in % Buchstaben korrekt für Netzgrößen zwischen 5 (= 2 065 Parameter) und 200 (= 79 870 Parameter[16]) verborgenen Neuronen aufgetragen. Das „deutsche" Netz wurde für 30 und das „englische" für 100 Iterationen trainiert[17], die abgebildeten Ergebnisse sind jeweils Durchschnittswerte aus den letzten 10 Iterationen. Wie zu erwarten, steigen die Erkennungsraten auf den Trainingsmengen stetig mit größer werdenden Netzen. Interessanterweise leiden die Ergebnisse auf den Testdaten auch bei den größeren Netzen noch nicht unter zu starkem Auswendiglernen („Overfitting") der Netze. Ab etwa $n_H = 50$ verborgenen Neuronen stellen sich nur noch geringe Verbesserungen der Erkennungsraten ein. Da kleinere Netze weniger Speicher- und Rechenressourcen benötigen, ist $n_H = 50$ für beide Datenbanken eine günstige Netzgröße, wenngleich auch für KA–Alph etwas unterdimensioniert.

[16] Jeweils im „deutschen Fall" mit $n_P = 70$

[17] Da die deutschen Trainingsdaten wesentlich umfangreicher sind, werden weniger Trainingsiterationen benötigt, um in den Sättigungsbereich zu kommen.

6.7.2 Art und Breite der Zeitverzögerungen

Mit einer Fensterbreite von $2d_E + 1 = 3$ Zeitschritten in der Eingabe- und $2d_H + 1 = 5$ in der verborgenen Schicht betrachtet das TDNN einen zeitlichen Kontext von insgesamt 7 Eingabevektoren. Wird dieser Gesamtkontext bereits vollständig in der Eingabeschicht ($d_E = 3$) und dafür ohne Verzögerungen in der verborgenen Schicht ($d_H = 0$) berücksichtigt (Abbildung 6.20(links)), erhält man eine einfachere Netzwerkarchitektur, die einem MLP entspricht. In den folgenden Experimenten wird diese MLP-Architektur mit der des TDNN verglichen. Zusätzlich werden für das MLP verschiedene Kontextbreiten zwischen $d_E = 0$ (nur ein Eingabevektor, kein zusätzlicher Kontext) und $d_E = 5$ (insgesamt 11 Zeitschritte Kontext) untersucht.

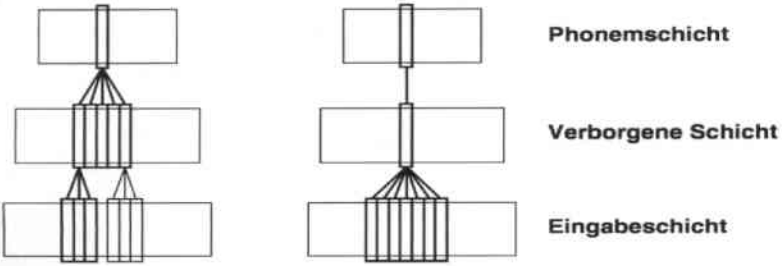


Abbildung 6.20: Netz-Architekturen mit einer Kontextbreite von 7 Eingabevektoren, links mit, rechts ohne Verzögerungen in der verborgenen Schicht

Eine Verlagerung der Zeitverzögerungen in die Eingabeschicht reduziert die Gesamtzahl der Parameter[18], ebenso eine Verkleinerung des Kontextes. Um vergleichbare experimentelle Bedingungen zu erreichen, wurde dieser Effekt durch eine jeweils entsprechend höhere Anzahl von Neuronen in der verborgenen Schicht ausgeglichen (Tabelle 6.9).

Für Kontextbreiten von insgesamt 1,3,5,7,9 und 11 Vektoren (ausschließlich in der Eingabeschicht) wurde jeweils ein Netz für 30 Iterationen auf den KA–Alph- und für 30 Iteration auf den RM–Spell-Daten auf Phonemebene trainiert. Die (wie zuvor auf den letzten 10 Iterationen gemittelten) Erkennungsraten sind in Abbildung 6.21 als durchgezogene Kurven zu sehen. Die Erkennungsraten für ein TDNN, das ebenfalls 7 Vektoren Kontext (3 in der Eingabe-, 5 in der verborgenen Schicht) berücksichtigt, sind als einzelne Punkte abgetragen.
Aus den Graphiken wird ersichtlich, daß die Fensterbreite nicht kleiner als 5 gewählt werden sollte. Die besten Ergebnisse werden konsistent über beide Datenbanken für Fensterbreiten zwischen 5 und 9 erzielt, wobei in drei von vier Fällen eine Breite von 7 die beste Wahl ist.
Die TDNN- und MLP-Variante mit jeweils einer Kontextbreite von 7 unterscheiden sich

[18] Da der größere Teil der Parameter zwischen verborgener und Phonemschicht liegt

nicht wesentlich. Die MLP-Variante schneidet, mit Ausnahme auf den KA–Alph-Daten nach Training auf Wortebene, geringfügig besser ab.

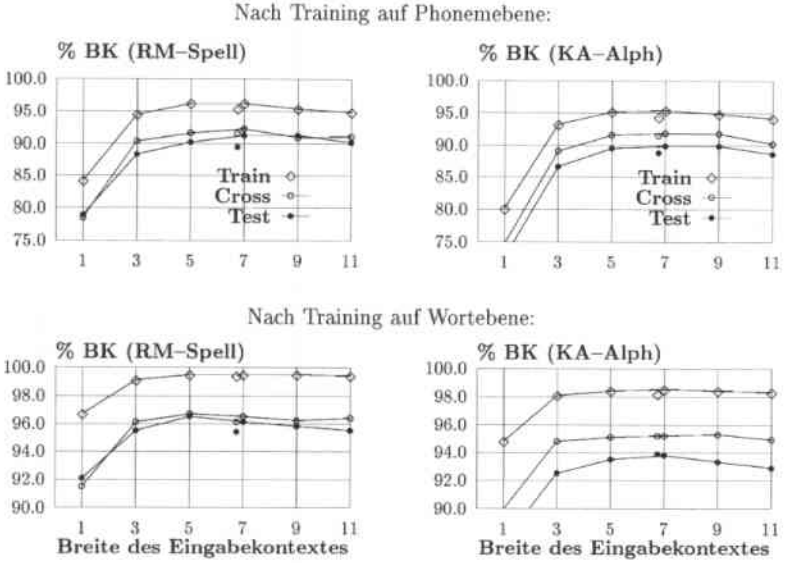


Abbildung 6.21: % Buchstaben korrekt in Abhängigkeit von der Breite des Eingabekontextes (ausschließlich in der Eingabeschicht). Die einzelnen Punkte sind die entsprechenden Ergebnisse für ein TDNN mit einer Gesamtkontextbreite von 7 (3 und 5) Sprachvektoren.

			RM–Spell ($n_P = 59$)		KA–Alph ($n_P = 70$)	
Netz ($n_E = 16$)	d_E	d_H	n_H	Parameter	n_H	Parameter
Standard-TDNN(7)	1	2	50	17259	50	20020
MLP-11	5	0	73	17287	81	20077
MLP-9	4	0	84	17195	93	20065
MLP-7	3	0	100	17259	109	20017
MLP-5	2	0	123	17279	132	20002
MLP-3	1	0	159	17231	168	20062
MLP-1	0	0	226	17235	230	20080

Tabelle 6.9: Dimensionierung verschiedener Netze mit unterschiedlich breiten Eingabefenstern. Um die Anzahl der Netzparameter vergleichbar zu halten, werden Netze mit einem breiteren Eingabefenster mit weniger verborgenen Neuronen ausgestattet.

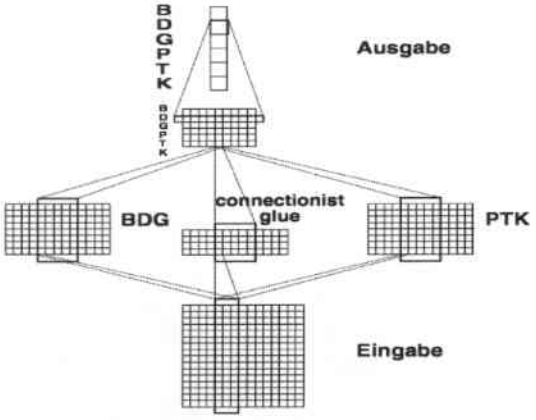


Abbildung 6.22: Modulares TDNN mit *connectionist glue*

6.8 Modulare Netze

In den bisherigen Netzwerken wurde sprecherunabhängig erkannt, indem alle Sprecher in einem gemeinsamen Netz trainiert wurden. Ist es möglich, bessere sprecherunabhängige Erkennung zu erzielen, indem sich Teilkomponenten des Netzes auf bestimmte Sprecher(gruppen) spezialisieren? Im Gegensatz zu monolithischen Netzwerken bestehen modulare Netzwerke aus Teilkomponenten, die individuell trainiert werden und/oder sich in ihrer Architektur unterscheiden. Modulare Netze kommen oft mit weniger Parametern aus[19] als entsprechende monolithische Netze und sind dadurch erst zu handhaben. Mit wachsender Rechenleistung tritt diese Motivation in den Hintergrund, und man versucht, mit modularen Systemen den Grad an Spezialisierung (manchmal auch an Redundanz) und damit die Klassifikationsleistung zu erhöhen. In [Wai89, WSS89b] untersucht Waibel eine Reihe von Techniken, um mit modularen TDNNs alle (statt nur drei) Konsonanten zu klassifizieren. Neben dem bekannten /dbg/-TDNN wurden fünf weitere TDNNs unabhängig voneinander darauf trainiert, jeweils eine Gruppe von zwei bis vier Konsonanten zu erkennen, und dann zu einem Netz verschaltet. Nur jeweils ein TDNN ist auf ein bestimmtes Phonem spezialisiert, die anderen Netzwerke erzeugen bei der „fachfremden" Eingabe bedeutungslose Werte, die die Klassifikation beeinträchtigen. Deshalb werden zusätzliche Verbindungen in Form eines weiteren TDNN eingeführt, die die Aktivierungen aller Phoneme beeinflussen, wie in Abbildung 6.22 illustriert. Diese als „connectionist glue" bezeichneten Verbindungen werden erst trainiert, nachdem die einzelnen Komponenten bereits eingelernt sind.

[19] In einem vollständig verbundenen Netzwerk wächst die Anzahl der Gewichte quadratisch mit der Anzahl der Knoten.

Gewöhnlich unterteilen modulare Netze nicht (wie im gerade beschriebenen Netz) den Ausgabe-, sondern den Eingaberaum, wie auch im **Meta-Pi-Netzwerk** von Hampshire und Waibel [HIW89b, HIW90a, HIW90b]. Dort sind die modularen Teilkomponenten nicht auf einzelne zu erkennende Phoneme, sondern auf unterschiedliche Sprecher spezialisiert. Statt einer additiven Beeinflussung durch den „connectionist glue" werden die einzelnen Komponenten durch eine multiplikative Verbindung linear kombiniert. Dazu werden 6 TDNNs auf jeweils einem von 4 männlichen und 2 weiblichen Sprechern auf der BDG-Task trainiert. Ihre Ausgaben werden gewichtet aufaddiert, wobei die Gewichtung durch ein zusätzliches Netz gelernt wird.

Von Jordan und Jacobs wurde ein Ansatz entwickelt, bei dem Klassifikatoren hierarchisch in einer Baumstruktur kombiniert werden (*„Hierachical Mixture of Experts"*) [JJ94].

6.8.1 Modulare MS-TDNNs

Hampshires Meta-Pi-TDNN [HIW90b] inspirierte zu einer Serie von Experimenten mit unterschiedlich modularisierten MS-TDNNs [HW93b], die als Spezialfall auch Hampshires Meta-Pi-Architektur enthalten. In Abbildung 6.23 sind die vier untersuchten Netzwerkstrukturen illustriert, die sich durch einen zunehmenden Grad an Spezialisierung unterscheiden. Im ersten Netz (POOLED) gibt es keine sprecherspezifischen Parameter, alle Daten werden mit einem gemeinsamen Netz trainiert. Das zweite Netz (BIASED) besitzt sprecherabhängige Bias-Gewichte, jeder Sprecher hat also gerade soviele spezifische Parameter, wie es Neuronen in der verborgenen und in der Phonemschicht gibt.

Im SHARED-Netz gibt es spezifische Verbindungen zwischen Eingabe- und verborgener Schicht für jeden Sprecher, die Gewichte in die Phonemschicht werden jedoch von allen Sprechern geteilt. Das INDIVIDUAL-Netz schließlich entspricht der Architektur des Meta-Pi-Netzes: Jedem Sprecher steht ein komplettes TDNN zur Verfügung, die Ausgaben werden auf Phonemebene zusammengeführt.

Wie in Abbildung 6.23 dargestellt, wird jedes *interne Sprechermodell* (*ISM*, also die jeweils sprecherspezifischen Parameter) durch eines der Auswahl-Neuronen (ISM-AN) angesteuert. Im BIASED-Netz bestehen die ISM aus den sprecherspezifischen Schwellwertgewichten, die mit den ISM-AN verbunden sind. Im SHARED- und INDIVIDUAL-Netz wirken die ISM-AN multiplikativ und bestimmen dadurch die Gewichtung der sprecherspezifischen Komponenten. Ein bestimmter Sprecher wird selektiert, indem das entsprechende ISM-AN auf 1, und die restlichen auf 0 gesetzt werden. Alternativ können die Sprechermodelle durch ISM-AN-Ausgaben a_i linear kombiniert werden, wobei als Randbedingungen $a_i \geq 0$ und $\sum_i a_i = 1$ gefordert werden.

In der Lernphase ist die Sprecheridentität für die Trainingssätze und damit die gewünschte Ausgabe der ISM-AN bekannt. In der Testphase muß eine geeignete Wahl automatisch getroffen werden. Dazu gibt es die in Abbildung 6.24 dargestellten zwei Möglichkeiten: **(a)** Unabhängig von der eigentlichen Erkennung wird ein zusätzliches

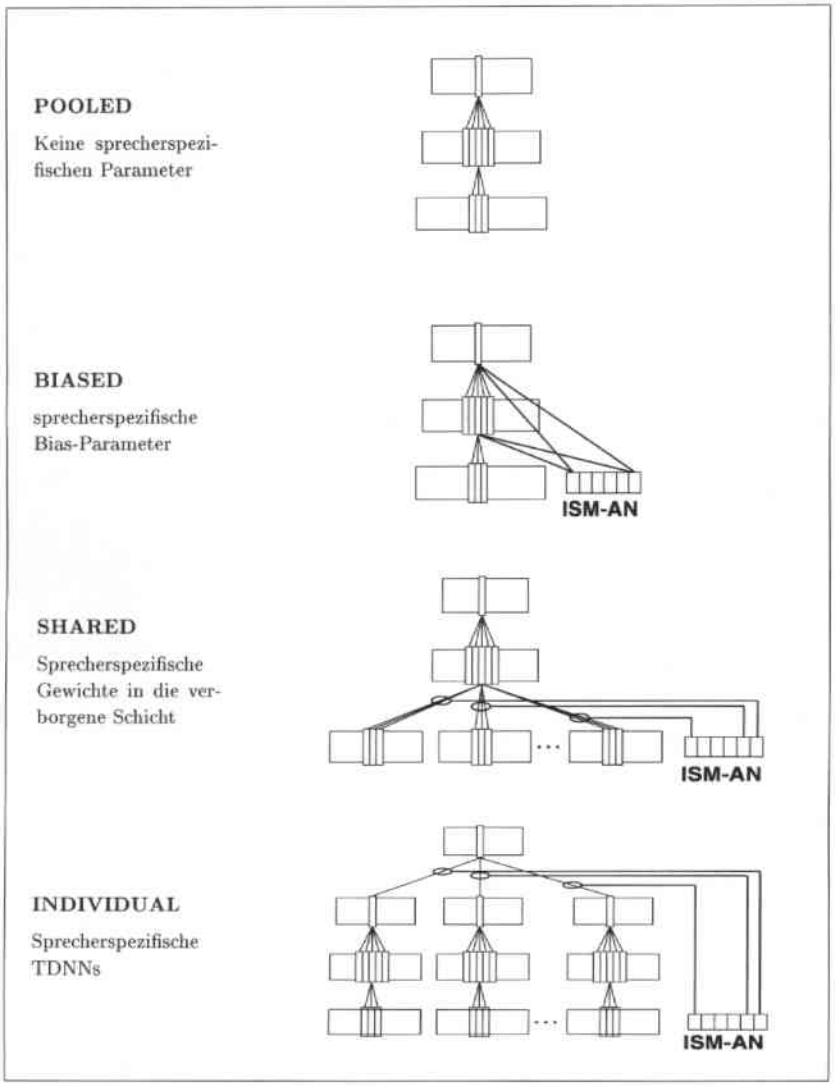


Abbildung 6.23: Vier Netzarchitekturen mit zunehmender Anzahl sprecherspezifischer Parameter, die mit „Internen-Sprecher-Modellen-Auswahlneuronen" (ISM-AN) angesteuert werden

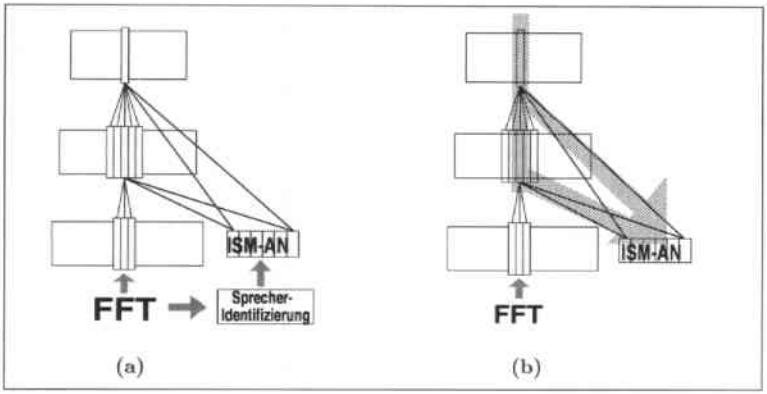


Abbildung 6.24: Anpassung der internen Sprechermodelle am Beispiel der BIASED-Architektur: „Sprecher-Identifizierung" (a) und „Tuning in" (b)

Netzwerk trainiert, um einen Sprecher zu identifizieren. Vor jedem Erkennungslauf werden dann die Ausgaben der ISM-AN durch dieses Netz bestimmt, oder **(b)** es erfolgt eine Adaption anhand einiger Äußerungen des unbekannten Sprechers („Tuning in"). Dazu werden die Gewichte der TDNNs „eingefroren", und anhand des üblichen Fehlerrückführungsverfahrens wird versucht, die ISM-AN so einzustellen, daß der Fehler auf den Adaptionsdaten minimiert wird. Mit der so gefundenen Einstellung werden dann die eigentlichen Testdaten erkannt. Bei unüberwachter Adaption ist die korrekte Transkription der Adaptionsdaten unbekannt. Ersatzweise kann die erkannte Transkription als korrekt betrachtet werden (in den Experimenten von [BC91] als „Phantom Targets" bezeichnet).

6.8.2 Experimentelle Ergebnisse

Mehrsprecher

Anhand dreier männlicher (mjmt, mdbs, maem) und dreier weiblicher (fcaw, flgt, fee) Sprecher der CMU–Alph-Datenbank wurden die verschiedenen Architekturen und Adaptionsmethoden getestet. Die Ergebnisse, gemittelt über alle 6 Sprecher und gemessen in % Buchstaben korrekt (auf ausgeschnittenen Buchstaben), sind in Tabelle 6.10 zusammengefaßt. In der ersten Spalte wurde die Sprecheridentität als bekannt vorausgesetzt. Mit einer zunehmenden Spezialisierung wird dabei eine bessere Erkennung erzielt. Das SHARED-Netz übertrifft sogar individuell trainierte MS-TDNNs. Ein ähnliches Bild ergibt sich, wenn der Sprecher durch ein zusätzliches Netzwerk identifiziert wird (2. Spalte, siehe auch Abbildung 6.24(a)). Die Erkennungsraten fallen zwar insgesamt etwas niedri-

Mehr-Sprecher	Sprecher bekannt	Sprecher identifiziert	Adaption überwacht	Adaption unüberwacht
POOLED	92.7			
BIASED	93.8	92.9	92.2	89.7
SHARED	95.9	94.5	95.0	78.5
INDIVID.	95.1	94.9	95.6	95.6

Tabelle 6.10: Im Mehr-Sprecher-System werden die Netze mit neuen Daten von denselben 6 Sprechern getestet

ger aus, sind aber immer noch deutlich besser als im Standardfall des POOLED-Netzes. Zur Adaption (3. und 4. Spalte, siehe auch Abbildung 6.24(b)) wurden 10 der insgesamt 400 Testsätze eingesetzt. Die überwachte Adaption funktioniert geringfügig besser als eine automatische Sprecheridentifizierung.

Neue Sprecher

Wie werden sich die auf 6 Sprecher spezialisierten Netze verhalten, wenn sie mit bisher „ungesehenen" Sprechern konfrontiert werden? Die gemittelten Ergebnisse von Experimenten mit 7 neuen Sprechern (5 männlich, 2 weiblich, je 50 Sätze oder etwa 250 Buchstaben) sind in Tabelle 6.11 zusammengefaßt. Eine Auswahl der „richtigen" Sprecher (Spalte 1) ist für neue Sprecher nicht mehr möglich, die automatische Sprecheridentifikation (Spalte 2) muß also einen ähnlichen oder eine Kombination möglichst ähnlicher Sprecher finden. Dies funktioniert jedoch in allen Fällen schlechter als das POOLED-Modell. Sowohl durch überwachte (Spalte 3) als auch durch unüberwachte Adaption (Spalte 4) kann jedoch eine Sprecherkombination gefunden werden, mit der das POOLED-Netz übertroffen wird. Während für die bekannten Sprecher ein zunehmender Spezialisierungsgrad förderlich ist, funktionieren die weniger spezialisierten Netze für die neuen Sprecher besser. Überwachte und unüberwachte Adaption erzielen bessere Ergebnisse als eine Identifikation der neuen Sprecher.

Neue Sprecher	Sprecher bekannt	Sprecher identifiziert	Adaption überwacht	Adaption unüberwacht
POOLED	81.3			
BIASED	–	79.5	83.2	82.2
SHARED	–	73.6	83.4	73.6
INDIVID.	–	66.2	72.3	69.3

Tabelle 6.11: Test auf neuen Daten von 7 neuen Sprechern

Resource Management

Geht man von einer Mehrsprecher-Erkennung zu einer sprecherunabhängigen über, ist eine Spezialisierung auf bestimmte Sprecher problematischer. Welche von potentiell be-

liebig vielen Sprechern sind als Modelle geeignet? Sollte man sich auf einzelne oder auf eine Gruppe von Sprechern als ein ISM festlegen? Versuche, auf den etwa 100 Trainingssprechern der RM–Spell-Datenbank mit Klusteralgorithmen automatisch eine gute Gruppierung von Sprechern zu finden, waren nur bedingt erfolgreich [HW93b]. Wir beschränken uns deshalb im folgenden Experiment auf die natürliche Einteilung in männliche und weibliche Sprecher. Jedes Netzwerk enthält zwei Subkomponenten, die sich jeweils auf Frauen- und Männerstimmen spezialisieren. Für die unterschiedlichen Spezialisierungsgrade der 4 Netzwerke ergeben sich die in Tabelle 6.12 aufgelisteten Ergebnisse. Das Geschlecht eines Sprechers kann mit beinahe 100% korrekt automatisch erkannt werden, deshalb sind die Erkennungsraten für bekannte und identifizierte Sprecher nahezu identisch.

RM–Spell	Geschlecht bekannt	Geschlecht identifiziert
POOLED	90.8	
BIASED	90.4	90.4
SHARED	92.0	92.0
INDIVIDUAL	91.3	91.1

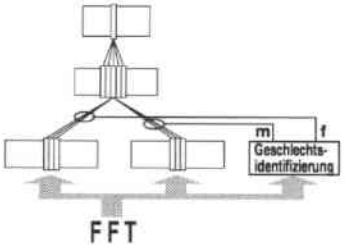


Tabelle 6.12: Buchstabenakkuratheit (kontinuierliche Sprache) auf den RM–Spell-Daten für die 4 unterschiedlichen Architekturen. Das MS-TDNN ist auf männliche und weibliche Sprecher spezialisiert, das Geschlecht des Sprechers wird durch ein weiteres Netz bestimmt.

6.9 Weitere Ergebnisse, Vergleiche

6.9.1 CMU–Alph-Daten

Neben den bereits in den vorausgehenden Experimenten getesteten Daten der Sprecher mjmt und mdbs umfaßt die CMU–Alph-Datenbank noch 4 weitere Sprecher, von denen für ein sprecherabhängiges Training ausreichend viele Daten vorliegen. In Tabelle 6.13 ist die Buchstabenakkuratheit für diese 6 Sprecher aufgelistet und mit verschiedenen Ergebnissen anderer Forscher verglichen.

Haffner beschreibt in [HFW91] seine MS-TDNN-Version und vergleicht sie mit Ergebnissen des SPHINX-Erkenners der Carnegie Mellon University (CMU).

Windheuser [WB93, PBW95] beschreibt Techniken, um optimale phonetische Ausgabedarstellungen für ein hybrides TDNN-HMM-System zu finden. Auf dem Sprecher

mjmt erzielt er bei bekannten Buchstabengrenzen (unterer Teil von Tabelle 6.13) 97.9% Buchstaben korrekt. Ebenfalls auf mjmt bei bekannten Buchstabengrenzen erreicht Iso [Iso92] 91.2% mit einem prädiktiven Ansatz (ähnlich den LPNNs, siehe Abschnitt 4.1.3).

Zwei weitere vergleichbare Ergebnisse auf den Sprechern mjmt[20] und mdbs finden sich in der Dissertation von Bodenhausen [Bod94], die sich hauptsächlich mit der automatischen Strukturierung von Netzwerkarchitekturen beschäftigt.

CMU–Alph-Daten, **kontinuierliche** Erkennung				
Sprecher	SPHINX [HFW91]	MS-TDNN Haffner [HFW91]	MS-TDNN Bodenhausen [Bod94]	MS-TDNN Hild
mjmt	96.0	97.5	97.4	**99.3**
mdbs	83.9	89.7	–	**95.3**
maem	–	–	–	**96.3**
fcaw	–	–	–	**98.8**
flgt	–	–	–	**86.9**
fee	–	–	–	**91.5**
CMU-Alph-Daten, **ausgeschnittene** Buchstaben				
	TDNN-HMM Windheuser [PBW95]	LPNN Iso [Iso92]	MS-TDNN Bodenhausen [Bod94]	MS-TDNN Hild
mjmt	97.9	91.2	–	**99.4**
mdbs	–	–	92.2	**98.5**

Tabelle 6.13: Sprecherabhängige Erkennung auf den englischen CMU–Alph-Daten

6.9.2 RM–Spell-Daten und SPHINX

In [HH92] wird das SPHINX-Spracherkennungssystem der CMU auf den RM–Spell-Daten getestet, um ein neues Konzept (Senone = subphonetische akustische Modellierung) zu erproben. Die dort veröffentlichten Ergebnisse sind in Tabelle 6.14 mit unseren verglichen.

Haffner erzielt in [HW92, Haf92] 94.3% korrekt auf ausgeschnittenen Buchstaben (entspricht unserem Ergebnis von 95.3% in Tabelle 6.5), macht wegen Problemen mit Einfügungen und Auslassungen aber keine Angaben über Ergebnisse bei kontinuierlicher Erkennung.

[20] Die in der Arbeit von Bodenhausen angegebenen 97.5% BA für das MS-TDNN des Autors wurden ohne Training auf Satzebene erzielt und sind deshalb um etwa 1.5 Prozentpunkte schlechter als das für Sprecher mjmt in Tabelle 6.13 mit 99.3% angegebene Ergebnis.

Resource-Management-Buchstabierdaten			
SPHINX [HH92]		MS-TDNN	
	+ Senone		geschlechtsspezifisch
88.7	90.4	**90.8**	**92.0**

Tabelle 6.14: Wortakkuratheit auf der sprecherunabhängigen RM-Spell-Task

6.9.3 Englische (OGI-)Telefondaten

Neben den mit 16 kHz aufgenommenen Daten (CMU-Alph, RM-Spell, KA-Alph) wurde der Buchstabiererkenner auch auf 8 kHz Telefondaten trainiert. In Kapitel 7 wird die Erkennung bei vorgegebenen Namenslisten unter anderem auf englischsprachigen Telefondaten untersucht. Im Vorgriff darauf sind in Tabelle 6.15 die Erkennungsergebnisse ohne Wortlisten auf der OGI-Telefondatenbank mit denen anderer Systeme (siehe Abschnitt 4.2) verglichen. Die OGI-Daten enthalten ohne (SLN) und mit Pausen (SLP) buchstabierte Namen und sind in Abschnitt 5.5, die experimentellen Rahmenbedingungen in Abschnitt 7.4.2 beschrieben.

Datenbank	SLN (ohne Pausen)		SLP (mit Pausen)	
Sprachmodell	ohne	Bigramme	ohne	Bigramme
MS-TDNN	88.2	91.0	90.6	92.3
OGI [FCR92]	-	-	87.0	-
Junqua [Jun97]	-	88.0	-	-
Spanias [LS96]	-	-	-	90.6

Tabelle 6.15: % BA auf der englischsprachigen OGI-Telefondatenbank

6.9.4 Deutsche Telefondaten

In einer Kooperation mit der Firma Siemens wurden das MS-TDNN und der Karlsruher JANUS-Erkenner (s.a. Abschnitt 8.2) auf deutschsprachigen Telefondaten getestet. Neben den beiden Erkennern sollten die Signalvorverarbeitungen aus Karlsruhe und von Siemens miteinander verglichen werden. Bei den Telefondaten handelt es sich um etwa 600 über das Telefon buchstabierte Namen, die in Kapitel 5 unter dem Namen BUCNAM beschrieben sind. Da diese Trainingsmenge sehr klein ist, wurde untersucht, inwieweit sich die wesentlich größere, aber mit 16 kHz und einem Nahbesprechungsmikrofon aufgenommene KA-Alph-Datenbank zum Trainieren eines Telefonerkenners nutzen läßt, wenn die Daten künstlich auf Telefonqualität transformiert werden. Mit einem Programm von Siemens wurden dazu die 10 000 Aufnahmen der KA-Alph-Daten von 16 kHz auf 8 kHz gewandelt, wobei auch die Frequenzcharakteristik des Telefonkanals sowie eine gröbere Quantisierung (8 statt 16 Bit pro Datenpunkt) simuliert wurden.

Das MS-TDNN wurde trainiert wie in den vorangehenden Abschnitten beschrieben. JANUS ist ein Sprachübersetzungssystem mit einem HMM-basierten kontinuierlichen Erkenner, der in Abschnitt 8.2 genauer beschrieben ist. JANUS kann natürlich auch Buchstaben erkennen, wenn sein Vokabular entsprechend spezifiziert wird. Mit einem auf der VERBMOBIL-Datenbank (Vokabular > 2000 Wörter) trainierten Erkenner wurden die Buchstabendaten phonetisch etikettiert. Damit konnte zuerst ein kontextunabhängiges und daraus ein kontextabhängiges HMM mit Wortmodellen trainiert werden[21].

Beide Erkenner wurden jeweils mit den 16 und 8 kHz Daten sowie der Vorverarbeitung aus Karlsruhe und von Siemens trainiert und getestet. Die Siemens-Vorverarbeitung nutzt ebenfalls Melscale-FFT-Koeffizienten, die im Frequenzbereich feiner aufgelöst sind (30 statt 16 Koeffizienten) und etwas anders normiert und berechnet werden (cepstrale Glättung). Aus den Ergebnissen in Tabelle 6.16 läßt sich ablesen, daß die unterschiedliche Vorverarbeitung keine signifikanten Auswirkungen hat. Je nach Aufgabe schneidet das MS-TDNN zwischen 1.5 und 3 Prozentpunkten besser ab. Gegenüber den 16 kHz Daten verliert das MS-TDNN etwa 2% und JANUS etwa 3% absolut.

Qualität	Vorverarbeitung	JANUS	MS-TDNN
16 kHz	Siemens	88.2	90.1
	Karlsruhe	88.6	90.0
8 kHz	Siemens	85.1	88.4
(künstlich)	Karlsruhe	86.0	88.2

Tabelle 6.16: Vergleichende Darstellung der beiden Erkenner

Die mit künstlichen Telefondaten trainierten Systeme sind auf entsprechenden Testdaten nicht wesentlich schlechter als die 16 kHz Systeme, zur Erkennung echter Telefondaten eignen sie sich jedoch nicht. Auf den 100 Testsätzen der BUCNAM-Daten erreicht das MS-TDNN mit der Siemens-Vorverarbeitung etwa 65%, die Karlsruher Vorverarbeitung erwies sich mit nur etwa 50% als weniger robust. In beiden Fällen lag das HMM auf den neuen Daten deutlich unter 50%. Ein auf der kleinen Trainingsmenge (< 500 Namen) der echten Telefondaten trainiertes MS-TDNN konnte immerhin 75% Erkennung auf der Testmenge erzielen[22].

6.9.5 Sonstige Datenbanken

Die Experimente von Reynolds und Tarassenko zur sprecherunabhängigen Buchstabiererkennung [RT92] wurden bereits in Abschnitt 4.1.1 beschrieben. Mit einer Trainings-

[21] An dieser Stelle einen herzlichen Dank an meinen Kollegen Martin Westphal, der sich etwa 2 Wochen Zeit nahm, um den JANUS-Erkenner auf der neuen Aufgabe zu optimieren.

[22] Die im Vergleich zu knapp 90% auf den englischen Telefondaten ziemlich schlechte Erkennung von nur 75% BA ist vermutlich in der kleinen Trainingsmenge der deutschen Telefondaten begründet.

und einer Testmenge von jeweils etwa 4000 isoliert gesprochenen Buchstaben wurde eine Erkennungsrate von 90% erreicht, die mit den etwa 95% Buchstaben korrekt des MS-TDNN auf der RM–Spell- oder der KA–Alph-Datenbank vergleichbar ist.

Ebenfalls dort erläutert sind die Experimente von Burr [Bur88a], der 85% bei sprecherabhängiger Erkennung erzielt, allerdings auf einer sehr kleinen Trainings- und Testmenge von 104 isolierten Buchstaben. In einem weiteren Experiment konnte durch eine Fokussierung auf den kritischen Anfang der Eingabe eine Erkennungsrate von 98.2% erzielt werden. Diese Ergebnisse können mit den sprecherabhängigen Erkennungsraten im unteren Teil von Tabelle 6.13 verglichen werden, die allerdings nicht auf isoliert, sondern auf kontinuierlich gesprochenen Buchstaben bei bekannten Grenzen erzielt wurden.

6.10 Zusammenfassung

Das *Multi-State-TDNN (MS-TDNN)* ist ein aus dem *Time-Delay Neural Network (TDNN)* entwickelter konnektionistischer Spracherkenner. Durch eine Verkettung der Phonemmodelle werden im MS-TDNN nicht (wie im TDNN) einzelne Phoneme, sondern Phonemsequenzen, also ganze Wörter bzw. in unserem Fall Buchstaben, erkannt. Zur zeitelastischen Anpassung der Buchstabenmodelle an die Spracheingabe ist der *Dynamic Time Warping (DTW)*-Algorithmus direkt in die MS-TDNN-Architektur integriert, wodurch diese diskriminativ auf Wort- und Satzebene trainiert werden kann.

Das MS-TDNN wird in drei Stufen trainiert und dabei schrittweise von der Phonemerkennung über die Erkennung von Einzelbuchstaben an die Erkennung von kontinuierlich gesprochenen Buchstabensequenzen herangeführt:

- Auf **Phonemebene** werden a posteriori Wahrscheinlichkeiten für eine Phonemklasse, gegeben ein Stück der Spracheingabe, gelernt. Die Wahl einer geeigneten Fehlerfunktion (Cross-Entropy- oder McClelland-Fehler) beschleunigt den Lernvorgang und erzielt bessere Ergebnisse als der konventionelle mittlere quadratische Fehler (MSE). Die a posteriori Phonemwahrscheinlichkeiten können direkt im MS-TDNN zur Buchstabenerkennung eingesetzt oder zu klassenbedingten Wahrscheinlichkeiten konvertiert und in einem hybriden TDNN-HMM-Ansatz genutzt werden. Dabei konnten mit dem hybriden TDNN-HMM-Ansatz bessere Ergebnisse erzielt werden als mit konnektionistischem MS-TDNN-Ansatz.

- Dieser Vorteil verliert sich jedoch auf **Wortebene**, wo der konnektionistische MS-TDNN-Ansatz ein direktes diskriminatives Training mit deutlicher Verbesserung der Erkennungsraten erlaubt. Da sich die Bewertungen der zu erkennenden Buchstaben aus gemeinsamen Phonemen speisen, ist ein Training auf „harte“ Sollwerte von 0 und 1 nicht sinnvoll. Vielmehr hat sich das CFM (Classification Figure of Merit)-Kriterium (oder gleichwertig GPD-Kriterium) bewährt, das nur den relativen Vorsprung der korrekten Ausgabe zum schärfsten Konkurrenten berücksichtigt. Auch bei der Präsentation der Muster im Training zahlt sich eine vorsichtige

Vorgehensweise aus, indem bereits sicher erkannte Muster nicht nochmals trainiert werden (*„Never change a winning team"*).

- Auf **Satzebene** sind die Grenzen der einzelnen Buchstaben nicht mehr bekannt, es werden kontinuierlich gesprochene Buchstabensequenzen erkannt. Zu Buchstabenverwechslungen kommen nun Einfügungen und Auslassungen als neue Fehlerquellen, die mit einer Längenmodellierung von Phonemen und Wörtern bekämpft werden. Mit Phonemübergangsstrafen und/oder einer phonemspezifischen Mindestdauer kann ein Großteil der hauptsächlich auftretenden Einfügungsfehler abgefangen werden. Schließlich kann auch auf Satzebene diskriminativ trainiert werden. Als Zwischenschritt zur völligen Freigabe aller Buchstabengrenzen hat sich das „Training über Wortgrenzen" bewährt.

Jede dieser Trainingsstufen und Modellierungsmaßnahmen wurde systematisch auf zwei sprecherabhängigen (mdbs, mjmt) und zwei sprecherunabhängigen (RM-Spell, KA-Alph) Datenbanken experimentell ausgewertet. Die wichtigsten der dabei erzielten Verbesserungen in der Erkennungsleistung sind in Tabelle 6.17 zusammengefaßt.

Training	RM-Spell	KA-Alph	mdbs	mjmt
% BK ausgeschnittene Einzelbuchstaben				
Training auf Phonemebene	89.4	89.9	97.1	99.2
+ hybrides NN-HMM	92.4	91.9	97.4	99.6
Training auf Wortebene	**95.3**	**94.0**	**98.5**	**99.4**
% BA kontinuierliche Erkennung				
Ohne Längenmodell	9.4	25.0	51.4	56.9
wortspez. Eingangsstrafe	84.7	84.8	90.9	91.6
phonemspez. Mindestdauer	88.4	87.8	95.0	96.4
Training auf Satzebene	89.6	**90.4**	94.8	98.2
+ Wortgrenzen-Training	**91.4**	89.7	**95.2**	**99.3**

Tabelle 6.17: Verbesserungen in der Erkennungsrate auf den Testmengen der vier Datenbanken durch die verschiedenen Stufen des Trainings

In der oberen Hälfte der Tabelle stehen die Ergebnisse nach dem Training auf Phonem- und auf Wortebene, gemessen in % Buchstaben korrekt bei bekannten Buchstabengrenzen (also ohne Einfügungs- und Auslassungsfehler). Der hybride TDNN-HMM-Ansatz schneidet im Training auf Phonemebene besser ab, konnte jedoch durch Training auf Wortebene nicht weiter verbessert werden. Im Gegensatz dazu gewinnt man mit dem MS-TDNN, abgesehen von den ohnehin bereits beinahe perfekt erkannten sprecherabhängigen Daten, mehrere Prozentpunkte durch das Training auf Wortebene.

In der unteren Hälfte der Tabelle sind die Ergebnisse für die kontinuierliche Erkennung, also inklusive Einfügungs- und Auslassungsfehlern, aufgelistet. Ganz offensichtlich kommt man ohne eine Längenmodellierung nicht aus. Mit einer phonemspezifischen

Mindestdauer oder wortspezifischen Eingangsstrafen können bereits durchaus zufriedenstellende Ergebnisse erreicht werden, die durch diskriminatives Training auf Satzebene weiter verbessert werden. Mit Ausnahme der KA–Alph-Daten führt das Training auf Satzebene zu besseren Ergebnissen, wenn zuvor das Wortgrenzen-Training durchgeführt wurde. Die schrittweise Verbesserung der einzelnen Trainingsstufen ist in graphischer Form am Beispiel der RM–Spell-Daten bereits in Abbildung 6.18 aufgezeigt.

Zur Dimensionierung des Netzes wurden zwei **Architekturparameter** untersucht: Als Anzahl benötigter Verbindungen (Gewichte) erweisen sich 50 verborgene Neuronen oder insgesamt etwa 20 000 Parameter bereits als ausreichend. Die Breite des Eingabekontextes sollte mehr als 3 Sprachvektoren (= 30 Millisekunden) betragen; die besten Ergebnisse werden mit einem zeitlichen Kontext von 5,7 oder 9 Vektoren erreicht.

Modulare Netzwerke erlauben eine Spezialisierung auf bestimmte Sprecher oder Sprechergruppen. In mehreren Experimenten konnten 6 Sprecher bei unbekannter Identität praktisch genausogut erkannt werden, wie wenn jeder Sprecher einzeln trainiert und getestet wurde. Allerdings ist diese Technik nur eingeschränkt auf sprecherunabhängige Erkennung übertragbar, da es schwierig ist, geeignete Aufteilungen der Sprecher (außer in weibliche und männliche) zu finden. Da die Modularisierung außerdem eine erhöhte Netz- und Trainingskomplexität bedingt, wurde dieser Ansatz nicht weitergeführt.

Hybride NN-HMM-Systeme berechnen mit neuronalen Netzen a posteriori Wahrscheinlichkeiten für Phoneme, die dann zu klassenbedingten Wahrscheinlichkeiten konvertiert und mit HMMs zeitlich verarbeitet werden. Von einer initialen Etikettierung der Daten abgesehen findet das Training der Netze unabhängig von der späteren Klassifikation im HMM statt. Im Gegensatz dazu ist im **MS-TDNN** die zeitliche Anpassung der Phonemsequenzen (basierend auf DTW, wie in einem HMM,) in die Netzarchitektur integriert. Dadurch kann das MS-TDNN direkt auf Wort- und Satzebene diskriminativ trainiert werden. Gegenüber hybriden Systemen, die nur auf Phonemebene diskriminativ trainiert werden, konnten damit bessere Ergebnisse erzielt werden. Ein gewisser Nachteil des direkten diskriminativen Trainings auf Wortebene ist jedoch, daß im MS-TDNN die Ausgaben der Phonemschicht nicht mehr als a posteriori Wahrscheinlichkeiten interpretiert werden können, wodurch eine mathematische Modellierung erschwert wird.

Vergleichssystem und Daten		MS-TDNN
JANUS (auf KA–Alph)	88.2	90.1
SPHINX (auf RM–Spell)	90.4	92.0
MS-TDNN Haffner (mjmt)	97.5	99.3

Tabelle 6.18: Vergleich des MS-TDNN mit anderen Erkennern (in % Buchstabenakkuratheit)

Die beschriebenen Techniken haben das MS-TDNN erstmals soweit verbessert, daß auch sprecherunabhängig kontinuierliche Buchstabensequenzen ausreichend gut erkannt werden können. Im **Vergleich** mit anderen Buchstabiererkennern hat das MS-TDNN konsistent sehr gut abgeschnitten. Die wichtigsten der in Abschnitt 6.9 beschriebenen Vergleiche sind in Tabelle 6.18 zusammengefaßt.

Kapitel 7

Sprachmodelle für Buchstabensequenzen

7.1 Einleitung

In den vorangegangenen Kapiteln ist gezeigt worden, wie die sprecherunabhängige Erkennungsgenauigkeit für kontinuierlich gesprochene Buchstaben durch Verbesserungen des MS-TDNN-Erkenners auf etwa 90% erhöht werden kann. Die Wahrscheinlichkeit, *alle* Buchstaben eines Namens richtig zu erkennen, ist natürlich geringer. Bei einer durchschnittlichen Wortlänge von 6 Buchstaben[1] kann man etwa $0.9^6 = 53\%$ korrekte Namen erwarten. Allerdings wurde zur Erzielung der bisher vorgestellten Ergebnisse keinerlei Wissen über die zu buchstabierenden Wörter (sogenannte Sprachmodelle) genutzt.

In diesem Kapitel wird beschrieben, wie die Erkennungsleistung durch den Einsatz mächtiger Sprachmodelle entscheidend verbessert werden kann. Insbesondere wenn zur Erkennung ausschließlich „legale" Namen erlaubt werden, können Namenserkennungsraten erzielt werden, die einen praktischen Einsatz möglich erscheinen lassen. In entsprechenden Experimenten wurden Nachnamen aus 14 Millionen Telefoneinträgen der Nordostküste der USA in Echtzeit mit knapp 90% korrekt erkannt.

Dabei sind die beiden wesentlichen Fragen, **welche Einschränkungen** für das Sprachmodell gewählt und **wo** bzw. **wie** diese im Erkenner eingesetzt werden. Unter Einschränkungen des Sprachmodells verstehen wir im weiteren Sinne alle Maßnahmen, die die Suche auf syntaktischer Ebene beeinflussen, z.B. konventionelle Bi- oder Trigramme, die unterschiedliche Buchstabenpaare bzw. -tripel mit unterschiedlichen Wahrscheinlichkeiten bewerten, oder härtere Restriktionen, die von vorneherein nur bestimmte Buchstabensequenzen erlauben. Weiterhin können diese Einschränkungen an unterschiedlichen Stellen im Erkennungsprozeß zum Tragen kommen.

[1] Die durchschnittliche Länge eines Karlsruher Nachnamens liegt bei 6.6, die der amerikanischen Nachnamen bei 6.5 Buchstaben.

Unter diesen Gesichtspunkten werden Verfahren basierend auf Nächster-Nachbar-Suche, N-Besten-Listen, N-Grammen sowie Graph- bzw. Baumsuche entworfen und experimentell ausgewertet. Als Testmenge für einen Vergleich der Methoden dienen etwa 1300 Aufnahmen aus der Karlsruher Buchstabierdatenbank. Diese Namen sind zufällig ausgewählt aus einer Liste von über 100 000 Nachnamen, auf die die Erkennung eingeschränkt werden soll. Als beste und gleichzeitig konzeptionell einfachste Methode stellt sich die Baumsuche heraus. Ihr wird deshalb ein eigener Abschnitt gewidmet, in dem Verfeinerungen der Suchtechnik auf einer zweiten und schwierigeren Aufgabenstellung, der Erkennung von über das Telefon buchstabierten Namen, exploriert werden. Als Sprachdatenbank wird dazu der öffentlich verfügbare und auch von anderen Gruppen genutzte „OGI Spoken and Spelled Telephone"-Korpus eingesetzt, wobei zur Erkennung Listen mit einem Umfang von bis zu 14 Millionen Namen zugrunde gelegt werden. Dabei hat es sich als hilfreich erwiesen, nicht nur die Liste gegebener Namen, sondern auch deren Auftrittshäufigkeiten zu nutzen.

Die erläuterten Verfahren erhöhen die Robustheit der Erkennung deutlich, arbeiten aber nach wie vor ausschließlich auf Buchstabensequenzen. Buchstabiereffekte, wie sie in spontansprachlichen Situationen auftreten, werden im nächsten Kapitel diskutiert.

7.2 Sprachmodelle und Buchstabieren

Die Sprachmodelle praktisch aller kontinuierlichen Spracherkenner mit größeren Vokabularen basieren auf den in Abschnitt 3.6 beschriebenen statistischen N-Gramm-Modellen. Deren Reichweite ist beschränkt, und die mit N exponentiell anwachsende Zahl möglicher N-Gramme erlaubt höchstens die Modellierung von Bi- oder Trigrammen. Aufgrund seines kleinen Vokabulars läßt das Alphabet weiterreichendere Modelle zu. Für viele Buchstabieranwendungen ist es darüber hinaus sinnvoll, die zu erkennenden Buchstabensequenzen auf eine Liste gegebener Namen einzuschränken. Nützliche Größen für solche Listen schwanken zwischen wenigen Hundert und Millionen von Namen, je nachdem ob es sich um die interne Telefonauskunft einer kleinen Firma oder die Auskunft für eine Großstadt handelt.

Neben dem Grad der Einschränkungen stellt sich als zweite Frage, **wo** die Einschränkungen des Sprachmodells zum Tragen kommen. In Abbildung 7.1 sind zwei Ansatzpunkte illustriert:

- **Nachbearbeitung bereits erkannter Hypothesen.** Ist die erkannte Hypothese h „illegal", kann man in der Liste erlaubter Namen die zu h ähnlichste Hypothese suchen. Alternativ kann der Erkenner in Form einer N-Besten-Liste mehrere Vorschläge generieren, unter denen dann nach einer legalen Hypothese gesucht wird.

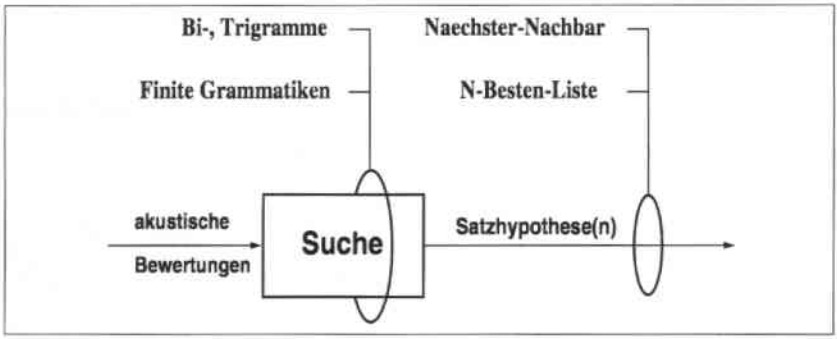


Abbildung 7.1: Einschränkungen im Suchprozeß oder nachgeschaltet

- **Direkte Integration der Einschränkungen in die Suche.** Die in klassischen Bigramm-Sprachmodellen gegebenen Wahrscheinlichkeiten werden direkt im Suchprozeß bei allen potentiellen Buchstabenübergängen verrechnet, können aber nur den jeweils letzten Buchstaben berücksichtigen. Stärkere Einschränkungen erlauben nur die Erkennung von Namen aus einer gegebenen Liste. Dies kann mit einer Suche auf einem Graphen realisiert werden, in dessen Zuständen die bis dahin durchlaufene Suchgeschichte kodiert wird.

Ähnliche Fragestellungen ergeben sich auch bei „konventionellen" Erkennern für große Vokabulare. Bigramme können direkt beim Übergang von einem Wort in das nächste berücksichtigt werden. Bei einem Erkennungsvokabular von 1000 Wörtern fallen zu jedem Zeitpunkt 1000 mal 1000 mögliche Wortübergänge an, die nur mit heuristischen Techniken wie Strahlsuche effizient durchsucht werden können. Mit Trigrammen erhöht sich die Zahl möglicher Wortübergänge um einen weiteren Faktor 1000, gleichzeitig wird die Verwaltung der partiellen Hypothesen komplexer, da diese nicht nur für jedes letzte Wort, sondern für alle letzten Wortpaare unterschiedlich sein können. Aus diesen Gründen werden Trigramme meist nicht direkt in der Suche, sondern in einem Nachverarbeitungsschritt eingesetzt, um einen bereits mit Bigrammen erzeugten Worthypothesengraphen neu zu bewerten.

7.3 Vergleich von Suchmethoden

7.3.1 Experimentelle Rahmenbedingungen

Bei den im folgenden beschriebenen, vergleichenden Experimenten wird die Suche auf eine Liste von insgesamt 111 882 (bzw. 32 267 unterschiedlichen) Karlsruher Nachna-

men eingeschränkt. Die Buchstabierung eines gegebenen Namens ist leider nicht immer eindeutig, sondern kann variiert werden. Neben den in Abschnitt 5.2 erwähnten drei bzw. zwei Aussprachemöglichkeiten für „ß" (Eszett, Scharf-S, Scharfes-S) und „-" (Strich, Bindestrich) werden gelegentlich Doppelbuchstaben nicht explizit, sondern mit „`doppel`" buchstabiert, also z.B. „`M A doppel N`". Unter Berücksichtigung aller dieser Varianten vergrößert sich die Menge der 32 267 unterschiedlichen Nachnamen auf eine Menge $S = \{s_1, s_2, \ldots\}$ von insgesamt $|S| = 43\,181$ Namen.

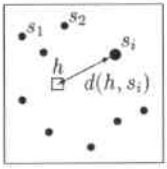
Der Erkenner wurde mit insgesamt 8 133 Namen (55 449 Buchstaben) von 70 Sprechern trainiert und entspricht dem in Kapitel 6 vorgestellten, auf den KA–Alph-Daten trainierten System. Die Testmenge umfaßt 1 316 Nachnamen (8 661 Buchstaben) aus S, buchstabiert von 23 weiteren Sprechern. Die Sprachdaten sind mit 16 kHz über ein Sennheiser-Nahbesprechungsmikrofon aufgenommen.

Das Ausgangssystem ohne Sprachmodell erzielt auf den 1316 Aufnahmen eine Buchstabenakkuratheit von 90.1% und eine Namensakkuratheit von 56.4%. In den folgenden Experimenten werden verschiedene Suchtechniken verglichen, bei ansonsten unverändertem Erkenner.

7.3.2 Nächster-Nachbar

Hier wie bei den N-Besten-Listen des nächsten Abschnittes werden die Einschränkungen erst nach dem eigentlichen Erkennungsprozeß auf syntaktischer Ebene wirksam. Ziel ist, für einen möglicherweise falsch erkannten Namen den besten Vertreter aus der Liste S der erlaubten Namen zu finden: Für eine erkannte Hypothese h soll der ähnlichste Name oder String $s^* \in S = \{s_1, s_2, \ldots\}$ gefunden werden, also

$$s^* = \underset{s_i \in S}{\operatorname{argmin}} \{d(h, s_i)\}$$



Dabei ist $d(s_i, s_j)$ das Abstandsmaß zwischen zwei Namen s_i und s_j. Ein direkter Vergleich Buchstabe für Buchstabe scheidet aus, denn nach diesem Maß wären beispielsweise die beiden Sequenzen „`T-I-S-C-H`" und „`T-E-I-S-C-H`" nur an einer Position identisch. Vielmehr muß genau wie bei der Bestimmung der Wortakkuratheit in Abschnitt 3.5.2 die minimale Anzahl von Einfügungen, Auslassungen und Verwechslungen gefunden werden, die die beiden Sequenzen unterscheidet, die sogenannte „Editierdistanz".

Zusätzlich kann man ausnutzen, daß manche Buchstabenpaare in ihrer akustischen Realisierung einander ähnlicher, d.h. mißverständlicher sind als andere. Beispielsweise wird `B` eher mit `D` als mit `X` verwechselt, und entsprechend sollte der Buchstabenabstand

	a	ae	b	c	d	e	f	g	h	i	j	k	l	m	n
a	3340	.	1	.	.	.	.	.	9	.	.	2	.	.	.
ae	.	391	2	.	.	36	2	.	.	.	.	.	19	1	2
b	.	1	1696	.	11	14	1	.	.	.	.	.	.	.	.
c	.	.	.	2005	.	.	.	3	.	.	.	.	1	.	.
d	.	1	40	4	1586	19	.	31	.	1	.	.	1	.	.
e	.	10	16	.	2	5465	.	4	.	66	.	.	3	.	2
f	.	.	.	.	.	.	1223	.	.	.	1	.	.	.	.
g	.	.	4	.	28	23	.	1653	.	8	.	.	.	.	2
h	25	.	.	1	.	.	.	.	2665	.	.	19	.	.	.
i	.	.	.	.	.	95	.	4	.	2822	.	.	.	.	.
j	.	.	.	.	.	.	.	.	.	.	619	.	.	.	.
k	5	.	.	.	.	.	.	.	4	.	.	1724	.	.	.
l	.	5	1	.	.	9	.	1	.	.	.	.	2841	3	23
m	.	2	.	.	.	1	.	.	.	1	.	.	7	1637	107
n	.	1	.	.	.	5	1	.	.	.	.	.	19	47	3438

Tabelle 7.1: Ausschnitt aus einer Konfusionsmatrix

$d_l(\mathtt{B},\mathtt{D})$ kleiner als $d_l(\mathtt{B},\mathtt{X})$ gewählt werden. Statt nun also zwei Buchstaben L_i und L_j nur als gleich (Abstand 0) oder ungleich (Abstand 1) zu bewerten, d.h. ein Buchstabenabstandsmaß

$$d_l(L_i, L_j) := \delta_{ij} \tag{7.1}$$

zu benutzen, definieren wir ein Abstandsmaß, das die Ähnlichkeit der Buchstaben berücksichtigt:

$$d_l(L_i, L_j) := 1 - P(L_i|L_j) \tag{7.2}$$

Dabei ist $P(L_i|L_j)$ die Wahrscheinlichkeit, daß der korrekte Buchstabe L_i ist, wenn L_j erkannt wurde. Diese Größen können direkt aus einem Testlauf auf dem Trainingsmaterial geschätzt werden, indem eine Statistik über die Verwechslungen zwischen allen Buchstabenpaaren geführt wird. Das Beispiel in Tabelle 7.1 zeigt einen Ausschnitt aus einer solchen Konfusionsmatrix.

Mit den „harten" 0/1 Strafen (Gl. (7.1)) wurden 75.6% der Namen korrekt (NK) erkannt, bei einer Buchstabenakkuratheit (BA) von 92.8%. Wird die auf dem Trainingsmaterial geschätzte Verwechselbarkeit gemäß Gl. (7.2) berücksichtigt, verbessern sich die Ergebnisse auf 85.0% NK bzw. 95.2% BA.

7.3.3 N-Besten-Listen

Werden vom Erkenner gleich mehrere Hypothesen berechnet und sind diese nach ihren (vom Erkenner bestimmten) Wahrscheinlichkeiten sortiert, spricht man von N-Besten-

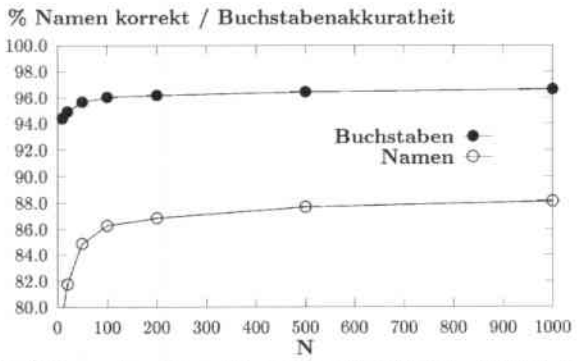


Abbildung 7.2: % Namen korrekt und Buchstabenakkuratheit in Abhängigkeit der Größe der N-Besten-Liste

Listen. Statt die beste Hypothese h_1 auf einen der gegebenen Namen abzubilden, kann man auch solange nach weiteren Hypothesen h_i fragen, bis eine legale gefunden wird. Sei also $H = (h_1, h_2, \ldots, h_N)$ die Liste der N besten Hypothesen, sortiert mit der besten Hypothese an erster Stelle. Dann wählen wir die Hypothese h^* als

$$h^* = \begin{cases} h_1 & \text{falls } H \cap S = \emptyset \\ h_{i^*}, \quad i^* = min\{i | h_i \in S\} & \text{sonst} \end{cases}$$

Kann keine der Hypothesen h_i in der Liste legaler Namen S gefunden werden, wählen wir (unabhängig von S) die Hypothese mit der besten Bewertung[2].

In den meisten Fällen (60.7%) ist bereits der (erst)beste Name legal, davon sind jedoch 5% inkorrekt. Der Anteil legaler, aber falsch erkannter Hypothesen steigt erwartungsgemäß in den unteren Positionen der Liste. Beispielsweise ist für $N = 100$ in 5.1% aller Fälle in der ganzen N-Besten-Liste kein legaler Eintrag. Dieser Zusammenhang ist in Abbildung 7.3 für alle Listenpositionen dargestellt. Naturgemäß steigt mit größeren N-Besten-Listen die Wahrscheinlichkeit, daß mindestens ein legaler Name gefunden wird. Zwar sinkt in den unteren Listenplätzen die Wahrscheinlichkeit dafür, daß ein legaler Name auch der korrekte ist. Dennoch dominieren die korrekten Namen, so daß die Erkennungsrate mit größeren N-Besten-Listen kontinuierlich ansteigt: Für $N = 50$ erhält man 85% der Namen korrekt. Spätestens bei $N = 500$ sättigt sich der Wert bei 88%, wie aus Abbildung 7.2 ersichtlich ist.

In [JLMG93, JLM93] (siehe Abschnitt 7.5) wird ein Verfahren vorgestellt, das nur kleine N-Besten-Listen berechnet, diese aber anschließend neu bewertet.

[2] Wird selbst in einer großen N-Besten-Liste kein legaler Name gefunden, könnte man dies als Rückweisung interpretieren, d.h. man hypothetisiert, daß der gesprochene Name nicht in der Liste legaler Namen vertreten ist. Alternativ könnte der Name, wie im vorangehenden Abschnitt, auf einen legalen Namen abgebildet werden – mit dieser Möglichkeit wurde jedoch nicht experimentiert.

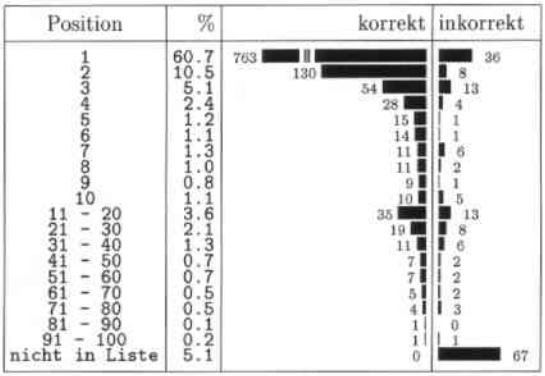


Abbildung 7.3: Das Histogramm zeigt, wie häufig ein legaler Name an erster, zweiter etc. Position in der N-Besten-Liste gefunden wurde und wie häufig dieser korrekt war

7.3.4 Einschränkungen direkt im Suchprozeß

In den beiden bislang vorgestellten Fällen wurden die gegebenen Einschränkungen auf syntaktischer Ebene durch Nachverarbeitungsschritte erzwungen. Die Einschränkungen können aber auch schon früher, nämlich direkt im Suchprozeß eingebracht werden. Die klassische Technik hierfür sind Bi- oder Trigramme, die allerdings nur ein schwaches Modell sind, da sie ja nur einen einzigen bzw. zwei Buchstaben zurückblicken können. Will man die Wahl des nächsten Buchstabens von *allen* bisher erkannten Buchstaben abhängig machen, muß die Suchgeschichte aller partiellen Hypothesen gespeichert werden. Dazu kann man die Suche als einen Durchlauf durch einen Graphen organisieren, in dem genau alle zu erkennenden Sequenzen repräsentiert sind.

Im Suchprozeß ist jeder zu erkennende Buchstabe L_i durch ein akustisches Wortmodell λ_i repräsentiert. In einer konventionellen Suche gibt es für jedes L_i genau eine Instanz des Modells λ_i. Diese sind, wie in Abbildung 7.4(links) illustriert, parallelgeschaltet und rückgekoppelt. Wenn die Suche das Ende eines Modells erreicht, springt sie zurück und kann in jedem der Wortmodelle neu beginnen. Entsprechend ist bei einem Buchstabenübergang auch nur der direkte Vorgänger bekannt.

Um auf die gesamte Suchgeschichte zurückzugreifen, kann man einen Graphen konstruieren, der explizit genau die Menge der zu erkennenden Buchstabensequenzen kodiert, wie in Abbildung 7.4(a)-(c) am Beispiel der Namen „B O B", „B O Y" und „B A Y" illustriert. Eine naive Konstruktion des Graphen erzeugt für jeden Namen eine eigene Sequenz von Buchstaben (Beispiel (a) in Abbildung 7.4). Gemeinsame Anfänge können zu einem Baum zusammengefaßt werden (b). Die ökonomischste Lösung ist ein minimaler Graph, bei dem auch noch gemeinsame Enden geeignet zusammengefaßt sind (c).

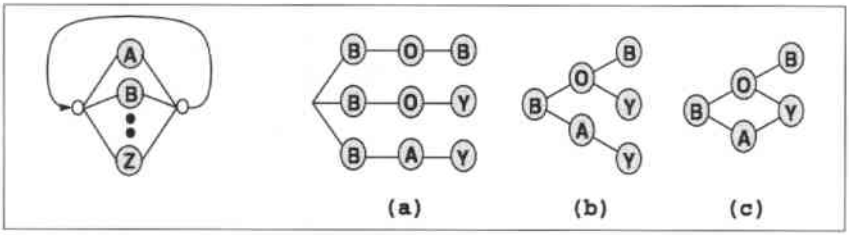


Abbildung 7.4: Konventionelle Suche (links) benötigt genau eine Instanz von jedem Wortmodell. Die drei rechten Graphen repräsentieren jeweils drei Namen als Liste (a), Baum (b) oder minimalen Graphen (c). Dazu werden Duplikate der Buchstabenmodelle benötigt.

Die Suche entspricht dann einer Traversierung des Graphen: In einem mit Buchstabe L_i markierten Knoten wird das entsprechende akustische Modell λ_i durchlaufen, und die Kanten definieren die erlaubten Übergänge in nachfolgende Modelle. Die Suchgeschichte, d.h. die bisher erkannte Buchstabensequenz, ist implizit in der Position des aktuellen Knotens kodiert.

Derselbe Buchstabe L_i kann im Graphen mehrfach an unterschiedlichen Positionen vorkommen, beispielsweise sind im Baum in Abbildung 7.4(b) die Buchstaben B und Y je zweimal vertreten. Da sich die Suchkontexte und damit die bis an diese Stelle akkumulierten Suchbewertungen unterscheiden, muß für jedes Vorkommen der individuellen $L_i^1, L_i^2, \ldots$ eine eigene Kopie $\lambda_i^1, \lambda_i^2, \ldots$ des entsprechenden Modells im Speicher gehalten werden. Jeder Knoten verbraucht Speicherplatz und Rechenzeit – möglichst kleine Strukturen müssen also bevorzugt werden. Das Einsparungspotential wird an folgendem Beispiel deutlich: Die für den Vergleich der Suchverfahren eingesetzte Namensliste enthält 43 181 unterschiedliche Namen mit 345 570 Buchstaben – dieselbe Anzahl von Knoten erfordert ein Graph in Listenform. Als Baum repräsentiert werden 141 066, als minimaler Graph nur 57 713 Knoten benötigt.

Ein minimaler Graph für die Namen „`Hof, Hose, Hase, Hast`“ ist in Abbildung 7.5 zusammen mit einer entsprechenden Suchmatrix abgebildet. Mit dem gegebenen Graphen kann der Suchpfad nur im akustischen Modell für H beginnen und von dort nur nach A oder O springen. Das E am Ende von „`Hase`“ und „`Hose`“ kann zusammengelegt werden – in diesem akustischen Modell werden dann die beiden Pfade für Hase und Hose konkurrieren. Das S jedoch muß dupliziert werden, sonst könnte ein nicht erlaubter Pfad „`H O S T`“ entstehen.

7.3.5 Zweistufige Suche im minimalen Graphen

In dem kleinen Beispiel in Abbildung 7.5 muß nur ein einziger Buchstabe „S“ dupliziert werden. Der minimale Graph für alle 43 000 Namen des Karlsruher Telefonbuchs hat

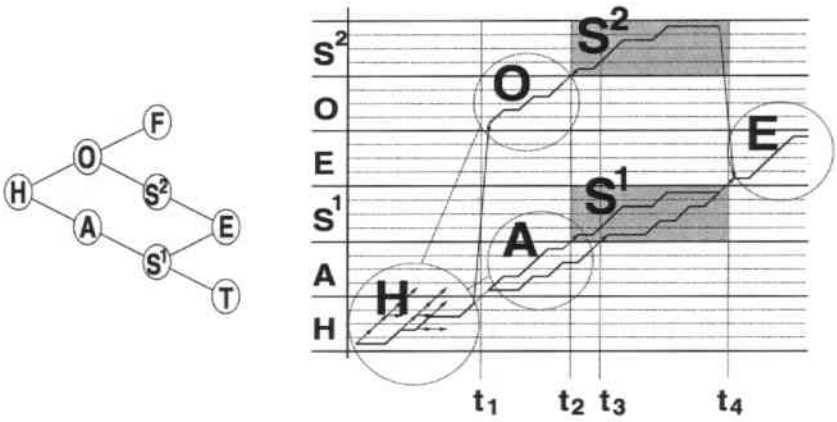


Abbildung 7.5: Ein minimaler Graph für die Namen „`Hof, Hose, Hase, Hast`" sowie eine entsprechende Suchmatrix mit Suchpfaden für „`Hose`" und „`Hase`"

jedoch bereits über 57 000 Knoten und benötigt eine entsprechend große Suchmatrix, die beispielsweise über 5000 Duplikate des Modells für E enthält.

Diese Kopien sind notwendig, um die individuell unterschiedlichen Suchgeschichten der einzelnen partiellen Pfade zu verwalten – für gleiche Ein- und Austrittszeitpunkte berechnen sich die Pfade innerhalb eines Buchstabenmodells jedoch völlig identisch. Im Beispiel in Abbildung 7.5 sind zwei mögliche Pfade durch das Modell S^1 von Buchstabe „S" gezeigt. Der obere beginnt bei t_2 und endet bei t_4, genauso wie ein dritter Pfad durch S^2. Zwar haben die beiden letzteren Pfade unterschiedliche Vorgeschichten und damit unterschiedliche akkumulierte Bewertungen – die lokale Bewertung für die Traversierung von S^1 und S^2 ist jedoch identisch und eine doppelte Berechnung ist eigentlich nicht erforderlich.

Diese Tatsache motivierte die Entwicklung der im folgenden beschriebenen, zweistufigen Suchtechnik[3]. Die prinzipielle Idee dabei ist, die (vom aktuellen Suchkontext unabhängige) Berechnung der Suchpfade innerhalb eines akustischen Buchstabenmodells von der Verwaltung der Suchgeschichte zu trennen. Statt in 5000 Duplikaten muß eine Traversierung durch ein E nur noch in einem Modell berechnet werden. Dafür benötigt man aber eine aufwendigere Verwaltung der partiellen Suchpfade. Dieses Konzept ist in Ab-

[3]Eine ähnliche Technik wurde bereits 1979 von Sakoe unter dem Namen „Two-Level DP-Matching" [Sak79] vorgeschlagen. Für die Suche in „normalen" Wortschätzen (die keine große Anzahl identischer Kopien besitzen) hat das „Two-Level DP-Matching" jedoch keinen Vorteil gegenüber dem konzeptionell einfacheren „One-Stage-DP-matching" [Ney84], das sich seit langem als Standard-Suchverfahren durchgesetzt hat.

bildung 7.6 visualisiert. In der Suchmatrix gibt es nur noch ein akustisches Modell pro Buchstabe. Sei t der aktuelle Zeitpunkt. In der ersten Stufe werden unabhängig vom Suchkontext für jeden Buchstaben L_i lokale Suchpfade π_{i,t_e} durch das Modell von L_i berechnet, die zum Zeitpunkt t beginnen und zu verschiedenen Zeitpunkten t_e enden können. Dabei werden mit $t_e \in \{t + mindur(L_i), \ldots, t + maxdur(L_i)\}$ alle sinnvollen Längen von L_i betrachtet.

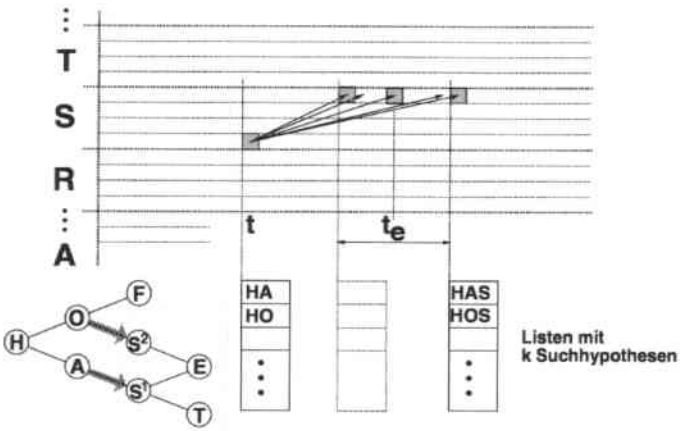


Abbildung 7.6: In der zweistufigen Suche gibt es nur noch ein akustisches Modell für jeden Buchstaben. Die lokalen Pfade innerhalb eines Buchstabenmodells werden in einer ersten Stufe vorausberechnet, und mit deren Bewertungen werden in der zweiten Stufe die Suchhypothesen gemäß der im Graphen gegebenen Struktur fortgesetzt.

In der zweiten Stufe werden die Suchpfade gemäß der im Graphen gegebenen Struktur fortgesetzt. Zum Zeitpunkt t ist die bisherige Suchgeschichte in einer Liste mit k Suchhypothesen gespeichert, wobei eine Suchhypothese aus einem partiellen Suchpfad (repräsentiert von einem bestimmten Knoten im Graphen) sowie dessen Bewertung besteht. Eine Suchhypothese wird verlängert, indem alle im Graphen erlaubten Nachfolge-Buchstaben des aktuellen Suchknotens durchquert werden. Dabei kann auf die bereits in der ersten Stufe vorausberechneten Pfade π_{i,t_e} zurückgegriffen werden, so daß die neuen Suchhypothesen zu den Zeitpunkten t_e direkt aus den alten Hypothesen sowie den Bewertungen der lokalen Pfade π_{i,t_e} bestimmt werden können. Im Beispiel in Abbildung 7.6 werden die beiden Hypothesen „H-A“ und „H-O“ zu „H-A-S“ und „H-O-S“ erweitert, indem jeweils das „S“ durchsprungen wird.

Es ist ausreichend, zu jedem Zeitpunkt etwa $k \geq 40$ aktive Hypothesen zu halten; ab $k < 20$ wird ein signifikanter Erkennungsverlust beobachtet. Die zweistufige Suche erzielt 92.7% Namen korrekt bei einer Buchstabenakkuratheit von 97.7% und schneidet damit

deutlich besser ab als die bisher vorgestellten Verfahren. Weitere Details zur zweistufigen Suche finden sich in der Diplomarbeit von Betz [Bet94] sowie in [BH95a].

7.3.6 Suche in Bäumen

Trotz der deutlich höheren Knotenzahl, die ein Baum gegenüber einem minimalen Graphen aufweist (141 066 gegenüber 57 713 Knoten für die Namen des Karlsruher Telefonbuches), konnte letztlich auf einer Baumstruktur die effizienteste und zugleich konzeptionell einfachste vollständig eingeschränkte Suche realisiert werden. In Verbindung mit einem neuen Konzept, der Berücksichtigung der a priori Wahrscheinlichkeiten einzelner Namen, wird die Baumsuche in Abschnitt 7.4 detaillierter beschrieben und anhand zusätzlicher Experimente auf einer zweiten Aufgabenstellung dokumentiert. Hier sollen unkommentiert die mit den Karlsruher Nachnamen erzielten Ergebnisse erwähnt werden: Ob Baum oder Graph – die Einschränkungen sind dieselben, und entsprechend unterscheidet sich die Baumsuche mit einer Namenserkennungsrate von 92.6% praktisch nicht von der Graphensuche. Werden zusätzlich die a priori Wahrscheinlichkeiten der einzelnen Namen genutzt, erreicht man mit der Baumsuche 94.1% korrekte Namen.

7.3.7 Vergleich der Ergebnisse

In den zuletzt betrachteten Experimenten werden verschiedene Methoden zur Einschränkung des Suchraums verglichen. Ganz ohne Sprachmodell erkennt man etwa 90% der Buchstaben korrekt, was in einer niedrigen Namenserkennung von knapp 60% resultiert. Konventionelle Bigramme können diesen Wert um 6 Prozentpunkte verbessern – ein nur kleiner Gewinn im Vergleich zu den anderen Methoden. Dafür sind Bigramme aber nicht strikt auf die gegebene Wortliste eingeschränkt, sondern können prinzipiell beliebige Buchstabensequenzen erkennen.

Die „**Nächster-Nachbar**-Methode" vergleicht eine bereits erkannte Hypothese mit allen legalen Kandidaten und wählt den ähnlichsten. Wird die Verwechslungsähnlichkeit von Buchstabenpaaren berücksichtigt, kann der Anteil korrekt erkannter Namen (% NK) von 75% auf 85% gesteigert werden. Weiterhin kann der Erkenner eine ganze ***N*-Besten-Liste** von Hypothesen erzeugen, um daraus die am besten bewertete legale Hypothese zu wählen. Für ausreichend große n (> 500) werden damit 88% NK erzielt.

Besser als eine syntaktische Nachbearbeitung bereits erkannter Hypothesen ist jedoch, die vollen Einschränkungen bereits zu einem früheren Zeitpunkt, nämlich **direkt im Suchprozeß**, einzusetzen. Dazu werden die zu erkennenden Namen in Form eines Graphen kodiert, der dann mit verschiedenen Techniken durchsucht werden kann. In den Graphen gibt es nur 26 unterschiedliche akustische Modelle, die sich aber tausendfach wiederholen. Dies wird von dem zweistufigen Suchverfahren ausgenutzt, indem die Verwaltung der Suchhypothesen von der Berechnung der Buchstabendurchgänge entkoppelt wird.

Die Ergebnisse der unterschiedlichen Methoden sind einander in Tabelle 7.2 gegenübergestellt, zusammen mit der zur Erkennung eines Namens[4] benötigten Zeit[5]. Bedingt durch die erforderliche Nachverarbeitung oder die aufwendigere Suche ist eine Erkennung mit Sprachmodellen deutlich zeitintensiver als ohne, aber immer noch in Echtzeit möglich.

	Sek./Name	% BA	% NK
Ohne Sprachmodell	0.4	90.1	56.4
Bigramme	–	92.0	62.1
Nächster Nachbar (0/1)	3.5	92.8	75.6
Nächster Nachbar (p)	3.5	95.2	85.0
N-Besten-Listen	7.0	96.1	88.1
minimaler Graph, zweistufig	3.2	97.7	92.7
Baum	2.6	98.2	92.6
Baum + Wahrscheinlichkeiten	1.6	98.5	94.1

Tabelle 7.2: Vergleich verschiedener Methoden zur Einschränkung des Sprachmodells auf eine Liste von Namen

7.4 Baumsuche

Zur Repräsentation einer Liste von Namen benötigt ein Graph in Baumstruktur im allgemeinen mehr als doppelt so viele Knoten wie ein minimaler Graph. Der entscheidende Vorteil eines Baumes ist jedoch, daß jedem Knoten ein eindeutiger Pfad zugeordnet ist und somit die Notwendigkeit einer Rückwärtsverzeigerung entfällt[6]. Gleichzeitig kann mittels einer Strahlsuche die Anzahl der zu durchsuchenden Knoten so stark reduziert werden, daß die Vorteile der in Abschnitt 7.3.5 beschriebenen zweistufigen Suche nicht mehr wirksam werden. Da keine Rückverzeigerungen gespeichert werden müssen, können alte, nicht mehr aktive Knoten eliminiert werden. Es ergibt sich so eine konzeptionell einfache Suche, die sehr effizient implementiert werden kann, wie in Abbildung 7.7 angedeutet: Jedem Knoten ist ein Buchstabe bzw. ein Duplikat des entsprechenden akustischen Modells zugeordnet. Außerdem repräsentiert jeder Knoten K eine Suchhypothese, nämlich genau die Buchstabensequenz vom Wurzelknoten bis zu K. Die Phonembewertungen des nächsten Zeitschrittes t werden nun allen (aktiven) Knoten zur Verfügung gestellt, woraus nach dem Prinzip der dynamischen Programmierung ein neuer, optimaler Pfad für jeden Zustand des akustischen Modells berechnet werden kann. Weiterhin konkurriert der Endzustand eines Knotens mit den Bewertungen der Anfangszustände

[4] Die durchschnittliche Länge der buchstabierten Namen liegt bei 3.5 Sekunden.

[5] Auf einer HP-735-Workstation

[6] In einem minimalen Graphen werden Kanten wieder zusammengeführt, wodurch auch gemeinsame Namensenden in Subgraphen zusammengefaßt werden. Ein Pfad durch einen solchen Graphen kann nur mit einer entsprechenden Rückwärtsverzeigerung rekonstruiert werden.

aller nachfolgenden Knoten. Zu jedem Zeitpunkt wird die wahrscheinlichste Buchstabensequenz durch den Knoten mit der besten Bewertung seines Endzustandes definiert. Somit braucht in jedem Knoten lediglich die jeweils akkumulierte Bewertung, also nur ein skalarer Wert pro Zustand des entsprechenden akustischen Modells λ_i, gespeichert zu werden.

Der größte in unseren Experimenten eingesetzte Baum repräsentiert 800 000 Namen mit etwa 2 Millionen Knoten – zuviele, um vollständig durchsucht zu werden. Deshalb wird eine Strahlsuche (*beam search*) eingesetzt, bei der die Suche entlang derjenigen Knoten nicht mehr weiterverfolgt wird, die in ihrer Bewertung zu weit hinter den besten Knoten zurückfallen. Ein Knoten K bleibt also nur aktiv, solange seine aktuelle Bewertung $s(K)$ innerhalb des durch den besten Knoten K^* und die Strahlbreite b definierten Bereiches liegt, d.h. falls gilt:

$$s(K) > s(K^*) - b$$

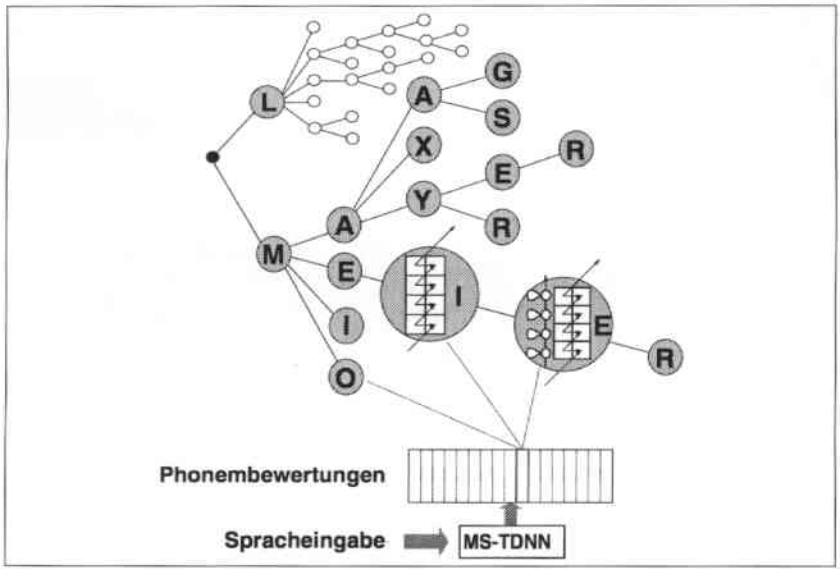


Abbildung 7.7: Baumsuche: Jeder Knoten entspricht einer Suchhypothese. Mit den aktuellen Phonembewertungen werden in jedem Zeitschritt neue akkumulierte Bewertungen der Suchpfade innerhalb der Knoten berechnet.

7.4.1 Wahrscheinlichkeitsannotierte Bäume

Die Einschränkung der Suche auf eine gegebene Namensliste ist eine starke Beschränkung, doch es gibt eine weitere, für die Suche in Graphen bzw. Bäumen bisher unberücksichtigte Informationsquelle: Gebräuchliche Namen wie „Schmitt" tauchen sehr häufig auf, andere nur selten. Die 100 häufigsten Namen decken über 10% der später verwendeten Liste von 14 Millionen (800 000 eindeutigen) ab. Man kann nun leicht die Häufigkeit jedes Namens s_i bestimmen und daraus eine Wahrscheinlichkeit $P(s_i)$ ableiten. Diese Wahrscheinlichkeiten können auf verschiedene, im folgenden beschriebene Weisen in der Suche berücksichtigen werden,

Finale Wahrscheinlichkeiten

Jeder Endknoten des Suchbaumes repräsentiert einen Namen s_i. Die direkteste Methode, die Wahrscheinlichkeiten der Namen in den Suchbaum zu integrieren, ist, jedem Endknoten die seinem Namen entsprechende Wahrscheinlichkeit $P(s_i)$ zuzuordnen. Dadurch werden die Einschränkungen erst ganz am Ende der Suche wirksam. Theoretisch sollte es gleichgültig sein, wo die Wahrscheinlichkeitsmasse im Baum verteilt ist, solange sich am Ende jedes einzelnen Pfades die richtigen Wahrscheinlichkeiten akkumulieren. Für die oft sehr großen Bäume muß in der Praxis eine Strahlsuche eingesetzt werden, um die Anzahl der gleichzeitig verfolgten Hypothesen einzuschränken. Dann kann es von Vorteil sein, das Wissen um die Einschränkungen schon früher einzusetzen, wie es bei den beiden folgenden Methoden geschieht.

Lokale Wahrscheinlichkeiten

Der Name $s_i \in S$ bestehe aus n_i Buchstaben

$$s_i = l_{i,1} l_{i,2} \ldots l_{i,n_i}$$

Dabei bezeichne $l_{i,j}$ den j-ten Buchstaben in Name s_i. Ein partieller Pfad $l_{i,1} l_{i,2} \ldots l_{i,k}$ bis zum k-ten Buchstaben von s_i bestimmt eindeutig einen Knoten im Baum. Wir bezeichnen die eindeutige Kante oder Transition, die in diesen Knoten führt, als $t_{i,k}$:

$$t_{i,k} \equiv l_{i,1} l_{i,2} \ldots l_{i,k-1} \xrightarrow{t_{i,k}} l_{i,k}$$

Statt nun $P(s_i)$ den terminalen Knoten zuzuordnen, kann man das Sprachmodell zu einem früheren Zeitpunkt in den Suchprozeß einbringen, indem man eine „lokale" Wahrscheinlichkeit $\alpha_{local}(t)$ für jede Transition t definiert:

$$\alpha_{local}(t_{i,k}) := P(l_{i,k} | l_{i,1} l_{i,2} \ldots l_{i,k-1})$$

$\alpha_{local}(t_{i,k})$ ist die relative Wahrscheinlichkeit, mit der vom gemeinsamen Vaterknoten ausgehend die Transition $t_{i,k}$ und nicht eine ihrer benachbarten Transitionen gewählt

wird. Sie entspricht gerade der bedingten Wahrscheinlichkeit für $l_{i,k}$, gegeben den gesamten linken Kontext $l_{i,1} \ldots l_{i,k-1}$, und kann bestimmt werden durch Abzählen der Pfade, die durch die Kante $t_{i,k}$ und ihre Nachbarkanten laufen[7]. Betrachtet man die aufmultiplizierten lokalen Wahrscheinlichkeiten entlang eines Pfades zu einem Endknoten, der s_i repräsentiert, ergibt sich korrekterweise $P(s_i)$:

$$\begin{aligned}\prod_{k=1}^{n_i} \alpha_{local}(t_{i,k}) &= P(l_{i,1}) \cdot P(l_{i,2}|l_{i,1}) \cdots P(l_{i,n_i}|l_{i,1} \ldots l_{i,n_i-1}) \\ &= P(l_{i,1} l_{i,2} \ldots l_{i,n_i}) = P(s_i)\end{aligned}$$

Frühe Wahrscheinlichkeiten

Der wahrscheinlichste Name bzw. Endknoten, der von einem bestimmten Knoten aus erreicht werden kann, ist als Maß für die Wichtigkeit des entsprechenden Pfades interpretierbar[8]. Wir definieren $\beta(t)$ als die Wahrscheinlichkeit des wahrscheinlichsten Namens, der von der Transition t aus erreicht werden kann. $\beta(t)$ wird rekursiv berechnet, indem ausgehend von den Blättern des Baumes jeweils die maximale Wahrscheinlichkeit benachbarter Knoten auf deren Vaterknoten übertragen werden:

$$\beta(t_{i,k}) := \begin{cases} P(s_i) & \text{falls } k = n_i \quad (\text{„Endknoten“}) \\ max_{t \in suc(t_{i,k})}\{\beta(t)\} & \text{sonst} \end{cases}$$

Dabei bezeichnet $suc(t)$ die Menge aller Kanten, die direkt auf t folgen können. Nun beginnen wir am Startknoten und definieren mit $\alpha_{early}(t)$ für jede Transition t die zur Endwahrscheinlichkeit $P(s_i)$ fehlenden Faktoren:

$$\alpha_{early}(t_{i,k}) := \begin{cases} \beta(t_{i,1}) & k = 1 \\ \frac{\beta(t_{i,k})}{\beta(t_{i,k-1})} & k > 1 \end{cases}$$

Auch hier multiplizieren sich die Wahrscheinlichkeiten entlang des Pfades nach s_i korrekt zu $P(s_i)$:

$$\prod_{k=1}^{n_i} \alpha_{early}(t_{i,k}) = \beta(t_{i,1}) \cdot \frac{\beta(t_{i,2})}{\beta(t_{i,1})} \cdots \frac{\beta(t_{i,n_i})}{\beta(t_{i,n_i-1})} = \beta(t_{i,n_i}) = P(s_i)$$

Abbildung 7.8 zeigt ein Beispiel für die drei beschriebenen Methoden, die in den folgenden Experimenten mit „Final“, „Lokal“ und „Früh“ gekennzeichnet werden.

[7] Dazu „durchläuft“ man den Baum mit alle Namen entsprechend ihrer Häufigkeit und führt entsprechende Statistiken in den einzelnen Knoten.

[8] Ein Dankeschön an Michael Finke, der mich zum Experimentieren mit den „frühen“ Wahrscheinlichkeiten inspirierte, indem er mich darauf aufmerksam machte, daß diese Technik zur Abschätzung von Monogramm-Wahrscheinlichkeiten in als Bäumen organisierten Wörterbüchern des JANUS-Spracherkenners eingesetzt wird.

Bisher haben wir ignoriert, daß ein Name s_i ein Präfix eines anderen Namens s_j sein kann, was zu Problemen bei den obigen Formeln führt, denn neben die Wahrscheinlichkeiten für die Folgetransitionen tritt auch noch die Wahrscheinlichkeit, in diesem Knoten zu terminieren. Man löst das Problem, indem an jeden Namen s_i ein virtueller Knoten mit einer expliziten „Ende-Markierung" l_{i,n_i+1} angehängt wird.

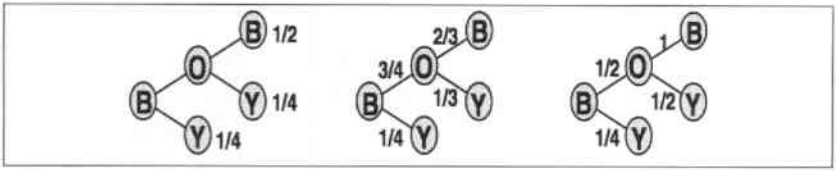


Abbildung 7.8: Finale (links), lokale (Mitte) und frühe (rechts) Zuweisung von Wahrscheinlichkeiten für einen Suchbaum, der die Namen (Bob, Boy, By) mit den Wahrscheinlichkeiten $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ repräsentiert.

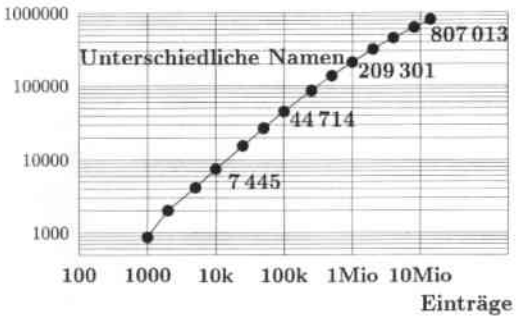
7.4.2 Experimentelle Rahmenbedingungen

Daten

Die prominenteste Anwendung für die Suche in großen, mit Wahrscheinlichkeiten versehenen Namenslisten ist die automatische Telefonauskunft. Entsprechend wurde für dieses Experiment der „Oregon Graduate Institute (OGI) Spelled and Spoken Word Telephone"-Korpus [CRF92] ausgewählt. Dort sind Aufnahmen von etwa 4000 Anrufen über das öffentliche amerikanische Telefonnetz zur Verfügung gestellt, darunter auch buchstabierte Vor- und Nachnamen.
Etwa 8% der Buchstabierungen enthalten Wörter außerhalb des Alphabets, wie beispielsweise „`c h e [sorry] c h a v [as in victor] e z`". Ohne diese Fälle ergeben sich gemäß der mit der Datenbank definierten Partitionierung der Anrufe in Trainings-, Kreuzvalidierungs- und Testmengen die in Tabelle 7.3 aufgelisteten Einteilungen. Dabei sind **SLP** und **SLN** buchstabierte Nachnamen mit und ohne Pausen zwischen einzelnen Buchstaben, **SFP** buchstabierte Vornamen und **ALP** buchstabierte Alphabete (A ... Z). Weitere Details zur OGI-Datenbank finden sich in Abschnitt 5.5.

Mengen (Typ der Daten)	Namen	Buchstaben
Training (SLN, SLP, SFP, ALP)	4132	39687
Dev. Test (SLN, SLP, SFP, ALP)	2063	15612
Test 1 (SLN)	685	4419
Test 2 (SLP)	305	1935

Tabelle 7.3: Trainings-, Kreuzvalidierungs- und Testmenge der OGI-Buchstabierdaten



Einträge	unterschiedl. Namen
685	596
1 000	870
2 500	1 990
5 000	4 078
10 000	7 445
25 000	15 571
50 000	26 787
100 000	44 714
250 000	85 258
500 000	135 550
1 000 000	209 301
2 000 000	313 320
4 000 000	452 903
8 000 000	630 718
14 000 000	807 013

Abbildung 7.9: Anzahl unterschiedlicher Namen bei wachsender Listengröße

Namenslisten

Als Namensliste wurden 14 Millionen Einträge[9] aus allen Telefonverzeichnissen des Nordostens der USA extrahiert. Der häufigste Name ist „`Smith`" und kommt beinahe 100 000 Mal vor. Insgesamt gibt es etwas über 800 000 eindeutige Namen. Über 40% dieser Namen erscheinen in den 14 Millionen Namen nur einmal!

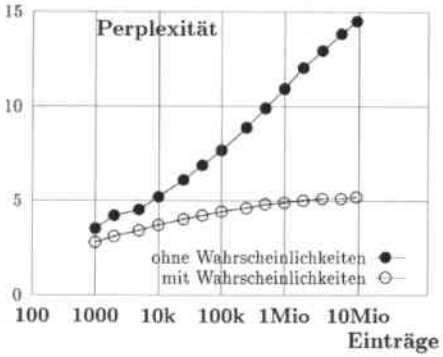
Um die Baumsuchverfahren abhängig von der Anzahl der Namen zu testen, wurden aus den 14 Millionen Namen durch zufälliges Auswählen (ohne Zurücklegen) Listen verschiedener Größen gezogen. Selbst mit den 14 Millionen Einträgen werden nicht alle Namen der SLN-Testmenge abgedeckt[10]. Deshalb wurde jede Liste zuerst mit den Namen der SLN- und SLP-Testmenge angefüllt, und dann mit den zufällig gezogenen Namen zu Listen mit einem Umfang zwischen 1000 und 14 Millionen Einträgen ergänzt. Natürlich enthalten die so gewonnenen Listen wieder viele doppelte Namen. Eine Aufstellung aller Listen mit der Anzahl der unterschiedlichen Namen findet sich in Abbildung 7.9. Die kaum abflachende Kurve zeigt, daß auch mit 14 Millionen Namen bei weitem noch nicht alle möglichen Namen abgedeckt sind.

Die Annotierung der Bäume mit Wahrscheinlichkeiten bewirkt eine starke Reduzierung der Perplexität, die sich auch positiv auf die Erkennungsraten auswirkt. In Abbildung 7.10 ist die Perplexität[11] für die SLN-Testmenge und für Bäume ohne und mit Wahrscheinlichkeiten aufgetragen.

[9] Es wurden nur die als „residential" gekennzeichneten privaten, aber keine Firmenanschlüsse ausgewählt.

[10] 49 Namen der SLN-Testmenge sind nicht in der Liste der 14 Millionen Namen enthalten!

[11] Die Perplexität in einem Baum kann man sich als den durchschnittlichen Verzweigungsfaktor seiner Kanten vorstellen (siehe auch Abschnitt 3.6).



Total	Perplexität	
	ohne Wahrscheinlichkeiten	mit
685	3.28	2.65
1 000	3.52	2.79
2 500	4.17	3.13
5 000	4.50	3.39
10 000	5.20	3.66
25 000	6.11	3.97
50 000	6.89	4.20
100 000	7.71	4.40
250 000	8.93	4.61
500 000	9.93	4.75
1 000 000	10.94	4.87
2 000 000	11.97	4.97
4 000 000	12.93	5.06
8 000 000	13.83	5.13
14 000 000	14.53	5.17

Abbildung 7.10: Größe der Namenslisten und Perplexität der SLN-Testmenge bei Bäumen ohne und mit Wahrscheinlichkeiten

Ergebnisse bei konventioneller Erkennung

Die Sprachdaten wurden über das öffentliche Telefonnetz gesammelt, d.h. es muß mit einer großen Bandbreite unterschiedlicher Aufnahmekanäle bzw. akustischer Charakteristika gerechnet werden. Es gibt in der Literatur zahlreiche Verfahren (RASTA, Mean Subtraction), um durch den Telefonkanal auftretende Störeffekte zu unterdrücken. Frühere Experimente mit diesen Verfahren [Bol95, Bir95] haben jedoch keine signifikanten Vorteile gegenüber unserer Standard-Vorverarbeitung gezeigt. Zur Vorverarbeitung wurde deshalb, wie auch in allen anderen Experimenten, alle 10 ms aus einem Analysefenster eine Kurzzeit-Spektralanalyse von 16 Melscale-FFT-Koeffizienten berechnet. Da die Telefondaten mit 8 kHz abgetastet sind, stehen für ein 16 ms breites Analysefenster statt 256 nur 128 Abtastwerte zur Verfügung.

Das MS-TDNN wurde gemäß der in Kapitel 6 beschriebenen Verfahren trainiert. Mit 100 Neuronen in der verborgenen Schicht ergab sich eine Gesamtzahl von etwa 34 000 Parametern. Die Erkennungsraten ohne Sprachmodell bzw. mit „nur" einem Bi- oder Trigramm-Modell (berechnet auf der Namensliste mit 8 Millionen Einträgen) sind in Tabelle 7.4 aufgelistet, ein Vergleich dieser Ergebnisse mit anderen Systemen wurde bereits in Tabelle 6.15 vorgestellt. Beachtenswerterweise liegen die Erkennungsraten nur geringfügig unter denen der 16 kHz Daten. Der Höreindruck einiger Aufnahmen läßt vermuten, daß die Sprecher im Durchschnitt kooperativer sind als die anderer Sprachdatenbanken.

In der SLP-Testmenge wurden die Sprecher zu Pausen zwischen den Buchstaben aufgefordert, was sich in den Ergebnissen widerspiegelt: Eine Fehleranalyse zeigt, daß in

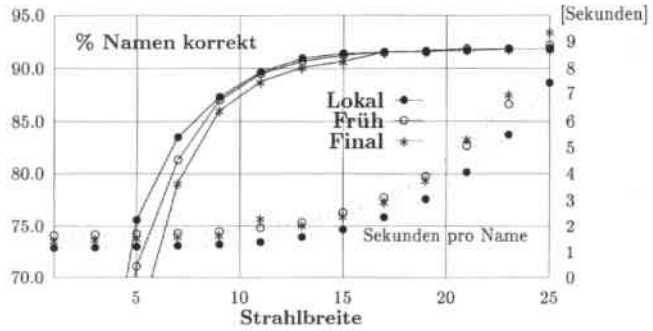


Abbildung 7.11: % Namen korrekt für die Baumsuche in einer Liste von 1 Million (200 000 unterschiedlichen) Namen. Die drei Methoden werden verglichen für unterschiedliche Strahlbreiten. Mit wachsenden Strahlbreiten steigen auch die Erkennungszeiten, die jeweils in den unteren Kurven abgetragen sind.

der SLN-Testmenge etwa 2.5, in der SLP-Testmenge aber nur 1.7 Prozentpunkte (bei insgesamt etwa 10% Fehlern) auf das Konto von Einfügungen und Auslassungen gehen.

	SLN		SLP	
Sprachmodell	BA	NK	BA	NK
Kein LM	**88.2**	53.7	**90.6**	60.0
Bigramme	91.0	62.8	92.3	69.2
Trigramme	92.5	70.2	92.6	72.5

Tabelle 7.4: Ergebnisse ohne bzw. mit Bi- und Trigramm-Sprachmodellen

7.4.3 Vergleich der Wahrscheinlichkeitsannotationen

Die drei Methoden der finalen, lokalen und frühen Zuweisung von Wahrscheinlichkeiten wurden an einem Baum getestet, der die Liste mit 1 Million Einträgen repräsentiert. Die 209 301 unterschiedlichen Namen dieser Liste entsprechen 1 527 476 Buchstaben, die in 542 103 Knoten repräsentiert werden. Gleichzeitig soll mit den Experimenten eine geeignete Strahlbreite[12] gefunden werden, denn ohne Strahlsuche ist ein Baum dieser Größe nicht handhabbar. Die Erkennungsraten für die drei Methoden sind für Strahlbreiten zwischen 5 und 25 in Abbildung 7.11 aufgezeichnet.

Wie erwartet gleichen sich die drei Methoden mit wachsender Strahlbreite an, und gleichzeitig steigt die zur Erkennung benötigte Rechenzeit. Insgesamt erzielen die „loka-

[12] Da die akustischen Bewertungen des Erkenners immer Werte zwischen 0 und 1 liefern, bedeutet ein Strahl beispielsweise der Breite 10, daß ein alternativer Pfad für mindestens 10 Zeitschritte (100 msec) sehr schlecht sein muß, bevor er abgeschnitten wird.

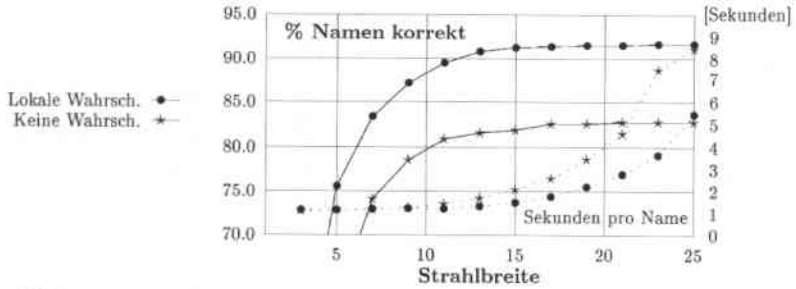


Abbildung 7.12: Einfluß der Strahlbreite und Erkennungszeit pro Name für Suche mit und ohne Wahrscheinlichkeiten. Die Erkennung mit Wahrscheinlichkeiten ist nicht nur besser, sondern auch schneller.

len" Wahrscheinlichkeiten die günstigsten Ergebnisse. Die finalen Wahrscheinlichkeiten schneiden am schlechtesten ab, denn das Sprachmodell kommt zu spät zum Einsatz, um einen akustisch schlecht bewerteten Pfad vor einer Verdrängung aus dem Suchstrahl zu bewahren.

In Abbildung 7.12 sind die lokalen Wahrscheinlichkeiten mit einer Baumsuche ganz ohne Wahrscheinlichkeiten verglichen. Dabei erzielen die lokalen Wahrscheinlichkeiten mit einem Vorsprung von beinahe 10 Prozentpunkten deutlich bessere Erkennungsleistungen. Außerdem sind die Erkennungszeiten kürzer: Durch eine bessere Fokussierung auf die wahrscheinlicheren Namen werden umgekehrt die weniger vielversprechenden Pfade früher von der Strahlsuche eliminiert.

7.4.4 Vergleich unterschiedlicher Listengrößen

Je stärker die Einschränkung des Suchraums, desto größer ist der Gewinn in den Erkennungsraten. In einem neuen Experiment wird die Baumsuche auf den Namenslisten unterschiedlicher Größe getestet, mit den oben als optimal gefundenen Einstellungen einer Strahlbreite von 15 sowie der „lokalen" Wahrscheinlichkeitsannotation. Die Ergebnisse sind in Abbildung 7.13 für die SLN- und SLP-Testmengen aufgetragen, und (nur für SLN) in Tabelle 7.5 auszugsweise den Ergebnissen ohne bzw. mit Bi- und Trigramm-Sprachmodell gegenübergestellt. Es ist ersichtlich, daß durch die Wahrscheinlichkeiten eine dramatische Fehlerreduzierung erreicht wird, die in der Perplexitätsreduzierung begründet liegt, die in Abbildung 7.10 aufgezeigt ist.

Zur Suche muß der gesamte Baum als Struktur im Speicher gehalten werden, die akustischen Modelle mit den Bewertungen für die partiellen Suchhypothesen werden jedoch dynamisch nur für die jeweils aktiven Knoten alloziert. So braucht die Suche in der Liste mit 1 Million Einträgen bzw. 200 000 unterschiedlichen Namen etwa 42 Megabyte

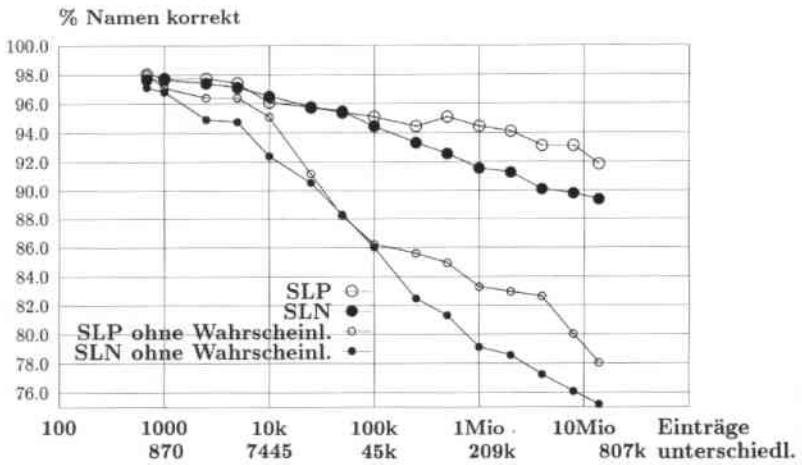


Abbildung 7.13: % Namen korrekt für Bäume mit und ohne Wahrscheinlichkeiten auf der SLN- (Spelled Last Names) und der SLP-Testmenge (Spelled Last Names with Pauses), in Abhängigkeit von der Größe der Listen (Einträge) bzw. der Anzahl unterschiedlicher Namen

Sprachmodell	SLN	
	BA	NK
Kein Sprachmodell	**88.2**	53.7
Bigramme	91.0	62.8
Trigramme	92.5	70.2
Baum 1 000	98.1	**97.7**
Baum 100 000	97.5	**94.4**
Baum 1 000 000	97.1	**91.5**
Baum 14 000 000	96.5	**89.3**
14 Mio., keine Wahrscheinlichkeiten	91.4	**75.2**

Tabelle 7.5: Vergleich von % Namen korrekt und % Buchstabenakkuratheit für die SLN-Testmenge mit verschiedenen Sprachmodellen

Hauptspeicher (Baum inklusive gesamtem Erkenner), und ein Name kann im Durchschnitt in weniger als 2 Sekunden erkannt werden[13]. Während der Suche sind von den mehr als einer halben Million Knoten zu jedem Zeitpunkt nur etwa 100 bis 1000 aktiv. Auch in dem Baum mit 2 Millionen Knoten für 800 000 Namen liegt die Erkennungszeit pro Name unter 4 Sekunden, was bei einer durchschnittlichen Länge der Aufnahmen von ca. 3.5 Sekunden etwa Echtzeit entspricht.

[13] Auf einer SUN-Ultra2-Workstation, mit einer Strahlbreite von 15

7.4.5 Erkennung von Straßennamen im Auto

Neben der Telefonauskunft eignet sich die Buchstabiererkennung aus vorgegebenen Namenslisten für eine weitere Anwendung: In Fahrzeugnavigationssystemen können Zieladressen durch Buchstabieren der Städte- und Straßennamen eingegeben werden. Zu dieser Aufgabenstellung werden derzeit erste Pilotversuche auf im Auto gesammelten Buchstabierungen (die in Abschnitt 5.6 beschriebenen VODIS-Daten) durchgeführt. Als Namensliste wurden alle im Stuttgarter Postleitzahlenverzeichnis aufgeführten 3 400 Straßennamen gewählt.

Im fahrenden Auto aufgenommene Daten sind mit zahlreichen Störgeräuschen, insbesondere Motor- und Fahrgeräuschen, behaftet, wodurch die Erkennung erschwert wird. In ersten Experimenten wurde mit einer Trainings-, Kreuzvalidierungs- und Testmenge von 50, 10 und 20 Sprechern[14] (jeder Sprecher buchstabierte das Alphabet und 5 Namen) eine Buchstabenakkuratheit (auf der Testmenge) von 73% bei einem am Hals des Fahrers angebrachten Nahbesprechungsmikrofon und von 60% mit einem Deckenmikrofon erreicht. Ein ganzer Name ist damit nur noch in etwa 20% der Fälle korrekt.

Werden nur die 3 400 Stuttgarter Straßennamen[15] zugelassen, erkennt man mit der Baumsuche auf der Testmenge 90% (Nahbesprechungsmikrofon) bzw. 77% (Deckenmikrofon) der buchstabierten Namen korrekt. Trotz der relativ niedrigen Erkennungsraten für einzelne Buchstaben werden also mit der Baumsuche immer noch gute Ergebnisse erzielt.

7.5 Verwandte Arbeiten

Im folgenden werden die Arbeiten anderer Forschungsgruppen zur Buchstabiererkennung auf eingeschränkten Namenslisten beschrieben[16]. Einige Beiträge stammen von Telekommunikationsunternehmen, die damit ihr Interesse an Anwendungen in Telefonauskunftssystemen dokumentieren.

Bellcore

Eine Untersuchung zur Erkennung sowohl fließend als auch buchstabiert gesprochener Nachnamen wird in [KSS95, PKP+96] von Kamm, Pepper und anderen beschrieben.

Zur Erkennung **fließend gesprochener Namen** wird der HTK-HMM-Erkenner[17] als kontextunabhängiger Phonemerkenner mit Hilfe der OGI-Datenbank trainiert. Als Test-

[14] Mit dem in Abschnitt 5.6 beschriebenen Umfang von 100 Sprechern wurden noch keine Experimente durchgeführt.

[15] Plus diejenigen in den Testdaten buchstabierten Straßen- und Personennamen, die nicht bei den Stuttgarter Straßennamen vertreten sind

[16] Für Vergleiche ohne Sprachmodelle siehe auch Abschnitt 6.9.

[17] Der von der Cambridge University vertriebene Spracherkenner („Hidden Markov Model Toolkit")

menge dienen 200 Aufnahmen, in denen jeweils Vor- und Nachname gesprochen sind[18]. Als Namenslisten werden zufällige Teilmengen aus 1.5 Millionen unterschiedlichen Einträgen in Telefonbüchern gezogen. In einem Vergleich der erkannten Phonemsequenz mit den Phonemumschriften der jeweiligen Liste wird der am besten passende Name gefunden, die genaue Prozedur ist nicht spezifiziert. Dabei wurden 82.5% korrekte Namen bei einer Liste von 200 Namen, und 16.5% bei einer Liste von 1.5 Millionen eindeutigen Namen erzielt.

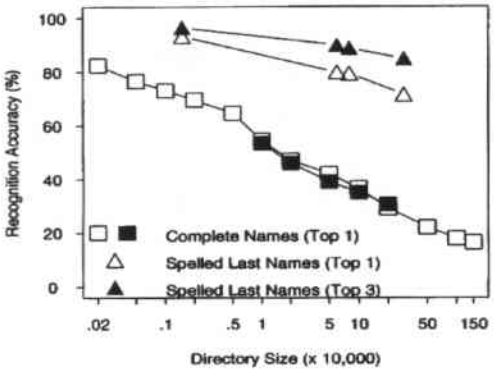


Abbildung 7.14: Erkennungsraten für gesprochene und buchstabierte Namen in den Bellcore-Experimenten, aus [KSS95]

Die Testmenge von 1242 **buchstabierten Namen** wurde mit dem Buchstabiererkenner des OGI [CFGJ91] auf Namenslisten mit bis zu 290 332 unterschiedlichen Namen[19] getestet. In [KSS95] ist die Suchtechnik nicht spezifiziert, nach persönlicher Auskunft der Autoren wurde zu einem zuerst ohne Sprachmodell erkannten Namen der ähnlichste Eintrag in der Namensliste gesucht, wobei die Verwechselbarkeit der Buchstaben berücksichtigt wurde. Die Graphik in Abbildung 7.14 weist für die Liste mit 290 332 unterschiedlichen Namen etwa 70% korrekte Namen aus.

CNET

In [JLMG93, JLM93] untersuchen D. Jouvet et al. von CNET (Center National d'Études des Telecommunications, die Forschungsabteilung der französischen Telekom) die Erkennung von Städtenamen. Trainings- und Testmenge bestehen aus jeweils einer Hälfte von etwa 3000 Städtenamen, die von 180 Sprechern auf französisch buchstabiert wurden.

[18] Ein kompletter Name (inkl. Vorname) wird selbstverständlich zuverlässiger erkannt als nur ein Nachname.

[19] Das entspricht etwa der Liste mit 1 Million (209 301 unterschiedlichen) Namen aus Abschnitt 7.4.2.

Wie in unseren eigenen Experimenten erzielen sie mit einer direkten Integration der Einschränkungen in die Suche die besten Ergebnisse, die aber nur auf eine Liste von 120 Städtenamen angewandt wird. Mit einem allgemeinem rekursiven Transitionsnetzwerk, dessen Terminalsymbole bei Bedarf dynamisch mit den entsprechenden akustischen Modellen expandiert werden [Dup93], konnten 5000 unterschiedliche Namen direkt in die Suche integriert werden.

Für größere Listen wurden zwei verschiedene syntaktische Nachbearbeitungen eingesetzt:

(1) Wie auch in unseren Versuchen in Abschnitt 7.3.3 wird der erste legale Name aus einer N-Besten-Liste gewählt.

(2) Jeder Name der gegebenen Liste wird als kleines HMM modelliert, welches Substitutionen, Einfügungen und Auslassungen für die Buchstaben des gegebenen Namens erlaubt. Dann wird derjenige Name als der korrekte bestimmt, dessen HMM den erkannten Namen mit der größten Wahrscheinlichkeit erzeugen kann. Die Parameter der HMMs werden mit Hilfe der auf der Trainingsmenge erkannten Namen eingestellt. Dieses Verfahren ist ähnlich, aber aufwendiger als das in Abschnitt 7.3.2 vorgeschlagene, bei dem statt HMM-Wahrscheinlichkeiten die Editierdistanz unter Berücksichtigung der Konfusionsmatrix berechnet wird.

Um obige Strategien zu kombinieren, berechnen die Autoren die ersten N (beispielsweise 8) Hypothesen (1) und bewerten diese dann wie in (2) neu. Die Ergebnisse der einzelnen und kombinierten Methoden sind in Tabelle 7.6 aufgeführt.

Größe der Wortlisten, Suchtechnik	% NK
ohne Wortliste (bei 84% BA)	40
120, in Suche integriert	99
5 000, in Suche integriert	97
5 000, (1) und (2) kombiniert	96
30 000, (1) bester in N-Besten-Liste	85.0
30 000, (2) Wahrscheinlichkeitsbewertung	87.6
30 000, (1) und (2) kombiniert	91

Tabelle 7.6: % Namen korrekt auf französisch buchstabierten Städtenamen, nach Jouvet et. al. (CNET)

Panasonic STL

J.C. Junqua et al. von Panasonic Speech Technology Laboratory (STL) stellen in [JVFM95] ihr System namens „SmarTspelL" vor. Als Erkenner dient ein konventionelles

Hidden-Markov-Modell, das in ein komplexes System mit insgesamt 4 Verarbeitungsstufen eingebettet ist. Im ersten Schritt wird eine Liste der $N = 20$ besten Hypothesen erzeugt, die in einem zweiten Schritt mit einem „selectively trained network" nachbearbeitet werden. In einem dritten Schritt werden diese Hypothesen der Liste der zu erkennenden Namen gegenübergestellt. Dann wird entweder der beste Kandidat ausgesucht oder eine erneute, nun auf die 20 besten Kandidaten eingeschränkte Erkennung durchgeführt. In einer neueren Version des Systems [Jun97] wird auf die spezialisierten Netzwerke verzichtet.

Die Sprachdaten sind eine Untermenge des OGI-Telefonkorpus. Die Trainingsmenge besteht aus 225 buchstabierten Alphabeten und 1300 buchstabierten Namen, die Kreuzvalidierungsmenge aus 558 und die Testmenge aus 491 Namen. Die Namen wurden aus der SLN-Kategorie (Nachnamen ohne explizite Pausen) gewählt, gemäß der in der Datenbank gegebenen Aufteilung in Trainings- und Testmengen. Es wurden nur Aufnahmen berücksichtigt, in denen keinerlei Geräusche transkribiert sind, deshalb stehen nur 491 Namen (statt 685 in unseren Experimenten) zur Verfügung[20].

Größe der Wortlisten	% NK
491	98.4
3 388	95.3
21 877	90.4
110 000	92.8

Tabelle 7.7: % korrekt erkannte buchstabierte Namen der SLN-Testmenge bei den Experimenten von Junqua [JVFM95, Jun97]

Tabelle 7.7 faßt die mit unterschiedlichen Wortlisten erzielten Ergebnisse zusammen. In [Jun97] wird ein weiteres Experiment mit einer Wortliste mit 110 000 Namen beschrieben, bei dem sogar ohne den letzten Verarbeitungsschritt (erneute Erkennung auf eingeschränkter Namensliste) eine Erkennungsrate von 92.8% Namen erzielt wurde. Dieses etwas inkonsistente Ergebnis (2.5% bessere Erkennungsrate bei größerer Wortliste) erklärt der Autor damit[21], daß die größere Wortliste disjunkt (keine Obermenge) zu den kleineren war. Ohne Wortliste (aber mit Bigrammen) wird in [JVFM95] eine Buchstabenerkennungsrate von 86.8% angegeben, in der zwar Auslassungs-, aber keine Einfügungsfehler (3.7%) berücksichtigt sind, woraus sich eine Buchstabenakkuratheit von 83.1% ergibt. In einer neueren Untersuchung [Jun97] experimentiert Junqua mit verschiedenen Vorverarbeitungen. Die besten Ergebnisse liegen bei 88.0% Buchstabenakkuratheit[22], erzielt mit einer cepstralen Mittelwertnormierung.

[20] Die Größe von Junquas Trainings- und Testmengen ist konsistent mit den in Tabelle 5.5 als „Qualitätsstufe 0" klassifizierten Daten, wohingegen für die Experimente in dieser Arbeit „Qualitätsstufe 0 bis 2" verwendet wird.

[21] Persönliche Mitteilung

[22] Das Ergebnis ist in der Veröffentlichung nur als Graphik dargestellt, genaue Rate nach persönlicher Kommunikation mit J.C. Junqua.

OGI

Die Gruppe um Ron Cole und Mark Fanty am Oregon Graduate Institute (OGI) hat bereits 1991 einen Ansatz vorgestellt [CFGJ91], der als Mittelweg gesehen werden kann zwischen direkter Integration in die Suche und einem Nachverarbeitungsschritt. Der bereits in den Abschnitten 4.2 und 6.9 vorgestellte OGI-Buchstabiererkenner segmentiert die Eingabe zuerst in einzelne Buchstaben, die dann mit einem statischen neuronalen Netz klassifiziert werden. Dabei erhält jeder Buchstabe eine Bewertung zwischen 0 und 1. Ein erkannter Name wird unter Berücksichtigung dieser Bewertung mit den erlaubten Namen verglichen. Um eine schnelle Suche zu ermöglichen, ist die Namensliste in einer Baumstruktur gespeichert, in der mit Hilfe der Buchstabenbewertung und einer Strafe für Einfügungen und Auslassungen der Name mit der besten Bewertung gesucht wird.

Das System in [CFGJ91] wurde auf 1020 buchstabierten Namen getestet. Die 34 Sprecher waren instruiert, kurze Pausen zwischen den einzelnen Buchstaben einzulegen, was aber in etwa 10% der Fälle nicht eingehalten wurde. Ohne Sprachmodell erzielte der Erkenner eine Buchstabenakkuratheit von 89.1%, die in 53.9% korrekten Namen resultierte. Bei bekannten Buchstabengrenzen wurden 93% der Buchstaben korrekt erkannt. Eingeschränkt auf eine Liste von 50 000 Namen (Perplexität 4) wurden 95.3% korrekte Namen erzielt.

In einem weiteren Experiment [FCR92] wurde dieselbe Technik auf Buchstabierungen mit Pausen evaluiert, die über das Telefon gesammelt wurden (einer Untermenge des später als „OGI Spoken and Spelled Telephone Corpus" bekannt gewordenen Datensatzes). Die Trainingsmenge besteht aus 400 buchstabierten Alphabeten, sowie 800 Vor- und Nachnamen von insgesamt 800 Sprechern. Als Testmenge wurden 100 Alphabete und 300 Namen von „neuen" Sprechern mit Pausen buchstabiert. Auf den 100 Alphabeten wurden 89%, auf den 300 Nachnamen 87% Buchstabenakkuratheit erzielt. Auf die Liste mit 50 000 Nachnamen eingeschränkt, wurden 90.7% korrekte Namen erkannt.

FAUST, Deutsche Telekom

Auf der EUROSPEECH'95 in Madrid wurde von Kaspar et al. [KFSW95] das System „FernsprechAUSkunft Telekom" (FAUST) vorgestellt, ein Demonstrator für eine kleine Telefonauskunft. Der Anrufer wird gebeten, den Stadtnamen und den Namen des gewünschten Teilnehmers zu nennen. In beiden Fällen kann auch buchstabiert werden. Das Telefonverzeichnis umfaßt insgesamt 5000 Namen, wird aber durch die Wahl einer von 25 möglichen Städten noch einmal stark reduziert. Über den eingesetzten Buchstabiererkenner und Erkennungsraten werden keine Angaben gemacht.

7.6 Zusammenfassung

Ein einzelner Buchstabe in einer kontinuierlich gesprochenen Buchstabensequenz wird in etwa 90% der Fälle korrekt erkannt. Die Wahrscheinlichkeit, alle Buchstaben, also etwa einen ganzen Eigennamen, richtig zu erkennen, ist geringer und liegt unter 60%. Nutzt man mit Hilfe von Sprachmodellen Wissen über die zu erkennenden Namen, kann dieser Wert beachtlich verbessert werden, wie in Tabelle 7.8 anhand einiger experimenteller Eckdaten aufgezeigt ist. Selbst wenn man die kompletten Telefonbücher der Nordostküste der USA zugrunde legt (14 Millionen Einträge/800 000 unterschiedliche Namen), können mit geeigneten Suchtechniken knapp 90% der Namen korrekt erkannt werden.

Sprachmodell (Größe der Wortlisten)	KA-Alph % NK	OGI (SLN) % NK
ohne Sprachmodell	56.4	53.7
100 000 Einträge	94.1	94.4
14 Millionen Einträge	–	89.3

Tabelle 7.8: Einschränkung des Sprachmodells auf Listen von Namen

In einer ersten Serie von Experimenten (s.a. Zusammenfassung in Abschnitt 7.3.7) wurden verschiedene Suchtechniken auf einer Liste von 110 000 Namen verglichen [BH95a]. Ein Weg besteht darin, die **Einschränkungen erst nach der eigentlichen Erkennung** auf syntaktischer Ebene zu aktivieren: Bei der „Nächster-Nachbar-Suche" wird die erkannte Hypothese mit allen erlaubten Namen verglichen und der ähnlichste gewählt. Dabei ist es hilfreich, wenn bei der Berechnung der Distanzen die Verwechselbarkeit einzelner Buchstaben berücksichtigt wird. In der „N-Besten-Suche" erzeugt der Erkenner ein große Auswahl an Hypothesen, unter denen die am besten bewertete legale Hypothese gewählt wird.

Erfolgreicher ist es, die Einschränkungen schon früher, **direkt im Suchprozeß**, zu nutzen. Auch die klassischen N-Gramm-Sprachmodelle werden direkt in der Suche wirksam. Allerdings betrachten Bi- und Trigramme nur die letzten ein oder zwei zurückliegenden Wörter und sind daher für unsere Anwendung nicht mächtig genug.
Um den vollen Kontext zu berücksichtigen, kann man in einem Netzwerk (als minimaler Graph oder Baum repräsentiert) genau die Menge aller zu erkennenden Namen kodieren. Auf dem minimalen Graphen wurde ein zweistufiges Suchverfahren entwickelt, das die Verwaltung der Suchhypothesen von der Berechnung der Buchstabendurchgänge entkoppelt, um nicht jedes einzelne der im Graphen tausendfach duplizierten akustischen Buchstabenmodelle erneut durchsuchen zu müssen.

Letztlich hat sich aber die **Suche in einer Baumstruktur** als beste Technik erwiesen. Ein Baum ist zwar wesentlich größer als ein entsprechender minimaler Graph, dafür besitzt jedes Blatt einen eindeutigen Pfad zum Wurzelknoten. Damit erspart man sich die

Notwendigkeit einer Rückwärtsverzeigerung, ohne die in einem Graphen (mit zusammengeführten Kanten) der durchlaufene Pfad zum besten Endknoten nicht wiedergefunden werden könnte. Dadurch wird die Baumsuche konzeptionell einfacher und trotz der größeren Baumstruktur effizienter, da ohnehin immer nur ein kleiner Teil der Knoten aktiv durchsucht werden muß. Die Ergebnisse dieser Experimente wurden bereits in Tabelle 7.2 zusammengefaßt.

Mit der aus dem Vergleich am erfolgreichsten hervorgegangenen Baumsuche wurden auf den buchstabierten Namen der OGI-Telefondatenbank weitere Experimente [HW96] durchgeführt. Als Listen erlaubter Namen wurden unterschiedlich große Teilmengen eines umfangreichen Telefonverzeichnisses mit 14 Millionen Einträgen bei etwa 800 000 unterschiedlichen Namen getestet. Neben den in den Listen gegebenen Namen kann auch die Information über deren Häufigkeit gewinnbringend genutzt werden. Diese Nutzung der a priori Wahrscheinlichkeiten einzelner Nachnamen wurde erstmals in der vorliegenden Arbeit systematisch untersucht. Dazu wurden drei Konzepte zur Annotierung der Suchbäume mit Wahrscheinlichkeiten entworfen und evaluiert.

Vergleichbare Experimente			MS-TDNN + Baumsuche	
	Liste	% NK	Liste	% NK
Junqua [JVFM95, Jun97]	-	**88.0**†	-	**91.0**†
	3 388	**95.3**	4 078	**97.1**
	21 877	**90.4**	26 787	**95.5**
	110 000	**92.8**	100 000	**94.5**
CNET [JLM93]	5 000	**97.0**	4 078	**97.1**
	30 000	**91.0**	26 787	**95.5**
OGI [FCR92]	-	**87.0**‡	-	**88.2**‡
	50 000	**90.7**	85 258	**93.3**
Bellcore [KSS95]	290 332	**70**	313 320	**91.2**
			807 013	**89.3**

Tabelle 7.9: Vergleich verschiedener Systeme zur Erkennung buchstabierter Namen. Die mit † gekennzeichneten Erkennungsergebnisse wurden mit einem Erkenner ohne Wortlisten, aber mit Bigrammen ermittelt und geben nicht % Namen korrekt (NK), sondern % Buchstabenakkuratheit (BA) an, ‡ kennzeichnet % BA ganz ohne Sprachmodell.

In Tabelle 7.9 sind die Ergebnisse der MS-TDNN-Baumsuche den im letzten Abschnitt beschriebenen Experimenten anderer Forscher gegenübergestellt[23]. Obwohl die Erkennungsraten nur bedingt miteinander verglichen werden können (da sie in allen Fällen mit anderen Wortlisten und in manchen Fällen mit anderen akustischen Daten ermittelt wurden), darf festgestellt werden, daß das MS-TDNN in allen Vergleichen besser abschneidet.

[23] Siehe auch Tabelle 6.15 für einen Vergleich von Ergebnissen ohne Wortlisten

Kapitel 8

Buchstabieren in spontaner Sprache

Wir haben in den vorangehenden Kapiteln einen Buchstabiererkenner betrachtet, der zusammenhängend gesprochene Buchstabensequenzen mit geringen Fehlerraten erkennt, insbesondere wenn es sich dabei um buchstabierte Namen aus einer bekannten Namensliste handelt. Was passiert jedoch, wenn eine Äußerung neben Buchstaben auch „normale" Wörter enthält? Wie erkennt man Buchstabensequenzen inmitten einer Konversation?

8.1 Problemstellung

Wann werden wir mit der Situation konfrontiert, Buchstaben zusammen mit fließendem Text zu erkennen? Man kann sich nicht darauf verlassen, daß ein Benutzer beispielsweise einer Telefonauskunft ausschließlich buchstabiert, selbst wenn er in einer restriktiven Dialogführung explizit dazu aufgefordert wird. Solche Abweichungen vom Erlaubten mögen selten auftreten[1], andererseits ist eine natürlichsprachliche Benutzerschnittstelle mit möglichst wenigen Einschränkungen aber ohnehin zu bevorzugen. In einer völlig freisprachlichen Telefonauskunft würde man – statt sich durch einen starren Dialog zu hangeln – zum Beispiel einfach sagen:

„Bitte verbinden Sie mich mit Herrn Müller, M, Ü, L, L, E, R."

Allgemein muß man immer dann mit Buchstabierungen rechnen, wenn Namen oder Adressen übermittelt werden:

„Wie komme ich in die Albstraße, A, L, B ?"

Hier werden Buchstabierungen von spontaner Sprache eingeleitet, in sie eingebettet, von ihr kommentiert und/oder korrigiert. Weiterhin können Füllwörter auftreten („und dann L"), oder es werden kompliziertere Konstrukte wie etwa „H wie Haus" benutzt. Aus der Fülle dieser Phänomene, die in Abschnitt 1.3 klassifiziert sind, untersuchen wir folgende drei Problemkreise:

[1] Immerhin in etwa 8% aller Fälle in den OGI-Spell-Daten

- **In spontane Sprache eingebettete Buchstabierungen**, wie im Fall der beiden zuletzt angeführten Beispielsätze. Buchstabieren in spontaner Sprache ist ein bisher kaum erforschtes Gebiet, infolgedessen stehen nur wenige Sprachdaten zur Verfügung. Die einzige uns bekannte Datenbank mit einer einigermaßen umfangreichen Menge von Buchstabierungen in spontaner Sprache ist der VERBMOBIL (VM)-Korpus (siehe Abschnitt 5.8), auf dem einige der im folgenden beschriebenen Experimente durchgeführt wurden. Dabei sind zwei Probleme zu lösen:
  Zum einen wird ein **Sprachmodell** für die Buchstabiersätze benötigt. Dies kann aus den Trainingstexten des VM-Korpus gewonnen werden, allerdings sind dort nur wenige Buchstabierbeispiele verfügbar. Indem die Buchstabierungen zu einem Subsprachmodell „Buchstabensequenz" verallgemeinert werden, können Übergänge innerhalb von Buchstabensequenzen unabhängig von den gegebenen Trainingsdaten modelliert werden.
  Die **akustische Modellierung** der Buchstaben kann natürlich von dem Erkenner übernommen werden, der auch den fließenden Text erkennt. Da jedoch der MS-TDNN-Erkenner für Buchstaben bessere Ergebnisse geliefert hat und außerdem stärkere Sprachmodelle benutzen kann, versucht man den „Buchstabierspezialisten" mit dem spontansprachlichen Erkenner zu kombinieren.

- **Buchstabenspezifikationen oder Paraphrasierungen**
  Auch in der Buchstabierung selbst können Nicht-Buchstaben in Form von Assoziationen oder Füllwörtern vorkommen, wie in dem Beispiel „`H wie in Haus, Ida, Ludwig, und dann D wie Dora`". Lösungsansätze zu dieser Problemstellung sind in Abschnitt 8.7 diskutiert. Auf eine experimentelle Untersuchung dieser Phänomene konnte im Rahmen dieser Arbeit nicht eingegangen werden, insbesondere da es zu diesem Themenbereich noch keine Sprachdaten gibt.

- **Buchstabierte und fließend gesprochene Wörter**
  Wie zuvor sind hier Buchstabierungen und „normale" Sprache vermischt, allerdings ist die Situation sehr einfach strukturiert: Ohne sonstigen Text wird lediglich ein Name zuerst fließend, und danach buchstabiert gesprochen. Damit ist dieselbe Information auf zwei unterschiedliche Arten repräsentiert. Kann diese Redundanz zur Erzielung besserer Erkennungsleistung ausgenutzt werden? Anhand einer in Karlsruhe gesammelten Datenbank von knapp 3 000 fließend und buchstabiert gesprochenen Namen werden in Abschnitt 8.8 einige Experimente zu dieser Fragestellung beschrieben.

„Normale", fließend gesprochene Sprache wird nicht mit dem Buchstabiererkenner, sondern mit dem spontansprachlichen JANUS-Erkenner erkannt, der auf den VM-Daten trainiert wurde und im nächsten Abschnitt kurz beschrieben ist.

8.2 Der JANUS-Spracherkenner

JANUS ist ein Sprach-zu-Sprach-Übersetzungssystem, das gemeinsam an der Universität Karlsruhe und der Carnegie Mellon University, Pittsburgh, entwickelt wird [LWL+97]. Teile des JANUS-Projektes werden seit Mitte 1993 im Rahmen des BMBF Verbundprojektes VERBMOBIL gefördert. Im Zusammenhang mit der Buchstabiererkennung wird nur die deutschsprachige Erkennerkomponente von JANUS benötigt, das Gesamtsystem ist im folgenden kurz umrissen.

Das JANUS-Sprachübersetzungssystem

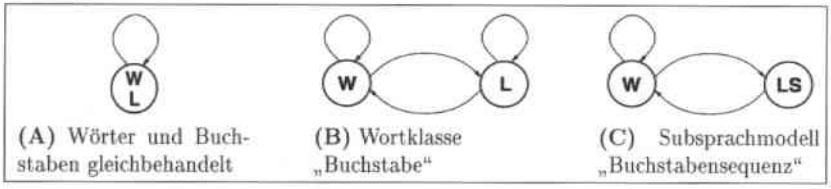
JANUS übersetzt spontansprachliche Äußerungen aus einer der Eingabesprachen Deutsch oder Englisch wahlweise in eine der Zielsprachen Deutsch, Englisch, Spanisch, Japanisch oder Koreanisch. Die in der Eingabesprache erkannte Satzhypothese wird mit unifikationsbasierten Parsingalgorithmen analysiert und in eine sprachunabhängige, interne Zwischensprache, die *Interlingua*, übersetzt und dort nicht wortgetreu, sondern sinngemäß in Form verschiedener aufgabenspezifischer Konzepte repräsentiert. Ausgehend von der Interlingua werden die Sätze in der Zielsprache generiert und akustisch synthetisiert.

Der JANUS-Spracherkenner

Die Spracherkennungskomponente des JANUS-Systems ist in der Lage, kontinuierlich gesprochene Sätze sprecherunabhängig zu erkennen. Die Vokabulargröße liegt derzeit bei etwa 5 000 Wörtern für das spontansprachliche Terminabsprachen-Szenario von VERBMOBIL. In „spontaner" Sprache findet man zahlreiche Effekte wie Wortabbrüche, Wiederholungen oder Verzögerungen. Wortverschleifungen und dialektbedingte Einflüsse treten stärker auf, und kaum ein Satz wird grammatikalisch korrekt zu Ende geführt. Die Erkennung spontaner Sprache stellt somit wesentlich höhere Anforderungen als die gelesener Sprache und wird gegenwärtig nur auf relativ eng begrenzten Domänen eingesetzt, wie etwa im Terminabsprachen-Szenario.

Der JANUS-Erkenner basiert auf semi-kontinuierlichen oder voll kontinuierlichen *Hidden Markov Modellen*. Die Emissionswahrscheinlichkeiten der akustischen Modelle werden mit bis zu 5 000 32-modalen Gauß-Mischverteilungen über einem Merkmalsraum berechnet, dessen Dimension je nach Sprachvorverarbeitung zwischen 16 und 48 liegt. Die akustischen Spracheinheiten sind Subpolyphone (Segmente von Phonemen in einem breiten Kontext, typischerweise Triphone), die mit Hilfe eines divisiven Ballungsalgorithmus und eines Entscheidungsbaumes von ursprünglich mehreren 100 000 auf schließlich 5 000 akustische Modelle geballt werden.

Die Emissionswahrscheinlichkeiten werden in eine vokabularspezifische Matrix eingetragen, in der durch dynamisches Programmieren (DTW) die besten Satzhypothesen



(A) Wörter und Buchstaben gleichbehandelt

(B) Wortklasse „Buchstabe"

(C) Subsprachmodell „Buchstabensequenz"

Abbildung 8.1: Buchstaben im Sprachmodell

gesucht werden. Die Suche von JANUS ist auf die Erkennung kontinuierlich gesprochener Sprache bei großen Wortschätzen optimiert und arbeitet in mehreren Durchgängen. Gemeinsame Anfangsphoneme der Wörter des Erkennungsvokabulars können zu einer Baumstruktur zusammenfaßt werden, um den Suchaufwand zu reduzieren. Das Suchergebnis ist ein Worthypothesengraph (WHG) oder eine aus dem WHG extrahierte Liste der N-Besten-Hypothesen.

Die neueste JANUS-Version ist objektorientiert programmiert und in das `Tcl/Tk`-Toolkit eingebettet. `Tk` bietet elegante Visualisierungs- und Interaktionsmöglichkeiten, während die Skriptsprache `Tcl` eine flexible Steuerung des Erkenners erlaubt, mit der Einfluß auf die elementaren Datenstrukturen genommen und mächtige Trainingsalgorithmen programmiert werden können.

8.3 Sprachmodelle für eingebettete Buchstabensequenzen

Sollen in spontane Sprache eingebettete Buchstabierungen erkannt werden, umfaßt das Vokabular $V = V_W \cup V_L$ neben den „normalen" Wörtern $V_W := \{w_1, w_2, \ldots\}$ die Buchstaben $V_L = \{A, B, \ldots\} := \{L_1, L_2, \ldots\}$. Um letztere in das Sprachmodell zu integrieren, können die drei in Abbildung 8.1 illustrierten Methoden (A) - (C) angewandt werden:

(A) Im einfachsten Fall werden die Buchstaben wie Wörter behandelt. Ein entsprechendes Sprachmodell kann direkt aus den Trainingstexten generiert werden. Allerdings werden die Übergänge $P(L_j|w_i)$, $P(L_j|L_i)$ und $P(w_j|L_i)$ von jedem Wort in jeden Buchstaben, innerhalb von Buchstaben und von den Buchstaben zurück zu Wörtern nur schlecht modelliert, wenn wie im VM-Korpus nur einige Hundert Buchstabierbeispiele verfügbar sind.

(B) Wir betrachten eine Buchstabierung als vom umgebenden Text unabhängig[2] und fassen alle Buchstaben in einer Klasse <L> zusammen. Wenn man jeden Buchstaben im Trainingstext durch das Symbol <L> ersetzt, erhält man neue Wahrscheinlichkeiten $P(w_j|w_i)$, $P(\text{<L>}|w_i)$, $P(w_j|\text{<L>})$ und $P(\text{<L>}|\text{<L>})$. Die Wahrscheinlichkeit $P(L_j|w_i)$ wird

[2] Ausnahme: Es wird ein Name buchstabiert, der unmittelbar zuvor fließend gesprochen wurde.

dann berechnet als Wahrscheinlichkeit, überhaupt einen Buchstaben nach w_i anzutreffen, multipliziert mit der Wahrscheinlichkeit, daß es sich im Falle eines Buchstabens um L_j handelt, also $P(L_j|w_i) = P(\texttt{<L>}|w_i) \cdot P_{\texttt{<L>}}(L_j)$.
Somit hängt die Schätzung von $P(L_j|w_i)$ nicht mehr davon ab, welche Buchstaben nach w_i (zufällig gerade) in den Trainingstexten beobachtet wurden. Weiterhin können statt der wenigen Trainingsbeispiele zur Schätzung der $P(L_j|L_i)$ externe Quellen, beispielsweise beliebig große Listen von Nachnamen, hinzugezogen werden.

(C) Statt jeden einzelnen Buchstaben in eine Klasse abzubilden, kann auch die gesamte Buchstabensequenz als Klasse `<LS>` modelliert werden. Man erhält dann Statistiken für $P(w_j|w_i), P(\texttt{<LS>}|w_i)$ und $P(w_j|\texttt{<LS>})$. Im Unterschied zu (B) handelt es sich bei `<LS>` nicht um eine Wortklasse, sondern um ein Untersprachmodell, in dem beispielsweise auch die Länge oder die Position einzelner Buchstaben berücksichtigt werden kann.

Eine zusätzliche Wortklasse oder ein Subsprachmodell, wie sie durch `<L>` bzw. `<LS>` definiert werden, kann prinzipiell während eines Erkennungslaufs direkt in der Suche berücksichtigt werden, allerdings erfordert dies eine explizite und aufwendige Modellierung der Suchalgorithmen. In JANUS ist diese Option (noch) nicht vorgesehen, deshalb benutzen wir die allgemeinere, wenngleich weniger effiziente Methode, das Untersprachmodell gewissermaßen anhand des Standard-N-Gramm-Sprachmodells zu simulieren, wie im nächsten Abschnitt beschrieben.

8.3.1 Buchstabierungen als Untersprachmodell

Der im folgenden genauer betrachtete Fall (C) ist nicht nur auf Buchstabensequenzen, sondern allgemeiner auf häufig auftretende, in sich geschlossene Phrasen anwendbar, wie beispielsweise Terminangaben: Etwa kann „`Treffen wir uns am Montag um sechzehn Uhr dreißig`“ zu „`Treffen wir uns am <Wochentag> um <Uhrzeit>`“ oder sogar zu „`Treffen wir uns <Termin>`“ generalisiert werden. Analog werden Buchstabensequenzen durch das Symbol `<LS>` ersetzt:

„`Mein Name ist Quell, geschrieben Q U E L L, ich hätte gerne ...`“
„`Mein Name ist Quell, geschrieben <LS>, ich hätte gerne ...`“

Im Sprachmodell tritt an die Stelle von `<LS>` dann ein zweites, unabhängiges Untersprachmodell. Formal betrachtet haben wir die folgende Situation:

Gegeben ist ein Text T über dem Vokabular $V_W = \{\vdash, \dashv, w_1, w_2, \ldots\} \cup \{\texttt{<LS>}\}$. Die Symbole $\vdash$ und $\dashv$ stehen für Satzanfang und -ende. Das Subsprachmodell `<LS>` steht beispielsweise für eine Zeitangabe oder Buchstabensequenz. Das aus T berechnete Sprachmodell LM_W enthält die Wahrscheinlichkeiten:

$P_W(w_j\|w_i)$	Transition innerhalb „normaler“ Wörter
$P_W(\texttt{<LS>}\|w_i)$	Transition in das Subsprachmodell hinein
$P_W(w_j\|\texttt{<LS>})$	Transition aus dem Subsprachmodell heraus

<LS> repräsentiert die Buchstabensequenz, also einen Text über einem Vokabular $V_L = \{\vdash, \dashv, L_1, L_2, \ldots\}$, mit einem von LM_W unabhängigen Sprachmodell LM_L:

$P_L(L_j\|L_i)$	Transitionen innerhalb des Subsprachmodells
$P_L(L_j\|\vdash)$	Wahrscheinlichkeit für L_j als erstes Wort von <LS>
$P_L(\dashv\|L_i)$	Wahrscheinlichkeit für L_i als letztes Wort von <LS>

Gesucht wird ein neues Sprachmodell LM über dem Vokabular $V = V_L \cup V_W$. Um LM_W und LM_L zu einem gemeinsamen Sprachmodell LM zu verschmelzen, müssen die folgenden Wahrscheinlichkeiten berechnet werden:

$$P(w_i) := P_W(w_i) \tag{8.1}$$
$$P(L_i) := P_L(L_i|\vdash) \cdot P_W(\texttt{<LS>}) \tag{8.2}$$

$$P(w_j|w_i) := P_W(w_j|w_i) \tag{8.3}$$
$$P(L_j|w_i) := P_W(\texttt{<LS>}|w_i) P_L(L_j|\vdash) \tag{8.4}$$
$$P(w_j|L_i) := P_L(\dashv|L_i) P_W(w_j|\texttt{<LS>}) \tag{8.5}$$
$$P(L_j|L_i) := P_L(L_j|L_i) \tag{8.6}$$

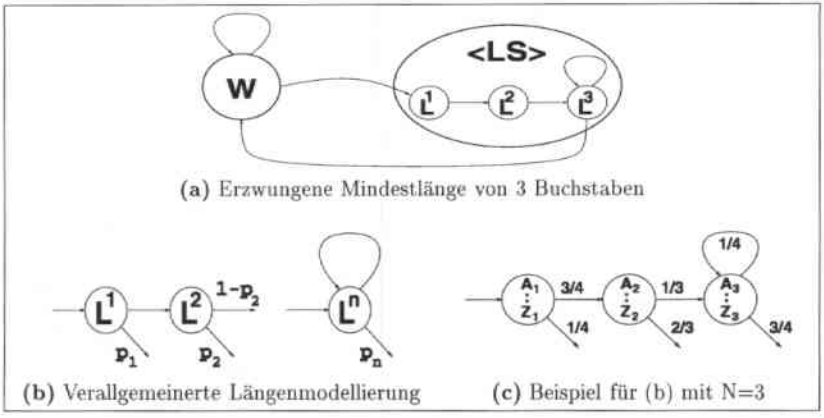
Beispielsweise wird (8.4) hergeleitet, indem wir $P(L_j|w_i) = P(L_j, \texttt{<LS>}|w_i)$ als bedingte Wahrscheinlichkeiten $P(L_j|\texttt{<LS>}, w_i) \cdot P_W(\texttt{<LS>}|w_i)$ schreiben. Wird der Beginn L_j einer Buchstabensequenz als vom vorausgehenden Wort w_i unabhängig angenommen[3], vereinfacht sich der erste Faktor zu $P_L(L_j|\vdash)$.

Für **Trigramme** gelten ähnliche Formeln:

$$P(w_k|w_i, w_j) := P_W(w_k|w_i, w_j) \tag{8.7}$$
$$P(L_k|w_i, w_j) := P_L(L_k|\vdash) \cdot P_W(\texttt{<LS>}|w_i, w_j) \tag{8.8}$$
$$P(w_k|w_i, L_j) := P_W(w_k|w_i, \texttt{<LS>}) \cdot P_L(\dashv|\vdash, L_j) \tag{8.9}$$
$$P(L_k|w_i, L_j) := P_L(L_k|\vdash, L_j) \tag{8.10}$$
$$P(w_k|L_i, w_j) := P_W(w_k|\texttt{<LS>}, w_j) \tag{8.11}$$
$$P(L_k|L_i, w_j) := P_L(L_k|\vdash) \cdot P_W(\texttt{<LS>}|\texttt{<LS>}, w_j) \tag{8.12}$$
$$P(w_k|L_i, L_j) := P_W(w_k|\texttt{<LS>}) \cdot P_L(\dashv|L_i, L_j) \tag{8.13}$$
$$P(L_k|L_i, L_j) := P_L(L_k|L_i, L_j) \tag{8.14}$$

Analog zu (8.4) stellt (8.8) die Wahrscheinlichkeit für den Anfang einer Buchstabensequenz dar. $P_L(\dashv|\vdash, L_j)$ in (8.9) ist die Wahrscheinlichkeit, daß eine mit L_j beginnende Sequenz nur einen Buchstaben lang ist. Befindet man sich bereits in einer Buchstabensequenz wie in (8.10), dann spielt das vorausgehende Wort w_i keine Rolle mehr.

[3] Diese Annahme ist dann falsch, wenn unmittelbar vor der Buchstabensequenz der entsprechende Eigenname gesprochen wurde. Meist werden Eigennamen aber als „Neues-Wort"-Modell erkannt, so daß diese Information vom Sprachmodell ohnehin nicht genutzt werden kann.



(a) Erzwungene Mindestlänge von 3 Buchstaben

(b) Verallgemeinerte Längenmodellierung (c) Beispiel für (b) mit N=3

Abbildung 8.2: Längenmodellierung für Buchstabensequenzen

Wenn wir keine aufeinanderfolgenden und keine leeren Buchstabensequenzen erlauben[4], werden die zu fordernden Bedingungen $\sum_{x \in V_W \cup V_L} P(x|w_i) = 1$ und $\sum_{x \in V_W \cup V_L} P(x|w_i w_j) = 1$ erfüllt, wie in Abschnitt A.3 gezeigt, zusammen mit einigen Bemerkungen zur Berechnung der Backoff-Wahrscheinlichkeiten.

8.3.2 Längenmodellierung

Das Subsprachmodell <LS> erlaubt eine Modellierung der Buchstabensequenzen unabhängig von den Buchstabierbeispielen in den Trainingstexten. Werden beispielsweise Nachnamen buchstabiert, kann zur Berechnung der Buchstabenbigramme auf beliebig große externe Namenslisten zurückgegriffen werden. Ebenso kann die Länge von Buchstabensequenzen explizit in <LS> modelliert werden.

Einzelne kurze Buchstaben werden leicht an falschen Stellen eingefügt. Andererseits treten die Buchstaben praktisch immer in Gruppen auf, was eine Längenmodellierung der Buchstabensequenzen motivierte. In einem ursprünglichen Ansatz wurde eine Mindestlänge von drei Buchstaben erzwungen, indem ein einmal in eine Buchstabensequenz eingetretener Suchpfad mindestens 3 Buchstaben durchlaufen muß, bevor er die Sequenz wieder verlassen kann. Diese Randbedingung kann im Sprachmodell implementiert werden, wenn es zu jedem Buchstaben L_i drei Kopien L_i^1, L_i^2, L_i^3 gibt, die, wie in Abbildung 8.2(a) illustriert, der Reihe nach durchlaufen werden müssen.

[4] Zwei aufeinanderfolgende <LS>-Symbole würden automatisch zu einem verschmolzen werden. Leere Buchstabensequenzen tauchen erst gar nicht als <LS> auf.

Das verallgemeinerte Modell in Abbildung 8.2(b) erzwingt keine konstante Mindestlänge, sondern kann nach jedem Buchstaben verlassen werden, allerdings mit unterschiedlichen Wahrscheinlichkeiten. Im Zustand L^k verläßt man das Modell mit der Wahrscheinlichkeit p_k oder kommt mit der Wahrscheinlichkeit $1 - p_k$ in den nächsten Zustand. Sei $r_i := P(l = i)$ die Wahrscheinlichkeit für eine Buchstabensequenz der Länge $l = i$. Offensichtlich hat gemäß diesem Modell eine Sequenz der Länge eins gerade die Wahrscheinlichkeit $r_1 = p_1$. Für $k = 2, \ldots, n$ kann r_k rekursiv berechnet werden:

$$r_k = p_k \prod_{i=1}^{k-1}(1-p_i) = \frac{p_k}{p_{k-1}} \cdot \underbrace{p_{k-1} \prod_{i=1}^{k-2}(1-p_i)}_{r_{k-1}}(1-p_{k-1}) = r_{k-1} \cdot \frac{p_k(1-p_{k-1})}{p_{k-1}} \qquad (8.15)$$

Die Wahrscheinlichkeit für Sequenzen gleich oder länger als n ist

$$P(l \geq n) = \sum_{i=n}^{\infty} r_i = \sum_{i=0}^{\infty} r_n(1-p_n)^i = r_n \frac{1}{p_n} = p_n \prod_{i=1}^{n-1}(1-p_i)\frac{1}{p_n} = \prod_{i=1}^{n-1}(1-p_i)$$

p_n hat also keinen Einfluß auf $P(l \geq n)$, bestimmt aber, wie schnell die Wahrscheinlichkeiten r_k für $k = n, n+1, \ldots$ abklingen. Die Parameter p_k für eine gewünschte Längenverteilung $(r_1, r_2, \ldots, r_{n-1}, P(l \geq n))$ lassen sich leicht aus (8.15) bestimmen. Das Beispiel in Abbildung 8.2(c) modelliert Längen von einem, zwei oder mehr Buchstaben mit den Wahrscheinlichkeiten $\frac{1}{4}$, $\frac{3}{4}\frac{2}{3} = \frac{1}{2}$ und $\sum_{i=0}^{\infty} \frac{3}{4}\frac{1}{3}(\frac{1}{4})^i\frac{3}{4} = \frac{1}{4}$.

Für ein Längenmodell mit n expliziten Zuständen muß jeder Buchstabe L_i n-fach als $L_i^1, \ldots, L_i^n$ im Sprachmodell vertreten sein. Dabei können Buchstabenbigramme $P(L_j|L_i)$ wie folgt integriert werden:

$$\begin{aligned}
P(L_j^1|\vdash) &:= P(L_j|\vdash) \\
P(L_j^{k+1}|L_i^k) &:= P(L_j|L_i) \cdot (1-p_k)\ ,\ k = 1, \ldots, n-1 \\
P(L_j^n|L_i^n) &:= P(L_j|L_i) \cdot (1-p_n) \\
P(\dashv|L_i^k) &:= P(\dashv|L_i) \cdot p_k\ ,\ k = 1, \ldots, n
\end{aligned}$$

Alle weiteren Übergänge werden explizit verboten (indem die entsprechenden Bigramme auf sehr kleine Wahrscheinlichkeiten gesetzt werden), um unerlaubte „Abkürzungen" im Modell zu verhindern.

Ein solches Längenmodell mit $n = 5$ expliziten Zuständen wurde im JANUS-Erkenner in der VERBMOBIL-Evaluation 1996 eingesetzt [FGH+97]. In der Längenverteilung hatten Sequenzen der Länge eins und zwei sehr kleine Wahrscheinlichkeiten, während die Länge vier am wahrscheinlichsten war, da in der Trainingsmenge viele Firmenakronyme der Länge vier beobachtet wurden. Damit konnte auf den Buchstabensegmenten eine Verbesserung um einen Prozentpunkt erreicht werden.

8.3.3 Eigennamen

Im VM-Korpus wird meist buchstabiert, wenn sich ein Gesprächspartner seinem Gegenüber vorstellt. Solche Buchstabiersätze haben typischerweise die Form
„<..Text..> <mein-Name> <..Text..> <mein-Name-buchstabiert> <..Text..>"

Eigennamen stellen für die Erkennung ein spezielles Problem dar. Da ihre Anzahl praktisch unbegrenzt ist, können sie nicht ohne weiteres in das Erkennervokabular aufgenommen, sondern müssen gesondert behandelt werden. Auf der Ebene des Sprachmodells bietet sich ein ähnlicher Ansatz wie bei den Buchstabensequenzen an. Individuelle Eigennamen werden zu einer Wortklasse <PN> zusammengefaßt. Damit können aus den Trainingstexten allgemeinere und somit robustere Bigramme der Form $P(\texttt{<PN>}|\text{„Herr"})$ geschätzt werden. Akustisch wird <PN>, wie in Abschnitt 8.4.2 beschrieben, als „Neues Wort" modelliert.

Eine korrekte Erkennung oder zumindest Identifizierung der Eigennamen ist auch für die Erkennung der Buchstabensequenz wichtig, denn oft schließt sich diese direkt an den Eigennamen an und kann durch die Folgefehler eines falsch erkannten Namens in Mitleidenschaft gezogen werden. In den weiter unten beschriebenen experimentellen Ergebnissen konnte durch eine Modellierung der Eigennamen die Erkennungsrate innerhalb der Buchstabensequenzen um etwa 5 Prozentpunkte gesteigert werden.

8.4 Akustische Modellierung

Natürlich kann auch JANUS Buchstaben erkennen, indem diese wie alle anderen Wörter mit ihren phonetischen Transkriptionen in das Erkennervokabular aufgenommen werden. Es soll nun untersucht werden, wie sich das MS-TDNN als spezialisierter Buchstabiererkenner gewinnbringend in den spontansprachlichen Erkenner integrieren läßt. Der Ansatz dabei ist, mit JANUS ein Buchstabensegment zu identifizieren, das dann mit dem MS-TDNN reklassifiziert wird.

8.4.1 „Mumble"-Wörter

Der JANUS-Erkenner identifiziert Buchstabensegmente, die jedoch aufgrund von Erkennungsfehlern nicht notwendigerweise nur aus Buchstaben bestehen. Nicht-Buchstaben können vom Buchstabiererkenner nicht erkannt werden und werden deshalb automatisch auf Buchstaben abgebildet. Um diese Fehlerquelle abzuschwächen, wurde ein sogenanntes „Mumble"-Wort als allgemeines Modell zum Auffangen von Nicht-Buchstabenwörtern in das Vokabular des Buchstabiererkenners aufgenommen. Das Mumble-Wort besteht aus 5 neuen Zuständen und wurde trainiert, indem die Buchstabiersätze der VM-Trainingsmenge unter die Buchstabierungen der KA–Alph-Datenbank gemischt wurden. Dazu wurden alle „Nicht-Buchstaben" dieser Sätze als Trainingsmaterial für die akustischen Modelle des Mumble-Wortes eingesetzt.

8.4.2 Akustische Modelle für Eigennamen

Eigennamen wurden als Wortklasse <PN> ins Sprachmodell aufgenommen. Akustisch werden sie als *„Neue Wörter"* modelliert. Wir benutzen dazu eine Technik, wie sie von Kemp und Jusek in [KJ96] vorgeschlagen wurde: Ein neues Wort wird als Folge von Silben definiert, die ihrerseits als ein Netzwerk aus Phonemen modelliert sind. Dabei wird ausgenutzt, daß es im Deutschen zwar potentiell Millionen von Eigennamen, aber nur eine begrenzte Anzahl von etwa 10 000 Silben gibt [KJ96], die Anzahl möglicher Phonemsequenzen also entsprechend eingeschränkt ist. Das in Abbildung 8.3 stark schematisiert dargestellte Silbenmodell aus [KJ96] wurde nach linguistischen Gesichtspunkten entworfen. Ähnliche Modelle kann man bauen, indem Bi- oder Trigramme über den Phonemsequenzen von potentiellen Eigennamen berechnet werden.

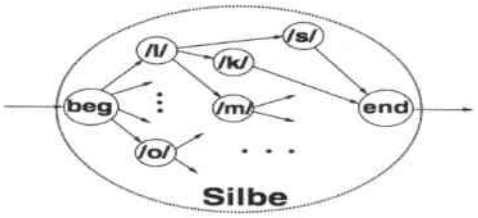


Abbildung 8.3: Silbe eines neuen Wortes, repräsentiert als Netzwerk aus Phonemen

Ein konkretes Silbenmodell mit 464 Zuständen (Phonemen) und insgesamt 11391 Transitionen wurde uns freundlicherweise von Kemp und Jusek zur Verfügung gestellt und wird in den noch zu beschreibenden Experimenten als „Neues-Wort"-Modell eingesetzt.

8.5 Integration der reklassifizierten Ergebnisse

Die Zusammenarbeit von JANUS und MS-TDNN funktioniert wie folgt: Zuerst erkennt JANUS den vollständigen Buchstabiersatz mit den oben beschriebenen Sprachmodellen für Buchstabensequenzen. Danach werden die von JANUS als Buchstabensegmente erkannten Sprachabschnitte dem MS-TDNN zugeführt und erneut erkannt.

Reklassifikation der 1-Besten-Hypothesen

Im einfachsten Fall wird nur die beste von JANUS erkannte Hypothese zur Reklassifikation der Buchstabensegmente herangezogen. JANUS liefert neben der Wortsequenz auch die Grenzen jedes Wortes, anhand derer die Buchstabensegmente aus der Spracheingabe ausgeschnitten werden können. Diese werden dem MS-TDNN zugeführt, erneut

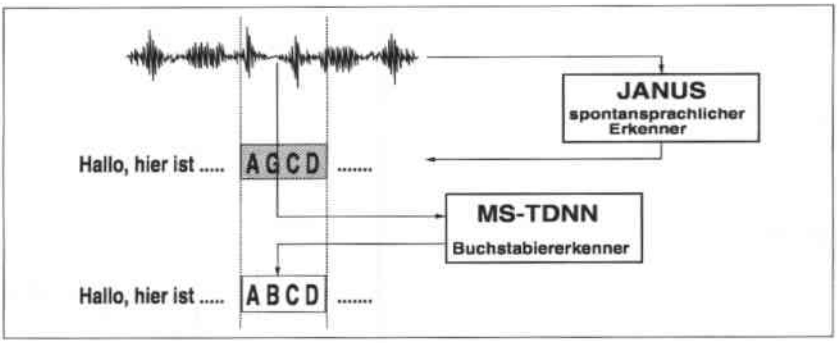


Abbildung 8.4: Reklassifikation von Buchstabensegmenten mit dem MS-TDNN

erkannt und ersetzen dann in der Hypothese die alten Buchstabensequenzen, wie in Abbildung 8.4 illustriert.

Reklassifikation von Wort-Hypothesengraphen

Im BMBF-Verbundprojekt VERBMOBIL werden spontansprachliche Terminabsprachen erkannt und übersetzt. Der VM-Forschungsprototyp besteht aus zahlreichen Software-Modulen, die von den universitären und industriellen Projektpartnern entwickelt und vom DFKI[5] zu einem Gesamtsystem integriert wurden. Neben Teilen von JANUS ist auch der Buchstabiererkenner in das Projekt eingebunden. Zwischen den Modulen des Prototypen werden Erkennungsergebnisse in einem standardisierten Format als *Worthypothesengraphen (WHG)* ausgetauscht. Ein WHG repräsentiert die Ergebnisse der Suche in einer informationsreicheren, allgemeineren Form als die beste oder eine Liste der besten Hypothesen. Ein Beispiel für einen WHG ist in Abbildung 8.5 gezeigt. An jeder Kante des WHG stehen ein hypothetisiertes Wort und seine akustische Bewertung. Die Knoten indizieren den zeitlichen Verlauf als den Beginn oder das Ende einer Kante bzw. eines Wortes. Aufgrund der Bewertungen an den Kanten kann aus dem WHG die beste Hypothese (oder allgemeiner eine N-Besten-Liste von Hypothesen) extrahiert werden. Dabei erlaubt der WHG eine wesentlich kompaktere Darstellung als ein explizites Auflisten der N besten Hypothesen.

Ein Buchstabensegment in einer einzelnen Satzhypothese zu ersetzen ist einfach. Im WHG stellt sich die Situation schwieriger dar:

- Typischerweise gibt es eine relativ große Anzahl möglicher Buchstabierpfade im WHG. Statt einer einzelnen Reklassifikation werden nun Hunderte benötigt.

[5] Deutsches Forschungszentrum für Künstliche Intelligenz

- Ein Buchstabierpfad im Graphen kann nicht ohne weiteres durch die reklassifizierte Sequenz ersetzt werden, da der originale Pfad mit zahlreichen anderen ein- und ausgehenden Kanten verbunden ist, und es nicht immer eindeutig ist, wo diese Kanten in der neuen Sequenz anzubringen sind.
- Die akustischen Bewertungen des MS-TDNN- und des JANUS-Erkenners sind nicht zueinander kompatibel.

Um Buchstabensegmente in einem WHG zu reklassifizieren, gehen wir folgendermaßen vor: Zuerst werden im WHG alle Subgraphen bestimmt, die ausschließlich Buchstaben (sowie die Wörter für Stille und diverse Geräusche, um eine zu starke Fragmentierung zu vermeiden,) enthalten (Abbildung 8.6 oben). In jedem Subgraphen gibt es einige Ein- und Austrittsknoten und dazwischen eine große Anzahl verschiedener Buchstabierpfade. Diese werden ersetzt durch Pfade, die von jedem Eintrittsknoten zu jedem Austrittsknoten laufen und auf dem entsprechenden akustischen Segment mit dem MS-TDNN reklassifiziert wurden (Abbildung 8.6 unten). Werden für einen gegebenen Eintrittsknoten die n Austrittsknoten zeitlich sortiert, können mit einer inkrementellen Vorgehensweise alle n entsprechenden Sprachsegmente in einem Durchlauf erkannt werden. Um kompatibel mit den akustischen Bewertungen von JANUS zu bleiben, erhalten die Pfade die JANUS-Bewertung des jeweils besten Buchstabenpfades, der im originalen WHG zwischen einem bestimmten Paar von Ein- und Austrittsknoten gefunden wurde.

Die Darstellung im WHG erlaubt eine Verarbeitung der unterschiedlichen Buchstabensequenzen in nachfolgenden Modulen, unabhängig davon, ob es die originalen oder die ersetzten Pfade sind. Insbesondere ist denkbar, die gefundenen Buchstabensequenzen mit einem eventuell früher erkannten Eigennamen abzustimmen. Umgekehrt könnte anhand der gefundenen Buchstabensequenzen die Aussprache für einen Eigennamen approximiert werden, was dessen Identifikation erleichtern würde.

8.6 Experimentelle Resultate

Das MS-TDNN wurde mit der KA–Alph-Datenbank, wie in Abschnitt 6 beschrieben, trainiert. JANUS wurde mit 7335 Sätzen, davon 255 Buchstabiersätze, der VM-Trainingsmenge trainiert.

Die Testmenge bestand aus 115 neuen Buchstabiersätzen von der VM CD-ROM 4. Die VM-Sprachdaten wurden nicht in Karlsruhe, also unter anderen Bedingungen aufgenommen als die KA–Alph-Daten. JANUS wurde mit einem wie in Abschnitt 8.3 beschriebenen Sprachmodell (Buchstabier-Subsprachmodell aus externer Namensliste, 3 erzwungene Buchstaben) und wahlweise mit oder ohne „Neues-Wort"-Modelle betrieben. Beschränkt man die Auswertung auf die Buchstabensegmente, erhält man die in Tabelle 8.1 aufgelisteten Ergebnisse. In Spalte 1 stehen die Ergebnisse auf den von JANUS erkannten Buchstabensegmenten. In den Spalten 2 bis 4 wurden die Buchstabensegmente mit

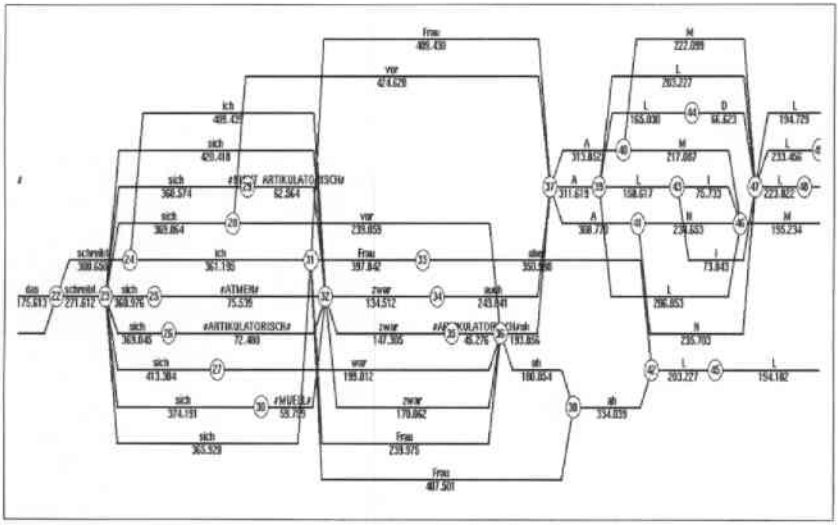


Abbildung 8.5: Ausschnitt aus einem Worthypothesengraphen (WHG)

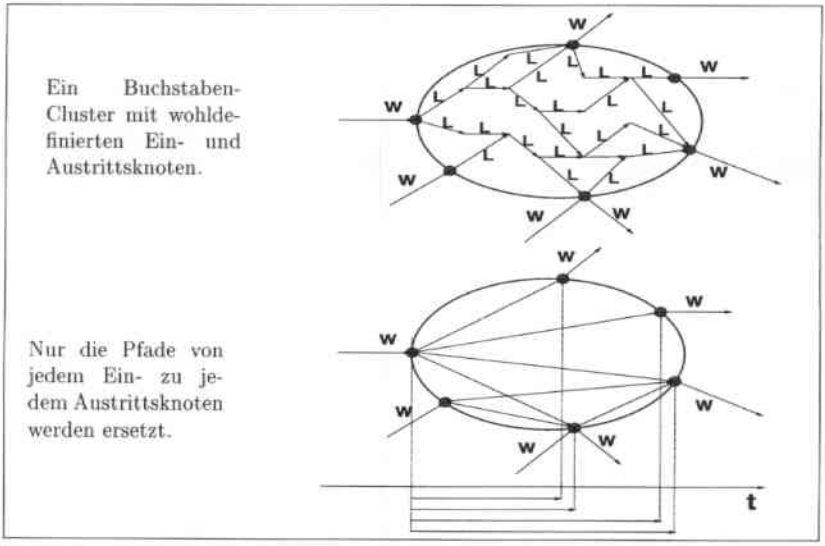


Abbildung 8.6: Reklassifikation von Buchstaben-Clustern im WHG

	JANUS	+ MS-TDNN-Reklassifikation		
			+ Nachtraining	+ „Mumble"-Wörter
3 erzwungene Buchstaben	**77.0**	79.1	82.0	83.0
+ „Neues-Wort"-Modelle	**78.3**	79.8	82.2	**83.5**

Tabelle 8.1: Buchstabenakkuratheit auf Buchstabensegmenten

dem MS-TDNN reklassifiziert. Um vergleichbare Ergebnisse zu erhalten, wurde dasselbe Sprachmodell wie in JANUS eingesetzt. Spalte 2 zeigt die Auswirkung der Reklassifikation mit einem MS-TDNN, das ausschließlich auf den Karlsruher Buchstabierdaten trainiert wurde. Da alle VM-Daten von anderen Aufnahmeorten stammen, wurde der Buchstabiererkenner mit den Buchstabensegmenten aus den VM-Trainingsdaten nachtrainiert. Obwohl es sich dabei nur um 255 Buchstabierungen handelte, erwies sich ein Nachtrainieren als hilfreich (Spalte 3). Wird dem MS-TDNN mit den „Mumble"-Wörtern die Möglichkeit gegeben, Nicht-Buchstaben zu identifizieren und als solche zu ignorieren, ergibt sich eine weitere Verbesserung (Spalte 4). Die ersten Experimente zu dieser Aufgabenstellung wurden in [HW95] vorgestellt, allerdings mit einem früheren JANUS-System und entsprechend niedrigeren Erkennungsraten.

Eine erzwungene Mindestlänge von drei Buchstaben hat keine Auswirkungen auf die Erkennung der Buchstabiersätze, vermeidet aber die Erkennung von Buchstaben in Sätzen, die keine enthalten. In einem Test auf 331 Nicht-Buchstabensätzen wurde mit einer erzwungenen Mindestlänge von drei Buchstaben nur einmal versehentlich ein Block von drei Buchstaben fälschlich eingefügt; ohne diese Rahmenbedingung waren es insgesamt neun Buchstaben.

8.7 Spezifizierte Buchstaben, Assoziationen

Gesprochene Buchstaben sind nicht immer einfach zu verstehen – deshalb werden vom Sprecher oft zusätzliche Bemühungen unternommen, sie verständlich zu machen, wie an einigen Beispielen in Tabelle 8.2 illustriert ist. Insbesondere in sicherheitsrelevanten Anwendungen wie dem Flugverkehr wird statt der verwechselbaren Buchstaben das *Funkeralphabet* (Bravo Charlie Delta ...) benutzt, das in Anhang B.3 in mehreren Versionen aufgeführt ist. Wer mit solchen speziellen Konventionen nicht vertraut ist, behilft sich mit allgemeinen Lautassoziationen, etwa „A wie (in) Apfel". Die mit diesen Konstrukten verbundene Redundanz erlaubt dem Menschen eine robustere Kommunikation.

Im folgenden sind einige Überlegungen angestellt, wie solche Konstrukte erkannt werden

D	nur ein Buchstabe
Dora	nur ein assoziiertes Wort
D wie Dora	
D wie in Dora	Buchstabe und Assoziation
Hotel India Lima Delta	Englisches Funkeralphabet
Heinrich Ida Ludwig Dora	Deutsches Funkeralphabet
H wie in Haus, I, L, D wie Dora	Mischform

Tabelle 8.2: Beispiele für spezifizierte Buchstaben

könnten. Eine experimentelle Untersuchung dieser Phänomene war im Rahmen dieser Arbeit nicht möglich, insbesondere da zu diesem Themenbereich keine Sprachdaten zur Verfügung standen.

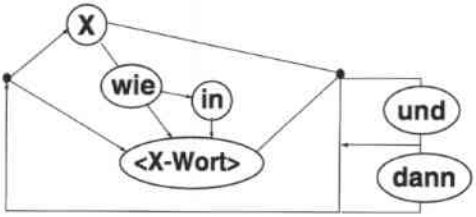


Abbildung 8.7: Grammatik für spezifizierte Buchstabierungen

Eine kleine Grammatik, die eine Erkennung von allgemeinen Buchstabierungen der Form „`H wie in Haus, E, R, M wie in Maus, und dann Anton Nordpol Nordpol`“ erlaubt, ist in Abbildung 8.7 dargestellt. Wenn man nur das Funkeralphabet zuläßt, entspricht `<A-Wort>` „`Anton`“, `<B-Wort>` „`Berta`“ usw. Ohne diese Beschränkung hat man das Problem eines offenen Vokabulars, denn dann kann `<X-Wort>` jedes Wort mit Anfangsbuchstabe X sein.

Mangels Daten kann nur darüber spekuliert werden, ob vielleicht manche Wörter bevorzugt für Lautassoziationen herangezogen werden:

- Wörter aus bestimmten Kategorien, wie zum Beispiel Vornamen, Städte oder Länder.
- Wörter, in denen der zu assoziierende Laut deutlich artikuliert ist. Das A in „Ameise“ ist gedehnt und daher besser zu verstehen als das A in „Abfall“.
- Wörter sollten nicht nur mit denselben Buchstaben, sondern auch mit denselben Phonemen wie das entsprechende Buchstabenwort beginnen: „`V wie Vogel`“ statt „`V wie Vase`“, „`E wie Esel`“ statt „`E wie Ei`“, obwohl interessanterweise in den Funkeralphabeten (siehe Tabelle B.6) diese Regel im Fall „`Viktor`“ nicht zutrifft.

Buchst.	Anfangsphonem (Häufigkeit)	Buchst.	Anfangsphonem (Häufigkeit)
a	A(3594) AU(1694) AH(145)	p	P(2077) F(169)
b	B(2653)	q	K(162)
c	K(62) CH(11)	r	R(2226)
d	D(1225)	s	SCH(4351) Z(2225) S(161) ZU(11)
e	AE(1268) AI(760) EH(338) EU(30)	t	T(2484)
f	F(1627)	u	U(2049) UH(189)
g	G(1768)	v	F(2339) V(293)
h	H(1876)	w	V(2168)
i	I(840) IE(106)	x	–
j	J(217) D(10)	y	J(11)
k	K(2698)	z	T(1702)
l	L(1587)	ä	AE(84) AEH(23) EU(12)
m	M(2140)	ö	OEH(45) OE(14)
n	N(1367)	ü	UEH(631)
o	O(258) OH(232)		

Tabelle 8.3: Anfangsphoneme von Wörtern, die mit den Buchstaben A ... Z beginnen

- Kurze, einfache, alltägliche Wörter werden bevorzugt: „`M wie Maus`“ ist wahrscheinlicher als „`M wie Müllverbrennungsanlage`“.

Eine andere Alternative ist, daß man erst gar nicht versucht, alle potentiellen Kandidaten für `<A-Wort>` ... `<Z-Wort>` zu erraten, sondern diese gleich als „Neues-Wort“-Modell behandelt. Statt eines konkreten Wortes erhält man dann eine Sequenz von Phonemen oder Silben. Um daraus den umschriebenen Buchstaben zu erkennen, kann man entweder die Phonem- oder Silbensequenz auf das am besten passende Wort abbilden, oder man zieht das erste Phonem der Sequenz heran, um daraus auf den intendierten Buchstaben zu schließen. Allerdings ist dies nicht immer eindeutig möglich: Beispielsweise beginnen „Kaufmann“ und „Quelle“ mit demselben Phonem. In Tabelle 8.3 sind die Häufigkeiten der Anfangsphoneme zu einem bestimmten Anfangsbuchstaben aufgelistet, bestimmt anhand eines Wörterbuchs mit 50 000 Einträgen[6].

8.8 Fließend gesprochene und buchstabierte Namen

Häufig wird ein Name fließend gesprochen, bevor er buchstabiert wird. Dann liegt dieselbe Information[7] in zwei unterschiedlichen Repräsentationen vor. Kann diese zusätzliche Redundanz genutzt werden?

[6] Anfangsphoneme, die weniger als zehn Mal auftreten, sind nicht berücksichtigt.

[7] Genaugenommen liegt beim buchstabierten Namen mehr Information vor, da selbst ein optimal erkannter (verstandener) fließend gesprochener Name mehrere mögliche Schreibweisen haben kann.

Wir betrachten zuerst drei Szenarien mit steigendem Schwierigkeitsgrad, in denen ein Benutzer nach einem Namen gefragt wird. Die gewünschte Information (fließend gesprochener und buchstabierter Name) kann auf folgende Arten vorliegen:

1. Als zwei getrennte Aufnahmen, etwa im Dialog einer Telefonauskunft:
   „`Bitte nennen Sie den Namen:`" → „**Maier**"
   „`Bitte buchstabieren Sie den Namen:`" → „**M A I E R**"

2. An einem Stück fließend gesprochen und buchstabiert:
   „`Bitte sprechen und buchstabieren Sie den Namen`"
   → „**Maier, M A I E R**"

3. Verstreut in freiem Text:
   „`Wen möchten Sie sprechen?`"
   → „Bitte verbinden Sie mich mit **Maier**, geschrieben **M A I E R**."

Die ersten beiden, als Szenario 1 und 2 bezeichneten Fälle sind im folgenden genauer untersucht. Zu Szenario 2 wurden knapp 3000 Sprachaufnahmen gesammelt, in denen 57 Sprecher einen Namen erst fließend und dann buchstabiert in einer Aufnahme sprechen. Zu diesen Daten wurden halbautomatisch die Trenngrenzen zwischen den fließenden und den buchstabierten Namen bestimmt, so daß damit auch Szenario 1 simuliert[8] werden kann.

Es ist nicht immer einfach, korrekt zu buchstabieren – in etwa 3% der Fälle wurde der (schriftlich vorliegende) Nachname falsch buchstabiert. Neben schlichten Konzentrationsfehlern sind typische Fehler vergessene Buchstaben, wie in „`karolus k r o l u s`" (interessanterweise passiert dies insbesondere dann oft, wenn der vergessene Buchstabe lautlich bereits ein Teil des vorangegangenen Buchstabens war, also etwa A nach K, oder E nach D), oder klanglich verwechselte Buchstaben, wie in den Fällen (echte Beispiele) „`campos k a m p o s`" oder „`vogel f o g e l`". Weitere Details zu dieser Sprachdatenbank finden sich in Abschnitt 5.7.

Szenario 3 kann auf Szenario 1 reduziert werden, indem jeweils der fließend gesprochene und der buchstabierte Name im gesprochenen Satz lokalisiert und dann als getrennte Aufnahmen behandelt werden. Natürlich kommt damit erschwerend hinzu, daß diese Sprachsegmente nicht immer fehlerfrei gefunden und zudem nicht isoliert, sondern im Kontext gesprochen werden.

Die Erkennung fließend gesprochener Eigennamen ist ein Problem für sich. Allein in Deutschland gibt es ca. eine Million unterschiedliche Nachnamen. Bevor ein Name in das Erkennervokabular aufgenommen werden kann, muß natürlich seine Aussprache, also seine phonetische Transkription bekannt sein. Im Rahmen des von der Europäischen

[8] Die Situation entspricht nicht ganz Szenario 1, da trotz bekannter Grenze Koartikulationseffekte zwischen den in einem Atemzug fließend gesprochenen und buchstabierten Namen auftreten können.

Union geförderten ONOMASTICA-Projektes [SFSJ93] wurden umfangreiche Aussprachewörterbücher für Europäische Eigennamen[9] gesammelt, von denen inzwischen in 11 Sprachen fast jeweils eine Million in verschiedenen Qualtitätsstufen transkribiert wurden. Der Universität Karlsruhe wurde für Forschungszwecke von der Deutschen Telekom und der TU Berlin die Hälfte aller verfügbaren deutschen Nachnamen zur Verfügung gestellt.

Nachfolgend werden einige erste Experimente zur Erkennung gesprochener und buchstabierter Nachnamen vorgestellt, die im Rahmen einer Diplomarbeit von Michael Meyer [Mey97] untersucht und in [MH97] veröffentlicht werden. Wie bei den Experimenten mit buchstabierten Nachnamen wird davon ausgegangen, daß die Liste zu erkennender Namen bekannt ist. Je nach deren Größe ergeben sich Situationen, die unterschiedliche technische Lösungen erfordern, welche in den beiden folgenden Abschnitten untersucht werden:

- Sind die Namenslisten relativ klein (z.B. $< 10\,000$) und ihre Transkription bekannt, können sie explizit in das Wörterbuch des Erkenners aufgenommen und wie bei einem „normalen" Einzelwort-Erkennungsproblem behandelt werden.
- Größere Listen (z.B. 200 000 Namen) kann das Wörterbuch des Erkenners nicht mehr fassen. Abhilfe schafft ein Erkenner, der nur Phoneme und/oder Buchstaben erkennt und die gegebenen Einschränkungen in vollem Umfang erst in einem zweiten Erkennungsdurchlauf nutzt. Mit zunehmender Listengröße werden nicht mehr für alle Namen phonetische Transkriptionen bekannt sein.

8.8.1 Kleine Namenslisten

Von den 2780 gesammelten Sprachaufnahmen mit fließend gesprochenen und buchstabierten Eigennamen sind durch die uns zur Verfügung gestellte Untermenge des ONOMASTICA-Wörterbuches etwa die Hälfte abgedeckt. Diese 1337 Namen dienen als Testmenge für die folgenden Versuche.

Getrennte Erkennung

Zur **Erkennung der fließend gesprochenen Eigennamen** wurde der JANUS-Erkenner eingesetzt, und zwar in der Version, mit der die Universität Karlsruhe an der VERBMOBIL-Evaluation 1996 teilnahm, und die mit einer Wortakkuratheit von über 85% bei einem Vokabular von 5 000 Wörtern als bestes System hervorging[10]. Von den 1337 Eigennamen konnte derselbe Erkenner 60% korrekt erkennen. Obwohl es sich dabei nicht um kontinuierliche Sprache, sondern um Einzelworterkennung handelt, mußte

[9] Nicht nur Familien-, sondern auch Straßen- und Städtenamen

[10] Für die hier beschriebenen Versuche wurde ein zwei Prozentpunkte schlechteres, aber dafür doppelt so schnelles System benutzt.

Abel Abendschein Adams Adler Agha Akkoc Aksu Albiez Alesi Alexakis Alilovic Allgeier Alphan Ammersbach Anselm Apostolidis Appelt Artuso Asmus Attrasch Aubert Augustin Avci Aydogan Azad Böhm Böhme Böhnke Büchner Bühler Bürk Bacher Baier Baltz Baranowski Baron Barteczko Barthlott Bartholomaeus Bartl Bastian Baumann Baumgärtner Baumhakl Baumhauer Baumheier Bauser Baydaroglu Bayer Bayram Beck Becker Bednar Bein Beinert Beisert Bello Benda Benthien Benvegnu Berber Berger Bergner Berndt Besserdich Bestelmeyer Beyer Bicheler Bieler Bier Birn Bischof Bitsch Bittner Black Blankenburg Blaschke Bleile Blenk Bochnig Bok Bollin Borchardt Bordt Borger Born Bornhauser Borrmann Boudjema Brand Brandstetter Brehm Breiller Brennenstuhl Brinkmann Brkic Broch Bruch Bruder Bruker

Tabelle 8.4: Liste der ersten 100 der 1337 Nachnamen der Testmenge

etwas überraschend eine deutliche Leistungseinbuße hingenommen werden, die aber auf eine ganze Reihe von Ursachen zurückgeführt werden kann. Die Perplexität der in etwa gleichwahrscheinlichen Eigennamen liegt bei über 900 und ist damit um einen Faktor 20 größer als die der VM-Daten. Außerdem ist der Erkenner nicht auf Einzelwörtern, sondern auf kontinuierlicher und spontaner Sprache trainiert. Erschwerend kommt hinzu, daß der Erkenner ausschließlich Wörter erkennen muß, auf denen er niemals trainiert wurde, zumal unklar ist, inwieweit die ONOMASTICA-Transkriptionen mit den Konventionen des JANUS-Wörterbuches konsistent sind. Weiterhin enthalten Eigennamen viele untypische Phonemsequenzen. Dies trifft insbesondere auf ausländische Familiennamen zu, deren Aussprache oft recht uneinheitlich gehandhabt wird, die aber in den Daten zu einem nicht unbeträchtlichen Teil vertreten sind, wie aus Tabelle 8.4 ersichtlich ist.

Bei der **Erkennung der buchstabierten Namen** treten obige Probleme nicht auf. Um die 1337 Nachnamen in buchstabierter Form zu erkennen, werden ihre phonetischen Transkriptionen in das Wörterbuch von JANUS eingetragen, also etwa [`Lang E L - AH - E N - G EH`] für den Namen „`Lang`", wobei die Bindestriche für optionale Pausen stehen. Mit diesem Wörterbuch konnte JANUS 93.3% aller buchstabierten Namen korrekt erkennen. Das auf den KA-Alph-Daten trainierte MS-TDNN konnte aus der Liste dieser 1337 Namen mit der in Abschnitt 7.4 beschriebenen Baumsuche 96.5% der buchstabierten Namen korrekt erkennen.

Kombinierte Erkennung

Wie kann nun die Erkennung der buchstabierten und fließend gesprochenen Namen kombiniert werden? Die Darstellung beider Sprecharten ist nicht so orthogonal wie man zuerst denken mag. Schließlich repräsentiert die Aussprache der einzelnen Buchstaben in etwa diejenigen Laute, für die sie im gesprochenen Wort stehen: Die akustischen Realisierungen von „`Haus`" und „`H A U S`" sind recht ähnlich. Ausnahmen hiervon sind einerseits diejenigen Buchstaben, die nicht direkt einen Laut verkörpern, wie „Ypsilon" oder das englische „Double-U", und andererseits bestimmte Buchstabenkombina-

tionen, die einen eigenen Laut verkörpern oder stumme Buchstaben enthalten, wie etwa „sch,ch,ck,th,pf,ph,ie". Inwieweit diese Zusammenhänge zwischen den Lautrealisierungen im gesprochenen und buchstabierten Namen in Form expliziter Regeln zu einer Kombination der Erkennungsergebnisse genutzt werden können, ist fraglich. Im folgenden ist der weit weniger komplexe Ansatz beschrieben, die jeweiligen Hypothesen ausschließlich auf der Grundlage ihrer akustischen Bewertungen zu kombinieren.

Wir betrachten zuerst **Szenario 1**, also eine getrennte Aufnahme beider Sprecharten. Sei $Y_B(i)$ die Bewertung eines buchstabierten Namens i aus der N-Besten-Liste des MS-TDNN und $Y_F(i)$ die Bewertung desselben Namens fließend gesprochen in der N-Besten-Liste von JANUS. Aus beiden N-Besten-Listen kann eine neue berechnet werden, indem die Bewertungen linear kombiniert werden:

$$Y(i) = \lambda * Y_B(i) + (1 - \lambda) * Y_F(i) \tag{8.16}$$

Als Erkennungsergebnis zählt dann der Name i mit der höchsten neuen Bewertung $Y(i)$. Obwohl die Buchstabiererkennung mit 95.6% deutlich besser war als die Erkennung der gesprochenen Namen mit nur 60%, konnte mit einem λ-Faktor[11] in der Umgebung von 0.9995 eine kleine Verbesserung um 0.5 Prozentpunkte erreicht werden. Eine gewinnbringendere Technik schränkt die Erkennung der fließend gesprochenen Namen von vorneherein auf diejenigen ein, die sich bereits in der N-Besten-Liste des MS-TDNN befanden. Bei einer Gewichtung mit $\lambda = 0.96$ werden dabei 97.7% der Namen korrekt erkannt. Abbildung 8.8 zeigt die Erkennungsraten der kombinierten N-Besten-Liste in Abhängigkeit von der Gewichtung λ. Dabei entspricht $\lambda = 1$ gerade den 96.5% bei ausschließlicher Buchstabiererkennung.

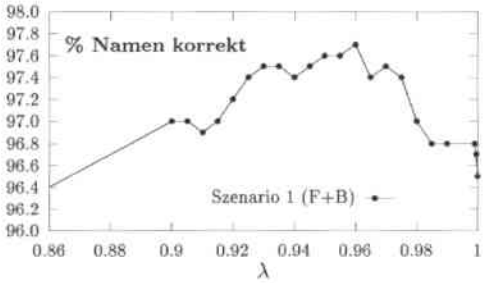


Abbildung 8.8: Mit λ gewichtete Kombination der N-Besten-Listen gesprochener und buchstabierter Namen

[11] Es ist klar, daß das Hauptgewicht auf die bessere Buchstabiererkennung fallen muß. Es muß aber auch berücksichtigt werden, daß die Bewertungen beider Erkenner nicht kompatibel sind, denn JANUS liefert etwa 10-fach höhere Werte als das MS-TDNN. Dieser Effekt trägt zusätzlich zu einem hohen λ-Faktor bei.

In **Szenario 2** liegt der gesprochene und buchstabierte Name in *einer* Aufnahme vor, und die Grenze zwischen beiden ist nicht mehr bekannt. In diesem Fall wird das Wörterbuch von JANUS mit Einträgen der Form [Lang_L_A_N_G L A NG - E L - AH - E N - G EH] gefüllt. Diese Erkennung der gesamten Aufnahme fällt mit 86.1% schlechter aus als auf dem buchstabierten Teil allein! Zwar wird durch die gesprochenen Namen zusätzliche Information zur Verfügung gestellt, allerdings ist deren Erkennung mit 60% unzuverlässig, und sie kann eine korrekte, allein auf der Buchstabierung getroffene Entscheidung überschreiben. Natürlich ist die Aufgabe insgesamt schwieriger, da die Grenzen zwischen gesprochenem und buchstabiertem Wort nicht a priori bekannt sind.

Nach der Erkennung ist eine Grenze bekannt, und man könnte eine λ-Gewichtung wie in (8.16) versuchen. Dies ist jedoch problematisch, da in den einzelnen Hypothesen der N-Besten-Liste die beiden Teile (gesprochen und buchstabiert) unterschiedlich lang sind, und daher automatisch diejenigen Hypothesen bevorzugt werden, in denen der höher gewichtete Teil möglichst lang ist. Deshalb wurde zur Gewichtung aller Hypothesen immer dieselbe Grenze der besten Hypothese benutzt, womit eine Verbesserung auf 89.6% erzielt werden konnte. Wird stattdessen das identifizierte Buchstabensegment mit dem MS-TDNN erkannt, ergibt eine Gewichtung 95.8% korrekte Namen.

8.8.2 Große Namenslisten

Ab einer bestimmten Größe ist es nicht mehr sinnvoll, alle zu erkennenden Namen explizit im Wörterbuch des Erkenners zu halten. Dann kann man sich behelfen, indem in einem ersten Schritt mit einer „gröberen" Erkennung die Anzahl möglicher Hypothesen auf eine ausreichend kleine Anzahl eingeschränkt wird, auf die dann die bereits oben beschriebenen Techniken für kleinere Wortlisten angewandt werden können. Um dies zu erreichen, wurden die gesprochenen Namen in Phoneme und die buchstabierten Namen in ihre Einzelbuchstaben aufgelöst, so daß JANUS nur noch eine Sequenz von Phonemen und Buchstaben zu erkennen braucht, also mit einem Erkennervokabular von etwa 100 Wörtern (bestehend aus Phonemen und Buchstabenmodellen) auskommt. Ein spezielles Sprachmodell sorgt dafür, daß am Anfang der Äußerung nur Phoneme und dann, nach einem Wechsel, nur noch Buchstaben erkannt werden. Dazu wurden entsprechende Trigramme aus Phonem- und Buchstabensequenzen der etwa 200 000 Namen der ONOMASTICA-Datenbank berechnet.

Im Gegensatz zu einem Wörterbuch mit den kompletten Namen wird die mit einem solchen Sprachmodell erkannte Phonem-/Buchstabensequenz nur selten unmittelbar als Name interpretierbar sein. Außerdem besteht bei der Erkennung keine Kopplung mehr zwischen gesprochenem und buchstabiertem Namen. Die erkannten Sequenzen können jedoch auf die 100 oder 1000 ähnlichsten Namen abgebildet werden, die dann (mit vollständigen Transkriptionen im Wörterbuch) erneut erkannt werden. Mit den so gewonnenen neuen, besseren Grenzen können die Buchstabensegmente wieder mit dem MS-TDNN erkannt und wie oben mit einer λ-Gewichtung mit den fließenden Namen

kombiniert werden. Wird eine große Liste mit 100 000 unterschiedlichen Namen zugrundegelegt, erzielt man auf dem buchstabierten Namen allein 87.1% und mit der Kombination 89.5% bzw. 88.1% korrekte Namen bei bekannter bzw. unbekannter Grenze[12]. Diese Ergebnisse sind mit denen für kleine Wortlisten in Tabelle 8.5 zusammengefaßt.

	Erkennung **einzeln**		Erkennung **kombiniert**	
	gesprochen (F)	buchstabiert (B)	Szenario 1 (F+B)	Szenario 2 (FB)
1337 Namen im Wörterbuch	60.0	96.5	97.7	95.8
Zweistufig, 100 000 Namen	-	87.1	89.5	88.1

Tabelle 8.5: Zusammenfassung der Ergebnisse für die Erkennung fließend gesprochener und buchstabierter Namen

8.8.3 Flexible Erkennung

Buchstabierte Namen können deutlich besser als fließend gesprochene erkannt werden. Mit einer Kombination beider Sprecharten konnte aufgrund der dominanten Rolle der Buchstabiererkennung nur für getrennte Aufnahmen (Szenario 1) eine Verbesserung der Erkennung erreicht werden.

Dessen ungeachtet erlauben die oben untersuchten Techniken eine *flexiblere* und damit benutzerfreudlichere Erkennung: Wird der Erkenner so betrieben, daß jeder zu erkennende Name alternativ *nur* buchstabiert, oder gesprochen *und* buchstabiert hypothetisiert werden kann, dann können diese beiden Fälle zu 99% korrekt unterschieden werden.

In einem Experiment dazu wurden dem Erkenner alle 1337 Namen sowohl nur buchstabiert als auch gesprochen und buchstabiert zur Erkennung präsentiert, wobei nicht a priori bekannt war, um welche Version es sich jeweils handelte. Mit 95.5% Namen korrekt konnte dabei annähernd dieselbe Erkennungsrate erzielt werden wie bei reinem Buchstabieren.

8.9 Zusammenfassung

Nicht immer ist es möglich oder wünschenswert, buchstabierte Namen ausschließlich als pure Buchstabensequenzen zu erkennen. Namen werden oft in einem Atemzug zuerst gesprochen und dann buchstabiert oder von weiterem Text begleitet. Auch während des Buchstabierens können Füllwörter („`und dann`") oder Assoziationen („`A wie Auto`") auftreten. Wir haben diese Phänomene unter dem Begriff „Buchstabieren in spontaner

[12]Leider waren für die 100 000 unterschiedlichen Namen keine Häufigkeitsverteilungen bekannt. Die Erkennungsrate von 87.1% muß daher mit den unteren beiden Kurven in Abbildung 7.13 (Erkennung für Telefonauskunft) verglichen werden.

Sprache" zusammengefaßt und davon die folgenden drei Schwerpunkte genauer untersucht.

Eingebettete Buchstabensequenzen werden von „normalem" Text begleitet, der nicht mit dem Buchstabiererkenner, sondern nur mit einem Spracherkenner für große Wortschätze erkannt werden kann. Um dessen Sprachmodell auf die Erkennung von Buchstabensequenzen einzustellen, wurden die Buchstabensequenzen in einem Untersprachmodell <LS> zusammengefaßt. Damit erreicht man zweierlei: Zum einen können Wort-Buchstaben- und Buchstaben-Wort-Übergänge robuster geschätzt werden, da mit <LS> statt vieler konkreter Buchstabensequenzen nur ein einziges Symbol betrachtet werden muß. Außerdem können die Übergänge innerhalb der Buchstabensequenz nun unabhängig von dem eigentlichen Trainingstext mit zusätzlichen, externen Texten geschätzt werden. Zum anderen können die so erkannten Buchstabensequenzen für einen zweiten Erkennungsschritt an den spezialisierten Buchstabiererkenner weitergereicht und korrigiert wieder in die Hypothese eingefügt werden.

Buchstaben können mit **Wortassoziationen** näher spezifiziert werden („`H wie Haus`"). Solche Konstrukte kann man mit relativ einfachen Sprachmodellen nachbilden – das eigentliche Problem besteht darin, daß zusätzlich zu den Wörtern des Funkeralphabets beliebige Mengen weiterer Wörter benutzt werden können, die mit ihren Anfangsbuchstaben die gewünschte Assoziation herstellen. Da ein unbegrenzter Wortschatz in der Praxis nicht möglich ist, muß auf phonetischer Ebene erkannt oder eine geeignete Untermenge erlaubter Assoziationswörter antizipiert werden.

Ein sowohl **fließend gesprochener als auch buchstabierter Name** beinhaltet zusätzliche Redundanz. Allerdings können buchstabierte Namen deutlich besser erkannt werden als fließend gesprochene. Daher können die vom fließenden Namen stammenden zusätzlichen akustischen Anhaltspunkte nur bedingt genutzt werden, denn umgekehrt besteht die Gefahr, daß eine aufgrund der Buchstabierung getroffene korrekte Entscheidung von einem falsch erkannten fließenden Namen überschrieben wird. Mit einer Gewichtung der Bewertungen zugunsten des buchstabierten Namens kann die Erkennung gegenüber der nur buchstabierter Namen geringfügig verbessert werden.
Ab einem bestimmten Umfang können nicht mehr alle Eigennamen in das Wörterbuch des Erkenners aufgenommen werden. Dann wird mit einem speziellen Sprachmodell der fließend gesprochene und buchstabierte Name zuerst nur als Sequenz von Phonemen und Buchstaben erkannt, die anschließend zu einer Liste mit den 100 oder 1000 ähnlichsten Namen erweitert wird. Auf diese Kandidatenliste können in einem zweiten Erkennungsdurchgang alle für kleine Namenslisten erprobten Techniken angewandt werden.
Ebenfalls von Vorteil ist die flexiblere Erkennung, die mit den (zur Kombination beider Sprecharten) untersuchten Algorithmen realisiert werden kann: Wird wahlweise nur buchstabiert, oder gesprochen und buchstabiert, kann dabei praktisch genausogut erkannt werden, wie wenn a priori bekannt ist, daß nur buchstabiert wird.

Kapitel 9

Systeme

Während der Entstehung dieser Arbeit wurden eine Reihe von Demonstrationssystemen entwickelt, von denen die zwei interessantesten kurz vorgestellt werden sollen. Das erste System ist ein allgemeiner Buchstabiererkenner, der auf einem Laptop installiert wurde und dank einer schnellen schritthaltenden Erkennung praktisch simultan zur Spracheingabe Namen aus großen Wortlisten erkennen kann, hier demonstriert anhand aller Stuttgarter Straßennamen. Dabei können lange Namen vorausschauend vervollständigt werden, noch bevor sie zu Ende buchstabiert wurden.
Im zweiten System ist der Buchstabiererkenner in ein Telefonauskunftssystem integriert. Ein Anrufer wird durch einen Dialog geführt, kann den Namen des gewünschten Teilnehmers buchstabieren und erfährt dann dessen Telefonnummer.

Der Beschreibung dieser beiden Systeme sind noch einige Details zur Implementierung und schritthaltenden Verarbeitung des Erkenners vorangestellt.

9.1 Schritthaltende Verarbeitung mit Vorausschau

Implementierung

Der MS-TDNN-Buchstabiererkenner ist in der Programmiersprache `C` implementiert und in das `Tcl/Tk`-Toolkit eingebettet. Die elementaren Objekte und Datenstrukturen des Erkenners sind in einzelnen Modulen gekapselt und können mit Befehlen gesteuert werden, die in einer hierarchischen Menustruktur organisiert sind und von der `Tcl`-Skriptsprache aufgerufen werden können. Dadurch wird eine flexible und interaktive Programmierung erlaubt, und in Verbindung mit `Tk` können leicht graphische Benutzerschnittstellen realisiert werden, wie die des noch vorzustellenden Demonstrationssystems in Abbildung 9.3.

Schritthaltende Verarbeitung

Statt, wie in Abschnitt 6.2 beschrieben, die komplette Eingabe Schicht für Schicht durch das MS-TDNN zu propagieren, kann man genausogut einen Sprachvektor nach dem nächsten von der Eingabe- zur Ausgabeschicht schleusen. Werden zusätzlich Vorverarbeitung und Suche auf eine inkrementelle Verarbeitung umgestellt, erhält man eine *schritthaltende* Erkennung mit zwei wesentlichen Vorteilen: Zum einen muß der Erkenner nicht mehr bis zum Ende der Aufnahme warten, sondern kann bereits während der Spracheingabe mit ihrer Verarbeitung beginnen und entsprechend früher zu einem Ergebnis kommen. Außerdem können (vorausgesetzt, der Erkenner ist schnell genug,) bereits während noch gesprochen wird, partielle Hypothesen ausgegeben werden. Mit der Baumsuche aus Abschnitt 7.4 ergibt sich sogar die interessante Möglichkeit, einen Namen automatisch zu vervollständigen, noch bevor er zu Ende buchstabiert wurde.

Die Architektur des schritthaltenden MS-TDNN ist in Abbildung 9.1 zu erkennen. Die einzelnen Schichten sind als Ringpuffer organisiert, wodurch Eingaben von unbegrenzter Länge verarbeitet werden können. Sobald ein neues Stück Sprachsignal im Eingangspuffer ankommt, wird es zu FFT-Koeffizienten vorverarbeitet und in der FFT-Schicht abgelegt. Nachdem die FFT-Koeffizienten normalisiert und längere Pausen herausgeschnitten worden sind, gelangen sie in die Eingabeschicht des MS-TDNN und von dort als Phonembewertung in die Ausgabeschicht. Die Phonembewertungen werden an alle aktiven Knoten der Baumsuche verteilt, und die partiellen Suchhypothesen werden erweitert. Die Informationsverarbeitung der schritthaltenden Erkennung ist mit der einer Schicht-für-Schicht-Erkennung identisch, mit einer Ausnahme: Die FFT-Koeffizienten können nun nicht mehr über die gesamte Länge der Eingabe normalisiert werden. Um dennoch zu einer vergleichbaren Skalierung zu kommen, kann die Eingabe mittels vorgegebener, konstanter Minimums-/Maximumswerte auf den gewünschten Wertebereich von $[-1,+1]$ transformiert werden, oder die $min-, max-$Werte der Transformation $[min, max] \rightarrow [-1,+1]$ werden mit einem gleitenden Fenster dynamisch adaptiert. Dabei sollte allerdings darauf geachtet werden, daß innerhalb von Sprechpausen (Stille) nicht adaptiert wird.

Vorausschau

Die Baumsuche schränkt gegenüber einer freien Erkennung den Suchraum stark ein und ermöglicht dadurch eine wesentlich robustere Erkennung. Da im Baum alle zu erkennenden Namen repräsentiert sind, kennt man zu einer partiellen Hypothese alle möglichen Erweiterungen. Wenn diese eindeutig sind, kann man die Fortführung eines Namens ein Stück weit oder sogar ganz vorhersagen, so daß ein langer Name nicht bis zu seinem Ende buchstabiert werden muß. In Abbildung 9.2 ist illustriert, in welchen Situationen ein Name wie weit ergänzt werden kann.

Untenstehende Beispiele zeigen die Auswirkungen einer Vorausschau. Es liegt eine Liste von etwa 500 englischen Nachnamen zugrunde; buchstabiert wurde der Name

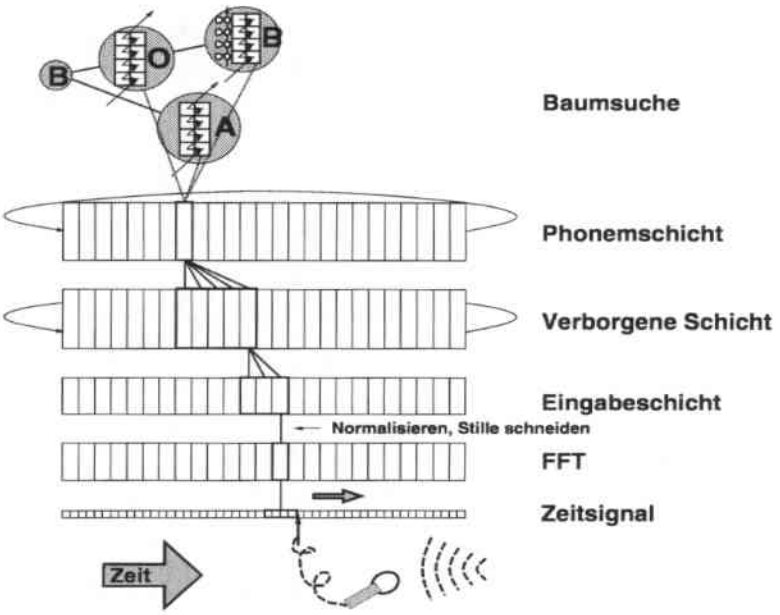


Abbildung 9.1: Schritthaltende Erkennung mit Baumsuche

„E L L I N G S O N" mit einer Gesamtlänge von etwa 6.5 Sekunden. Nach etwa 2.5 Sekunden verarbeiteter Spracheingabe findet die vorausschauende Baumsuche die folgende Liste der zehn besten Suchhypothesen:

```
12 nodes left in last frames (2.39 sec)
 [1] 198.67:   e l l i | ...
 [2] 197.22:   e l l n | e r)
 [3] 193.19:   e l l | ...
 [4] 190.08:   e l l n e r |
 [5] 189.25:   e l l n e | r)
 [6] 188.81:   e l l i n | g s o n)
 [7] 188.59:   e l l i o | t t)
 [8] 182.79:   c o l l i | n s)
 [9] 181.77:   e l l i s |
[10] 178.98:   e n g l i | s h)
```

Ein senkrechter Strich „|" kennzeichnet die Position, bis zu der tatsächlich erkannt wurde. Texte rechts davon sind automatische Ergänzungen, wobei die Klammer „)" einen Endknoten, also eine vollständige Hypothese, markiert. Punkte bedeuten, daß keine eindeutige Weiterführung möglich ist. Die korrekte Hypothese befindet sich bereits

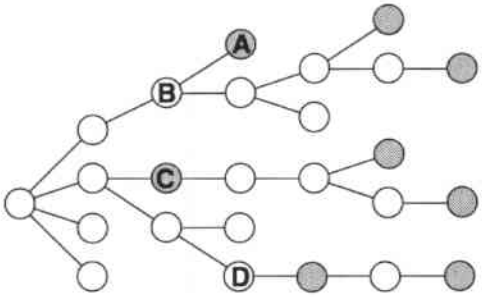


Abbildung 9.2: Schematischer Ausschnitt aus dem Suchbaum. Endknoten sind dunkel markiert. In A ist bereits ein Endknoten erreicht, die Hypothese ist vollständig. In B ist keine Erweiterung möglich, in C kann der Pfad um zwei Knoten eindeutig verlängert werden, allerdings nicht zu einer vollständigen Hypothese wie in D.

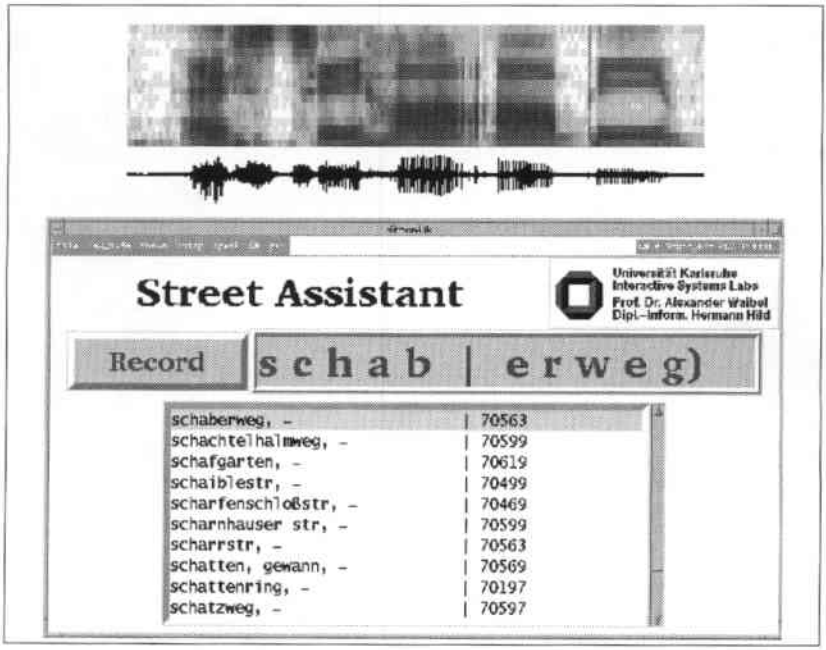


Abbildung 9.3: Demonstrationssystem. Nachdem die ersten fünf Buchstaben „S C H A B" gesprochen worden sind, ergänzt das System automatisch den Straßennamen und zeigt die entsprechende Postleitzahl an. Die Erkennung erfolgt in Echtzeit: Sprachsignal, Spektrogramm sowie die bisher erkannte Hypothese werden praktisch simultan zur Spracheingabe angezeigt.

in der Liste, allerdings erst auf Position sechs. Sie gelangt bereits nach knapp 3 Sekunden auf Position eins und bleibt dort bis zum Ende der Suche nach 6 Sekunden:

```
10 nodes left in last frames (2.79 sec)
[1] 230.36:   e l l i n | g s o n)
[2] 225.69:   e l l i o | t t)
[3] 225.68:   e l l i | ...
[4] 224.17:   e l l i s |
[5] 220.03:   e l l i o t | t)
[6] 219.67:   e l l i n g | s o n)
[7] 214.48:   c o l l i n | s)


2 nodes left in last frames (6.47 sec)
[1] 550.48:   e l l i n g s o n |
[2] 532.28:   e l l i n g s o | n)
```

Das MS-TDNN mit Baumsuche arbeitet selbst für Namenslisten mit über 100 000 Einträgen schneller als Echtzeit. Zusammen mit der schritthaltenden Verarbeitung erlaubt dies eine Erkennung praktisch simultan zur gesprochenen Eingabe. Dadurch kann man bereits während der Erkennung das Ergebnis beobachten und die Buchstabierung vorzeitig abbrechen, wenn der gewünschte Name vorausschauend bestimmt wurde.

9.2 Erkennung von Straßennamen

Abbildung 9.3 zeigt die graphische Benutzeroberfläche des Demonstrationssystem in einer Momentaufnahme während einer Erkennung. Der Sprecher hat in diesem Augenblick den Beginn des Straßennamens „`Schaberweg`" buchstabiert. Die Wellenform und das Spektrogramm der Sprache werden simultan zu ihrer Eingabe dargestellt, zusammen mit den bisher erkannten Buchstaben. Mit den ersten fünf erkannten Buchstaben konnte der korrekte vollständige Straßennamen erraten werden, er braucht daher nicht mehr zu Ende buchstabiert zu werden.

Das System ist auf die Erkennung aller (etwa 4000) Stuttgarter Straßennamen eingestellt. Mit Hilfe der Menuleiste kann ohne Zeitverlust auf andere, voreinstellbare Namenslisten umgeschaltet werden. Die Aufnahme wird durch einen Knopf gesteuert, den man mit der Maus oder auf einem drucksensiblen Bildschirm während des Sprechens gedrückt hält („push-while-talking"). Auf Wunsch kann das erkannte Ergebnis laut ausgesprochen werden.

9.3 Telefonauskunft

Mit dem auf den englischen OGI-Telefondaten trainierten Erkenner wurde ein Telefonauskunftssystem realisiert, mit dem beispielsweise alle Telefonnummern der Universität Karlsruhe abgefragt werden können. Ein Anrufer wird durch einen Dialog geführt,

in dessen Verlauf er den Nachnamen und eventuell den Vornamen des gewünschten Teilnehmers buchstabiert. Der genaue Ablauf ist im Flußdiagramm in Abbildung 9.4 festgehalten. Neben den Buchstaben müssen dazu auch „yes" und „no" erkannt werden, die dem Erkennervokabular als zwei weitere Wörter hinzugefügt und mit Sprachmaterial der OGI-Datenbank trainiert wurden. Die Ansagen des Systems können im voraus durch einen menschlichen Sprecher aufgenommen werden, was allerdings nicht sehr praktikabel ist[1], wenn das System auch den erkannten Namen ansagen soll. Qualitativ nicht ganz so hochwertig, jedoch wesentlich flexibler ist es, Sprachsynthese einzusetzen, die schon lange als kommerzielles Produkt erhältlich ist.

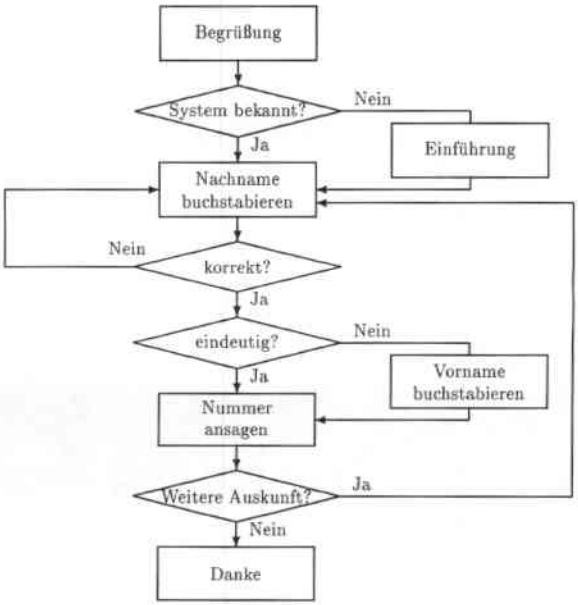


Abbildung 9.4: Vereinfachter Ablauf eines Dialoges für eine Telefonauskunft

Ähnliche, bisher nicht realisierte Anwendungen sind automatische E-Mail-Leser. Ein Anrufer meldet sich an, verifiziert sich durch Buchstabieren seines Namens und eines Paßwortes und bekommt daraufhin über das Telefon seine E-Mail vorgelesen. Umgekehrt ist auch denkbar, auf einem zentralen Rechner per Telefon Nachrichten für andere Nutzer zu hinterlegen, die davon mit einer E-Mail benachrichtigt werden.

[1] Zumal ein Anrufer verwirrt werden kann, wenn er nicht weiß, ob er mit einem Menschen oder einer Maschine spricht.

Kapitel 10

Zusammenfassung

Mit dem heutigen Stand der Technik können in der Spracherkennung große Mengen von Namen – etwa Millionen von Eigennamen oder Adressen – nur in buchstabierter Form robust erkannt werden. In dieser Arbeit wird ein Buchstabiererkenner vorgestellt, der genau zur Lösung dieser Aufgabe entwickelt worden ist. Es werden verschiedene Aspekte der Buchstabiererkennung behandelt. Zuerst wird ein konnektionistischer Spracherkenner mit dem Ziel entwickelt, möglichst hohe Erkennungsgenauigkeit auf beliebigen Buchstabensequenzen zu erreichen (Kapitel 6). Davon ausgehend werden Techniken zur Suche und Sprachmodellierung entworfen und untersucht, um das in großen Namenslisten gegebene Wissen möglichst vollständig und effizient zu nutzen (Kapitel 7). In Verbindung mit einem spontansprachlichen Erkenner werden schließlich Buchstabierungen erkannt, bei denen ein Name nicht nur buchstabiert, sondern auch fließend gesprochen wird, oder die von einleitenden Texten umgeben sind (Kapitel 8).

10.1 Die wichtigsten Ergebnisse und Beiträge

Die wichtigsten Ergebnisse und eigenen Beiträge zu jedem dieser drei thematischen Schwerpunkte sind im folgenden noch einmal kurz skizziert. Eine technisch detaillierte Zusammenfassung findet sich jeweils am Ende der Kapitel 6, 7 und 8.

Die **akustische Komponente** des konnektionistischen Erkenners wird durch ein Multi-State Time-Delay Neural Network (MS-TDNN) realisiert, einer Weiterentwicklung des TDNN. Die zeitliche Anpassung der akustischen Modelle an die Spracheingabe („dynamic time warping") ist im MS-TDNN direkt in die Netzwerkarchitektur integriert. Damit kann der Erkenner nicht nur (wie die meisten hybriden NN-HMM-Systeme) auf Phonemebene, sondern auch auf Wort- und Satzebene diskriminativ trainiert werden, was sich insbesondere für die Erkennung der kurzen und leicht verwechselbaren Buchstaben als vorteilhaft herausstellt.

Das MS-TDNN lernt in mehreren Trainingsphasen, die anhand systematischer Untersuchungen auf vier Sprachdatenbanken entwickelt und abgestimmt wurden. Dabei wird das System schrittweise von der Phonem- über die Wort- an die Satzerkennung herangeführt. Eine sorgfältige Auswahl der Fehlerfunktionen verbessert die Erkennungsleistung und berücksichtigt die Besonderheiten der MS-TDNN-Architektur. Insbesondere hat sich für das Training auf Wortebene die „CFM (Classification Figure of Merit)"-Fehlerfunktion bewährt, die nicht auf absolute Sollwerte 0 und 1, sondern auf einen relativen Vorsprung der korrekten Ausgabe abzielt. Durch zusätzliche Maßnahmen zur Zeitdauermodellierung der akustischen Modelle, beispielsweise eine phonemspezifische Mindestdauer, konnte erstmals eine sprecherunabhängige Erkennung von kontinuierlich[1] gesprochenen Buchstabensequenzen mit einer Buchstabenakkuratheit von über 90% erreicht werden. Für sprecherabhängige Erkennung konnten sogar bis zu 99% Buchstabenakkuratheit erzielt werden. Außerdem wurden diverse Architekturparameter sowie Aspekte der Netzwerk-Modularisierung untersucht.

Mit **Sprachmodellen** wird der Tatsache Rechnung getragen, daß verschiedene Buchstabenfolgen mit unterschiedlichen Wahrscheinlichkeiten auftreten. Bedingt durch die Größe der Erkennervokabulare schätzen konventionelle, statistische N-Gramm-Sprachmodelle (Bi- und Trigramme) die Wahrscheinlichkeit für das nächste Wort lediglich in Abhängigkeit von ein oder zwei unmittelbar vorausgehenden Wörtern.
Mit nur 26 Buchstaben im Vokabular ist es jedoch möglich, zur Sprachmodellierung in der Buchstabiererkennung größere Kontexte zu berücksichtigen. Dazu wurden mehrere Methoden entwickelt und evaluiert, die den vollständigen Suchkontext nutzen, um buchstabierte Namen aus sehr großen, gegebenen Namenslisten zu erkennen. Dabei stellten sich diejenigen Verfahren als die besten heraus, bei denen die durch das Sprachmodell gegebenen Einschränkungen nicht erst in einem Nachverarbeitungsschritt, sondern bereits verzahnt mit der Suche durch die akustischen Modelle zum Tragen kommen. Unter diesen erwies sich die Baumsuche als die konzeptionell eleganteste und gleichzeitig effizienteste Technik. Eine weitere signifikante Verbesserung konnte dadurch erzielt werden, daß die Erkennung nicht nur auf eine große Liste von Namen eingeschränkt wird, sondern erstmals auch deren a priori Wahrscheinlichkeit („`Maier`" ist ein häufigerer Name als „`Westphal`") berücksichtigt.
Ohne Sprachmodell werden bei einer Buchstabenakkuratheit von etwa 90% nur in etwa 60% der Fälle alle einzelnen Buchstaben eines buchstabierten Namens richtig erkannt. Mit der effizienten Baumsuche können komplette buchstabierte Namen selbst dann noch in Echtzeit mit etwa 90% korrekt erkannt werden, wenn der Erkennung Listen mit Millionen von Einträgen zugrundeliegen.

Unter dem Begriff „**Buchstabieren in spontaner Sprache**" haben wir diejenigen Buchstabier-Phänomene zusammengefaßt, bei denen nicht ausschließlich Buchstaben, sondern zusätzlich Füllwörter, sonstiger begleitender Text oder buchstabierte Namen auch fließend gesprochen werden.

[1] d.h. fließend, ohne Pause gesprochene Buchstaben

Um in „normalen" Text eingebettete Buchstabensequenzen (mit dem Karlsruher spontansprachlichen JANUS-Erkenner) zu erkennen, wurden spezielle Sprachmodelle entwickelt, die Buchstabensequenzen als unabhängiges Untersprachmodell repräsentieren. Dadurch können im N-Gramm-Sprachmodell Wort-Buchstaben- und Buchstaben-Wort-Übergänge unabhängig von den zufällig in den Trainingstexten beobachteten Buchstabierungen geschätzt werden, und die Modellierung innerhalb einer Buchstabensequenz kann auf externe Quellen zurückgreifen.
Weitere Experimente betrafen die Erkennung von Namen, die sowohl fließend gesprochen als auch buchstabiert wurden. Die durch die unterschiedlichen Sprecharten gegebene Redundanz kann ausgenutzt werden, indem die N-Besten-Listen des MS-TDNN- und des JANUS-Erkenners kombiniert werden. Da jedoch buchstabierte Namen deutlich besser als fließend gesprochene erkannt werden, muß eine Kombination stark zugunsten der Buchstabiererkennung gewichtet werden, um eine, wenngleich nur geringe, Verbesserung zu erzielen. Um auch sehr große Listen fließend gesprochener Namen handhaben zu können, wird ein Name zuerst nur als Phonemsequenz erkannt, aus der anschließend eine Kandidatenliste gewonnen wird. Erst diese vorselektierte Liste ist klein genug, um mit ihren vollen phonetischen Transkriptionen in das Wörterbuch des Erkenners aufgenommen werden zu können. Zusätzlich erlauben die beschriebenen Techniken eine flexiblere Erkennung, bei der wahlweise nur buchstabiert, oder gesprochen und buchstabiert wird.

Dank der einfachen und homogenen Struktur neuronaler Netze ist der vorgestellte MS-TDNN-Erkenner schnell, kompakt und kommt mit der relativ geringen Anzahl von etwa 20 000 Parametern aus. Mit einer schritthaltenden Erkennung konnte ein Demonstrationssystem realisiert werden, das auch auf kleinen Rechnersystemen (Laptop) in Echtzeit praktisch simultan zur Spracheingabe erkennt. Außerdem wurde ein Auskunftssystem für Telefonnummern implementiert.

Mit dem MS-TDNN konnten auf gleichen oder ähnlichen Sprachdaten in allen Fällen die **Ergebnisse vergleichbarer Systeme übertroffen** werden. Damit wurde der Nachweis erbracht, daß mit einem konnektionistischen Erkenner exzellente „state-of-the-art"-Erkennungsraten erzielt werden können.
Weiterhin konnte demonstriert werden, daß mit geeigneten Suchtechniken selbst Namenslisten, die die kompletten Telefonverzeichnisse vieler amerikanischer Großstädte enthalten, schnell und mit guten Erkennungsraten durchsucht werden können.
Mit der Buchstabiererkennung in spontaner Sprache wurde Neuland betreten, ein Vergleich mit anderen Systemen ist daher nicht möglich.

10.2 Ausblick

Es sollen nun noch einige interessante Möglichkeiten zur Verbesserung des Erkenners diskutiert werden, die im Rahmen dieser Arbeit nicht experimentell untersucht wurden.

Im allgemeinen kann die Erkennungsleistung durch eine **kontextabhängige akustische Modellierung** gesteigert werden. Zu einem gewissen Grad sind die Phoneme im MS-TDNN-Erkenner bereits kontextabhängig, denn durch das buchstabenspezifische Übergangsphonem in fast allen Buchstabenmodellen ist eine kontextabhängige Modellierung innerhalb der Buchstaben zumindest teilweise gegeben. Gerade bei den kurzen Buchstaben sollte akustischer Kontext jedoch auch über die Buchstabengrenzen hinweg modelliert werden. Da viele Buchstaben mit den gleichen Phonemen beginnen oder enden, ist die Anzahl dabei zu berücksichtigender Kontexte günstigerweise relativ klein.

Ein Großteil der Erkennungsfehler geht auf Verwechslungen innerhalb der schwer unterscheidbaren Buchstabengruppen (z.B. {N,M} oder {D,B,G}) zurück. In einem nachgeschalteten Erkennungsschritt könnte man **spezifisch trainierte Netzwerke** einsetzen, die nur innerhalb einer verwechselbaren Buchstabengruppe klassifizieren müssen und daher ihre Aufmerksamkeit besser auf die jeweils kritischen Merkmale fokussieren können.

Zur Erkennung von Namen ausschließlich aus einer gegebenen Liste wird durch die Baumsuche mit a priori Namenswahrscheinlichkeiten ein **Sprachmodell** realisiert, das bereits optimal im Sinne einer Nutzung aller vorhandenen Informationen ist. Allerdings werden dadurch auch beliebige „ungültige" Eingaben immer auf einen Namen der Liste abgebildet. Hier sollte ein Mechanismus zur **Rückweisung** ungültiger Eingaben entwickelt werden. Beispielsweise kann man parallel zur Baumsuche mit einem „Neuer-Name"-Modell beliebige Sequenzen erkennen. Ergibt sich damit eine ausreichend bessere Bewertung als mit der Baumsuche, liegt der Verdacht auf einen listenfremden Namen nahe.

Das Buchstabieren von langen und/oder nicht schriftlich vorliegenden Namen ist nicht immer einfach und kann den Sprecher überfordern. Für eine benutzerfreundliche Eingabe am Telefon sollte eine Dialogstruktur gefunden werden, bei der ein langer Name in mehreren Teilen buchstabiert und gegebenenfalls korrigiert werden kann.

Will man etwa in Auskunftssystemen statt einer strengen Dialogführung eine benutzerfreundlichere freie Buchstabiererkennung, so wird man auf die Erkennung **spontansprachlicher Buchstabiereffekte** nicht verzichten können. Diese Phänomene wurden bereits in Abschnitt 1.3 und Kapitel 8 vorgestellt und teilweise experimentell untersucht. Als zukünftig zu lösende praxisrelevante Probleme bieten sich die Erkennung von Funkeralphabeten und Wortassoziationen („Bravo, Charlie, A wie Alex") sowie von nur teilweise buchstabierten Namen („Maier mit A I") an.

Anhang A

Mathematische Ergänzungen

A.1 Fehlerfunktionen, Transferfunktionen und ihre Ableitungen

Fehlerfunktionen

Im folgenden sind die MSE-, CE-, CFM- und GPD-Fehlerfunktionen mit ihren Ableitungen aufgeführt. Es sei $\boldsymbol{y} = (y_1 \dots y_n)$ die Ausgabe des Klassifikators und $\boldsymbol{t} = (t_1 \dots t_n)$ die Sollausgabe (*target*).

Mean Square Error (MSE)

$$\begin{aligned} E &= \frac{1}{2}||\boldsymbol{t} - \boldsymbol{y}||^2 = \frac{1}{2}\sum_j (t_j - y_j)^2 \\ \frac{\partial E}{\partial y_i} &= y_i - t_i \end{aligned} \tag{A.1}$$

Cross Entropy (CE)

$$\begin{aligned} E &= -\sum_j t_j \log(y_j) + (1 - t_j)\log(1 - y_j) \\ \frac{\partial E}{\partial y_i} &= -\left(t_i \frac{1}{y_i} + (1 - t_i)\frac{-1}{1 - y_i}\right) = \frac{1 - t_i}{1 - y_i} - \frac{t_i}{y_i} = \frac{y_i - t_i}{y_i(1 - y_i)} \end{aligned} \tag{A.2}$$

McClelland Error

$$\begin{aligned} E &= -\sum_j \log(1-(t_j-y_j)^2) \\ \frac{\partial E}{\partial y_i} &= \frac{-1}{1-(t_i-y_i)^2}(-2)(t_i-y_i)(-1) = \frac{-2(t_i-y_i)}{1-(t_i-y_i)^2} \end{aligned} \tag{A.3}$$

CFM

$$E = \frac{1}{C-1}\sum_{j=1, j\neq k}^{C} s(d_j), \quad s(x) = \frac{1}{1+e^{-\alpha x+\beta}} \tag{A.4}$$

$s(x)$ ist die parametrisierte Sigmoidfunktion aus (A.6). $d_j := y_k - y_j$ ist die Distanz zwischen y_j und der Ausgabe y_k der korrekten Klasse k. Für $i \neq k$ lautet die Ableitung nach y_i:

$$\frac{\partial E}{\partial y_i} = \frac{1}{C-1}\frac{\partial s(d_i)}{\partial d_i}\cdot\frac{\partial d_i}{\partial y_i} \overset{(A.7)}{=} \frac{-\alpha}{C-1}s(d_i)(1-s(d_i))$$

und für $i = k$:

$$\frac{\partial E}{\partial y_k} = \frac{1}{C-1}\sum_{j\neq k}\frac{\partial s(d_j)}{\partial d_j}\cdot\frac{\partial d_j}{\partial y_k} = \frac{\alpha}{C-1}\sum_{j\neq k} s(d_j)(1-s(d_j))$$

Das Fehlersignal für den Knoten mit der korrekten Ausgabe hat denselben Betrag wie die Summe aller inkorrekten Knoten, denn es ist

$$\frac{\partial E}{\partial y_k} = \sum_{j\neq k}\frac{\alpha}{C-1}s(d_j)(1-s(d_j)) = -\sum_{j\neq k}\frac{\partial E}{\partial y_i}$$

GPD

Mit $y_j := g_j(\boldsymbol{x})$ ergibt sich das Mißklassifikationsmaß d aus (2.26) zu

$$\begin{aligned} d &= -y_k + \frac{1}{\eta}\log\left[\frac{1}{C-1}\sum_{j\neq k} e^{y_j^\eta}\right] \\ \frac{\partial d}{\partial y_i} &= \begin{cases} -1 & \text{falls } i = k \\ \frac{1}{\eta}\frac{1}{\sum_{j\neq k} e^{y_j^\eta}} e^{y_i^\eta}\cdot\eta = \frac{e^{y_i^\eta}}{\sum_{j\neq k} e^{y_j^\eta}} & \text{falls } i \neq k \end{cases} \end{aligned}$$

Wie für die CFM-Funktion ist die Summe der Fehlersignale aller inkorrekten Ausgaben bis auf das Vorzeichen identisch mit dem Fehlersignal der korrekten Ausgaben:

$$\sum_{i \neq k} \frac{\partial d}{\partial y_i} = \sum_{i \neq k} \frac{e^{y_i^\eta}}{\sum_{j \neq k} e^{y_j^\eta}} = 1 = -\frac{\partial d}{\partial y_k}$$

Wird d mit dem von der L_p-Norm abgeleiteten Maß wie in (2.23) definiert,

$$d = -y_k + \left[\frac{1}{C-1} \sum_{j \neq k} y_j^\eta\right]^{\frac{1}{\eta}}$$

dann ergibt sich $\partial d / \partial y_k = 1$ und für $i \neq k$:

$$\frac{\partial d}{\partial y_i} = \left(\frac{1}{C-1}\right)^{\frac{1}{\eta}} \cdot \frac{1}{\eta} (\sum_{j \neq k} y_j^\eta)^{\frac{1-\eta}{\eta}} \cdot \eta \cdot y_i^{\eta-1} = \left(\frac{1}{C-1}\right)^{\frac{1}{\eta}} \cdot \left[\frac{y_i}{(\sum_{j \neq k} y_j^\eta)^{\frac{1}{\eta}}}\right]^{\eta-1}$$

Transferfunktionen

Sigmoidfunktionen kann man als differenzierbare Schwellwertfunktionen interpretieren, wie es der Name „sigmoid" = „signum-ähnlich" bereits andeutet. Die klassische Sigmoidfunktion σ ist

$$y = \sigma(x) = \frac{1}{1 + e^{-x}}$$

Mit $r := 1 + e^{-x}$, und damit $y = 1/r$, lautet die Ableitung

$$\frac{\partial y}{\partial x} = \frac{-1}{r^2}(-e^{-x}) = \frac{1 + e^{-x} - 1}{r^2} = \frac{r-1}{r^2} = y - y^2 = y(1-y) \tag{A.5}$$

Die Steigung und Position der „Schwelle" kann durch zwei Parameter α und β eingestellt werden:

$$y = \frac{1}{1 + e^{-\alpha x + \beta}} \tag{A.6}$$

$$\frac{\partial y}{\partial x} = \alpha y(1-y) \tag{A.7}$$

Gelegentlich wird statt $\sigma(x)$ auch der Tangens Hyperbolicus $\tanh(x)$ als Transferfunktion eingesetzt. $\tanh(x)$ realisiert eine Sigmoidfunktion im Wertebereich $[-1, 1]$ und ist bis auf Stauchung und Verschiebung identisch mit $\sigma(x)$:

$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} = \frac{2 - (1 + e^{-2x})}{1 + e^{-2x}} = \frac{2}{1 + e^{-2x}} - 1 \\ &= 2\sigma(2x) - 1 \end{aligned}$$

Betrachten wir nun die Ableitung $\partial E/\partial x_i$ nach der Aktivierung x_i (statt der Ausgabe y_i) für ein Neuron i mit einer Sigmoid-Transferfunktion. Allgemein gilt

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial x_i} = \frac{\partial E}{\partial y_i} y_i(1 - y_i)$$

Mit Cross Entropy als Fehler vereinfacht sich der Ausdruck zu

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial x_i} \overset{(A.2)}{=} \frac{y_i - t_i}{y_i(1 - y_i)} \cdot y_i(1 - y_i) = y_i - t_i$$

Bei der **Softmax-Funktion** ist die Ausgabe y_i nicht nur von x_i, sondern von allen x_j abhängig

$$y_i = \text{softmax}(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Die Ausgaben der Softmax-Funktion sind auf 1 normiert, d.h. $\sum_i y_i = 1$. Mit $r := \sum_j e^{x_j}$ ist $y_i = e^{x_i}/r$, und es gilt:

$$\frac{\partial y_j}{\partial x_i} = \frac{e^{x_i}\delta_{ij} r - e^{x_j} e^{x_i}}{r^2} = \frac{e^{x_i}}{r}(\delta_{ij} - \frac{e^{x_j}}{r}) = y_i(\delta_{ij} - y_j) \tag{A.8}$$

Somit ist $\partial y_i/\partial x_i = y_i(1 - y_i)$ identisch mit der Ableitung der Sigmoidfunktion (A.5). Da jedoch $\partial y_i/\partial x_j \neq 0$, wird die Ableitung $\partial E/\partial y_i$ etwas aufwendiger:

$$\begin{aligned}\frac{\partial E}{\partial x_i} &= \frac{\partial E}{\partial \boldsymbol{y}} \cdot \frac{\partial \boldsymbol{y}}{\partial x_i} = \sum_j \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \overset{(A.8)}{=} \sum_j \frac{\partial E}{\partial y_j} \cdot y_i(\delta_{ij} - y_j) \\ &= \sum_j \frac{\partial E}{\partial y_j} y_i \delta_{ij} - \sum_j \frac{\partial E}{\partial y_j} y_i y_j = y_i(\frac{\partial E}{\partial y_i} - \sum_j \frac{\partial E}{\partial y_j} y_j)\end{aligned}$$

A.2 Fehlerrückpropagierung im TDNN

Im TDNN berechnet sich die Ableitung des Fehlers nach den Gewichten im wesentlichen wie im Standard-Backpropagationverfahren. Der Unterschied liegt darin, daß nun ein einzelnes Gewicht nicht nur ein, sondern eine ganze Reihe von Neuronen betrifft. Entsprechend müssen die zu jedem Zeitpunkt aufgesammelten Gewichtsänderungen gemittelt werden.

Wie zuvor bezeichnen wir die Aktivierung eines Neurons mit x und seine Ausgabe mit y. Für ein Neuron der **Phonemschicht** mit der Ausgabe $y^{\text{P}}_{t,i}$ (also die Bewertung von Phonem i zum Zeitpunkt t) gilt nach Gl. (6.3) und (6.4):

$$x^{\text{P}}_{t,i} = b^{\text{P}}_i + \sum_{d=-d_{\text{H}}}^{d_{\text{H}}} \sum_{j=1}^{n_{\text{H}}} w^{\text{P},d}_{i,j} \cdot y^{\text{H}}_{t+d,j} \tag{A.9}$$

$$y^{\text{P}}_{t,i} = \sigma(x^{\text{P}}_{t,i}) \tag{A.10}$$

Analoge Formeln gelten gemäß (6.1) und (6.2) für Neuronen $x^{\mathrm{H}}_{t,i}$ der **verborgenen Schicht**. Daraus ergeben sich die später benötigten Ableitungen:

$$\begin{aligned}\frac{\partial y^{\mathrm{P}}_{t,i}}{\partial w^{\mathrm{P},d}_{i,j}} &= \frac{\partial y^{\mathrm{P}}_{t,i}}{\partial x^{\mathrm{P}}_{t,i}} \cdot \frac{\partial x^{\mathrm{P}}_{t,i}}{\partial w^{\mathrm{P},d}_{i,j}} \\ &\overset{(A.9,A.10)}{=} \frac{\partial \sigma(x^{\mathrm{P}}_{t,i})}{\partial x^{\mathrm{P}}_{t,i}} \cdot \frac{\partial (b^{\mathrm{P}}_i + \sum_{d'=-d_{\mathrm{H}}}^{d_{\mathrm{H}}} \sum_{j'=1}^{n_{\mathrm{H}}} w^{\mathrm{P},d'}_{i,j'} \cdot y^{\mathrm{H}}_{t+d',j'})}{\partial w^{\mathrm{P},d}_{i,j}} \\ &= \sigma'(x^{\mathrm{P}}_{t,i}) \cdot y^{\mathrm{H}}_{t+d,j} \end{aligned} \tag{A.11}$$

und analog

$$\frac{\partial y^{\mathrm{H}}_{t,i}}{\partial w^{\mathrm{H},d}_{i,j}} = \sigma'(x^{\mathrm{H}}_{t,i}) \cdot y^{\mathrm{E}}_{t+d,j} \tag{A.12}$$

Wir berechnen nun zuerst die Ableitung des Fehlers nach den in die Phonemschicht führenden Gewichten

$$\begin{aligned}\frac{\partial E}{\partial w^{\mathrm{P},d}_{i,j}} &= \frac{\partial E}{\partial \mathbf{Y}^{\mathrm{P}}} \cdot \frac{\partial \mathbf{Y}^{\mathrm{P}}}{\partial w^{\mathrm{P},d}_{i,j}} = \sum_t \sum_{k=1}^{n_{\mathrm{P}}} \frac{\partial E}{\partial y^{\mathrm{P}}_{t,k}} \cdot \frac{\partial y^{\mathrm{P}}_{t,k}}{\partial w^{\mathrm{P},d}_{i,j}} \overset{(*)}{=} \sum_t \frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} \cdot \frac{\partial y^{\mathrm{P}}_{t,i}}{\partial w^{\mathrm{P},d}_{i,j}} \\ &\overset{(A.11)}{=} \sum_t \frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} \sigma'(x^{\mathrm{P}}_{t,i}) \cdot y^{\mathrm{H}}_{t+d,j} \end{aligned} \tag{A.13}$$

In Schritt $(*)$ verschwindet die Summe über k, da die Gewichte $w^{\mathrm{P},d}_{i,j}$ nur Einfluß auf die i-te Zeile haben, also $\partial y^{\mathrm{P}}_{t,k}/\partial w^{\mathrm{P},d}_{i,j} = 0$ für $k \neq i$. Wird die Ausgabe y^{O}_i gemäß (6.7) als $\frac{1}{F}\sum_t y^{\mathrm{P}}_{t,i}$ berechnet, gilt wie bereits in (6.8) bestimmt (wobei F die Anzahl der Vektoren der Phonemschicht war):

$$\frac{\partial E}{\partial y^{\mathrm{P}}_{t,i}} = \frac{1}{F} \frac{\partial E}{\partial y^{\mathrm{O}}_i} \tag{A.14}$$

Die Berechnung der Ableitung nach den Gewichten in die verborgene Schicht erfolgt zuerst analog zu (A.13):

$$\begin{aligned}\frac{\partial E}{\partial w^{\mathrm{H},d}_{i,j}} &= \frac{\partial E}{\partial \mathbf{Y}^{\mathrm{H}}} \cdot \frac{\partial \mathbf{Y}^{\mathrm{H}}}{\partial w^{\mathrm{H},d}_{i,j}} = \sum_t \frac{\partial E}{\partial y^{\mathrm{H}}_{t,i}} \cdot \frac{\partial y^{\mathrm{H}}_{t,i}}{\partial w^{\mathrm{H},d}_{i,j}} \\ &\overset{(A.12)}{=} \sum_t \frac{\partial E}{\partial y^{\mathrm{H}}_{t,i}} \sigma'(x^{\mathrm{H}}_{t,i}) \cdot y^{\mathrm{E}}_{t+d,j} \end{aligned} \tag{A.15}$$

$\partial E/\partial y^{\mathrm{H}}_{t,i}$ ist etwas aufwendiger zu bestimmen, da $y^{\mathrm{H}}_{t,i}$ aufgrund der Time-Delays auch zu t benachbarte Vektoren in der Phonemschicht beeinflußt.

$$\frac{\partial E}{\partial y^{\mathrm{H}}_{t,i}} = \frac{\partial E}{\partial \mathbf{Y}^{\mathrm{P}}} \cdot \frac{\partial \mathbf{Y}^{\mathrm{P}}}{\partial y^{\mathrm{H}}_{t,i}} = \sum_{t_p} \sum_{k=1}^{n_{\mathrm{P}}} \frac{\partial E}{\partial y^{\mathrm{P}}_{t_p,k}} \cdot \frac{\partial y^{\mathrm{P}}_{t_p,k}}{\partial y^{\mathrm{H}}_{t,i}}$$

$$\overset{(**)}{=} \sum_{d=-d_H}^{d_H} \sum_{k=1}^{n_P} \frac{\partial E}{\partial y^P_{t-d,k}} \cdot \frac{\partial y^P_{t-d,k}}{\partial x^P_{t-d,k}} \cdot \frac{\partial x^P_{t-d,k}}{\partial y^H_{t,i}} \tag{A.16}$$

Die Summe über alle Zeitschritte t_p in der Phonemschicht kann im Schritt $(**)$ verkürzt werden, denn $y^H_{t,i}$ hat nur Einfluß auf die Vektoren $\mathbf{y}^P_{t-d_H} \cdots \mathbf{y}^P_{t+d_H}$.

$$\begin{aligned} \frac{\partial x^P_{t-d,k}}{\partial y^H_{t,i}} &= \frac{\partial (b^P_k + \sum_{d'=-d_H}^{d_H} \sum_{j=1}^{n_H} y^H_{t-d+d',j} \cdot w^{P,d'}_{k,j})}{\partial y^H_{t,i}} \\ &= \frac{\partial (\sum_{d'=-d_H}^{d_H} y^H_{t-d+d',i} \cdot w^{P,d'}_{k,i})}{\partial y^H_{t,i}} \\ &= w^{P,d}_{k,i} \end{aligned} \tag{A.17}$$

Der erste Faktor in (A.16) ist das bereits in (A.14) angegebene, aus der höheren Schicht kommende Fehlersignal, der zweite die Ableitung der Transferfunktion, $\sigma'(x^P_{t-d,k})$. Mit (A.17) wird aus (A.16):

$$\frac{\partial E}{\partial y^H_{t,i}} = \sum_{d'=-d_H}^{d_H} \sum_{k=1}^{n_P} \frac{\partial E}{\partial y^P_{t-d',k}} \cdot \sigma'(x^P_{t-d',k}) \cdot w^{P,d'}_{k,i} \tag{A.18}$$

Dies besagt gerade, daß $y^H_{t,i}$ von all jenen Neuronen ein Fehlersignal bekommt, mit denen es im Vorwärtsschritt mit einem der zeitverzögerten Gewichte verbunden war, wie es in Abbildung A.1 veranschaulicht ist. Die Ableitungen nach den Schwellwertgewichten b^P_i und b^H_i erfolgt vollkommen analog. Insgesamt ergeben sich mit (A.13, A.14, A.15, A.18) die in Tabelle 6.1 zusammengestellten Ableitungen.

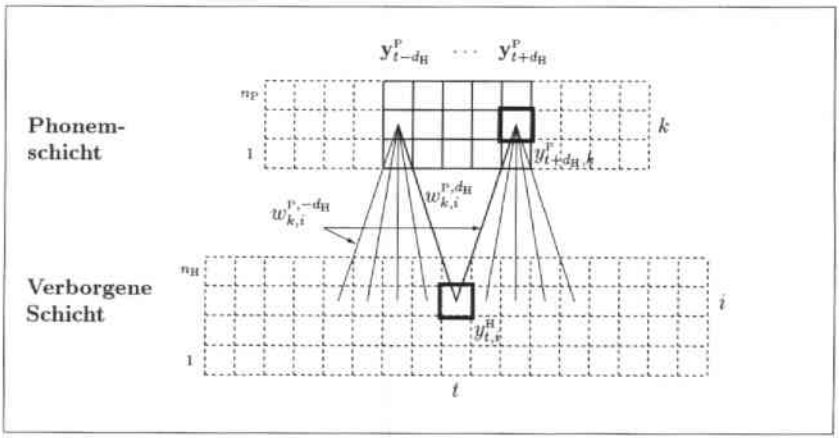


Abbildung A.1: Fehlerrückpropagierung entlang der Time-Delays im TDNN.

A.3 Sprachmodelle

Ein Sprachmodell muß die Bedingung $\sum_{w_j \in V} P(x|w_i) = 1$ erfüllen. Wir zeigen dies für das in Abschnitt 8.3.1 aus LM_W und LM_L neu generierte Sprachmodell LM, wobei gefordert werden muß, daß keine leeren oder aufeinanderfolgenden Symbole <LS> auftreten, also $P_L(\dashv \mid \vdash) = P_W(\texttt{<LS>}|\texttt{<LS>}) = 0$ gilt. Satzanfang ($\vdash$) und Satzende ($\dashv$) werden in der Summation aus V_L ausgenommen, da sie bereits in V_W enthalten sind.

Monogramme

$$\begin{aligned}
& \sum_{w_j \in V_W} P(w_j) + \sum_{L_j \in V_L \setminus \{\vdash,\dashv\}} P(L_j) \\
\overset{(8.1,8.2)}{=} & \sum_{w_j \in V_W} P_W(w_j) + \sum_{L_j \in V_L \setminus \{\vdash,\dashv\}} P_L(L_j|\vdash) \cdot P_W(\texttt{<LS>}) \\
= & \quad 1 - P_W(\texttt{<LS>}) + P_W(\texttt{<LS>}) \sum_{L_j \in V_L \setminus \{\vdash,\dashv\}} P_L(L_j|\vdash) \\
= & \quad 1 - P_W(\texttt{<LS>}) + P_W(\texttt{<LS>}) \cdot 1 = 1
\end{aligned}$$

Bigramme

$$\begin{aligned}
& \sum_{w_j \in V_W} P(w_j|w_i) + \sum_{L_j \in V_L \setminus \{\vdash,\dashv\}} P(L_j|w_i) \\
\overset{(8.3,8.4)}{=} & \sum_{w_j} P_W(w_j|w_i) + \sum_{L_j} P_W(\texttt{<LS>}|w_i) P_L(L_j|\vdash) \\
= & \quad 1 - P_W(\texttt{<LS>}|w_i) + P_W(\texttt{<LS>}|w_i) \sum_{L_j} P_L(L_j|\vdash) \\
= & \quad 1 - P_W(\texttt{<LS>}|w_i) + P_W(\texttt{<LS>}|w_i) = 1
\end{aligned}$$

$$\begin{aligned}
& \sum_{w_j \in V_W} P(w_j|L_i) + \sum_{L_j \in V_L \setminus \{\vdash,\dashv\}} P(L_j|L_i) \\
\overset{(8.5,8.6)}{=} & \sum_{w_j} P_L(\dashv|L_i) P_W(w_j|\texttt{<LS>}) + \sum_{L_j} P_L(L_j|L_i) \\
= & \quad P_L(\dashv|L_i) \sum_{w_j} P_W(w_j|\texttt{<LS>}) + \sum_{L_j} P_L(L_j|L_i) \\
= & \quad P_L(\dashv|L_i) + 1 - P_L(\dashv|L_i) = 1
\end{aligned}$$

Trigramme

Beispielsweise ergibt sich für (8.7) und (8.8):

$$\begin{aligned}
&\sum_{w_k \in V_W} P(w_k|w_i, w_j) + \sum_{L_k \in V_L \setminus \{\vdash, \dashv\}} P(L_k|w_i, w_j) \\
= \;& 1 - P_W(\texttt{<LS>}|w_i, w_j) + \sum_{L_k} P_L(L_k|\vdash, \vdash) \cdot P_W(\texttt{<LS>}|w_i, w_j) \\
= \;& 1 - P_W(\texttt{<LS>}|w_i, w_j) + P_W(\texttt{<LS>}|w_i, w_j) \sum_{L_k} P_L(L_k|\vdash, \vdash) \\
= \;& 1 - P_W(\texttt{<LS>}|w_i, w_j) + P_W(\texttt{<LS>}|w_i, w_j) = 1
\end{aligned}$$

Entsprechendes kann in analoger Weise für die drei übrigen Fälle gezeigt werden.

Backoff-Wahrscheinlichkeiten

Für die Monogramm- und Backoff-Wahrscheinlichkeiten gilt:

$$P(w_i) := P_W(w_i) \tag{A.19}$$

$$b(w_i) := b_W(w_i) \tag{A.20}$$

$$P(L_i) := P_L(L_i|\vdash) \cdot P_W(\texttt{<LS>}) \tag{A.21}$$

$$b(L_i) := b_W(\texttt{<LS>}) \cdot P_L(\dashv|L_i) \tag{A.22}$$

Die Backoff-Wahrscheinlichkeiten $b(w_i)$ und $b(L_i)$ werden benötigt, wenn für ein Wortpaar mangels Daten kein Bigramm geschätzt werden konnte. War dies beispielsweise für $P_W(w_j|\texttt{<LS>})$ und damit für $P(w_j|L_i)$ der Fall, so wird letzteres ersetzt durch

$$P(w_j|L_i) = P(w_j) \cdot b(L_i) \overset{(A.22)}{=} P_W(w_i) \cdot b_W(\texttt{<LS>}) \cdot P_L(\dashv|L_i)$$

Dies entspricht aber genau (8.5), wenn dort das Bigramm $P_W(w_j|\texttt{<LS>})$ durch $P_W(w_j) \cdot b_W(\texttt{<LS>})$ ersetzt wird. Die Definition von $b(L_i)$ gewährleistet also auch im Falle eines Backoffs eine korrekte Berechnung von $P(w_j|L_i)$, hat aber zur Folge, daß $b(L_i)$ nicht gleichzeitig als Backoff für $P(L_j|L_i)$ dienen kann. Um diesen Konflikt aufzulösen, müssen im Sprachmodell alle Bigramme $P(L_j|L_i)$ explizit aufgelistet werden, um einen Zugriff auf $b(L_i)$ wegen fehlender Buchstabenbigramme zu vermeiden.

Anhang B

Tabellen

B.1 Detaillierte Ergebnisse

Die folgenden vier Tabellen listen die Erkennungsraten (% korrekt) für die Phonemklassifikation einzelner Vektoren und für Buchstaben auf, wie sie nach dem Training auf Phonem- und Wortebene erzielt werden. Dabei wurde jeweils mit allen Kombinationen der vier Fehlerfunktionen (Mean Square Error (MSE), Cross Entropy (CE), McClelland Error (MCL) und Classification Figure of Merit (CFM)) und der zwei Transferfunktionen (Sigmoid- und Softmaxfunktion) trainiert und getestet.

Die Ergebnisse sind für die zwei sprecherunabhängigen Datenbanken (RM–Spell und KA–Alph) sowie für die einzelnen Sprecher mdbs und mjmt jeweils für die Trainings- (Train), Kreuzvalidierungs- (Cross) und Testmenge (Test) aufgelistet.

		Sigmoid			Softmax		
RMspell	(rm-1a, it=100(10).fr)	Train	Cross	Test	Train	Cross	Test
MSE		76.0	74.4	75.4	76.5	74.2	75.7
CE		76.5	74.8	76.0	76.2	73.9	75.1
MCL		77.7	75.7	76.8	78.1	75.6	76.3
CFM		76.6	74.9	75.9	72.8	71.6	72.7
		Sigmoid			Softmax		
KAalph	(ka-1a, it=50(10).fr)	Train	Cross	Test	Train	Cross	Test
MSE		78.3	74.3	76.3	78.5	73.9	76.2
CE		77.6	73.3	75.7	77.5	72.9	75.5
MCL		78.9	74.3	76.7	79.3	74.9	77.1
CFM		78.0	74.0	76.1	76.1	71.5	74.2
		Sigmoid			Softmax		
mdbs	(mdbs-1a, it=100(10).fr)	Train	Cross	Test	Train	Cross	Test
MSE		77.8	75.0	75.9	78.8	75.8	77.0
CE		81.0	78.5	78.5	80.7	78.1	78.2
MCL		82.0	79.4	79.7	82.3	79.8	80.1
CFM		79.0	76.1	76.8	82.9	79.4	80.0
		Sigmoid			Softmax		
mjmt	(mjmt-1a, it=100(10).fr)	Train	Cross	Test	Train	Cross	Test
MSE		79.3	78.5	77.9	80.7	79.3	78.9
CE		83.0	81.8	80.6	83.3	82.0	80.6
MCL		84.1	83.1	82.2	84.7	82.7	81.5
CFM		84.5	83.1	82.0	85.5	83.6	82.6

Tabelle B.1: Phonemerkennungsraten (Vektoren) nach Training auf Phonemebene

		Sigmoid			Softmax		
RMspell	(rm-1a, it=100(10))	Train	Cross	Test	Train	Cross	Test
MSE		90.8	86.6	85.7	89.5	84.0	84.7
CE		93.3	89.3	88.7	94.0	90.0	88.8
MCL		94.7	90.9	89.5	95.3	91.5	89.4
CFM		91.8	88.7	86.2	86.2	82.0	81.0
		Sigmoid			Softmax		
KAalph	(ka-1a, it=50(10))	Train	Cross	Test	Train	Cross	Test
MSE		92.4	89.5	87.0	92.3	89.4	86.3
CE		91.9	89.2	86.9	92.9	91.0	87.4
MCL		93.8	90.1	88.2	94.7	92.0	89.9
CFM		92.3	90.2	87.7	90.8	88.6	85.5
		Sigmoid			Softmax		
mdbs	(mdbs-1a, it=100(10))	Train	Cross	Test	Train	Cross	Test
MSE		95.6	90.3	92.8	96.3	91.4	93.5
CE		98.2	95.6	95.9	98.2	96.4	96.8
MCL		98.7	96.5	96.5	98.8	97.3	97.1
CFM		96.5	93.6	93.6	98.4	97.0	96.8
		Sigmoid			Softmax		
mjmt	(mjmt-1a, it=100(10))	Train	Cross	Test	Train	Cross	Test
MSE		98.4	98.1	97.3	96.5	94.9	96.2
CE		99.3	99.0	98.9	99.5	99.2	99.2
MCL		99.5	99.2	99.0	99.5	99.1	99.2
CFM		99.4	99.2	99.0	99.4	99.2	99.0

Tabelle B.2: Buchstabenerkennungsraten nach Training auf Phonemebene

RM–Spell (rm-test-1a2.summary)	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MCL	96.5	93.4	93.5	97.0	94.5	92.4
CE	96.0	93.5	94.2	95.4	92.2	92.0
MSE	93.1	90.4	91.7	92.1	87.9	89.5
CFM	94.3	91.0	91.0	82.4	79.4	81.2
KA–Alph (ka-test-1a2.summary)	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MCL	94.0	91.8	89.0	95.7	94.1	91.9
CE	94.6	92.8	91.9	94.9	93.2	91.0
MSE	94.5	92.8	89.9	93.9	92.1	90.2
CFM	94.0	93.0	89.3	88.6	87.2	83.3
mdbs (mdbs-test-1a2.summary)	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MCL	99.1	97.9	98.3	98.7	97.9	97.4
CE	99.3	98.3	98.6	99.1	97.9	98.7
MSE	96.2	93.6	93.9	94.7	93.2	93.9
CFM	96.0	95.3	94.6	98.8	98.3	97.2
mjmt (mjmt-test-1a2.summary)	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MCL	99.5	99.6	99.3	99.6	99.6	99.6
CE	99.4	99.4	99.3	99.6	99.6	99.4
MSE	97.9	98.6	97.6	95.9	93.6	95.3
CFM	99.5	99.0	99.2	99.6	99.6	99.4

Tabelle B.3: Buchstabenerk. nach Training auf Phonemebene mit hybridem NN-HMM

RM–Spell (rm-1b2, it=100(10))	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MSE	97.9	93.7	93.7	97.9	93.6	92.2
CE	98.2	93.4	92.4	97.8	91.6	90.6
MCL	98.5	94.5	94.4	98.7	94.4	92.7
CFM	99.4	96.1	95.4	99.2	95.5	94.0
KA–Alph (ka-1b, it=50(10))	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MSE	97.1	93.9	92.0	97.2	94.2	91.7
CE	96.9	93.5	91.0	97.1	93.4	90.6
MCL	97.3	93.9	92.0	97.5	94.0	91.5
CFM	98.4	95.4	94.0	98.3	95.0	93.0
mdbs (mdbs-1b, it=100(10))	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MSE	99.3	97.6	97.8	99.5	97.5	98.4
CE	99.8	98.6	98.2	99.9	98.2	98.6
MCL	99.8	98.7	98.6	100.0	98.7	98.4
CFM	100.0	99.0	98.6	100.0	98.6	98.7
mjmt (mjmt-1b2, it=100(10))	Sigmoid			Softmax		
	Train	Cross	Test	Train	Cross	Test
MSE	99.7	99.4	99.4	99.7	99.4	99.4
CE	99.7	99.2	99.3	99.7	99.2	99.3
MCL	99.7	99.6	99.4	99.7	99.6	99.4
CFM	99.7	99.7	99.4	99.7	99.7	99.4

Tabelle B.4: Buchstabenerkennungsraten nach Training auf Wortebene

B.2 Phonetische Modellierung des Alphabets

Buchstabe	**Englische** Phonemumschrift	**Deutsche** Phonemumschrift
@	si1 si2	si1 si2
a	eyI eyF	? ahI ahF
ae		? aeI aeF
b	bI b-iy iy	bI b-eh ehF
c	sI s-iy iy	tI s s-eh ehF
d	dI d-iy iy	dI d-eh ehF
e	iyI iy	? ehI ehF
f	ehI eh-f fF	? aeI ae-f fF
g	jhI jh-iy iy	gI g-eh ehF
h	eyI ey-ch chF	hI h-ah ahF
i	ayI ay ayF	? ieI ieF
j	jhI jh-ey eyF	jI j-o o-t tF
k	kI k-ey eyF	kI k-ah ahF
l	ehI eh-l lF	? aeI ae-l lF
m	ehI eh-m mF	? aeI ae-m mF
n	ehI eh-n nF	? aeI ae-n nF
o	owI ow owF	? ohI ohF
oe		? oeI oeF
p	pI p-iy iy	pI p-eh ehF
q	kI k-y y-uw uw	kI k-uh uhF
r	aaI aa-r rF	? aeI ae-r rF
s	ehI eh-s sF	? aeI ae-s sF
sz		? aeI ae-s tI s-t t-ae ae-t tF
ss		sch a ae-r fF ? aeI ae-s sF
se		sch a ae-r fF ae-s sF ? aeI ae-s sF
t	tI t-iy iy	tI t-eh ehF
u	yI y-uw uw	? uhI uhF
ue		? ueI ueF
v	vI v-iy iy	fI f-au auF
w	dI d-ah ah b ax y-uw uw	vI v-eh ehF
x	ehI eh-k k-s sF	? iI i-k k-s sF
y	wI w-ay ay ayF	? ueI p s i l o nF
z	zI z-iy iy	tI s-t t-ae ae-t tF
-s		sch tI rF ieI ch
-b		bI ieF nF dI d-eh sch tI rF ieI ch
dp		dI ohI p p-eh lF

Tabelle B.5: Aussprachewörterbuch für das Alphabet. Im Deutschen gibt es zusätzlich die Umlaute (ae, oe, ue), Eszett (sz), Scharf-S (ss), Scharfes-S (se) für ß sowie Strich (-s), Bindestrich (-b) und doppel (dp).

B.3 Nationale/Internationale Funkeralphabete

Es gibt eine Vielzahl von Funkeralphabeten, auch phonetisches, Radio-, Buchstabier- oder Telefonalphabete genannt. Auf dem Internet unter der Adresse

`http://www.cl.cam.ac.uk/users/bck1/phon.full.html`

sind Funkeralphabete für etwa 30 Länder aufgelistet sowie lokale Varianten von Armee, Polizei und Fernsprechdiensten. Die in Tabelle B.6 aufgelisteten Funkeralphabete entstammen ebenfalls dieser Quelle. Das internationale NATO-Alphabet (Spalte 1) existiert seit 1955 und ist anerkannt bei der internationalen zivilen Luftfahrt (+ FAA) sowie der „International Telecommunication Union". Die deutsche Version stammt von der Deutschen Bundespost. Gelegentlich wird statt Siegfried auch Samuel, und statt Zeppelin auch Zacharias aufgeführt. Ein anderes internationales Alphabet (Spalte 4) stützt sich hauptsächlich auf Städtenamen.

Int. NATO	Deutsch	Britisch	Internat.
Alfa	Anton	Andrew	Amsterdam
-	Ärger	-	-
Bravo	Berta	Benjamin	Baltimore
Charlie	Cäsar	Charlie	Casablanca
-	Charlotte	-	-
Delta	Dora	David	Denmark
Echo	Emil	Edward	Edison
Foxtrott	Friedrich	Frederick	Florida
Golf	Gustav	George	Gallipoli
Hotel	Heinrich	Harry	Havana
India	Ida	Isaac	Italy
Juliet	Julius	Jack	Jerusalem
Kilo	Kaufmann	King	Kilogram
Lima	Ludwig	Lucy	Liverpool
Mike	Martha	Mary	Madagascar
November	Nordpol	Nelli	New York
Oscar	Otto	Oliver	Oslo
-	Ökonom	-	-
Papa	Paula	Peter	Paris
Quebec	Quelle	Queenie	Quebec
Romeo	Richard	Robert	Roma
Sierra	Siegfried	Sugar	Santiago
-	Schule	-	-
Tango	Theodor	Tommy	Tripoli
Uniform	Ulrich	Uncle	Uppsala
-	Übermut	-	-
Victor	Viktor	Victor	Valencia
Whisky	Wilhelm	William	Washington
Xray	Xanthippe	Xmas	Xanthippe
Yankee	Ypsilon	Yellow	Yokohama
Zulu	Zeppelin	Zebra	Zürich

Tabelle B.6: Nationale/Internationale Funkeralphabete

Literaturverzeichnis

[Bak75] **J. K. Baker**. The Dragon System - an Overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23:24–29, 1975.

[BBDS93] **H. Bourlard, J.-M. Boite, B. D'hoore und M. Saerens**. Performance Comparison of Hidden Markov Models and Neural Networks for Task Dependent and Independent Isolated Word Recognition. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, S. 1925–1928, Berlin, September 1993.

[BC91] **J. S. Bridle und S. J. Cox**. RecNorm: Simultaneous Normalisation and Classification Applied to Speech Recognition. In *Advances in Neural Information Processing Systems 3 (NIPS*90)*, S. 234–240. Morgan Kaufmann, 1991.

[Bel57] **R. Bellman**. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Bet94] **M. Betz**. Sprachmodelle für einen Buchstabiererkenner. Diplomarbeit, Universität Karlsruhe, Mai 1994.

[BH93] **D. Boiteau und P. Haffner**. Connectionist Segmental Post-Processing of the N-Best Solutions in Isolated and Connected Word Recognition. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1933–1936, Berlin, September 1993.

[BH95a] **M. Betz und H. Hild**. Language Models for a Spelled Letter Recognizer. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 856–859, Detroit, MI, Mai 1995.

[BH95b] **U. Bodenhausen und H. Hild**. Automatic Construction of Neural Networks for Special Purpose Speech Recognition Systems. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 5, S. 3327–3330, Detroit, MI, Mai 1995.

[Bir95] **S. Bird**. Recognition of Telephone Quality Speech. Diplomarbeit, Universität Karlsruhe, Mai 1995.

[BM94] **H. Bourlard und N. Morgan**. *Connectionist Speech Recognition - A Hybrid Approach*. Kluwer Academic Press, Boston, MA, 1994.

[BMHW93a] **C. Bregler, S. Manke, H. Hild und A. Waibel**. Bimodal Sensor Integration on the Example of "Speech-Reading". In *Proc. IEEE International Conference on Neural Networks*, Band 2, S. 667–670, San Francisco, CA, März 1993.

[BMHW93b] **C. Bregler, S. Manke, H. Hild und A. Waibel**. Improving Connected Letter Recognition by Lipreading. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 557–561, Minneapolis, MN, April 1993.

[BMWR92] **H. Bourlard, N. Morgan, C. Wooters und S. Renals**. CDNN: A Context Dependent Neural Network for Continuous Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 349–352, San Francisco, CA, März 1992. IEEE.

[Bod94] **U. Bodenhausen**. *Automatic Structuring of Neural Networks for Spatio-Temporal Real-World Applications*. Dissertation, Universität Karlsruhe, Juni 1994.

[Bol95] **E. Bolten**. PLP und RASTA-PLP: Zwei Verfahren zur Vorverarbeitung von Sprachsignalen. Studienarbeit, Universität Karlsruhe, Mai 1995.

[Bri90] **J. S. Bridle**. Alpha-Nets: A Recurrent 'Neural' Network Architecture with a Hidden Markov Model Interpretation. *Speech Communication*, 9(1):83–92, 1990.

[Bro87] **P. F. Brown**. The Acoustic-Modeling Problem in Automatic Speech Recognition. Technical Report CMU-CS-87-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, Mai 1987.

[Bur88a] **D. J. Burr**. Experiments on Neural Net Recognition of Spoken and Written Text. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1162–1168, Juli 1988.

[Bur88b] **D. J. Burr**. Speech Recognition Experiments with Perceptrons. In D. Z. Anderson (Hrsg.), *Neural Information Processing Systems*, S. 144–153. American Institute of Physics, New York, NY, 1988.

[BW90] **H. Bourlard und C. J. Wellekens**. Links between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178, Dezember 1990.

[BW91] **U. Bodenhausen und A. Waibel**. Learning the Architecture of Neural Networks for Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, Mai 1991.

[CFGJ91] **R. Cole, M. Fanty, M. Gopalakrishnan und R. D. T. Janssen**. Speaker-Independent Name Retrieval from Spellings Using a Database of 50,000 Names. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 325–328, Toronto, Canada, Mai 1991. IEEE.

[CFMG90] **R. A. Cole, M. Fanty, Y. Muthumamy und M. Gopalakrishnan**. Speaker-Independent Recognition of Spoken English Letters. In *International Joint Conference on Neural Networks*, San Diego, CA, Juni 1990. IEEE.

[CHM+93] **M. Cohen, F. Horacio, N. Morgan, D. Rumelhart und V. Abrash.** Context-Dependent Multiple Distribution Phonetic Modeling with MLPs. In S. J. Hanson, J. D. Cowan und C. L. Giles (Hrsg.), *Advances in Neural Information Processing Systems 3 (NIPS*90)*, Band 5, S. 649–657, San Mateo, CA, 1993. Morgan Kaufmann.

[CRF91] **R. Cole, K. Roginsky und M. Fanty.** English Alphabet Recognition with Telephone Speech. In *EUROSPEECH'91 (2nd European Conference on Speech Communication and Technology)*, Genua, 1991. IEEE.

[CRF92] **R. Cole, K. Roginsky und M. Fanty.** A Telephone Speech Database of Spelled and Spoken Names. In *Proc. International Conference on Speech and Language Processing*, S. 891–893, Banff, Alberta, Oktober 1992. IEEE.

[CSM86] **R. Cole, R. Stern und L. Moshé.** Performing Fine Phonetic Distictions: Templates versus Features. In J. S. Perkell und D. H. Klatt (Hrsg.), *Invariance and Variability in Speech Processes*, S. 325–345. Lawrence Erlbaum Assoc., 1986. (Auch in [WL90], S. 214–224).

[CT91] **T. M. Cover und J. A. Thomas.** *Elements of Information Theory.* John Wiley & Sons, Inc., New York, 1991.

[Dal87] **N. A. Daly.** Recognition of Words from their Spellings: Integration of Multiple Knowledge Sources. Diplomarbeit, Massachusetts Institute of Technology, 1987.

[DB95] **T. G. Dietterich und G. Bakiri.** Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DH73] **R. O. Duda und P. E. Hart.** *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[DMW94] **P. Duchnowski, U. Meier und A. Waibel.** See Me, Hear Me: Integrating Automatic Speech Recognition and Lip-Reading. In *Proc. International Conference on Speech and Language Processing*, Band 2, S. 547–550, Yokohama, Japan, September 1994.

[Dup93] **P. Dupont.** Dynamic Use of Syntactical Knowledge in Continuous Speech Recognition. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1959–1962, Berlin, September 1993.

[EZ87] **J. L. Elman und D. Zipser.** Learning the Hidden Structure of Speech. Technical Report ICS Report 8701, Institute for Cognitive Science, University of California, San Diego, Februar 1987.

[FBC95] **M. Fanty, E. Barnard und R. Cole.** Alphabet Recognition. In *Handbook of Neural Computation*, S. F2.1:1–8. IOP Publishing Ltd and Oxford University Press, 1995.

[FC90] **M. Fanty und R. Cole.** Speaker-Independent English Alphabet Recognition: Experiments with the E-Set. In *Proc. International Conference on Speech and Language Processing*, Kobe, Japan, November 1990.

[FC91] **M. Fanty und R. Cole**. Spoken Letter Recognition. In R. P. Lippman, J. Moody und D. S. Touretzky (Hrsg.), *Advances in Neural Information Processing Systems 2 (NIPS*88*, San Mateo, CA, November 1991. Morgan Kaufmann.

[FCR92] **M. Fanty, R. Cole und K. Roginski**. English Alphabet Recognition With Telephone Speech. In J. E. Moody, S. J. Hanson und R. P. Lippmann (Hrsg.), *Advances in Neural Information Processing Systems 5 (NIPS*91)*, S. 199–206. Morgan Kaufmann, 1992.

[FGH+97] **M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries und M. Westphal**. The Karlsruhe-Verbmobil Speech Recognition Engine. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 83–86, München, April 1997.

[FP87] **K. Földes-Papp**. *Vom Felsbild zum Alphabet*. Belser Verlag, Stuttgart, Zürich, 1987.

[Gis90] **H. Gish**. A Probabilistic Approach to the Understanding and Training of Neural Network Classifiers. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 1361–1364, April 1990.

[Haa90] **H. Haarmann**. *Universalgeschichte der Schrift*. Campus Verlag, Frankfurt/New York, 1990.

[Haf92] **P. Haffner**. Connectionist Word-Level Classification in Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 621–624, San Francisco, CA, März 1992. IEEE.

[Haf93a] **P. Haffner**. $\alpha\beta$-TDNN Implement "Fuzzy" Connectionist Time Alignment in Speech Recognition. In *Proc. International Conference on Artificial Neural Networks*, Amsterdam, September 1993. Springer-Verlag.

[Haf93b] **P. Haffner**. Connectionist Speech Recognition with a Global MMI Algorithm. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1929–1932, Berlin, September 1993.

[Haf94] **P. Haffner**. A New Probabilistic Framework for Connectionist Time Alignment. In *Proc. International Conference on Speech and Language Processing*, Band 3, S. 1559–1562, Yokohama, Japan, September 1994.

[HFW91] **P. Haffner, M. Franzini und A. Waibel**. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 105–108, Toronto, Canada, Mai 1991. IEEE.

[HH92] **M.-Y. Hwang und X. Huang**. Subphonetic Modeling with Markov States - Senone. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 33–37, San Francisco, CA, März 1992. IEEE.

[HIP90] **J. B. Hampshire II und B. Pearlmutter**. Equivalence Proofs for Multilayer Perceptron Classifiers and the Bayesian Discriminant Function. In D. S.

Touretzky, J. L. Elman, T. J. Sejnowski und G. E. Hinton (Hrsg.), *Proc. of the 1990 Connectionist Models Summer School*, San Mateo, CA, April 1990. Morgan Kaufmann.

[HIW89a] **J. B. Hampshire II und A. Waibel**. A Novel Objective Function for Improved Phoneme Recognition Using Time Delay Neural Networks. In *International Joint Conference on Neural Networks*, S. 235–241, Washington, DC, Juni 1989.

[HIW89b] **J. B. Hampshire II und A. Waibel**. Connectionist Architectures for Multi-Speaker Phoneme Recognition. Technical Report CMU-CS-89-167, Carnegie Mellon University, August 1989.

[HIW90a] **J. B. Hampshire II und A. Waibel**. Connectionist Architectures for Multi-Speaker Phoneme Recognition. In D. S. Touretzky und R. P. Lippmann (Hrsg.), *Advances in Neural Information Processing Systems 2 (NIPS*89)*, S. 203–210, San Mateo, CA, 1990. Morgan Kaufmann.

[HIW90b] **J. B. Hampshire II und A. Waibel**. The Meta-Pi Network: Connectionist Rapid Adaptation for High-Performance Multi-Speaker Phoneme Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, April 1990.

[HIW90c] **J. B. Hampshire II und A. Waibel**. A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Neural Networks*, 1(2):216–228, Juni 1990.

[HL88] **W. Y. Huang und R. P. Lippmann**. Neural Net and Traditional Classifiers. In D. Z. Anderson (Hrsg.), *Neural Information Processing Systems*, S. 387–396. American Institute of Physics, New York, NY, 1988.

[HN90] **R. Hecht-Nielsen**. *Neurocomputing*. Addison-Wesley Publishing Company, Reading, MA, USA, 1990.

[HRRC95] **M. M. Hochberg, S. J. Renals, A. J. Robinson und G. D. Cook**. Recent Improvements to the ABOTT Large Vocabulary CSR System. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 69–72, Detroit, USA, Mai 1995. IEEE.

[Hua92] **X. Huang**. Speaker Normalization for Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 465–468, San Francisco, CA, März 1992. IEEE.

[HW91] **P. Haffner und A. Waibel**. Time-Delay Neural Networks Embedding Time Alignment: A Performance Analysis. In *EUROSPEECH'91 (2nd European Conference on Speech Communication and Technology)*, S. 1439–1442, Genua, September 1991.

[HW92] **P. Haffner und A. Waibel**. Multi-State Time Delay Neural Networks for Continuous Speech Recognition. In J. E. Moody, S. J. Hanson und R. P. Lippmann (Hrsg.), *Advances in Neural Information Processing Systems 5 (NIPS*91)*, S. 135–142. Morgan Kaufmann Publishers, Inc., 1992.

[HW93a] **H. Hild und A. Waibel**. Connected Letter Recognition with a Multi-State Time Delay Neural Network. In S. J. Hanson, J. D. Cowan und C. L. Giles (Hrsg.), *Advances in Neural Information Processing Systems 4 (NIPS*92)*, S. 712–719. Morgan Kaufmann, 1993.

[HW93b] **H. Hild und A. Waibel**. Multi-Speaker/Speaker-Independent Architectures for the Multi-State Time Delay Neural Network. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 255–258, Minneapolis, MN, April 1993.

[HW93c] **H. Hild und A. Waibel**. Speaker-Independent Connected Letter Recognition with a Multi-State Time Delay Neural Network. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 2, S. 1481–1484, Berlin, September 1993.

[HW95] **H. Hild und A. Waibel**. Integrating Spelling Into Spoken Dialogue Recognition. In *EUROSPEECH'95 (4th European Conference on Speech Communication and Technology)*, Band 3, S. 1977–1979, Madrid, September 1995.

[HW96] **H. Hild und A. Waibel**. Recognition of Spelled Names over the Telephone. In *Proceedings Fourth International Conference on Speech and Language Processing*, Band 1, S. 346–349, Philadephia, PA, Oktober 1996.

[Iso92] **K.-I. Iso**. Speech Recognition Using Dynamical Model of Speech Production. Technical Report CMU-CS-92-187, Carnegie Mellon University, Juli 1992.

[IW90] **K.-I. Iso und T. Watanabe**. Speaker-Independent Word Recognition Using A Neural Prediction Model. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, April 1990. IEEE.

[IW91] **K.-I. Iso und T. Watanabe**. Large Vocabulary Speech Recognition Using Neural Prediction Model. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 57–60, Toronto, Canada, Mai 1991. IEEE.

[Jel76] **F. Jelinek**. Continuous Speech Recognition by Statistical Methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976.

[Jel90] **F. Jelinek**. Self-Organized Language Modeling for Speech Recognition. In **A. Waibel und K.-F. Lee** (Hrsg.), *Readings in Speech Recognition*, S. 450–506. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[JFC91] **R. D. T. Janssen, M. Fanty und R. Cole**. Speaker-Independent Phonetic Classification in Continuous English Letters. In *International Joint Conference on Neural Networks*, Seattle, WA, Juli 1991.

[JJ94] **M. I. Jordan und R. A. Jacobs**. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 6(2):181–214, Dezember 1994.

[JLM93] **D. Jouvet, M. N. Lokbani und J. Monné**. Application of the N-Best Solutions Algorithm to Speaker-Independent Spelling Recognition over the Telephone.

In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 2081–2084, Berlin, September 1993.

[JLMG93] **D. Jouvet, A. Lainé, J. Monné und C. Gagnoulet**. Speaker-Independent Spelling Recognition over the Telephone. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 235–238, Minneapolis, MN, April 1993. IEEE.

[JMMR85] **J. Jordan, B. Mertens, J. Mrosek und C. Richartz**. Bilder/Schriften/Alphabete. Staatliche Museen Preußischer Kulturbesitz, 1985.

[Jun97] **J.-C. Junqua**. SmarTspelL: A Multipass Recognition System for Name Retrieval over the Telephone. *IEEE Transactions on Speech and Audio Processing*, 5(2):173–182, März 1997.

[JVFM95] **J.-C. Junqua, S. Valente, D. Fohr und J.-F. Mari**. An N-Best Strategy, Dynamic Grammars and Selectively Trained Neural Networks for Real-Time Recognition of Continuously Spelled Names over the Telephone. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 852–855, Detroit, Michigan, Mai 1995. IEEE.

[KFSW95] **B. Kaspar, G. Fries, K. Schuhmacher und A. Wirth**. FAUST - A Directory Assistance Demonstrator. In *EUROSPEECH'95 (4th European Conference on Speech Communication and Technology)*, Band 2, S. 1161–1164, Madrid, Spain, September 1995.

[KJ96] **T. Kemp und A. Jusek**. Modelling Unknown Words in Spontaneous Speech. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 530–533, Atlanta, GA, Mai 1996. IEEE.

[KLJ91] **S. Katagiri, C.-H. Lee und B.-H. Juang**. New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent method. IEEE Workshop Neural Networks for Signal Processing, August 1991.

[KRR96] **D. J. Kershaw, A. J. Robinson und S. J. Renals**. The 1995 Abbot Hybrid Connectionist-HMM Large-Vocabulary Recognition System. In *ARPA Workshop on Spoken Language Technology*. Morgan Kaufmann, 1996.

[KSS95] **C. A. Kamm, C. R. Shamieh und S. Singhal**. Speech Recognition Issues for Directory Assistance Applications. *Speech Communication*, 17:303–311, November 1995.

[Lan89] **K. Lang**. *A Time-Delay Neural Network Architecture for Speech Recognition*. Dissertation, Carnegie Mellon University, Juli 1989. CMU-CS-89-185.

[LBM95] **M. Lennig, G. Bielby und J. Massicotte**. Directory Assistance Automation on Bell Canada: Trial Results. *Speech Communication*, 17:227–234, 1995.

[LCVC93] **P. Le Cerf und D. Van Compernolle**. Speaker Independent Small Vocabulary Speech Recognition using MLPs for Phonetic Labelling. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 1, S. 143–146, Berlin, September 1993.

[Lee89] **K.-F. Lee**. *Automatic Speech Recognition: the Development of the SPHINX System*. Kluwer Academic Publishers, Boston, 1989.

[Lev90] **E. Levin**. Word Recognition Using Hidden Control Neural Network Architecture. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, April 1990. IEEE.

[LG87] **R. P. Lippmann und B. Gold**. Neural-Net Classifiers Useful for Speech Recognition. In *IEEE First International Conference on Neural Networks*, Band 4, S. 417–425, San Diego, CA, Juni 1987.

[Lip89] **R. P. Lippmann**. Review of Neural Networks for Speech Recognition. *Neural Computation*, 1(1):1–38, März 1989. (Auch in [WL90], S. 374–392).

[LS96] **P. C. Loizou und A. S. Spanias**. High-Performance Alphabet Recognition. *IEEE Transactions on Speech and Audio Processing*, 4(6):430–445, November 1996.

[LWH90] **K. J. Lang, A. Waibel und G. E. Hinton**. A Time-Delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks*, 3(1):23–43, 1990.

[LWL+97] **A. Lavie, A. Waibel, L. Levin, M. Finke, D. Gates, M. Gavaldá, T. Zeppenfeld und P. Zhan**. JANUS-III: Speech-to-Speech Translation in Multiple Languages. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, München, April 1997.

[MB95] **N. Morgan und H. Bourlard**. Neural Networks for Statistical Recognition of Continuous Speech. *Proceedings of the IEEE*, 83(5):742–770, 1995.

[MBG+95] **N. Morgan, H. Bourlard, S. Greenberg, H. Hermansky und S.-L. Wu**. Stochastic Perceptual Models of Speech. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 397–400, Detroit, Michigan, 1995.

[Mey97] **M. Meyer**. Erkennung fließend gesprochener und buchstabierter Namen. Diplomarbeit, Universität Karlsruhe, März 1997.

[MFW95] **S. Manke, M. Finke und A. Waibel**. NPen++: A Writer Independent, Large Vocabulary On-Line Cursive Handwriting Recognition System. In *International Conference on Document Analysis and Recognition*, Montreal, August 1995. IEEE.

[MGR93] **A. Mellouk, P. Gallinari und F. Rauscher**. Prediction and Discrimination in Neural Networks for Continuous Speech Recognition. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1603–1606, Berlin, September 1993.

[MH97] **M. Meyer und H. Hild**. Recognition of Spoken and Spelled Proper Names. Erscheint in *EUROSPEECH'97 (5th European Conference on Speech Communication and Technology)*, Rhodos, Griechenland, September 1997.

[MHD96] **U. Meier, W. Hürst und P. Duchnowski**. Adaptive Bimodal Sensor Fusion for Automatic Speechreading. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 833–836, Atlanta, GA, Mai 1996. IEEE.

[MIKT90] **E. McDermott, H. Iwamida, S. Katagiri und Y. Tohkura**. Shift-Tolerant LVQ and Hybrid LVQ-HMM for Phoneme Recognition. In Waibel und Lee (Hrsg.), *Readings in Speech Recognition*, S. 425–438. Morgan Kaufmann, 1990.

[MP69] **M. Minsky und S. Papert**. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, Mass., 1969.

[NAH+95] **J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals und T. Robinson**. Speaker-Adaptation for Hybrid HMM-ANN Continuous Speech Recognition System. In *EUROSPEECH'95 (4th European Conference on Speech Communication and Technology)*, Band 3, S. 2171–2174, Madrid, September 1995.

[Ney84] **H. Ney**. The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):263–271, April 1984. (Auch in [WL90], S. 188–196).

[PBW95] **P. Puzrla, F. Bimbot und C. Windheuser**. Distributed Binary Representations for Word Recognition by TDNN-DTW Hybrid Systems. In *EUROSPEECH'95 (4th European Conference on Speech Communication and Technology)*, Band 3, S. 2175–2178, Madrid, September 1995.

[PF93] **B. Petek und A. Ferligoj**. On Use of Discriminant Analysis in Predictive Connectionist Speech Recognition. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1611–1614, Berlin, September 1993.

[PKP+96] **D. J. Pepper, C. A. Kamm, S. Patel, C. R. Shamieh, S. Singhal, E. S. Soper und K. M. Yang**. Voice-DQ: A Directory Assistance System for a Corporate Sized Directory. In *AVIOS Proceedings*, 1996.

[Rab89] **L. R. Rabiner**. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, Februar 1989. (Auch in [WL90], S. 267–296).

[RAB+93] **A. J. Robinson, L. Almeida, J.-M. Boite, H. Bourlard, F. Fallside, M. Hochberg, D. Kershaw, P. Kohn, Y. Konig, N. Morgan, J. P. Neto, S. Renals, M. Serens und C. Wooters**. A Neural Network Based, Speaker Independent, Large Vocabulary, Continuous Speech Recognition System. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1941–1944, Berlin, September 1993.

[Rig93] **G. Rigoll**. Joint Optimization of Multiple Neural Codebooks in a Hybrid Connectionist-HMM Speech Recognition System. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1727–1729, Berlin, September 1993.

[RL91] **M. D. Richard und R. P. Lippmann**. Neural Network Classifiers Estimate Bayesian A Posteriori Probabilities. *Neural Computation*, 3(4):461–483, 1991.

[RLRW79] **L. R. Rabiner, S. E. Levinson, A. E. Rosenberg und J. G. Wilpon**. Speaker-Independent Recognition of Isolated Words Using Clustering Techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27(4):336–349, August 1979.

[RM86] **D. E. Rumelhart und J. L. McClelland**. *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, Band I and II. MIT Press, Cambridge, MA, 1986.

[RM93] **S. Renals und D. MacKay**. Bayesian Regularisation Methods in a Hybrid MLP-HMM System. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 3, S. 1719–1722, Berlin, September 1993.

[RMCF92] **S. Renals, N. Morgan, M. Cohen und H. Franco**. Connectionist Probability Estimation in the Decipher Speech Recognition System. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 601–604, San Francisco, CA, März 1992. IEEE.

[RNR95] **G. Rigoll, C. Neukirchen und J. Rottland**. Large Vocabulary Speaker-Independent Continuous Speech Recognition with a New Hybrid System Based on MMI-Neural Networks. In *EUROSPEECH'95 (4th European Conference on Speech Communication and Technology)*, Band 3, S. 1659–1662, Madrid, März 1995.

[Rob94] **T. Robinson**. An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, 1994.

[Roj93] **R. Rojas**. *Theorie der neuronalen Netze*. Springer-Verlag, Berlin, 1993.

[RT92] **J. Reynolds und L. Tarassenko**. Spoken Letter Recognition with Neural Networks. *International Journal of Neural Systems*, 3(3):219–235, 1992.

[Sak79] **H. Sakoe**. Two-Level DP-Matching – A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27(6):588–595, Dezember 1979. (Auch in [WL90], S. 180–187).

[SC78] **H. Sakoe und S. Chiba**. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26(1):43–49, Februar 1978. (Auch in [WL90], S. 159–165).

[SFSJ93] **M. Schmidt, S. Fitt, C. Scott und M. A. Jack**. Phonetic Transcription Standards for European Names (ONOMASTICA). In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 1, S. 279–282, Berlin, September 1993.

[SI87] **H. Sakoe und K.-I. Iso**. Dynamic Neural Network — A New Speech Recognition Model Based on Dynamic Programming and Neural Network. Technical Report 87, IEICE, NEC Corporation, Dezember 1987.

[SIY+89] **H. Sakoe, R. Isotani, K. Yoshida, K.-I. Iso und T. Watanabe**. Speaker-Independent Word Recognition Using Dynamic Programming Neural Networks. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 29–32, Glasgow, Scotland, Mai 1989. (Auch in [WL90], S. 439–442).

[SR75] **R. W. Schafer und L. R. Rabiner**. Digital Representation of Speech Signals. *Proceedings of the IEEE*, 63(4):662–677, 1975. (Auch in [WL90], S. 49–64).

[SR86] **T. J. Sejnowski und C. R. Rosenberg**. NETtalk: A Parallel Network that Learns to Read Aloud. Technical Report JHU/EECS-86/01, John Hopkins University, Juni 1986.

[ST95] **E. G. Schukat-Talamazzini**. *Automatische Spracherkennung*. Vieweg, Braunschweig, 1995.

[Teb93] **J. Tebelskis**. Performance Through Consistency: Connectionist Large Vocabulary Continuous Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 259–262, Minneapolis, MN, April 1993. IEEE.

[Teb95] **J. Tebelskis**. *Speech Recognition using Neural Networks*. Dissertation, Carnegie Mellon University, Pittsburgh, USA, Mai 1995. CMU-CS-95-142.

[TH87] **D. W. Tank und J. J. Hopfield**. Concentrating Information in Time: Analog Neural Networks with Applications to Speech Recognition Problems. In *IEEE First International Conference on Neural Networks*, Band 4, S. 455–468, San Diego, CA, Juni 1987.

[TW88] **S. Tamura und A. Waibel**. Noise Reduction Using Connectionist Models. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, New York, NY, April 1988. IEEE.

[TW90] **J. Tebelskis und A. Waibel**. Large Vocabulary Recognition Using Linked Predictive Neural Networks. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, April 1990. IEEE.

[TW93] **J. Tebelskis und A. Waibel**. Performance Through Consistency: MS-TDNN's for Large Vocabulary Continuous Speech Recognition. In *Advances in Neural Information Processing Systems 4 (NIPS*92)*, S. 696–703, San Mateo, CA, 1993. Morgan Kaufmann.

[TWPS91a] **J. Tebelskis, A. Waibel, B. Petek und O. Schmidbauer**. Continuous Speech Recognition by Linked Predictive Neural Networks. In R. Lippmann, J. Moody und D. Touretzky (Hrsg.), *Advances in Neural Information Processing Systems 2 (NIPS*88*, S. 199–205, San Mateo, CA, 1991. Morgan Kaufmann.

[TWPS91b] **J. Tebelskis, A. Waibel, B. Petek und O. Schmidbauer**. Continuous Speech Recognition Using Linked Predictive Neural Networks. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 61–64, Toronto, Canada, Mai 1991. IEEE.

[Wai89] **A. Waibel**. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation, MIT-Press*, 1(1):39–46, März 1989.

[WB93] **C. Windheuser und F. Bimbot**. Phonetic Features for Spelled Letter Recognition with a Time Delay Neural Network. In *EUROSPEECH'93 (3rd European Conference on Speech Communication and Technology)*, Band 2, S. 1489–1492, Berlin, September 1993.

[WHH+87] **A. Waibel, T. Hanazawa, G. Hinton, K. Shikano und K. Lang**. Phoneme Recognition Using Time-Delay Neural Networks. Technical Report TR-1-0006, ATR Interpreting Telephony Research Laboratories, Oktober 1987.

[WHH+89] **A. Waibel, T. Hanazawa, G. Hinton, K. Shikano und K. Lang**. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-37(3):328–339, März 1989. (Auch in [WL90], S. 393–405).

[WL90] **A. Waibel und K.-F. Lee** (Hrsg.). *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, CA, 1990.

[WSS89a] **A. Waibel, H. Sawai und K. Shikano**. Consonant Recognition by Modular Construction of Large Phonemic Time-Delay Neural Networks. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 112–115, Glasgow, Scotland, Mai 1989.

[WSS89b] **A. Waibel, H. Sawai und K. Shikano**. Modularity and Scaling in Large Phonemic Neural Networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-37(12):1888–1898, Dezember 1989.

[WY83] **A. Waibel und B. Yegnanarayana**. Comparative study of nonlinear time warping techniques in isolated word speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31(6):1582–1586, Dezember 1983.

[ZHW93] **T. Zeppenfeld, R. Houghton und A. Waibel**. Improving the MS-TDNN for Word Spotting. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 475–478, Minneapolis, MN, April 1993. IEEE.

[ZW97] **P. Zhan und M. Westphal**. Speaker Normalization Based on Frequency Warping. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, München, April 1997. IEEE.

Abbildungsverzeichnis

Tabellenverzeichnis

Sach- und Personenverzeichnis