Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular

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

Dissertation

von

Ivica Rogina
aus Zapresić

Tag der mündlichen Prüfung:	26. Juni 1997
Erster Gutachter:	Prof. Dr. A. Waibel
Zweiter Gutachter:	Prof. Dr. H.-H. Nagel

Berichte aus der Informatik

Ivica Rogina

Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular

Shaker Verlag
Aachen 1998

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Rogina, Ivica:
Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular / Ivica Rogina. - Als Ms. gedr. -
Aachen : Shaker, 1998
(Berichte aus der Informatik)
Zugl.: Karlsruhe, Univ., Diss., 1997
ISBN 3-8265-3281-3

Als Manuskript gedruckt. Printed in Germany.

ISBN 3-8265-3281-3
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

Danksagung

Ich möchte mich bedanken bei Herrn Prof. Dr. Alexander Waibel für die Betreuung meiner Arbeit. Er gab mir wichtige wissenschaftliche Impulse und ermöglichte mir durch einen Auslandsaufenthalt an der Carnegie Mellon University in Pittsburgh, USA, wertvolle Einblicke in die Praxis der Spracherkennungsforschung in den wichtigsten Labors der Welt. Seine stete Forderung nach Perfektion und sein Vertrauen auf den Erfolg war mir eine Hilfe bei der Mitwirkung an der Entwicklung eines der weltweit besten Spracherkenner.

Herrn Prof. Dr. Hans-Hellmut Nagel danke ich für die Übernahme des Korreferendariats und für seine wertvollen Ratschläge zur zügigen Durchführung der Promotion. Seine zahlreichen Anregungen halfen mir sehr bei der Fertigstellung der Dissertation.

Mein Dank gilt auch meinen Kollegen, Monika Woszczyna, Martin Westphal und ganz besonders Michael Finke für ihre hervorragende Leistung bei der Programmierung des JANUS-Spracherkenners und ihr ständiges Interesse an meiner Arbeit. Michael Finke war mit seiner Fachkompetenz stets eine Quelle vieler guter Ideen.

Mit Tanja Schultz, Jürgen Fritsch und Tilo Sloboda durfte ich interessante Aufgaben auf dem Gebiet der Spracherkennung bearbeiten, deren Ergebnisse, wenn auch nicht Bestandteil der vorliegenden Arbeit, sehr inspirierend waren.

Mit Petra Geutner, Thomas Kemp, Thomas Schaaf und Hermann Hild konnte ich in Karlsruhe sehr fruchtbare Diskussionen und interessante Fachgespräche über spezielle Themen der Spracherkennung führen. Dafür

möchte ich ihnen danken. Während meines Aufenthaltes an der Carnegie Mellon University gaben mir Torsten Zeppenfeld, Bernhard Suhm, Klaus Ries, Joe Tebelskis und Arthur McNair hilfreiche Anregungen.

Danke auch an die Kolleginnen und Kollegen, die in den Interactive Systems Labs an Themen außerhalb der Spracherkennung arbeiten. Rainer Stiefelhagen, Uwe Meier, Stefan Manke und insbesondere unsere geduldigen und hilfsbereiten Sekretärinnen Silke Dannenmeier, Ingrid Gemen und Sonja Seitz sorgten für ein angenehmes Arbeitsklima. Ohne unsere tatkräftigen Systemadministratoren Markus Baur, Frank Dreilich und Martin Klein wäre so manches Experiment den Launen der Hardware zum Opfer gefallen.

Meinem Kollegen Hermann Hild danke ich für seine Ratschläge und zahlreichen aufmunternden Gespräche zum Schreiben der Dissertation, sowie für das Korrekturlesen derselben.

Schließlich gebührt mein Dank noch meiner Frau Susanne Kaufmann für ihre moralische Unterstützung während einer auch für sie durch ihre eigene Promotion bedingte anstrengende Zeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Schon Ende der vierziger, Anfang der fünfziger Jahre [DBB52] [OB56] wurden Forschungsarbeiten durchgeführt, um aus digitalisierten Sprachsignalen deren Inhalt zu extrahieren. Die ersten Experimente wurden noch auf dem reinen Signal gemacht. Später ging man dazu über, das Signal vorzuverarbeiten, so daß als Merkmalsraum für die Erkennung nicht mehr der Zeitbereich, sondern der Frequenzbereich des Signals verwendet wurde. Die erstaunlich guten Leistungen von Experten [KS73], die anhand einer vorliegenden Spektralanalyse eines Stückes Sprache das Gesagte korrekt herauslesen konnten, ermunterte die Forscher dazu, das Spektrum als das Merkmal der Wahl für die automatische Spracherkennung zu verwenden.

Die frühen Forschungsarbeiten wurden mit Aufnahmen von Vokalen eines Sprechers durchgeführt [Fry59] [FF59]. Erst später fing man an, größere Spracheinheiten zu erkennen. Als es möglich war, ganze Wörter zu erkennen [Ita75] [RLRW79], entstanden verschiedene Anwendungen, wie zum Beispiel die Steuerung von Geräten oder die Identifizierung von Sprechern. Der Benutzer mußte jedes zu erkennende Wort ein oder mehrere Male sprechen, damit der Erkenner Referenzmuster anlegen konnte. Während der Erkennung wurde dann das Gesprochene mit den gespeicherten Referenzen verglichen und die Klasse des am besten passenden Musters identifiziert.

In den siebziger Jahren wurde die Erkennung von kontinuierlicher sowie sprecherunabhängiger Sprache vorangetrieben [Kla77]. Die Größen der verwendeten Wortschätze und die Vielfalt der akustischen Eigenschaften verschiedener Sprecher machten es nicht mehr praktikabel, für jedes zu

erkennende Wort Referenzmuster zu sammeln und abzuspeichern. Die angelegten Referenzmuster bezogen sich nun auf kleinere Spracheinheiten, wie Phoneme, aus denen jedes Wort des Erkennervokabulars konkateniert werden konnte. Der Einsatz kontinuierlicher Sprache brachte einen weiteren Schwierigkeitsgrad mit sich. Nun waren die Grenzen der einzelnen Wörter nicht mehr vorgegeben und mußten vom Erkenner selbst gefunden werden, so daß neuartige Fehler wie das Nichterkennen gesprochener Wörter oder das fälschlicherweise Erkennen nicht gesprochener Wörter auftraten. Außerdem werden die Wörter in kontinuierlicher Sprache anders ausgesprochen, die Satzmelodie fängt an eine Rolle zu spielen, und an den Wortgrenzen treten Koartikulationseffekte auf. Zur Lösung all dieser Probleme boten sich Hidden Markov Modelle als die geeignetste Lösung an. Sie boten die Möglichkeit, komplexe Spracheinheiten wie Wörter oder Sätze aus kleineren Einheiten leicht zusammenzusetzen. Suchtechniken ermöglichten es, unter Zuhilfenahme von Sprachmodellen Wortfolgen zu erkennen, die nie zuvor in der Entwicklung der Erkenner oder der Sprachmodelle beobachtet wurden.

Verschiedene Anwendungen für das Erkennen sprecherunabhängiger Sprache wurden entwickelt. Dazu gehörten so einfache Dinge wie das Bedienen von Geräten, aber auch so komplizierte wie die Übersetzung in eine andere Sprache [LWL+97]. Lange Zeit war eine ausreichend gute Erkennung nur bei einer Einschränkung der Domäne gesichert. Das Diktieren beliebiger Texte war immer noch nur mit vorgegebenen kurzen Pausen zwischen je zwei Wörtern sinnvoll möglich. Bis heute hat sich auf dem Markt noch kein Produkt etabliert, das es erlaubt, beliebige Diktate von beliebigen Sprechern mit zufriedenstellender Genauigkeit zu erkennen. Die Anforderungen an ein Diktiersystem sind normalerweise so hoch, daß jedes mißverstandene Wort als unakzeptabler Fehler angesehen wird. Bei anderen Anwendungen, bei denen die Sprache dazu verwendet wird, eine bestimmte Aktion zu initiieren, können Fehlerkennungen toleriert werden, solange immer noch die korrekte Aktion durchgeführt wird. Diktierte Sprache ist in der Regel anders als spontane Sprache, weil der Diktierende sorgfältiger spricht und versucht, grammatikalische Fehler und Störungen wie Geräusche oder Stottern zu vermeiden. Das heißt, daß die Erkennung spontaner Sprache eine etwas größere Herausforderung darstellt, was sich auch in der Praxis durchweg in Form schlechterer Erkennungsraten bemerkbar macht. Dennoch lassen sich die meisten Erkenntnisse, die in der Forschung mit diktierter Sprache gewonnen werden, auch auf die Erkennung spontaner Sprache anwenden.

Einer der wichtigsten Motoren der Spracherkennungsforschung waren die Programme der „Defense Advanced Research Projects Agency" (DARPA) der US Regierung, von deren Seite zunehmend Wert darauf gelegt wird, Sprache unter widrigen Bedingungen (schlechte Aufnahmequalität, laute Hintergrundgeräusche, Nichtmuttersprachler, Telefonsprache etc.) zu erkennen. Obwohl diese Ziele erstrebenswert sind, ist die Leistung der weltbesten Erkenner auf sogenannter „sauberer" Sprache noch lange nicht gut genug, um das Problem als gelöst zu betrachten. Aus Untersuchungen [EP95] [DGDP96] weiß man, daß Menschen beliebige saubere Aufnahmen in ihrer Muttersprache mit weniger als einem Prozent Fehler erkennen können. Davon sind die heutigen Spracherkenner noch eine Größenordnung entfernt. Während die wesentlichen Unterschiede bei den Erkennern für problematische Sprache in den Techniken der Adaption an die aktuellen Aufnahmebedingungen bestehen, unterscheiden sich die Erkenner für saubere diktierte Sprache vor allem in der Gestaltung ihrer Parameterräume und den verwendeten Verfahren zur Schätzung der Parameter. Verschiedene Arten, Parameter zu allozieren oder miteinander zu koppeln, führen zu verschiedenen Arten von Hidden Markov Modellen.

Die Motivation für die vorliegende Arbeit war gegeben durch den Wunsch, einen Erkenner zu entwickeln, der mit uneingeschränktem Vokabular unbegrenztes Diktieren ermöglichen sollte. Dazu war es nötig, die Fehlerrate bei großen Vokabularen deutlich zu senken. Dem Autor gelang es, den Fehler von anfangs über 22% auf unter 8% zu dritteln. Auf absehbare Zeit wird es keinen Spracherkenner geben, der in allen Domänen und unter beliebigen Randbedingungen gut arbeitet. Um aber zu ermöglichen, daß für eine definierte, klar umrissene Aufgabe ein Erkenner auch von Nichtfachleuten gebaut werden kann, besteht ein Interesse daran, den Anteil der sogenannten „Schwarzen Magie" im Entwurf von Spracherkennern zu verringern. Viele Entwurfsvariablen und Trainingsmethoden sind mathematisch nicht fundiert, unbegründet oder widersprechen sogar der Theorie. Allein schon die Verwendung von Hidden Markov Modellen erster Ordnung macht eine nicht begründete Annahme über die Natur der Sprache, indem eine Unabhängigkeitsvermutung getroffen wird, die besagt, daß die Eigenschaften eines Zustandes nicht von den zeitlich davorliegenden Zuständen abhängen. Viele Entwurfsentscheidungen werden von Forschern durch auf Erfahrung basierende Schätzungen getroffen, oder in zeitrauben-

den Versuchsreihen herbeigeführt. Meist werden solche „fudge factors" in den Veröffentlichungen nicht erwähnt. Oft werden nur die Entwurfsentscheidungen beschrieben, aber nicht der Prozeß, der zur Entscheidung führte.

In der vorliegenden Arbeit wird beschrieben, wie der JANUS-Spracherkenner der Universität Karlsruhe zu einem System entwickelt wurde, das nun zu den derzeit besten der Welt zählt. Verschiedene Verfahren werden vorgestellt, mit deren Hilfe die Erkennungsrate verbessert, der Parameterraum kompaktifiziert, und Entwurfsentscheidungen vereinfacht werden können.

Kapitel 2 führt in die Problematik der Erkennung kontinuierlicher Sprache mit Hidden Markov Modellen ein. Dort werden die in den folgenden Kapiteln verwendeten Begriffe und Notationen vorgestellt.

In Kapitel 3 werden die in HMM-Erkennern verwendeten Parameterräume und deren verschiedene Erscheinungsformen beschrieben. Es wird erklärt, wie durch Kopplung der Parameter unterschiedliche Kontinuierlichkeitsgrade von diskreten über semikontinuierliche bis hin zu voll kontinuierlichen HMMs entstehen.

Die Wall Street Journal Datenbasis, die von der DARPA viele Jahre als Vergleichstest für Diktiersystem verwendet wurde, wird in Kapitel 4 vorgestellt. Soweit nicht explizit anders erwähnt, wurden alle hier berichteten Fehlerraten auf der offiziellen Testmenge vom November 1994 oder einer repräsentativen Untermenge davon gemessen.

Kapitel 5 beschreibt den JANUS-Erkenner als Werkzeug, so wie er für alle hier beschriebenen Experimente verwendet wurde. Selbstverständlich ist ein großes Softwaresystem wie JANUS eine gemeinschaftliche Entwicklung vieler. Der Autor war jedoch für die Programmierung großer Teile des Systems verantwortlich.

In Kapitel 6 werden Untersuchungen über verschiedene Arten, den Parameterraum zu gestalten, beschrieben. Es wird ein neuartiges Verfahren vorgestellt, das es ermöglicht, die Größe des Parameterraumes automatisch zu optimieren, das dabei die Übereinstimmung der Welt mit ihrem Modell gegenüber vorher verwendeten Methoden verbessert und schließlich die Feh-

lerrate signifikant senkt.

Dabei wurden zwei Probleme angegangen: die Frage nach einem geeigneten Distanzmaß für die Ballung akustischer Modelle, und die Frage nach einer geeigneten Anzahl akustischer Modelle. Das bisher in JANUS und anderen Spracherkennern verwendete Distanzmaß hatte zum Ziel, den Informationsgehalt der Systemparameter zu maximieren. In dieser Arbeit wird die Hypothese, daß eine Maximierung der Wahrscheinlichkeit der Trainingsdaten bei gegebenem akustischen Modell bessere Ergebnisse erzielt, bestätigt. Darüber hinaus wird gezeigt, daß durch Verwenden einer Kreuzvalidierungsmenge ein Haltekriterium für den Ballungsprozeß definiert werden kann, wodurch die Entscheidung für die zu verwendende Anzahl der akustischen Modelle automatisiert werden kann.

Kapitel 7 befaßt sich mit der Kompaktifizierung der Parameterräume. Verschiedene Ansätze werden untersucht und verglichen. Es wird gezeigt, daß es möglich ist, den Parameterraum wesentlich zu verkleinern, dabei die Erkennungsgeschwindigkeit zu steigern und auch noch die Fehlerrate zu senken.

Die Erfahrung mit Spracherkennern hat – wie schon in [DH73] festgestellt – gezeigt, daß größere Parameterräume bessere Erkennungsraten hervorbringen, solange die zur Verfügung stehenden Trainingsdaten in etwa proportional zur Parameterraumgröße vorhanden sind. Dennoch ist man an kompakteren Parameterräumen interessiert, weil so der benötigte Speicher und Rechenzeitaufwand eingeschränkt werden kann. In dieser Arbeit wurde untersucht, inwieweit der Parameterraum eines trainierten Erkenners ohne negative Seiteneffekte reduziert werden kann. Dabei wurde die Hypothese bestätigt, daß die Generalisierungsfähigkeit und damit auch die Erkennungsleistung erhöht werden kann, wenn die Zahl der Systemparameter durch Radialisierung von Kovarianzmatrizen deutlich verringert wird.

Kapitel 8 beschreibt die Komponenten und die Entwicklungsphasen des JANUS-Diktiererkenners vom Frühjahr 1997, der eine Fehlerrate von unter 8% erzielte, ein Wert, der derzeit zu den bestmöglichen Erkennungsraten auf der Teststichprobenmenge gehört. Einen wichtigen Schritt auf dem Weg der Verbesserung stellte die Verwendung von Polyphonen dar. Zuvor waren die Phoneme als Triphone modelliert, also in Abhängigkeit von den unmittelbaren Nachbarphonemen. Durch den Einsatz von Polyphonen konnte die Hypothese bestätigt werden, daß die Modellierung breiterer Kontexte eine

bessere Erkennungsleistung zur Folge hat. Weitere bedeutsame Beiträge wurden durch die Vergrößerung des Erkennervokabulars, die Optimierung des Aussprachelexikons und den Einsatz eines Adaptionsverfahrens erzielt. Die Beiträge der einzelnen Verbesserungen sind erfahrungsgemäß nicht additiv. Andererseits sind manche Verbesserungen erst zusammen mit anderen wirklich effektiv. Eine vollständige Untersuchung der synergetischen Auswirkungen verschiedener Kombinationen von Verbesserungen ist aus rein praktischen Gründen, vor allem wegen des enormen Rechenzeitbedarfs, nicht möglich. Deshalb ist das in dieser Arbeit präsentierte Ergebnis nicht als endgültig zu verstehen. Es stellt eine Momentaufnahme eines Erkenners dar, der sich in ständiger Weiterentwicklung befindet.

Kapitel 2

Akustische Modellierung

2.1 Das Spracherkennungsproblem

Sprache erkennen bedeutet, zu einem gegebenen Sprachsignal, das in der Regel als eine zeitliche Folge von Merkmalen vorliegt, diejenige Wortfolge zu finden, für die die Wahrscheinlichkeit gemäß des erkennerinternen Modells der Sprache am größten ist. Üblicherweise wird das Spracherkennungsproblem als ein zweigeteiltes Problem beschrieben, bei dem der eine Teil das „Sprachmodell", der andere Teil die „akustische Modellierung" genannt werden. Die Trennung dieser beiden Probleme veranschaulicht die Gleichung 2.1.

$$P(W|X) = \frac{p(X|W) \cdot P(W)}{p(X)} \tag{2.1}$$

Hierbei ist $P(W|X)$ die Wahrscheinlichkeit, daß die Wortfolge W gesprochen wurde, unter der Voraussetzung, daß das Signal X beobachtet wird. $p(X)$ ist die a-priori Dichte dafür, daß das Signal X überhaupt beobachtet werden kann, und $P(W)$ ist die a-priori Wahrscheinlichkeit dafür, daß die Wortfolge W gesprochen wird. $p(X|W)$ ist somit die bedingte Dichte dafür, daß das Signal X beobachtet wird, wenn die Wortfolge W gesprochen wurde.

Sprache erkennen bedeutet nun, für ein gegebenes Signal X die wahrscheinlichste Wortfolge $\bar{W}$ zu finden, also:

$$\bar{W} = \underset{W}{\operatorname{argmax}}\, P(W|X) \tag{2.2}$$

$$= \underset{W}{\operatorname{argmax}} \frac{p(X|W) \cdot P(W)}{p(X)}$$
$$= \underset{W}{\operatorname{argmax}} (p(X|W) \cdot P(W))$$

Den Teil eines Spracherkenners, der $P(W)$ berechnet, nennt man das „Sprachmodell" und den Teil, der $p(X|W)$ berechnet, nennt man die „akustische Modellierung". Das Sprachmodell beschäftigt sich also mit der Problematik der Wahrscheinlichkeit von Wortfolgen, unabhängig davon, wie das Signal einer Sprachaufnahme aussieht. Im Gegensatz dazu ist die Aufgabe der akustischen Modellierung zu berechnen, mit welcher Wahrscheinlichkeit sich eine gesprochene Wortfolge so anhört wie ein vorliegendes Signal.

2.2 Methoden der akustischen Modellierung

Falls die Menge der erlaubten Wortfolgen endlich und klein ist, ist es möglich, für jedes erlaubte W ein eigenes $p(X|W)$ zu modellieren. Viele Einzelworterkenner verwenden diese Technik, indem sie für jede erlaubte Wortfolge ein oder mehrere Referenzmuster speichern und $p(X|W)$ aus der Distanz von X zu den Referenzmustern berechnen.

Bei der Erkennung von kontinuierlicher Sprache ist es aber wegen der unendlichen Menge von denkbaren Wortfolgen nicht möglich, für jede Wortfolge ein eigenes $p(X|W)$ zu modellieren. Statt dessen werden Wortfolgen in Folgen kleinerer Spracheinheiten (z.B. Wörter, Silben, Phoneme oder Phonemsegmente) unterteilt. Durch diese Unterteilung ergeben sich allerdings einige zusätzliche Probleme. Es wird ein sogenanntes Aussprachelexikon benötigt, das jede Wortfolge auf die entsprechenden Untereinheiten abbildet. Ein weiteres Problem ist das der zeitlichen Zuordnung (engl. time alignment). Während bei der Abbildung einer Sprachaufnahme auf eine Wortfolge der Anfang und das Ende der Wortfolge durch die Aufnahme gegeben sind, müssen bei der Erkennung von Untereinheiten die Anfänge und Enden derselben erst gefunden werden.

Die Erkennung von kontinuierlicher Sprache wird zusätzlich dadurch erschwert, daß Koartikulationseffekte bei Wortübergängen auftreten. Dasselbe Wort wird oft sehr unterschiedlich gesprochen, je nachdem in welchem Kontext es auftritt. Die Laute am Anfang und am Ende des Wortes

werden oft mit den Randlauten der angrenzenden Wörter verschliffen. Manchmal wird sogar das komplette Wort anders ausgesprochen. Diese wortübergreifenden Koartikulationseffekte treten in verschiedenen Sprachen verschieden stark auf, ganz besonders stark im amerikanischen Englisch.

Zwei Arten von Erkennungsfehlern treten bei der Erkennung kontinuierlicher Sprache im Gegensatz zur Erkennung isolierter einzelner Wörter zusätzlich auf, nämlich Auslassungen und Einfügungen einzelner Wörter. Während bei der isolierten Worterkennung mit vorgegebenen Wortgrenzen nur Vertauschungen vorkommen können, werden bei der kontinuierlichen Spracherkennung auch Wörter erkannt, die nicht gesprochen worden sind, und tatsächlich gesprochene Wörter werden überhaupt nicht erkannt. So ist es zum Beispiel auch möglich, eine Fehlerrate über 100% zu erhalten, wenn der Satz „Hallo." fälschlicherweise erkannt wird als „Halt, oh.". Dann werden nämlich auf einem zu erkennenden Wort zwei Fehler gemacht, und die Fehlerrate beträgt somit 200%.

Alle derzeit bekannten Spracherkenner für kontinuierliche Sprache basieren auf der Idee der Hidden-Markov-Modelle (HMM) oder einer Abwandlung derselben. Lediglich die Art der Berechnung der Emissionswahrscheinlichkeiten variiert. Die verbreitetsten Methoden sind Mixturen multivariater Normalverteilungen (manchmal auch Laplace-Verteilungen) und auf mehrschichtigen Perzeptronen basierende künstliche neuronale Netze. Die Spracherkenner, die künstliche neuronale Netze zur Berechnung der Emissionswahrscheinlichkeiten verwenden, werden „hybrid" genannt, im Gegensatz zu den „reinen" HMM Ansätzen, die nur parametrische Wahrscheinlichkeitsverteilungen verwenden.

Obwohl die Mehrzahl der in der Forschung benutzten Spracherkenner parametrische Wahrscheinlichkeitsverteilungen schätzen [PFGP96] [PGF+95], sind künstliche neuronale Netze in ihrer Erkennungsleistung durchaus konkurrenzfähig [KRR96].

Die vorliegende Arbeit befaßt sich ausschließlich mit reinen HMMs. Wenn im folgenden auf die Definition von HMMs eingegangen wird, so geschieht dies nicht zur Einführung in die HMM-Problematik, sondern lediglich zur Vorstellung der verwendeten Nomenklatur. Eine gute Einführung in die Theorie und Praxis der Spracherkennung mit HMMs liefern zum Beispiel

[Rab89] und [HJ89].

2.3 Warum Hidden-Markov-Modelle?

Man kann Sprache als einen stochastischen Prozeß betrachten. Es ist offensichtlich, daß dasselbe Phonem von verschiedenen Menschen verschieden ausgesprochen wird, und selbst derselbe Mensch spricht es zu verschiedenen Zeiten unterschiedlich aus. Hidden-Markov-Modelle sind geeignet, um zustandsgebundene stochastische Prozesse zu modellieren, bei denen bestimmten Zuständen Wahrscheinlichkeitsverteilungen für Beobachtungen aus dem Merkmalsraum sowie Wahrscheinlichkeiten für Übergänge in andere Zustände zugeordnet sind.

Ein Beispiel für ein einfaches Hidden-Markov-Modell ist ein Modell des Wetters. Man nimmt dazu an, daß das Wetter sich in bestimmten Zuständen befinden kann. Der Einfachheit halber seien dies nur zwei Zustände: „gutes Wetter" und „schlechtes Wetter". Mit einer bestimmten Wahrscheinlichkeit folgt gutem Wetter schlechtes Wetter, und mit der Restwahrscheinlichkeit folgt gutem Wetter wieder gutes Wetter. Diese Wahrscheinlichkeiten sind die Übergangswahrscheinlichkeiten. Nun kann man auch jeden Tag die Temperatur messen (das heißt beobachten). Die zu erwartende Temperaturverteilung wird an Gutwettertagen anders als an Schlechtwettertagen ausfallen. Diese Verteilungen nennt man Emissionswahrscheinlichkeiten, weil das Wetter die Temperatur „emittiert". Abbildung 2.1 zeigt die typischerweise gewählte Darstellung für ein HMM wie das eben vorgestellte.

Wenn nun ein Modell des Wetters entworfen werden soll, dann wird man für die Schätzung der Parameter der Übergangs- und Emissionswahrscheinlichkeiten einige Trainingsdaten verwenden, also eine Reihe gemessener Temperaturen und die dazugehörige Information, ob sie bei gutem oder schlechtem Wetter gemessen wurden. Die drei fundamentalen HMM-Probleme sind definiert als das Trainieren eines HMMs, das Berechnen der Wahrscheinlichkeit für eine Zustandsfolge bei gegebener Beobachtung, also zum Beispiel die Wahrscheinlichkeit dafür, daß drei Tage lang schönes Wetter ist bei Temperaturen von 20 Grad, und schließlich das Problem des Findens der wahrscheinlichsten Zustandsfolge bei gegebener Beobachtung.

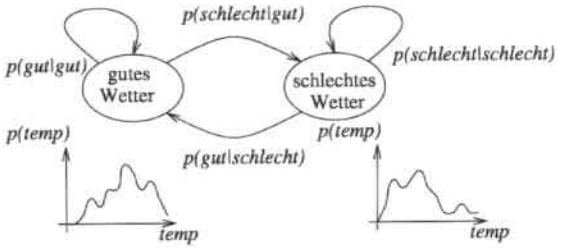


Abbildung 2.1: Ein einfaches HMM

2.4 Die HMM-Problematik

Ein HMM erster Ordnung ist definiert als ein Fünftupel λ mit folgenden Komponenten:

- S die Menge aller HMM-Zustände, $S = \{s_1 \dots s_n\}$
- π die Wahrscheinlichkeitsverteilung, die jedem Zustand s_i die Wahrscheinlichkeit $\pi(s_i)$ zuordnet, mit der s_i als erster Zustand einer Zustandsfolge auftritt
- A die Matrix der Übergangswahrscheinlichkeiten, $A = (a_{ij})$, wobei $a_{ij} = p(s_j|s_i)$ die bedingte Wahrscheinlichkeit dafür ist, daß dem gegebenen Zustand s_i der Zustand s_j folgt
- B die Menge der Emissionswahrscheinlichkeitsverteilungen, $B = \{b_1 \dots b_n\}$, wobei $b_i(x)$ die bedingte Wahrscheinlichkeit oder Dichte dafür ist, daß im gegebenen Zustand s_i der Punkt x des Merkmalsraumes beobachtet wird
- V die Menge der beobachtbaren Merkmale, die entweder diskret sein kann ($V = \{v_1 \dots v_{|V|}\}$) oder auch kontinuierlich ($V = \mathbb{R}^d$)

In einigen Literaturstellen findet man die Definition der Emissionswahrscheinlichkeiten als Funktion der Zustandsübergänge, so daß $B = (b_{ij})$, wobei $b_{ij}(x)$ die Wahrscheinlichkeit dafür ist, daß x beim Übergang vom Zustand s_i zum Zustand s_j beobachtet wird. In ihrer Modellierungsmächtigkeit sind diese beiden Ansätze aber gleich.

Die Emissionswahrscheinlichkeit $b_i(x)$ läßt sich auch schreiben als $p(x|s_i)$. Die Bestimmung der bedingten Wahrscheinlichkeit $p(x_1 \dots x_T|\lambda)$, eine Merkmalsfolge $x_1 \dots x_T$ bei gegebenem HMM λ zu beobachten, ist das sogenannte Evaluierungsproblem der HMMs.[1] Es gilt:

$$p(x_1 \dots x_T|s_{i_1} \dots s_{i_T}, \lambda) = \prod_{t=1}^{T} b_{i_t}(x_t)$$

$$p(s_{i_1} \dots s_{i_T}|\lambda) = \pi(s_{i_1}) \cdot \prod_{t=2}^{T} a_{i_{t-1}i_t}$$

und somit

$$\begin{aligned} p(x_1 \dots x_T|\lambda) &= \sum_{i_1 \dots i_T} p(x_1 \dots x_T|s_{i_1} \dots s_{i_T}, \lambda) \cdot p(s_{i_1} \dots s_{i_T}|\lambda) \\ &= \sum_{i_1 \dots i_T} \pi(s_{i_1}) b_{i_1}(x_1) \cdot \prod_{t=2}^{T} a_{i_{t-1}i_t} b_{i_t}(x_t) \end{aligned} \tag{2.3}$$

Neben dem Evaluierungsproblem gibt es noch das Dekodierungsproblem und das Optimierungsproblem. Das Dekodierungsproblem ist die Frage nach der wahrscheinlichsten Zustandsfolge[2] bei einer gegebenen Beobachtung $x_1 \dots x_T$. Gesucht ist also:

$$\underset{i_1 \dots i_T}{\operatorname{argmax}} \quad p(s_{i_1} \dots s_{i_T}|x_1 \dots x_T, \lambda) \tag{2.4}$$

Das Dekodierungsproblem wird gelöst durch den sogenannten „Forward-Backward"-Algorithmus [BE67] [BS68] oder den „Viterbi"-Algorithmus [Vit67] [For73]. Eine Lösung des Dekodierproblems wird gerne als ein Pfad in einer Matrix wie in Abbildung 2.2 dargestellt.

Das Optimierungsproblem ist definiert als das Finden von $\hat{\lambda} = \operatorname{argmax}_\lambda p(x_1 \dots x_T|\lambda)$. Eine analytische Lösung für dieses Problem existiert nicht. Man begnügt sich aber damit, eine iterative Methode

[1] in späteren Kapiteln wird die Bedingung λ gelegentlich weggelassen, wenn kein Zweifel über das zugrundeliegende HMM vorliegt

[2] oder einer in welcher Beziehung auch immer „optimalen" Zustandsfolge

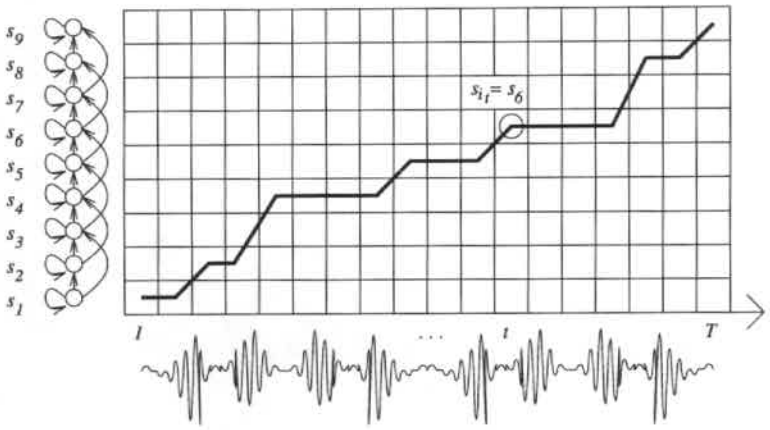


Abbildung 2.2: Lösung des Dekodierproblems: Viterbi-Pfad

zu verwenden, die zu einem gegebenen λ ein $\bar{\lambda}$ findet, so daß die Wahrscheinlichkeit, eine Merkmalsfolge $x_1 \ldots x_T$ zu beobachten, mit den Parametern von $\bar{\lambda}$ größer ist als mit den Parametern von λ, also $p(x_1 \ldots x_T|\bar{\lambda}) > p(x_1 \ldots x_T|\lambda)$.

Für die Modellierung der Emissionswahrscheinlichkeiten durch Mixturen von Normalverteilungen läßt sich beweisen [HAJ90], daß der weiter unten beschriebene EM-Algorithmus (Expectation Maximization) das Optimierungsproblem löst.

2.5 Emissionswahrscheinlichkeiten mit Gauß-Mixturen

Bei den „reinen" HMMs werden die Emissionswahrscheinlichkeiten $p(x|s_i)$ modelliert mit gewichteten Mixturen von Normalverteilungen, also:

$$p(x|s_i) = \sum_{k=1}^{n_i} \gamma_i(k) \cdot N(x, \mu_i(k), \Sigma_i(k)) \tag{2.5}$$

wobei $N(x, \mu, \Sigma)$ die Normalverteilung mit Mittelwertsvektor μ und Kovarianzmatrix Σ an der Stelle x darstellt. $\gamma_i(k)$ ist das k-te Mixturgewicht, mit dem die k-te Normalverteilung gewichtet in die Summe eingeht. In der Spracherkennung haben sich die Begriffe Codebuch (engl. Codebook) und Mixturgewichteverteilung (engl. mixture weight distribution) etabliert. Die Mixturgewichteverteilung eines Zustandes s_i ist γ_i, und das Codebuch dieses Zustandes besteht aus μ_i und Σ_i, also aus n_i Referenzvektoren mit n_i zugehörigen Kovarianzmatrizen. Oft wird für „Mixturgewichteverteilung" einfach kurz „Verteilung" gesagt.

Bisher wurde davon gesprochen, daß jedem HMM-Zustand ein Codebuch und eine Verteilung entspricht. Dabei wurde außer acht gelassen, daß es üblich ist, verschiedene HMM-Zustände auf dieselbe Art akustisch zu modellieren, insbesondere wenn keine Ganzwortmodelle verwendet werden, sondern kleinere Spracheinheiten wie z.B. Phoneme. In diesen Fällen kann es vorkommen, daß zwei verschiedene HMM-Zustände s_i und s_j beide für die gleiche Spracheinheit stehen, die in einem Wort oder Satz mehrfach vorkommt. Dann gilt auch, daß $p(x|s_i) = p(x|s_j)$. Eigentlich müßte an dieser Stelle eine weitere Indirektion $p(x|s_i) = p(x|M(s_i))$ eingeführt werden, welche HMM-Zustände auf ihre zugehörigen eindeutigen akustischen Modelle abbildet. Der Autor verzichtet allerdings zur Vereinfachung und besseren Lesbarkeit darauf in der Erwartung, daß der Leser auch ohne diese Indirektion versteht, was mit $p(x|s_i)$ gemeint ist.

Vereinfachungen der Rechnung

In der Praxis werden die Emissionswahrscheinlichkeiten sehr selten genau so wie in Formel 2.5 berechnet. Üblich ist zum Beispiel für $\Sigma_i(k)$ diagonale Matrizen zu verwenden, also die (in der Regel falsche) Modellannahme zu treffen, daß die einzelnen Dimensionen des Merkmalsraumes unkorreliert sind.

Eine andere Methode, die Berechnung der Emissionswahrscheinlichkeiten zu vereinfachen, besteht in der so genannten „top-1" Methode, bei der statt der gewichteten Summe aller Normalverteilungen einer Mixtur nur der Wert der größten Normalverteilung benutzt wird, das heißt:

$$p(x|s_i) = \gamma_i(k_{\max}) \cdot N(x, \mu_i(k_{\max}), \Sigma_i(k_{\max})) \tag{2.6}$$

wobei

$$k_{\max} = \underset{k}{\operatorname{argmax}}\, N(x, \mu_i(k), \Sigma_i(k)) \tag{2.7}$$

Diese Approximation vermindert den Rechenaufwand, weil zum einen weniger Werte aufaddiert werden müssen, vor allem aber deshalb, weil viele $N(x, \mu_i(k), \Sigma_i(k))$ gar nicht erst vollständig berechnet werden müssen. Die Berechnung einer einzelnen Normalverteilung geht nämlich über eine Summe über die Dimensionen des Merkmalsraumes, die abgebrochen werden kann, sobald schon an der Zwischensumme erkannt wird, daß kein neues Maximum gefunden werden kann. Sobald dann der Wert $k_{\max}$ feststeht, wird angenommen, daß für alle $k \neq k_{\max}$ gilt $N(x, \mu_i(k), \Sigma_i(k)) = 0$. Diese Annahme ist deshalb begründet, weil bei hochdimensionalen Merkmalsräumen der Wert der zweitgrößten Normalverteilung einer Mixtur meist eine oder mehre Größenordnungen kleiner als der Wert der größten ist.

2.6 Das Dekodierungsproblem: Der Forward-Backward-Algorithmus

Die unmittelbare Lösung des Dekodierungsproblems von Gleichung 2.4 liefert der Viterbi-Algorithmus. Der Forward-Backward-Algorithmus ist seine Verallgemeinerung.

Wir interessieren uns für die Wahrscheinlichkeit $\alpha_t(j) = p(x_1 \ldots x_t, s_{i_t} = s_j|\lambda)$, also die Wahrscheinlichkeit, bei gegebenem HMM λ die Beobachtung $x_1 \ldots x_t$ zu machen und dann im Zustand s_j zu sein. Diese Wahrscheinlichkeit läßt sich induktiv definieren.

So ist $\alpha_1(j)$ die Wahrscheinlichkeit, eine Merkmalssequenz x_1, die aus einem einzigen Vektor besteht, zu beobachten und dabei im Zustand s_j zu sein. Sie ist einfach das Produkt der Emissionswahrscheinlichkeit, x_1 in s_j zu beobachten, und der Wahrscheinlichkeit, daß s_j als erster Zustand vorkommt:

$$\alpha_1(j) = \pi_j \cdot b_j(x_1) \tag{2.8}$$

Induktiv gilt dann:

$$\alpha_t(j) = \left(\sum_{k=1}^{n} \alpha_{t-1}(k) \cdot a_{kj} \right) \cdot b_j(x_t) \tag{2.9}$$

Das heißt, $\alpha_t(j)$ ist das Produkt aus der Wahrscheinlichkeit, x_t in s_j zu beobachten, und der Summe der Wahrscheinlichkeiten, auf irgendeine Weise aus einem vorhergehenden Zustand s_k nach s_j zu gelangen, gewichtet mit der Wahrscheinlichkeit, zuvor im Zustand s_k gewesen zu sein.

$\alpha_T(j)$ ist dann die Wahrscheinlichkeit, eine gesamte Beobachtung $x_1 \dots x_T$ zu machen und am Ende im Zustand s_j zu terminieren. Demnach gilt: bei gegebenem HMM λ ist die Wahrscheinlichkeit, $x_1 \dots x_T$ zu beobachten:

$$P(x_1 \dots x_T|\lambda) = \sum_{j=1}^{n} \alpha_T(j) \tag{2.10}$$

Die Berechnung von $\alpha_t(j)$ ist der „Forward"-Teil des Forward-Backward Algorithmus. Definieren wir $\beta_t(j)$ als $p(x_{t+1} \dots x_T|s_{i_t} = s_j, \lambda)$. Auch $\beta_t(j)$ läßt sich in Analogie zu $\alpha_t(j)$ induktiv berechnen:

$$\beta_T(j) = 1 \tag{2.11}$$

und für $t < T$

$$\beta_t(j) = \sum_{k=1}^{n} a_{jk} \cdot b_k(x_{t+1}) \cdot \beta_{t+1}(k) \tag{2.12}$$

$\beta_T(j)$ ist auf das neutrale Element der Multiplikation gesetzt, da es sich auf eine leere Beobachtung bezieht. Nun kann die Wahrscheinlichkeit, zum Zeitpunkt t im Zustand s_j zu sein, bei gegebener Beobachtung $x_1 \dots x_T$ und HMM λ berechnet werden als:

$$p(s_{i_t} = s_j|x_1 \dots x_T, \lambda) = \frac{p(x_1 \dots x_t, s_{i_t} = s_j|\lambda) \cdot p(x_{t+1} \dots x_T|s_{i_t} = s_j, \lambda)}{p(x_1 \dots x_T|\lambda)} \tag{2.13}$$

Diese Wahrscheinlichkeit läßt sich mit Hilfe von $\alpha_t(j)$ und $\beta_t(j)$ ausdrücken:

$$p(s_{i_t} = s_j | x_1 \dots x_T, \lambda) = \frac{\alpha_t(j) \cdot \beta_t(j)}{p(x_1 \dots x_T | \lambda)} = \frac{\alpha_t(j) \cdot \beta_t(j)}{\sum_{k=1}^{n} \alpha_t(k) \cdot \beta_t(k)} \tag{2.14}$$

Der Wert $p(s_{i_t} = s_j | x_1 \dots x_T, \lambda)$ ist für das Trainieren mit dem unten beschriebenen EM-Algorithmus von zentraler Bedeutung. Er bestimmt, mit welchem Anteil der Trainingsvektor x_t in die Schätzung der akustischen Parameter des HMM-Zustands s_{i_t} eingeht.

Für die Optimierung der Zustandsübergangswahrscheinlichkeiten wird zusätzlich die Wahrscheinlichkeit $\xi_t(k,j) = P(s_{i_{t-1}} = s_k, s_{i_t} = s_j | x_1 \dots x_T, \lambda)$ benötigt, also die Wahrscheinlichkeit, zum Zeitpunkt $t-1$ im Zustand s_k und zum Zeitpunkt t im Zustand s_j zu sein. Auch diese Wahrscheinlichkeit läßt sich mit Hilfe der α und β ausdrücken:

$$\xi_t(k,j) = \frac{\alpha_{t-1}(k) \cdot a_{kj} \cdot b_j(x_t) \cdot \beta_t(j)}{\sum_l \alpha_t(l) \cdot \beta_t(l)} \tag{2.15}$$

Strahlsuche

Typische Trainingssätze in der Wall Street Journal Datenbasis sind etwa 10 Sekunden lang und bestehen aus $T = 1000$ Mustervektoren. Die Anzahl der zugehörigen HMM-Zustände liegt in derselben Größenordnung. Dadurch ergeben sich sehr viele $\alpha_t(j)$ und $\beta_t(j)$, die nur mit sehr viel Aufwand komplett berechnet werden können.

Deshalb verwendet man üblicherweise eine Strahlsuche, bei der ein $\alpha_t(j)$ auf 0.0 gesetzt wird, falls $\alpha_t(j) < \epsilon \cdot \max_k \alpha_t(k)$. Wenn für ein t auf diese Art viele $\alpha_t(j)$ auf 0.0 gesetzt werden, so ergibt sich für den Zeitpunkt $t+1$ automatisch, daß einige $\alpha_{t+1}(k) = 0.0$ sein müssen, also „unerreichbar" sind. Dasselbe Prinzip wird auch auf den Rückwärtspaß und die $\beta_t(j)$ angewandt. Der Faktor ϵ wird Strahlbreite genannt. Für $\epsilon = 1.0$ findet eine vollständige Suche statt. Für kleinere Werte von ϵ wird der Suchraum eingeschränkt.

2.7 Das Dekodierungsproblem: Der Viterbi-Algorithmus

In der Praxis bringt der Forward-Backward-Algorithmus einige Probleme mit sich. Die vielfache Multiplikation der meist sehr kleinen Emissionswahrscheinlichkeiten (oftmals mehrere tausend) führt leicht zu numerischen Problemen, da der darstellbare Zahlenbereich schnell unterschritten wird. Während sich dieses Problem noch durch Rechnen im logarithmischen Bereich in den Griff bekommen läßt, ist der Zeitaufwand durch das häufige Logarithmieren und Exponentiieren erheblich. Außerdem ist die Verwendung einer Strahlsuche im Forward-Backward-Algorithmus problematisch, denn der eigentliche Vorteil der Strahlsuche, nämlich eine drastische Einschränkung des Suchraumes, kann dazu führen, daß der Vorwärtspfad, der durch die $\alpha_t(j)$ bestimmt ist, vom Rückwärtspfad der $\beta_t(j)$ so stark abweicht, daß eine Kombination der beiden wie in Gleichung 2.14 nicht mehr sinnvoll ist oder sogar dazu führt, daß für jedes Paar (j,t) gilt, daß entweder $\alpha_t(j) = 0$ oder $\beta_t(j) = 0$, also jeder Punkt des Suchraumes durch die Strahlsuche entweder beim Vorwärtspaß oder beim Rückwärtspaß ausgeblendet wurde. Abbildung 2.3 zeigt ein Beispiel, in dem der Vorwärts- und der Rückwärtspfad aneinander vorbeilaufen. Die dunklen Felder sind die mit hoher Wahrscheinlichkeit, die hellen diejenigen mit niedriger Wahrscheinlichkeit. Man sieht, daß sogar zwischen den beiden Pfaden weiße Felder entstehen können, die durch die Strahlsuche in beiden Richtungen ausgeblendet werden. Deshalb darf die Strahlbreite nicht zu klein gewählt werden, was zur Folge hat, daß der gesamte Algorithmus relativ lange Zeit benötigt.

Eine andere Variante der zeitlichen Zuordnung von Mustervektoren zu HMM-Zuständen ist der Viterbi-Algorithmus. Dieser bewertet nicht für jeden Zustand die Wahrscheinlichkeit, zu einem Zeitpunkt der aktuelle Zustand zu sein, sondern berechnet diejenige Zustandsfolge, deren Wahrscheinlichkeit bei gegebener Beobachtung am größten ist. Er löst also das Dekodierproblem aus Gleichung 2.4.

Das Ergebnis des Viterbi-Algorithmus ist also eine Zustandsfolge $s_{i_1} \dots s_{i_T}$. Sie kann als ein Spezialfall des Ergebnisses des Forward-Backward-Algorithmus betrachtet werden, nämlich für den Fall, daß

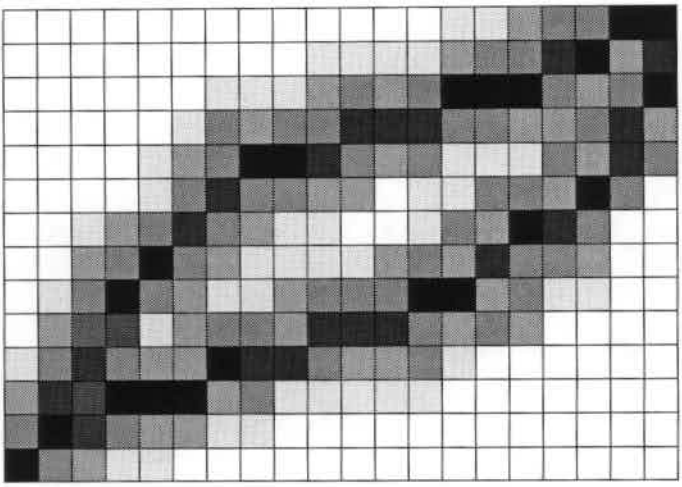


Abbildung 2.3: Ein schlechter Forward-Backward Pfad

$p(s_{i_t} = s_j | x_1 \dots x_T, \lambda) = \delta_{i_t,j}$. Das heißt, zu jedem Zeitpunkt t wird nur einem Zustand s_{i_t} die Wahrscheinlichkeit 1 zugewiesen, alle anderen haben die Wahrscheinlichkeit 0.

Das Viterbi-Verfahren ähnelt sehr stark den Verfahren der dynamischen Zeitanpassung („Dynamic Time Warping") oder der Methode zum Finden der minimalen Editierdistanz zwischen zwei Symbolfolgen.

Ziel des Viterbi Algorithmus ist es, die Folge

$$\begin{aligned} & \underset{i_1,\dots i_T}{\operatorname{argmax}} \quad p(s_{i_1} \dots s_{i_T} | x_1 \dots x_T, \lambda) \\ = & \underset{i_1 \dots i_T}{\operatorname{argmax}} \quad \frac{p(s_{i_1} \dots s_{i_T}, x_1 \dots x_T | \lambda)}{p(x_1 \dots x_T | \lambda)} \\ = & \underset{i_1 \dots i_T}{\operatorname{argmax}} \quad p(s_{i_1} \dots s_{i_T}, x_1 \dots x_T | \lambda) \end{aligned}$$

zu berechnen. Eine Lösung läßt sich induktiv entwickeln. Sei

$$q_t(j) = \max_{i_1 \ldots i_{t-1}} p(s_{i_1} \ldots s_{i_t} = j, x_1 \ldots x_t | \lambda) \tag{2.16}$$

die größte Wahrscheinlichkeit entlang eines partiellen Pfades $i_1 \ldots i_t$, der im Zustand s_j endet, die Beobachtung $x_1 \ldots x_t$ zu machen. $q_t(j)$ wird auch die „akkumulierte Wahrscheinlichkeit" des Zustandes s_j zum Zeitpunkt t genannt. Dabei ist

$$q_1(j) = \pi_j \cdot b_j(x_1) \tag{2.17}$$

und induktiv gilt:

$$q_{t+1}(j) = \left(\max_k q_t(k) a_{kj} \right) \cdot b_j(x_{t+1}) \tag{2.18}$$

Man sieht, daß $q_{t+1}(j)$ ganz analog zu $\alpha_{t+1}(j)$ im Forward-Backward-Algorithmus gebildet wird. Allerdings steht beim Viterbi-Algorithmus an der Stelle der Summe eine Maximierung (siehe Abbildung 2.4).

Auch im Viterbi-Algorithmus kann eine Strahlsuche verwendet werden, bei der $q_t(j)$ auf 0 gesetzt wird, falls $q_t(j) < \epsilon \cdot \max_k q_t(k)$.

Der Term $\text{argmax}_k\, q_t(k) a_{kj}$ wird im internen Sprachgebrauch als „Rückwärtszeiger" (backpointer) bezeichnet. Er gibt für jeden Zeitpunkt t und Zustand j denjenigen Vorgängerzustand an, von dem der optimale partielle Pfad gekommen sein muß. Wenn die obige Induktion bis zum Ende geführt ist, kann durch Rückverfolgen der Rückwärtszeiger der komplette optimale Pfad bestimmt werden.

Im allgemeinen Fall würde der Rückwärtsaufbau mit dem Zustand beginnen, der am wahrscheinlichsten der letzte Zustand ist. In der Praxis aber sieht die HMM-Struktur so aus, daß nur ein ganz bestimmter Zustand der letzte sein kann. Meist repräsentiert dieser letzte Zustand ebenso wie der einzig erlaubte Anfangszustand das akustische Stillemodell (siehe auch Abschnitt 5.2.3).

Das Ergebnis des Rückwärtspasses liefert also eine Zustandsfolge $(s_{i_1} \ldots s_{i_T})$. Diese kann als ein Spezialfall des Forward-Backward-Ergebnisses betrachtet werden, bei dem die Wahrscheinlichkeit, bei gegebener Beobachtung zum Zeitpunkt t im Zustand i_t zu sein, 1 ist und für alle anderen

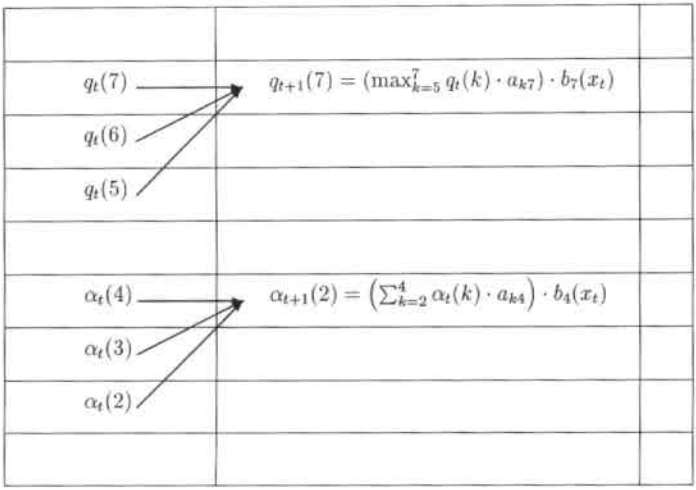


Abbildung 2.4: Unterschied zwischen Viterbi (oben) und Forward-Backward (unten)

Zustände 0 ist.

2.8 Das Optimierungsproblem: Der EM-Algorithmus

Der EM-Algorithmus (Expectation Maximization) [DLR77] ist eine Lösung für das HMM-Optimierungsproblem. Der EM-Algorithmus bestimmt, auf welche Art die Parameter eines HMMs verändert werden müssen, damit das veränderte HMM die zur Verfügung stehenden Trainingsdaten besser modelliert. An dieser Stelle soll der EM-Algorithmus nicht hergeleitet werden. Die Modifikationsregeln seien aber im folgenden aufgeführt.

Sei ein HMM λ gegeben. Die Trainingsdaten seien gegeben durch $x_1 \dots x_T$. Dann werden mit Hilfe des Forward-Backward-Algorithmus für alle HMM-Zustände die Einzelwahrscheinlichkeiten $c_{t,u} = p(i_t = u | x_1 \dots x_T, \lambda)$ bestimmt. Falls der Viterbi-Algorithmus für die zeitliche Zuordnung von Mustervektoren zu HMM-Zuständen verwendet wird, dann ist $c_{t,u} = \delta_{u,i_t}$, also 1 für alle Stellen auf dem Viterbi-Pfad und 0 überall sonst.

Nach dem EM-Algorithmus wird nun folgendes $\bar{\lambda}$ definiert: (Die Menge der Zustände S und das Vokabular V bleiben unverändert)

$$\begin{aligned} \bar{\pi}(s_u) &= c_{1,u} \\ \bar{a}_{uv} &= \frac{\sum_{t=2}^{T} \xi_t(u,v)}{\sum_{t=2}^{T} \sum_{w=1}^{n} \xi_t(u,w)} \\ \bar{b}_i(x) &= \sum_k \bar{\gamma}_i(k) \cdot N(x, \bar{\mu}_i(k), \bar{\Sigma}_i(k)) \end{aligned} \tag{2.19}$$

wobei

$$\begin{aligned} \bar{\gamma}_i(k) &= \frac{1}{T} \cdot \sum_{t=1}^{T} c_{t,i} \cdot h(i,k,x_t) \qquad (2.20) \\ \bar{\mu}_i(k) &= \frac{1}{T} \cdot \sum_{t=1}^{T} c_{t,i} \cdot h(i,k,x_t) \cdot x_t \\ \bar{\Sigma}_i(k) &= \frac{1}{T} \cdot \sum_{t=1}^{T} c_{t,i} \cdot h(i,k,x_t) \cdot x_t^2 - (\frac{1}{T} \cdot \sum_{t=1}^{T} c_{t,i} \cdot h(i,k,x_t) \cdot x_t)^2 \\ h(i,k,x_t) &= \frac{N(x, \bar{\mu}_i(k), \bar{\Sigma}_i(k))}{b_i(x_t)} \end{aligned}$$

Es läßt sich beweisen, daß mit den obigen Ersetzungen $p(x_1, \dots x_T | \bar{\lambda}) \geq p(x_1, \dots x_T | \lambda)$. Einen ausführlichen Beweis dazu findet man in [HAJ90].

Hier wurden die Regeln des EM-Algorithmus dafür angegeben, daß die gesamten Trainingsdaten aus einer einzige Beobachtungsfolge $x_1 \dots x_T$ bestehen. In der Praxis gibt es aber sehr viele Trainingsbeobachtungen. Dann werden die einzelnen $\lambda \rightarrow \bar{\lambda}$ Optimierungen über die verschiedenen Beobachtungen gemittelt.

2.9 Die Rolle der Suche

Für alle in dieser Arbeit vorgestellten Experimente wurde eine zeitsynchrone Suche verwendet. Der interessierte Leser sei auf [WF96] verwiesen, um mehr über die Details der JANUS-Suche zu erfahren. Hier werden nur die für diese Arbeit relevanten Aspekte angesprochen.

2.9.1 Suchfehler

Da der Suchraum, der durch alle erlaubten Wortfolgen definiert ist, nicht in vertretbarer Zeit vollständig durchsucht werden kann, ist es nötig, ihn auf geeignete Art und Weise einzuschränken. Das sogenannte „Pruning" entfernt die Teile des Suchraumes (ähnlich der Strahlsuche im Viterbi-Algorithmus, nur etwas komplizierter), deren bis zu einem gegebenen Zeitpunkt berechnete Wahrscheinlichkeit unter eine bestimmte Grenze fällt. Eine auf Null gesetzte Grenze würde in einer vollständigen Suche resultieren.

Man kann natürlich nicht ausschließen, daß ein einmal durch Pruning entfernter Teil des Suchraumes zu einem späteren Zeitpunkt eine so hohe Wahrscheinlichkeit haben würde, daß er mit anderen, noch nicht entfernten Suchraumteilen konkurrieren könnte. Da er aber nicht mehr betrachtet wird, kann Suchraumbeschneidung immer dann zu Erkennungsfehlern führen, wenn die am Ende wahrscheinlichste Hypothese irgendwann während des Suchvorganges aus dem Suchraum entfernt worden war und somit nur noch eine weniger wahrscheinliche gefunden werden kann. So kann es vorkommen, daß die am Ende gefundenen Hypothese nicht wie gewünscht die wahrscheinlichste ist. Diese Fälle nennt man Suchfehler.

In der vorliegenden Arbeit wurde das Problem der Suchfehler dadurch ausgeklammert, daß die Wahrscheinlichkeitsgrenzen für die Suchraumbeschneidung so niedrig gewählt wurden, daß die Häufigkeit von Suchfehlern auf nahezu Null gesenkt wird. Davon ist dann auszugehen, wenn trotz deutlicher Senkung der Beschneidungswahrscheinlichkeit immer noch dieselben Hypothesen gefunden werden.

2.9.2 Interaktion der Suche mit dem akustischen Modell

Die Suche verwendet das zuvor erwähnte statistische Sprachmodell. Während der Suche werden HMM-Emissionswahrscheinlichkeiten und Wortübergangswahrscheinlichkeiten miteinander kombiniert. Da in der Regel die Emissionswahrscheinlichkeiten und die Wortübergangswahrscheinlichkeiten stark voneinander abweichende Varianzen und Mittelwerte haben, führt dies dazu, daß die Suche entweder von dem akustischen Modell oder vom Sprachmodell dominiert wird, wenn diese beiden Wahrscheinlichkeiten nicht gewichtet werden, bevor sie miteinander multipliziert werden. Das heißt, daß die eigentliche Wahrscheinlichkeit einer Hypothese W nicht wie oben definiert als $P(W|X) = \frac{p(X|W) \cdot P(W)}{p(X)}$ angenommen wird, sondern vielmehr als

$$P'(W|X) = \frac{p(X|W) \cdot P(W)^z}{p(X)} \tag{2.21}$$

Der Exponent z wird Gewichtung des Sprachmodells genannt.

An dieser Stelle sei noch eine weitere Abweichung der tatsächlichen Implementierung von der reinen Theorie erwähnt. Da für die Berechnung von $P(W)$ an keiner Stelle explizit berücksichtigt wird, wie lang die Wortfolge W ist, sondern lediglich die einzelnen Wortübergangswahrscheinlichkeiten aufmultipliziert werden, verwendet man in der Suche eine sogenannte Wortübergangsstrafe q, so daß am Ende in der Implementierung des Spracherkenners gilt:

$$P'(W|X) = \frac{p(X|W) \cdot P(W)^z \cdot q^{|W|}}{p(X)} \tag{2.22}$$

Die Verwendung dieser beiden Werte z und q ist in der Spracherkennungsforschung üblich und wird auch explizit in den Regeln der (D)ARPA-Evaluationen beschrieben [PGF+95].

Kapitel 3

Parameterräume

In diesem Kapitel werden die Parameter der akustischen Modellierung im einzelnen vorgestellt und ihre Rolle erläutert. Im ersten Abschnitt wird auf die Parameter der Mixturen von multivariaten Normalverteilungen eingegangen. Danach folgt eine Diskussion der geeigneten Einheiten der Sprache, denen ein eigenes akustischen Modell zugeordnet werden kann. Da Hidden-Markov-Modelle auf unterschiedliche Arten realisiert werden können, werden verschiedene Kontinuierlichkeitsgrade aufgeführt und deren Unterschiede erläutert. Mögliche Kompaktifizierungen der Parameterräume durch Kopplung von Parametern werden danach besprochen. Schließlich wird auf die Architektur typischer Parameterräume eingegangen.

3.1 HMM-Emissionswahrscheinlichkeiten mit Mixturen von Normalverteilungen

Die mit Abstand am häufigsten vorkommende Art der Berechnung von HMM-Emissionswahrscheinlichkeiten ist die Mixtur multivariater Normalverteilungen:

$$p(x|s) = \sum_{k=1}^{n_s} \gamma_s(k) \cdot \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma_{s,k}|}} \cdot e^{-\frac{1}{2}(x-\mu_{s,k})^T \Sigma_{s,k}^{-1}(x-\mu_{s,k})} \tag{3.1}$$

Die Parameter dieser Dichtefunktion kann man in drei Gruppen einteilen: Die Mittelwertsvektoren μ_s, auch Referenzvektoren genannt, die Kovarianz-

matrizen Σ_s und die Mixturgewichte γ_s, wobei stets gilt $\sum_k \gamma_s(k) = 1$ und $\gamma_s(k) \geq 0$.

3.1.1 Mittelwertsvektoren

Die Mittelwertsvektoren sind die wichtigsten Parameter der akustischen Modellierung mit Normalverteilungen, in dem Sinne, daß die Erkennungsleistung von einer optimalen Schätzung der Referenzvektoren erfahrungsgemäß stärker abhängt als von den Kovarianzmatrizen und den Mixturgewichten. Je größer die Dimensionalität des Merkmalsraumes ist, desto stärker fällt auch der Einfluß der Referenzvektoren aus, da der durchschnittliche euklidische Abstand zweier Punkte im Einheitswürfel mit der Wurzel der Dimension wächst. Der durchschnittliche Wert eines Mixturgewichtes ist aber stets $1/n_s$, unabhängig von der Dimension.

3.1.2 Kovarianzmatrizen

Lange Zeit wurde der JANUS-Spracherkenner ganz ohne trainierte Kovarianzmatrizen betrieben, das heißt, als Kovarianzmatrizen wurde stets die Einheitsmatrix angenommen. Wenn die restlichen Parameter unter dieser Annahme trainiert werden, so steigt die Fehlerrate des Erkenners um ca. 10% bis 20% gegenüber einem Erkenner, der diagonale Kovarianzmatrizen verwendet. Auf die Verwendung von vollen Kovarianzmatrizen wurde im JANUS-Erkenner stets verzichtet, da der Zuwachs des Parameterraumes und der zusätzliche wesentliche erhöhte Rechenaufwand bei der Berechnung der Emissionswahrscheinlichkeiten in unerträgliche, nicht mehr handhabbare Größenordnungen steigt.

Erstmals wurden im JANUS-Erkenner radiale Kovarianzmatrizen erfolgreich eingesetzt (siehe Abschnitt 7.3). Diese reduzieren die benötigten Kovarianz-Parameter um den Faktor der Dimensionalität des Merkmalsraumes, ohne daß dabei die Erkennungsrate zu sehr leiden muß.

3.1.3 Mixturgewichte

Typische Werte einzelner Normalverteilungen, wie sie in dieser Arbeit verwendet wurden, liegen im Bereich zwischen 10^{-30} und 10^{-10} mit einer entsprechend großen Varianz. Gleichzeitig schwanken die Mixturgewichte nur

typischerweise zwischen 10^{-5} und 10^{0}. Im Hinblick auf diese Werte verwundert es nicht, daß die gemessenen Fehlerraten nur um wenige Prozentpunkte steigen, wenn alle Mixturgewichteverteilungen durch Gleichverteilungen ersetzt werden. Eine zufällige Initialisierung der Referenzvektoren dagegen würde den Erkenner total funktionsunfähig machen.

3.2 Geeignete Einheiten der Sprache

3.2.1 Wörter

In den Anfängen der Forschung auf dem Gebiet der Spracherkennung war das Wort die bevorzugte Spracheinheit, die akustisch modelliert wurde, weil die Erkennung von einzelnen Kommandowörtern lange Zeit die einzige realisierbare Anwendung war, bevor die Erkennung von kontinuierlicher Sprache handhabbar wurde. Auch heute noch verwenden viele Spracherkennungsprodukte wie zum Beispiel die sprachgesteuerte Bedienung von Geräten wie Autotelefone Wörter oder kurze Wortfolgen als akustisch modellierte Spracheinheiten.

Der Vorteil der akustischen Einheit „Wort" liegt darin, daß akustische Effekte, die sich über die Grenzen von Wortuntereinheiten wie z.B. Phoneme ausdehnen, automatisch mitmodelliert werden. Dadurch werden Koartikulationseffekte, die die beobachteten Muster der Wortuntereinheiten beeinflussen, quasi kontextabhängig modelliert, wobei eine Kontextbreite bis zu den Grenzen des Wortes verwendet wird. Die Problematik der Möglichkeit von verschiedenen Aussprachevarianten desselben Wortes kann in der Regel dadurch gelöst werden, daß entweder für so ein Wort verschiedene Referenzmuster abgespeichert werden, oder daß die verschiedenen Varianten wie eigenständige Wörter behandelt werden.

Die Verwendung von Wörtern als Spracheinheiten hat allerdings drei sehr gravierende Nachteile, nämlich den großen Ressourcenbedarf bei großem Vokabular, die schlechte Trainierbarkeit und die schwere bzw. kaum mögliche Vokabularmodifikation.

Der Ressourcenbedarf kann zu einem Problem werden, wenn die Zahl der zu erkennenden Wörter groß wird. In vielen flektierten Sprachen, wie zum

Beispiel im Deutschen ist die Zahl der verschiedenen Wörter in einem Text besonders hoch. Die Möglichkeit der Bildung zusammengesetzter Wörter oder selten verwendeter Neuschöpfungen wie „zugemilcht" als Bezeichnung für den Zustand einer Öffnung einer Kondensmilchdose, tun ihr Übriges zu diesem Problem. Die Frankfurter Rundschau verwendete in ihren Texten in einem Jahr ganze 650 000 verschiedene Wörter[1].

Die schlechte Trainierbarkeit ist ein weiterer Nachteil der Verwendung von akustischen Worteinheiten. Um ein Ganzwortmodell ausreichend zuverlässig zu trainieren, bedarf es mehrerer Beispielmuster. In fast allen heute erhältlichen Trainingskorpora für Spracherkennung, sei es die Wall Street Journal Datenbasis oder die Verbmobil Aufnahmen, kommt ein sehr großer Anteil der Wörter nur ein einziges Mal vor, eignet sich also nicht zur Ganzwortmodellierung. Eine speziell durchgeführte Sammlung von Einzelwortaufnahmen mit gehäufter Verwendung seltener Wörter wäre aber sehr teuer. Bei der Wall Street Journal Datenbasis besteht der Trainingskorpus aus insgesamt 13962 verschiedenen Wörtern, von denen 1231 nur ein einziges Mal darin auftauchen. Und weniger als die Hälfte aller Wörter kommt mindestens 10 mal vor.

Als gravierendster Nachteil der Ganzwortmodellierung gilt allerdings die kaum mögliche Modifikation des zu erkennenden Vokabulars. Zwar könnte man die einen oder anderen zusammengesetzten Wörter mit den trainierten Teilwörtern modellieren, aber wann immer dies nicht möglich ist, müßten für jedes neue, untrainierte Wort ein oder mehrere zusätzliche Beispielmuster aufgenommen werden. Dies ist in sehr vielen Anwendungen nicht tolerierbar und würde den Erkenner unbrauchbar machen. Gerade im Hinblick darauf, daß in naher Zukunft jeder interessierte Benutzer einen fertigen Spracherkenner unter anderem dadurch auf seine Zwecke anpassen wird, daß er das zu erkennende Vokabular selbst festlegen kann, ist es wichtig, die Möglichkeit der Vokabularmodifikation nicht zu erschweren.

[1] Die Angabe stammt aus [MOM+96], wo verschiedene große Datenbasen mit Nachrichtentexten verglichen werden

3.2.2 Phoneme

Silben wären eine Wortuntereinheit, die die Nachteile der Ganzwortmodellierung mildern würden. Silben wären auch Spracheinheiten, deren linguistische Motivation aus dem Bereich der Akustik kommt. Silben sind Worteinheiten, die im Ganzen betont oder unbetont sein können, sie bestehen aus nur einem vokalischen Laut mit ihn optional umgebenden Konsonanten. In der Spracherkennung hat sich die Verwendung von Silben als Spracheinheiten allerdings nicht durchgesetzt. Sie stellen immer noch ein größeres Trainierbarkeitsproblem dar als Phoneme. Phoneme sind auch akustisch linguistisch motivierte Spracheinheiten. Den Vorteil der Silbenmodellierung, nämlich die Beachtung der Koartikulationseffekte aufeinanderfolgender Phoneme, kann die Phonemmodellierung dadurch wett machen, daß die einzelnen Phoneme kontextabhängig modelliert werden. Ein typischer Phonemsatz für einen Erkenner der englischen Sprache besteht aus ca. 50 Phonemen. Diese Zahl gilt auch in etwa für die meisten anderen Sprachen.

Phoneme sind kleine Einheiten, so daß in der Regel sehr viele Trainingsdaten pro Phonem vorhanden sind. Die Trainierbarkeit stellt also hier kein Problem dar. Der Ressourcenbedarf ist durch die Menge der Phoneme beschränkt und wächst nicht mit der Größe des zu erkennenden Vokabulars. Der größte Vorteil der Phonemmodellierung ist allerdings die sehr leichte Erweiterbarkeit des Vokabulars. Wann immer ein neues Wort ins Vokabular aufgenommen werden soll, genügt es aus der Sicht der akustischen Modellierung, dieses Wort in das Aussprachelexikon aufzunehmen. Dort vermerkt man die bevorzugte Phonemsequenz, mit der das neue Wort ausgesprochen wird. Gegebenenfalls fügt man auch mehrere Aussprachevarianten ein.

3.2.3 Subphoneme

An verschiedenen Stellen des Spracherkenners können verschiedene Spracheinheiten verwendet werden. Wie oben erwähnt bieten sich wegen der leichten Erweiterbarkeit des Vokabulars Phoneme als Einheit im Aussprachelexikon an. Phoneme als Einheiten wurden aber ursprünglich von Linguisten eingeführt. In ihren Anfängen orientierte sich die Phonetik nach der menschlichen Perzeption. Manche akustischen Aspekte, die in der Perzeption nicht besonders prominent sind, wurden auch nicht besonders beachtet. Dazu gehört auch die Tatsache, daß der Klang vieler Phoneme

zeitlich nicht invariant ist. Manche Laute, wie zum Beispiel Vokalisierungen zwischen aufeinanderfolgenden Konsonanten oder Effekte, die durch die mechanischen Funktionen des Artikulationsapparates erzeugt werden, wie zum Beispiel Verschlußlaute, werden meist nicht einmal als Phoneme betrachtet. Der typische Klang des Anfangs eines Phonems unterscheidet sich meist von dem am Ende des Phonems. Meist hängt dieser zusätzlich noch von den angrenzenden Phonemen ab. Da der Artikulationsapparat des Menschen keine beliebig schnellen Bewegungen vollführen kann, entstehen zwangsläufig Koartikulationseffekte. Während der Schluß eines Phonems gesprochen wird, muß der Koartikulationsapparat schon die Aussprache des folgenden Phonems vorbereiten, damit die Sprache flüssig bleibt. Heute verwendet keiner der Weltklasse-Erkenner tatsächlich komplette Phoneme als akustisch atomare Einheiten. Vielmehr werden Subphoneme, in der Regel drei pro Phonem, verwendet, die dann auch noch nach Kontexten unterschieden und in Kontextklassen zusammengeballt werden.

3.2.4 Kontextabhängige Einheiten

Die oben erwähnten Spracheinheiten lassen sich alle auch durch ihren aktuellen Kontext feiner unterteilen. Wenn Phoneme in einem Kontext betrachtet werden, bei dem die beiden unmittelbaren Nachbarphoneme mit einbezogen werden, dann spricht man von Triphonen. Man beachte, daß ein Triphon nicht ein Phonem-Tripel ist, das aus drei Phonemen besteht, sondern ein einzelnes Phonem, indiziert durch zwei andere Phoneme. Wenn also Triphone als Spracheinheiten verwendet werden, so sind diese in ihrer zeitlichen Ausdehnung genauso groß wie kontextunabhängige Phoneme. Ein Satz aus m Phonemen besteht auch aus m Triphonen. Wenn die verwendete Phonemmenge aus n Phonemen besteht, dann sind insgesamt n^3 verschiedene Triphone denkbar. Von diesen kommen aber nur eine kleine Untermenge tatsächlich vor, weil viele Phonemfolgen niemals gesprochen werden (zum Beispiel **T K T**). Wenn zur Indizierung eines Phonems nicht nur die unmittelbaren Nachbarphoneme, sondern zwei in jede Richtung angrenzenden Phoneme verwendet werden, so spricht man von Quintphonen. Der Begriff Polyphon wird verwendet, um ein Phonem in einem Kontext unbestimmter Breite zu bezeichnen.

Denkbar ist natürlich auch die kontextabhängige Modellierung von

Silben, allerdings stellt sich hier schnell die Frage nach dem Sinn. Die Anzahl der möglichen Silben übersteigt bei weitem die Zahl der Phoneme, was schnell dazu führt, daß eine große Menge kontextabhängiger Silben zu modellieren wäre, in der die meisten Elemente nur sehr selten beobachtet werden. Außerdem ist davon auszugehen, daß insbesondere bei längeren Silben der Einfluß der gesamten Nachfolgesilbe auf den Anfang vernachlässigbar gering ist.

Eine kontextabhängige Modellierung von Subphonemen ist in den meisten heutigen Spracherkennern standardmäßig vorhanden. Allerdings werden einzelne Subphoneme nicht in Abhängigkeit von den angrenzenden Subphonemen, sondern in Abhängigkeit von den an das Gesamtphonem angrenzenden Phonemen modelliert. Ein akustisches Modell ist dann z.B. nicht der Mittelteil des Phonems **A**, dem der Anfangsteil des Phonems **A** vorangeht und das Endteil des Phonems **A** folgt, sondern z.B. das Mittelteil des Phonems **A**, wobei dem **A** ein **B** vorangeht und ein **D** folgt.

Wenn die Kontextbreite nur die unmittelbar angrenzenden Phoneme mit einbezieht, spricht man von Subtriphonen. Geballte Subtriphone wurden als erstes von Hwang [HH91] erfolgreich eingesetzt und werden dort „Senones" genannt.

3.3 Kontinuierlichkeitsgrade

Hidden-Markov Modelle werden danach unterschieden, ob ihr Merkmalsraum V diskret oder kontinuierlich ist. Bei diskreten Merkmalsräumen besteht die Berechnung der Emissionswahrscheinlichkeiten lediglich aus dem Nachschauen eines tabellarischen Eintrags, der vom Index des beobachteten Symbols und des akustischen Modells abhängt. Bei kontinuierlichen Merkmalsräumen fließt in die Berechnung auch die Distanz der Beobachtung zu den Referenzvektoren ein.

3.3.1 Diskrete HMMs

Sprachsignale sind von Natur aus analog. Sie werden zwar von einem Analog-Digital Wandler digitalisiert und selbst die Vektoren, die nach der Vorverarbeitung des digitalisierten Signals entstehen, lassen sich quanti-

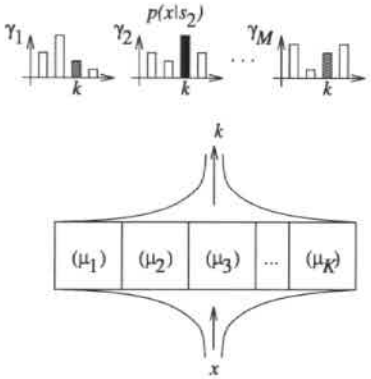


Abbildung 3.1: Diskretes HMM

sieren und so auf diskrete Indizes abbilden, aber für die Erkennung von kontinuierlicher Sprache mit großen Vokabularen haben sich kontinuierliche Merkmalsräume durchgesetzt. Allenfalls bei der Erkennung isolierter Einzelwörter aus einem kleinen Vokabular reicht die Auflösung eines diskreten Merkmalsraumes aus. Außer dem Algorithmus für die Quantisierung besteht der Parameterraum eines diskreten HMMs nur aus den diskreten Wahrscheinlichkeitsverteilungen. Die Emissionswahrscheinlichkeiten sind demnach auch echte Wahrscheinlichkeiten und keine Dichten. Diskrete HMMs sind zwar sehr schnell, da die Berechnung der Emissionswahrscheinlichkeiten sehr einfach ist, aber da ihre Modellierungsqualität sehr schlecht ist, werden sie in der kontinuierlichen Spracherkennung nicht verwendet.

Die Abbildung 3.1 zeigt die Berechnung der Emissionswahrscheinlichkeit in einem diskreten HMM. Das Codebuch dient lediglich dazu, aus einem kontinuierlichen Merkmalsvektor x einen diskreten Index k zu berechnen. Jedes akustische Einzelmodell i hat eine diskrete Wahrscheinlichkeitsverteilung γ_i über der Menge der Codebuchvektorindizes.

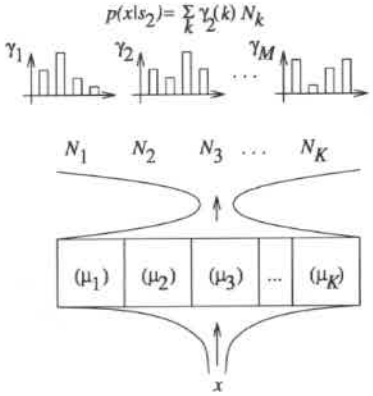


Abbildung 3.2: Semikontinuierliches HMM

3.3.2 Semikontinuierliche HMMs

Semikontinuierliche HMMs unterscheiden sich von diskreten HMMs dadurch, daß aus dem Quantisierungsprozeß die Distanzen zu den einzelnen Referenzvektoren gerettet werden. Das heißt, statt eines einzelnen Indexes werden n reelle Werte geliefert, die dann entsprechend der zugehörigen Verteilung gewichtet aufaddiert werden. Die Verteilungen werden daher auch Mixturgewichte genannt. Die reellen Werte sind meistens Funktionswerte von Normalverteilungen, selten werden auch andere Verteilungsarten wie z.B. die Laplace-Verteilung [ADNS94] verwendet.

Die Abbildung 3.2 zeigt die Berechnung der Emissionswahrscheinlichkeit für ein semikontinuierliches HMM. Das Codebuch liefert nicht nur einen diskreten Index, sondern K Dichtewerte von K Normalverteilungen an der Stelle x. Jedes akustische Einzelmodell i hat eine diskrete Wahrscheinlichkeitsverteilung γ_i über der Menge der Codebuchvektorindizes, die als Mixturgewichte für die gewichtete Addition der Dichtewerte dient.

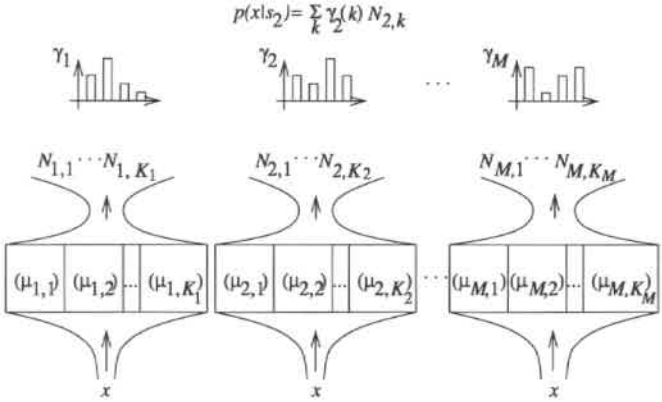


Abbildung 3.3: Voll kontinuierliches HMM

3.3.3 Kontinuierliche HMMs

Voll kontinuierliche HMMs haben den größten Parameterraum, da hier jedes einzelne akustische Modell sein eigenes Codebuch hat. Verschiedene akustische Modelle teilen sich keine Parameter der Gaußmischverteilungen. Voll kontinuierliche HMMs brauchen wegen ihres großen Parameterraumes auch viele Trainingsdaten. Weil in den letzten Jahren immer mehr sehr große Sprachdatenbasen angelegt wurden, wurde es immer leichter, voll kontinuierliche HMM-Erkenner zu trainieren, und diese haben sich in ihrer Erkennungsleistung gegenüber rein semikontinuierlichen durchgesetzt.

Die Abbildung 3.3 zeigt die Berechnung der Emissionswahrscheinlichkeit für ein voll kontinuierliches HMM. Jedes der M akustischen Einzelmodelle hat ein eigenes Codebuch, das K Dichtewerte von K Normalverteilungen an der Stelle x liefert. Jedes akustische Einzelmodell i hat auch eine eigene diskrete Wahrscheinlichkeitsverteilung als Mixturgewichte γ_i über der Menge der Indizes $1 \ldots K_i$ seines zugehörigen Codebuchs.

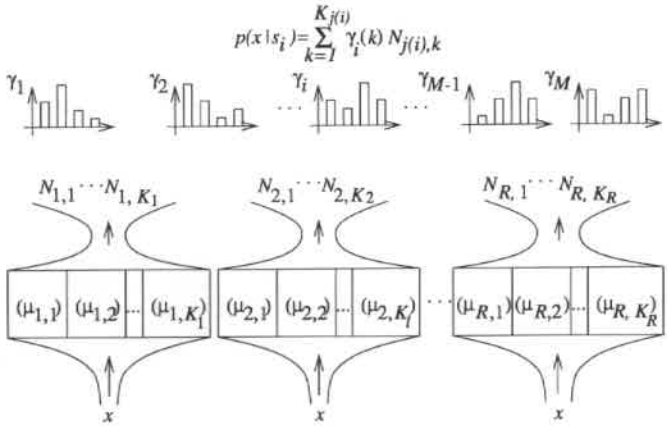


Abbildung 3.4: Beliebiger Kontinuierlichkeitsgrad

3.3.4 Feinere Kontinuierlichkeitsgrade

Zwischen den reinen semikontinuierlichen und den reinen kontinuierlichen HMMs gibt es verschiedene Feinheitsgrade. Häufig verwendet werden sogenannte phonetisch oder subphonetisch gekoppelte semikontinuierliche HMMs. Dabei gibt es ein eigenes Codebuch für jedes Phonem oder Subphonem. Alle kontextabhängigen Modelle eines (Sub)Phonems sind durch eine eigene Mixturgewichteverteilung über ihrem gemeinsamen Codebuch vertreten.

Aber auch automatisch bestimmte Codebuchkopplungen werden eingesetzt. Wenn kontextabhängige Modelle in einem Entscheidungsbaum divisiv geballt werden, dann kann zu einem Zeitpunkt während des Ballungsvorganges jedem Blatt des bis dahin aufgebauten Baumes, das heißt jeder bis dahin gefundenen Klasse, ein eigenes Codebuch zugeordnet werden. Danach wird die divisive Ballung fortgeführt. Am Ende wird jeder Kontextklasse eine Mixturgewichteverteilung zugeordnet, die über dem Codebuch definiert ist, das dem entsprechenden Vorgängerknoten zugewiesen worden war.

Abbildung 3.4 zeigt die Berechnung der Emissionswahrscheinlichkeit für ein HMM mit beliebig feinem Kontinuierlichkeitsgrad. Jedes der M akustischen Einzelmodelle hat eine eigene diskrete Wahrscheinlichkeitsverteilung γ_i über einem Codebuch $j(i)$ von $R \leq M$ Codebüchern.

3.4 Parameterkopplung

Man ist aus verschiedenen Gründen an der Kopplung von Parametern interessiert. Gekoppelte Parameter benötigen weniger Speicherplatz. Sie lassen sich robuster schätzen, und bei bestimmten Konstellationen kann durch Kopplung auch die benötigte Rechenzeit für die Berechnung der Emissionswahrscheinlichkeiten reduziert werden. Und schließlich kann die Generalisierungsfähigkeit der akustischen Modelle verbessert werden.

An dieser Stelle sei noch einmal die Formel für die Berechnung einer HMM-Emissionswahrscheinlichkeit der Beobachtung x im Zustand s_i [2] mittels voll kontinuierlicher Gauß-Mischverteilungen aufgeführt:

$$p(x|s_i) = \sum_{k=1}^{n_i} \gamma_{s_i}(k) \cdot \frac{1}{\sqrt{(2\pi)^d |\Sigma_{s_i,k}|}} \cdot e^{-\frac{1}{2}(x-\mu_{s_i,k})^T \Sigma_{s_i,k}(x-\mu_{s_i,k})} \quad (3.2)$$

Im voll kontinuierlichen Fall gilt :

$$\begin{aligned} \gamma_{s_i} &\neq \gamma_{s_j} \\ \mu_{s_i,k} &\neq \mu_{s_j,k} \quad \text{und} \\ \Sigma_{s_i,k} &\neq \Sigma_{s_j,k} \quad \forall i \neq j \end{aligned}$$

Im reinen semikontinuierlichen Fall gilt noch:

$$\begin{aligned} \gamma_{s_i} &\neq \gamma_{s_j} \quad \text{aber jetzt} \\ \mu_{s_i,k} &= \mu_{s_j,k} \quad \text{und} \\ \Sigma_{s_i,k} &= \Sigma_{s_j,k} \quad \forall i \neq j \end{aligned}$$

[2] s_i bezeichne hier das akustische Modell, mit dem ein HMM-Zustand modelliert wird. Das heißt, zwei verschiedene HMM-Zustände können durchaus mit demselben akustischen Modell modelliert werden.

Für ein HMM mit beliebig feinem Kontinuierlichkeitsgrad gilt:

$$\begin{array}{lll} & \gamma_{s_i} \neq \gamma_{s_j} & \\ \exists i,j: & \mu_{s_i,k} = \mu_{s_j,k} \quad \text{und} & \Sigma_{s_i,k} = \Sigma_{s_j,k} \\ \exists i,j: & \mu_{s_i,k} \neq \mu_{s_j,k} \quad \text{und} & \Sigma_{s_i,k} \neq \Sigma_{s_j,k} \end{array}$$

Neben den Kopplungsmethoden, die durch den Kontinuierlichkeitsgrad des verwendeten HMMs vorgegeben sind, sind auch andere Parameterkopplungen von Interesse:

3.4.1 Kopplung von Zustandsübergangswahrscheinlichkeiten

Eine korrekte Schätzung der Zustandsübergangswahrscheinlichkeiten hat sich in den letzten Jahren als eher weniger bedeutend herausgestellt. Der Hauptzweck dieser Parameter ist – neben der Definition der erlaubten Zustandsfolgen – die Modellierung der Aufenthaltsdauern in den einzelnen Zuständen. Im folgenden seien die drei wesentlichen Gründe für die geringe Bedeutung dieser Parameter aufgeführt:

Inkompatibilität zwischen Dichtewerten und Wahrscheinlichkeiten

Während die Zustandsübergangswahrscheinlichkeiten echte Wahrscheinlichkeiten sind, müssen sie während der Berechnung eines Viterbi- oder Forward-Backward-Pfades mit den Dichtewerten der Emissionswahrscheinlichkeiten multipliziert werden. Da erfahrungsgemäß die Varianz der Dichtewerte einer hochdimensionalen Normalverteilung viel größer ist als die Varianz der echten Wahrscheinlichkeiten, werden alle Entscheidungen während des Viterbi- oder Forward-Backward-Algorithmus von den Emissionswahrscheinlichkeiten dominiert. Der Einfluß der Zustandsübergangswahrscheinlichkeiten kann vernachlässigt werden.

Längenmodellierung durch die Emissionswahrscheinlichkeiten

Da fast alle Spitzenspracherkenner sehr einfache HMM-Topologien verwenden, bei denen drei bis sechs linear angeordnete Zustände verwendet werden,

wird die Aufenthaltsdauer in einem HMM-Zustand im wesentlichen von der Paßgenauigkeit der akustischen Modelle gesteuert. Ob ein Zustand verlassen und der Nachfolgezustand betreten werden sollte, kann ausreichend sicher nur mit Hilfe des akustischen Modells entschieden werden.

Explizite Längenmodellierung

Durch die Verwendung von Zustandsübergangswahrscheinlichkeiten können nur sehr eingeschränkte Wahrscheinlichkeitsverteilungen über die Aufenthaltsdauern in einzelnen Zuständen realisiert werden. Wenn die Wahrscheinlichkeit dafür, daß ein Zustand sein eigener Nachfolgezustand ist, p beträgt, dann berechnet sich die Wahrscheinlichkeit für einen Aufenthalt eines Viterbi-Pfades von t Zeiteinheiten als $p^{t-1} \cdot (1-p)$. Es handelt sich also um eine einfache Exponentialverteilung. Durch geschicktes Anordnen mehrerer Zustände, die dasselbe akustische Modell verwenden, läßt sich zwar auch jede beliebige kompliziertere Verteilung approximieren, jedoch lohnt sich dieser Aufwand nicht. Stattdessen bietet sich eher an, eine explizite Längenmodellierung einzusetzen. Dabei wird für jeden HMM-Zustand, bzw. für jede Äquivalenzklasse von Zuständen oder Zustandsfolgen, eine diskrete Wahrscheinlichkeitsverteilung von Aufenthaltslängen geschätzt. Nach Verlassen eines Zustandes oder einer Zustandsfolge wird dann die bis dahin akkumulierte Pfadwahrscheinlichkeit mit der Längenwahrscheinlichkeit verrechnet.

3.4.2 Einschränkungen der Parameterkopplung

In der Regel wird man nicht erlauben, daß jeder Parameter mit jedem anderen seiner Sorte gekoppelt werden kann. Das ist zum einen der Fall, weil die Parameter trotz Gleichartigkeit völlig verschiedene Bereiche des Parameterraumes belegen. So wird man eher davon absehen, die Kovarianzmatrizen aus einem Codebuch des Phonems **G** mit den Kovarianzmatrizen des Phonems **S** zu koppeln. Man wird also unter Einsatz von Wissen bestimmte Arten der Kopplung ausschließen. Ein weiterer Grund für die Einschränkung der möglichen Kopplungen ist die Vermeidung einer kombinatorischen Explosion. Es ist zum Beispiel mit der Leistung der heutigen Workstations ausgeschlossen, alle möglichen Kopplungen von 700 000 Subpolyphonen – ein nicht untypischer Wert – zu erlauben. Schon zu Beginn müßten ca. 245 Milliarden Distanzen berechnet werden.

Typische Einschränkungen für die Parameterkopplung werden so vorgenommen, daß keine Kopplungen über Phoneme hinweg erlaubt sind. Das heißt, es können keine Parameter eines Phonems mit denen eines anderen Phonems gekoppelt werden. Bei der Verwendung von subphonetischen Spracheinheiten, vermeidet man meist sogar die Kopplung von Parametern desselben Phonems, wenn sie zu verschiedenen Untereinheiten gehören.

3.4.3 Arten der Parameterkopplung

Kopplung durch A-priori-Wissen

Verschiedene gleichartige Parameter können auf verschiedene Arten gekoppelt werden. Ebenso wie Einschränkungen der möglichen Kopplungen durch A-priori-Wissen vorgenommen werden, werden auch einfach wissensbasiert bestimmte Parameter gekoppelt, ohne daß eine mathematische Untersuchung der Sinnhaftigkeit durchgeführt wird. So verwendet man keine kontextabhängigen akustischen Modelle für Stille oder für bestimmte Geräusche. Man nimmt einfach a priori an, daß die akustische Erscheinung der Stille nicht von den davor oder danach gesprochenen Worten abhängt. Als weiteres Beispiel des Einsatzes von A-priori-Wissen kann man die Zustandsübergangswahrscheinlichkeiten ansehen. Im JANUS-Erkenner werden diese für alle artikulatorischen akustischen Modelle einfach einmalig festgesetzt und dann nicht mehr verändert.

Kopplung über den Kontext

In die Entscheidung, ob zwei akustische Modelle gekoppelt werden sollen, kann auch Kontextinformation einfließen. Zwei Modelle können sich ihre gesamten akustischen Parameter teilen, wenn sie das gleiche Phonem oder Subphonem in verschiedenen aber bezüglich eines Maßes ähnlichen Kontexten modellieren. So bietet es sich an, Phoneme, deren zwei oder drei rechte und linke Nachbarn gleich sind, aber der weiter entfernte Kontext sich unterscheidet, gemeinsam zu modellieren, in der Erwartung, daß durch die verschieden weiten Kontexte der Unterschied der Akustik unbedeutend ist.

Datengetriebene Kopplung

Der statistisch korrekteste Weg ist die datengetriebene Kopplung. Dabei wird ein Distanzmaß zwischen verschiedenen Parametern oder Parametermengen definiert. Diejenigen Parametermengen, die eine sehr kleine Distanz haben, werden gemeinsam modelliert. Wenn die Parametermengen einzelnen akustischen Modellen entsprechen, spricht man von einer Ballung dieser Modelle. Modellballung wird in fast allen leistungsfähigen Spracherkennern eingesetzt. In der Regel handelt es sich dabei um eine hierarchische Ballung, meist unter Verwendung von Entscheidungsbäumen.

3.5 Architekturentwurf

Bei der Entwicklung eines HMM-basierten Spracherkenners müssen mehrere Entwurfsvariablen gefunden und optimiert werden. Entwurfsvariablen, die den Parameterraum der akustischen Modellierung betreffen, sind zum Beispiel die Anzahl der akustischen Modelle, die Anzahl der Mixturgewichteverteilungen und der Codebücher, der Kopplungsgrad der Kovarianzmatrizen, die Breite der modellierten Kontexte, die Anzahl der Zustände eines HMMs für die verwendeten Spracheinheiten, sowie deren Topologien. Weitere einzustellende Variablen sind die Größe der Codebücher, das heißt die Anzahl der Referenzvektoren je Codebuch und die Dimensionalität des Merkmalsraumes.

Auch auf dem Gebiet der temporalen Modellierung der HMMs gilt es Entwurfsentscheidungen zu treffen und Variablen einzustellen, wie zum Beispiel die Feinheit und den Kopplungsgrad der Übergangswahrscheinlichkeiten und eventuell vorhandene Parameter für explizite Längenmodellierung.

Oft ist es sehr aufwendig und ressourcenintensiv, verschiedene mögliche sinnvolle Werte für die Entwurfsvariablen durch Experimente auszuwerten und auf diese Art zu optimieren. Teilweise wird in manchen Bereichen der Spracherkennungsforschung sogar von einer Art schwarzen Magie gesprochen, wenn Forscher sogenannte „educated guesses“ (auf Erfahrung basierte Schätzungen) vornehmen. Nicht selten kann man beobachten, daß eine nur sehr wenig vom Optimum abweichende Einstellung bestimmter Architekturvariablen die Fehlerrate des Erkenners enorm erhöht.

Auf Erfahrung basierende Schätzungen sind auch bei den zahlreichen sogenannten „fudge factors“[3] üblich. Dabei handelt es sich um der Theorie eigentlich widersprechende Parameter, wie z.B. die Werte z und q in Gleichung 2.22. Solche Parameter werden aus verschiedenen Gründen für eine zufriedenstellende Funktion der Spracherkenner benötigt. Zum einen ist die getroffene Modellannahme (HMM erster Ordnung, Gauß-Mischverteilungen) nicht in völliger Übereinstimmung mit der Natur des Sprachproblems, zum anderen liegen in der Regel nicht so viele Trainingsdaten vor, daß das in der Statistik oft zitierte Gesetz der großen Zahl wirkt. Im Gegenteil, meistens wird der Parameterraum der Spracherkenner so weit aufgeblasen wie nur irgendwie trainierbar. Man tendiert eher dazu, ein paar schlecht geschätzte Parameter zuviel zu verwenden, und notfalls Glättungsmethoden einzusetzen, als zu wenige Parameter zu verwenden, die dann aber gut geschätzt werden, denn fehlende Information kann nicht ersetzt werden, während zuviel Information bei geeigneter Handhabung nicht schädlich sein muß. Schließlich ist ein weiterer Grund für die Verwendung von „fudge factors“ die oft inkorrekte Implementierung der mathematischen Grundlagen, bei der, um Rechenzeit zu sparen, Approximationen vorgenommen werden.

Das Ziel der Spracherkennungsforschung ist es, nicht nur fertige, funktionierende Produkte zu entwickeln, sondern auch Laien zu ermöglichen, einen für ihre speziellen Zwecke optimierten Spracherkenner selbst zu bauen, ohne dabei von ihnen zu verlangen, sich mit den verschiedenen Entwurfsvariablen auszukennen. Solange nicht abzusehen ist, daß fertige Spracherkennungsprodukte in jeder Umgebung, mit beliebigen Vokabularen, mit verschiedenen Sprachen und Dialekten sowie für viele verschiedene Anwendungsszenarien einwandfrei funktionieren, solange wird es interessant bleiben, daß Entwicklungsumgebungen existieren, mit denen auch Personen, die nicht auf dem Gebiet der Spracherkennung arbeiten, funktionierende Spracherkenner bauen können.

Für die Umsetzung eines Forschungsprototypen in ein fertiges Produkt ist es deshalb sinnvoll, Methoden zu finden, die es ermöglichen, Entwurfsvariablen automatisch zu optimieren.

[3] engl. fudge: schmierige Schokoladencreme

Während für die Suche der besten Parameter eines gegebenen parametrischen Modells etablierte und gut verstandene Algorithmen (z.B. Maximum Likelihood) existieren, gibt es keine Möglichkeit, uniform zu berechnen, welches parametrische Modell das beste ist. Man beschränkt sich daher darauf, zwischen zwei (oder mehr) verschiedenen Modellen dasjenige auszuwählen, das gemäß eines festgelegten Kriteriums das bessere ist. Für die Einstellung der Architekturvariablen bedeutet das, daß zum Beispiel an einem gegebenen System unter verschiedenen möglichen Modifikationen diejenige tatsächlich vorzunehmen ist, der die größte Gewinnerwartung zugeordnet wird.

Als Kriterien zur Bewertung einer Architekturmodifikation dienen in der Regel Maße, die basierend auf der Gesamtwahrscheinlichkeit der Trainingsdaten diejenige Architektur bevorzugen, für die die Gesamtwahrscheinlichkeit aller Trainingsdaten größer ist. Im folgenden werden solche Verfahren als „likelihood-basierte" Architekturoptimierungen bezeichnet. Andere Bewertungskriterien sind zum Beispiel solche, die andere prinzipiell wünschenswerte Eigenschaften berücksichtigen, wie den Informationsgewinn (das heißt den Entropieverlust), der durch eine Architekturmodifikation entsteht.

Bei allen likelihood-basierten Optimierungsverfahren besteht die Gefahr der Überanpassung, wenn versucht wird, die Gesamtwahrscheinlichkeit aller Trainingsdaten zu optimieren, weil dabei nicht berücksichtigt wird, wie sich die Architekturmodifikation auf andere, nicht zur Trainingsmenge gehörende Daten auswirkt.

Das Problem der Überanpassung ist auch bekannt aus der Theorie der künstlichen neuronalen Netze. Dort beobachtet man den Effekt, wenn ein neuronales Netz zu viele Epochen auf den Trainingsdaten trainiert wird. Wenn die Zahl der Parameter des Netzes ausreicht, so fängt es irgendwann an, die Trainingsdaten „auswendig" zu lernen, statt zu lernen, was deren Struktur ist. Für ungesehene Testdaten wird das im Netz gespeicherte Weltmodell ab diesem Zeitpunkt immer schlechter. Das Netz verliert an Generalisierungsfähigkeit. Das Problem wird dadurch gelöst, daß mit Hilfe einer Kreuzvalidierungsmenge der Zeitpunkt bestimmt wird, ab dem ein Leistungsabfall auf den Kreuzvalidierungsdaten eintritt. Diese Menge ist in der Regel eine Teilmenge der zur Verfügung stehenden Trainingsdaten,

die, nachdem sie bestimmt ist, nicht mehr zum Trainieren verwendet wird. Sobald der Effekt der Überanpassung einsetzt, wird der Trainingsprozeß abgebrochen.

Die Idee der Verwendung von Kreuzvalidierungsmengen läßt sich auch auf likelihood-basierte Optimierungsverfahren anwenden, sowohl um zu entscheiden, ob ein iterativer Optimierungsprozeß fortgeführt werden soll, als auch um zu bewerten, wie gut eine Architektur- bzw. Parametermodifikation ist. Die Bewertung einer Modifikation wird so vorgenommen, daß die Gesamtwahrscheinlichkeit der Kreuzvalidierungsmenge unter der Bedingung des modifizierten Systems berechnet wird. Wenn nun mehrere mögliche Systemmodifikationen antizipiert werden, dann sollte diejenige ausgeführt werden, die den größten Wahrscheinlichkeitsgewinn verspricht. Bei einem iterativen Optimierungsverfahren sollte das Verfahren so lange fortgesetzt werden, bis auf der Kreuzvalidierungsmenge kein Gewinn mehr festzustellen ist. Man sieht leicht ein, daß ohne eine Kreuzvalidierungsmenge ein Optimierungsverfahren, das die Zahl der Parameter mit jedem Schritt erhöht, auch die Gesamtwahrscheinlichkeit der Trainingsdaten beliebig weit erhöhen kann, im Extremfall so weit, daß jedes einzelne Trainingsdatum seinen eigenen Parameter im Parameterraum hat.

Typische Vorgehensweisen beim Architekturentwurf werden selten veröffentlicht. Bei den obligatorischen Systembeschreibungen der Spracherkenner, die an den offiziellen internationalen (D)ARPA Evaluationen teilnehmen, werden die Parameterräume zwar beschrieben, aber man findet kaum Informationen darüber, warum diese Architektur ausgewählt wurde. In der Praxis sieht zum Beispiel die Optimierung der Zahl der akustischen Modelle so aus, daß für eine Reihe verschiedener Werte ein Erkenner trainiert und danach evaluiert wird. Der am besten arbeitende Erkenner wird schließlich zum System der Wahl. Oft läßt sich dieses Vorgehen allerdings automatisieren, so daß die Hauptlast vor allem durch den Rechenzeitbedarf bestimmt wird.

Kapitel 4

Der Wall Street Journal Vergleichstest

4.1 Die Tradition der DARPA-Evaluationen

Seit Herbst 1992 führt die Defense Advanced Research Projects Agency, DARPA[1], der US-Regierung Evaluationen von Spracherkennern durch. Der Zweck dieser Evaluationen ist es, zum einen eine Vergleichbarkeit verschiedener Systeme zu ermöglichen, und zum anderen den Fortschritt durch gegenseitigen Austausch von Erfahrungen auf vergleichbaren standardisierten Tests zu beschleunigen. Bevor allgemeine, genormte Testumgebungen in öffentlichen Evaluationen verwendet wurden, war es in der Spracherkennung üblich, daß die einzelnen Forschungseinrichtungen jeweils ihre eigenen Testumgebungen und -bedingungen verwendeten. Die verschiedenen Veröffentlichungen waren so nur sehr schwer vergleichbar, und nützliche Neuerungen waren nur schwer einzuordnen. Die DARPA Tests sind heute zum internationalen de facto Standard geworden.

Ende der achtziger Jahre wurde deshalb die sogenannte „Resource Management Task" als erster offizieller Vergleichstest für sprecherunabhängige kontinuierliche Spracherkennung eingeführt. Der Test wurde mit einem fest vorgegebenen Vokabular von ca. 1000 Wörtern durchgeführt.

[1]Der Zusatz „Defense" war für eine kurze Zeit während der Jahre 1993 bis 1995 weggelassen worden. In dieser Zeit hieß die Agentur nur „ARPA"

Dabei wurde eine Wortpaargrammatik mit einer Perplexität[2] [Jel90] von 60 benutzt. Das Szenario der Resource Management Task war das eines Marine-Offiziers, der an einem Monitor sitzend mit Hilfe kontinuierlicher Sprache die Zustände und Variablen seiner Flotte erfragen und steuern konnte. Die zu sprechenden Sätze mußten aber einer vorgegebenen, durch einen endlichen Automaten definierten Grammatik genügen. Die Fehlerraten der meisten Evaluationsteilnehmer lagen schließlich im Bereich um 5%.

Die stark eingeschränkte Grammatik und das sehr kleine Vokabular machten die Resource Management Task ungeeignet für weitere Forschungen, so daß Anfang der neunziger Jahre zwei andere Vergleichstests eingeführt wurden, die Air Travel Information System Task, ATIS, und die Wall Street Journal Datenbasis, WSJ. Der ATIS-Test war die erste offizielle Evaluation für spontane Sprache. Während er neben der reinen Spracherkennung vor allem für die Evaluation von Zerteilern für natürliche Sprache gedacht war, war der Zweck des Wall Street Journal Tests die Erkennung sprecherunabhängiger Sprache mit sehr großen Vokabularen. Die Perplexität des Tests war je nach Testbedingungen mit 120 bis 160 wesentlich höher als bei der Resource Management Task.

Seit 1996 wird der Wall Street Journal Vergleichstest durch spontane Erkennungsaufgaben abgelöst. Für Diktiersysteme bleibt er jedoch weiterhin der Standard, an dem die Erkennerqualität gemessen wird.

4.2 Testbedingungen

Die letzte offizielle Evaluation von Diktiersystemen mit unverrauschten Aufnahmen fand im November 1994 statt. Im November 1995 fand noch eine Evaluation von Diktiersystemen statt, deren Ziel allerdings die Auswertung verschiedener Methoden zur Kompensation von Signalstörungen und Variationen durch verschiedene Mikrophone war. Danach wurden keine offiziellen Evaluationen für Diktiersysteme mehr durchgeführt. Die folgende Beschreibung der Testbedingungen bezieht sich also auf die Evaluation vom

[2]Die Perplexität ist ein Maß für die Schwierigkeit einer Testmenge. Dabei gilt grob: je höher die Perplexität, umso schwieriger die Erkennung.

November 1994.

Die Testmenge besteht aus 316 Äußerungen von 20 verschiedenen Sprechern, von denen zehn weiblich und zehn männlich waren. Die Gesamtdauer der Testmenge betrug ca. 53 Minuten reine Sprache. In dieser Zeit wurden 8186 Wörter gesprochen. Die Aufnahmen wurden alle in einem gewöhnlichen Bürozimmer mit einem sehr guten Nahbesprechungsmikrophon gemacht.

Die Evaluation wurde unter zwei verschiedenen Testbedingungen durchgeführt, der sogenannten **P0**- und der **C1**-Bedingung. Bei der **C1**-Bedingung waren die Trainingsdaten, das Erkennungsvokabular (20 000 Wörter) und das zu verwendende Sprachmodell genau vorgegeben. Sie sollte dazu dienen, aussagekräftige Vergleiche zwischen verschiedenen Arten der akustischen Modellierung machen zu können. Bei der **P0**-Bedingung war die Limitierung des Trainingsmaterials und die Gestaltung des Sprachmodells fallengelassen. Die Größe des Erkennungsvokabulars wurde auf 60 000 Wörter angehoben. Zusätzlich war es unter der **P0**-Bedingung erlaubt, die Kenntnis von Anfängen und Enden der einzelnen Zeitungsartikel zu verwenden, womit eine inkrementelle Adaption der Erkenner ermöglicht wurde. Das heißt, man durfte für die Erkennung eines Satzes alle Information aus den vom selben Sprecher stammenden früheren Sätzen verwenden. Die einzigen vorgegebenen Einschränkungen für die **P0**-Bedingung waren, daß alle Daten, die zum Trainieren des akustischen Modells und des Sprachmodells verwendet werden, zeitlich vor einem gegebenen Stichtag gesammelt worden und jedem Evaluationsteilnehmer zugänglich sein müssen. Der Stichtag wurde so gewählt, daß alle Aufnahmen der Evaluationsmenge zeitlich später lagen, so daß ausgeschlossen werden konnte, daß irgendwelche Informationen aus den Evaluationsdaten im Training der Erkenner verwendet werden konnten.

Die für die **C1**-Bedingung vorgegebenen Trainingsdaten bestanden aus den Wall Street Journal Korpora WSJ0 und WSJ1, die zusammen ca. 83 Stunden reine Sprache enthalten. Das zu verwendende Sprachmodell wurde von der Carnegie Mellon Universität aus einem 237 Millionen Worte umfassenden Textkorpus berechnet.

Neben den Evaluationen auf sauberer Sprache fanden auch Evaluationen auf Telefondaten statt und auf Aufnahmen, die mit Mikrophonen mit

schlechterer Qualität durchgeführt worden waren. Des weiteren wurden auch einzelne Aspekte der Spracherkennung wie die Adaption des Sprachmodells und die Optimierung der Erkennungsleistung auf der Sprache von Nichtmuttersprachlern evaluiert.

Ausdrücklich erlaubt war die Verwendung eines einzigen skalaren Parameters zur Gewichtung der Sprachmodellwahrscheinlichkeit mit den HMM-Emissionswahrscheinlichkeiten sowie die Verwendung eines skalaren Parameters zur Bewertung der Länge der erkannten Hypothese. Diese beiden Werte durften einmalig anhand einer Kreuzvalidierungsmenge optimiert werden und blieben dann im eigentlichen Test unverändert. Dies entspricht der Verwendung der Parameter z und q aus Gleichung 2.22.

4.3 Die Ergebnisse

Der JANUS-Erkenner erzielte bei der Evaluation vom November 1994 eine Fehlerrate von 22.8%. Die Fehlerrate des beste Teilnehmers war 7.2%.

Abbildung 4.1 zeigt die Fehlerraten der teilnehmenden Erkenner für 20 verschiedene Sprecher. Die verschiedenen Kurven ensprechen verschiedenen Sprechern. Man sieht, daß es einzelne Sprecher gibt, die allen Erkennern große Schwierigkeiten bereiten (im Jargon „goats" genannt), und andere Sprecher produzieren bei allen Erkennern durchweg gute Resultate („sheep").

Ein weiterer, die Erkennungsrate deutlich beeinflussender Faktor ist die Sprechgeschwindigkeit. Abbildung 4.2 zeigt die Fehlerraten von vier Erkennern in Abhängigkeit von der Sprechgeschwindigkeit. Die am deutlichsten schlechteste Erkennung erzielen alle Erkenner auf dem am schnellsten sprechenden Sprecher.

4.4 Andere Evaluationen

Der in der vorliegenden Arbeit verwendete Erkenner nahm auch bei anderen Evaluationen teil. Die Switchboard-Evaluationen werden auch von DARPA

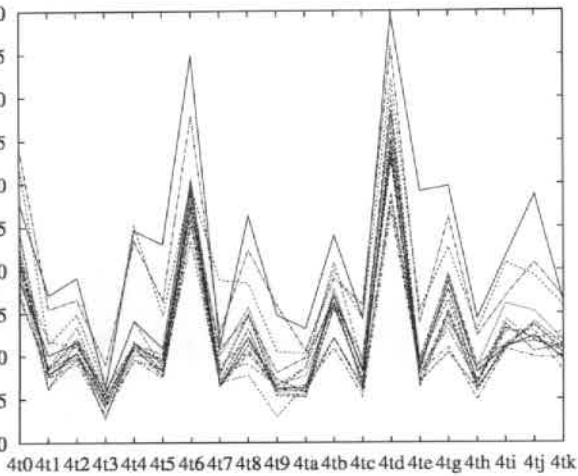


Abbildung 4.1: Fehlerrate (%) nach Sprechern[3] bei der Evaluation im November 1994

veranstaltet. Die alljährliche VERBMOBIL-Evaluation wird im Rahmen des vom deutschen Forschungsministerium geförderten VERBMOBIL Projektes durchgeführt. Bei beiden Testmengen handelt es sich um Aufnahmen spontaner Sprache. Bei Switchboard wurden die Aufnahmen über Telefonleitungen gemacht.

VERBMOBIL

Der JANUS-Spracherkenner nahm an allen drei VERBMOBIL-Evaluationen der Jahre 1994 bis 1996 teil und schnitt stets als bester aller Teilnehmer ab. Bei der Evaluation im September 1996 lag die Fehlerrate des JANUS-Erkenners bei weniger als 13% Fehlern, während der zweitbeste Erkenner über 19% Fehler machte [FGH+97].

[3] Der Sprecher 4tf wurde von der ARPA nicht in die Testmenge mit aufgenommen

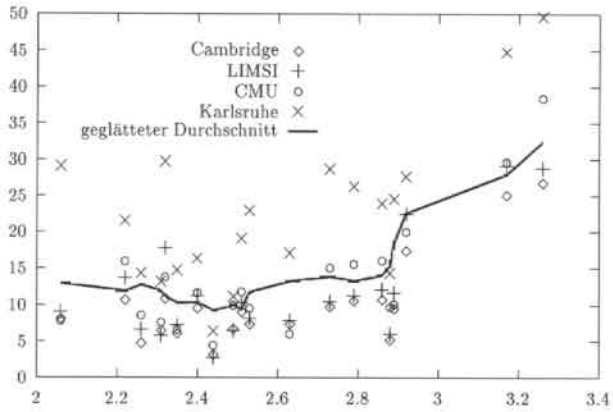


Abbildung 4.2: Fehlerrate (%) nach Sprechgeschwindigkeit (Worte/s) auf der Testmenge vom November 1994

Switchboard

Bei der Switchboard-Evaluation im April 1996 lag der JANUS-Erkenner an zweiter Stelle, und machte dabei mit 38% Fehlern nur statistisch insignifikant mehr Fehler als der beste Teilnehmer [FZ96] [Fin96]. Bei der Evaluation 1997 war die Fehlerrate der JANUS-Erkenners auf den Switchboard Daten mit 35% sogar die beste [ZFR+97].

Bei den erwähnten VERBMOBIL- und Switchboard-Evaluationen nahm die Cambridge University, die auf der Wall Street Journal Evaluation die besten Resultate hatte, nicht teil. Die Universität Hamburg, die am zweitbesten abschnitt, verwendete aber den HTK (Hidden-Markov Toolkit) Erkenner, mit dem die Cambridge University ihre DARPA-Evaluationen bestritt.

Kapitel 5

Der JANUS-Spracherkenner

In diesem Kapitel soll kurz der JANUS-Spracherkenner vorgestellt werden, der als Werkzeug und Entwicklungsumgebung für die hier vorgestellten Experimente verwendet wurde. Der Autor selbst war am Entwurf und der Entwicklung der meisten Teile des JANUS-Programms beteiligt.

5.1 Entwicklung des JANUS-Erkenners

Der hier verwendete JANUS-Erkenner wurde im internen Sprachgebrauch als JANUS 3 bezeichnet. Die Vorgängerversion (JANUS 2) wurde bei der Teilnahme an der ARPA-Evaluation vom November 1994 eingesetzt. Obwohl auch schon JANUS 2 durch beste Ergebnisse bei den Verbmobil-Evaluationen von 1994 und 1995 seine Leistungsfähigkeit bewies, schnitt er bei der ARPA-Evaluation mit dem schlechtesten von 15 Ergebnissen ab. Von Mitte 1995 bis Mitte 1996 wurde daraufhin ein vollständig neues Programmpaket entwickelt. Es wurde in eine Tcl/Tk-Umgebung [Wel95] eingebettet, die es ermöglichte, ohne Zugriff auf den Quellcode sowohl auf die elementaren Variablen und Parameter des Erkenners zuzugreifen als auch mächtige Funktionen und Operationen in einer Hochsprache zu programmieren. Dadurch erhielt JANUS 3 ein Maximum an Flexibilität und ist sehr einfach zu erweitern.

Einige der neu hinzugekommenen Eigenschaften waren die folgenden:

- Gegenüber JANUS 2 wurde die Möglichkeit des Einsatzes von Polyphonen (kontextabhängige Modellierung mit beliebig breitem Kontext) eingebaut.

- Phoneme können in JANUS 3 mit Markierungen versehen werden, die z.B. die Position des Phonems innerhalb eines Wortes oder den Grad der Betonung spezifizieren. Diese Markierungen können dann in den Entscheidungsbäumen abgefragt und als Basis für Entscheidungen verwendet werden.

- JANUS 3 erlaubt explizite Längenmodellierung von Polyphonen. Jedem Polyphon oder jeder Klasse von Polyphonen kann eine diskrete Wahrscheinlichkeitsverteilung über mögliche Längen zugeordnet und trainiert werden.

- Der abstrakte Datentyp der Entscheidungsbäume kann in JANUS 3 für verschiedene Objekte verwendet werden. So können nicht nur die akustischen Elementarmodelle, egal ob Gauß-Mischverteilungen oder mehrschichtige Perzeptronen, geballt werden, sondern auch die expliziten Längenmodelle und die HMM-Topologien der Polyphone.

5.2 Verwendung von JANUS

Um mit JANUS einen Spracherkenner zu trainieren, benötigt man eine Trainingsdatenbasis, bestehend aus digitalisierten Aufnahmen und einer möglichst genauen Transkription des Gesprochenen. Außerdem wird ein Aussprachelexikon gebraucht, das zumindest alle Wörter der Trainingsdaten sowie alle später zu erkennenden Wörter beinhaltet.

Wenn keine expliziten Entwicklungs- und Testaufnahmen vorgegeben sind, werden diese von den Trainingsdaten abgezweigt. Mit den übrigen Trainingsdaten werden in mehreren Phasen, zwischen denen verschiedene Entwurfsentscheidungen getroffen werden müssen, die Parameter der akustischen Modellierung geschätzt.

Für die Erkennung ist zusätzlich ein Sprachmodell sinnvoll. Ohne Sprachmodell lassen sich nur kleine Vokabulare bzw. Aufgaben mit niedriger Perplexität ausreichend gut erkennen. Das Sprachmodell wird in der Regel auf einer möglichst großen Menge Text, die aus der gleichen Domäne stammt wie die Trainingsdaten, oder auf den Trainingsdaten selbst geschätzt.

5.2.1 Trainingsverfahren

Die große Flexibilität des JANUS-Erkenners erlaubt viele verschiedene Arten des Trainings. In der Regel haben aber alle Arten einige wesentliche Schritte gemeinsam:

1. Trainieren eines Erkenners, der nur kontextunabhängige akustische Modelle mit relativ wenigen Parametern verwendet. Die initialen Parameter dieses Erkenners könnten zwar zufällig gewählt werden. Um eine schnellere und sicherere Konvergenz zu gewährleisten, werden sie in der Regel aber mit den Parametern eines anderen, bereits trainierten Erkenners für möglichst ähnliche akustische Randbedingungen initialisiert.

2. Das Trainieren der akustischen Parameter geschieht in mehreren Epochen. In einer Epoche werden dem System alle Trainingsdaten einmal präsentiert und die Parameter dem EM-Algorithmus entsprechend optimiert. Die optimale Anzahl der Epochen wird korrekterweise anhand einer Kreuzvalidierungsmenge bestimmt. Oft aber wird darauf verzichtet und ein auf Erfahrung basierender Wert genommen.

3. Um den aufwendigen Viterbi- oder Forward-Backward-Algorithmus seltener ausführen zu müssen, wird die von diesem Algorithmus gefundene zeitliche Zuordnung von Sprachvektoren zu HMM-Zuständen abgespeichert. So können weitere Trainingsschritte die zuvor gefundene Zuordnung in der Erwartung verwenden, daß sich die Zuordnung durch einen erneuten Aufruf des Zuordnungsalgorithmus nur unwesentlich verändern würde.

4. Wenn ein kontextunabhängiger Erkenner fertig ist, handelt es sich dabei meist um einen voll kontinuierlichen HMM-Erkenner, der für jedes Subphonem ein eigenes Codebuch und eine eigene Mixturgewichteverteilung verwendet. Danach wird für jedes kontextabhängige Subpolyphon,

von denen es hunderttausende geben kann, eine eigene Mixturgewichteverteilung eingeführt, die anfangs die gleiche ist wie die entsprechende kontextunabhängige. In einer oder mehreren erneuten Trainingsepochen werden die Parameter der vielen neu eingeführten Mixturgewichte trainiert. Das so entstehende System ist dann ein subphonetisch gekoppeltes semikontinuierliches HMM.

5. Wenn die vielen kontextabhängigen Mixturgewichte trainiert sind, werden sie mit Hilfe eines Ballungsverfahrens zu einigen tausend Klassen zusammengefaßt, die in einem kontext-befragenden Entscheidungsbaum angeordnet sind. An dieser Stelle gilt es festzulegen, wieviele Klassen entstehen sollen.

6. Ein komplett neuer kontextabhängiger Spracherkenner wird erzeugt, indem für jede durch die Ballung gefundene Klasse ein eigenes Codebuch und eine eigene Mixturgewichteverteilung verwendet wird. Die Initialisierung dieser Parameter erfolgt mit dem „Basic-ISODATA" oder k-Mittelwerte Algorithmus, der die zuvor abgespeicherten zeitlichen Zuordnungen verwendet. Nach der Initialisierung werden einige Epochen mit dem EM-Algorithmus trainiert. Das so entstehende System ist wieder ein voll kontinuierlicher HMM-Erkenner.

7. Mit dem bis dahin besten Erkenner werden erneut zeitliche Zuordnungen durch den Viterbi- oder Forward-Backward-Algorithmus berechnet und abgespeichert. In der Erwartung, daß diese genauer sind, wird in eine frühere Phase zurückgesetzt und alle nachfolgenden Trainingsphasen werden erneut durchgeführt. Dabei können dann auch Modifikationen zum Beispiel an der Art der Signalvorverarbeitung vorgenommen werden, die zuvor noch nicht sinnvoll oder möglich waren, wie etwa die Berechnung einer linearen Diskriminanzanalyse basierend auf den gefundenen Subpolyphonklassen.

5.2.2 Parameterräume in JANUS

Neben den Parametern für die akustische Modellierung mit Gauß-Mischverteilungen bietet JANUS auch die Möglichkeit, mehrschichtige Perzeptronen für die Berechnung von Emissionswahrscheinlichkeiten einzusetzen. Sowohl die Parameter der Gauß-Mischverteilungen als auch

die der Perzeptronen lassen sich mittels Entscheidungsbäumen divisiv ballen.

Ebenfalls divisiv ballen lassen sich explizite Längenmodelle (siehe Abschnitt 3.4.1) und Polyphontopologien. Mit dem Begriff *Topologie* werden die Zustände und Zustandsübergänge von Hidden-Markov-Modellen für einzelne Polyphone bezeichnet.

5.2.3 Aufbau von Satz-HMMs

Ein Satz-HMM bezeichnet die Folge von HMM-Zuständen, die eine Aufnahme (normalerweise ist das ein Satz) modellieren. Dazu werden ausgehend von den Worten der Transkription einer Aufnahme die zugehörigen Phoneme extrahiert und deren Topologien zusammengefügt.

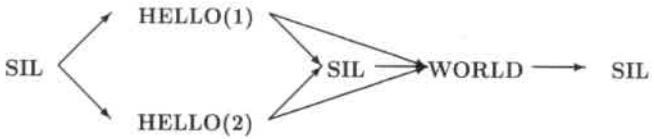
In der Regel liegen die Transkriptionen der Trainingsaufnahmen in einer suboptimalen Qualität vor. Das heißt, bestimmte akustisch bedeutsame Ereignisse sind nicht in den Transkriptionen beschrieben. Dazu gehören artikulatorische Geräusche (Atmen, Schmatzen) und nichtartikulatorische Geräusche (Tastaturgeklapper, Telefonklingeln) aber auch vor allem die akustische Stille. Echte Geräusche kommen in sorgfältig gesammelten Trainingsdaten eher selten vor (ca. 7 000 Geräusche unter 750 000 Wörtern) und sind meist von den Transkriptoren als solche markiert. Sie können dann ebenso wie andere Wörter akustisch modelliert werden und – insbesondere die artikulatorischen Geräusche – sogar ins Sprachmodell aufgenommen werden. Es wurde gezeigt, daß dadurch die Erkennungsleistung signifikant verbessert werden kann [SR95].

Die akustische Stille, die in jeder Aufnahme zumindest am Anfang und am Ende derselben kurz vorkommt, ist in der Regel nicht als solche in der Transkription markiert. Gerade bei der Verwendung von wortgrenzenüberschreitenden Polyphonen ergeben sich aber andere akustische Modelle, je nachdem ob zwischen zwei Wörtern eine Stille liegt oder ob sie ohne Pause aufeinanderfolgen. Deshalb erlaubt JANUS im Training ebenso wie in der Erkennung ein optionales Stillemodell zwischen zwei Wörtern.

Ein weiteres Problem vieler Transkriptionen ist die orthographische Schreibweise der Inhalte, aus der nicht hervorgeht, welche Aussprachevariante vom Sprecher tatsächlich gesprochen wurde. So steht zum Beispiel in einer

Transkription lediglich das Wort **FEBRUARY**, aber ob der Sprecher dafür die Phonemfolge **F EH B R UW R IY** oder **F EH B Y UW AX R IY** gesprochen hat, bleibt offen. Eine derart genaue Transkription von so großen Sprachdatenbasen wie der Wall Street Journal Datenbasis wäre extrem aufwendig und sehr teuer. Daher erlaubt JANUS sowohl im Training als auch während der Erkennung verschiedene Aussprachevarianten desselben Wortes.

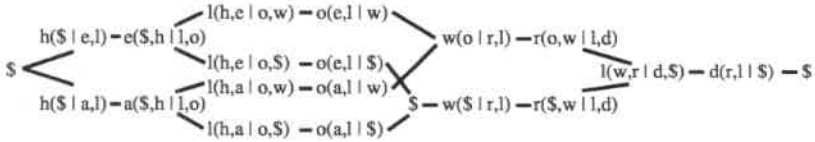
Ein Satz-HMM wird hierarchisch in drei Stufen aufgebaut. Auf der ersten Stufe wird ein Wortgraph aufgebaut, der sowohl die optionalen Stillen als auch die alternativen Aussprachen beachtet. Aus dem Satz **HELLO WORLD** wird dann zum Beispiel unter der Annahme, daß **HELLO** zwei verschieden Aussprachen erlaubt, der Graph



aufgebaut. Seien im Aussprachelexikon die folgenden Einträge

HELLO(1)	h e l o
HELLO(2)	h a l o
WORLD	w r l d
SIL	$

vorhanden, dann wird aus dem obigen Wortgraphen der folgende Polyphongraph aufgebaut:



Man sieht, daß vier verschiedene Modelle des Phonems **o** verwendet werden, je nachdem welche Aussprachevariante von **HELLO** gesprochen wird, und ob zwischen **HELLO** und **WORLD** eine Stille ist.

Im dritten Schritt wird aus dem Polyphongraphen ein HMM-Zustandsgraph berechnet. Dabei wird an die Stelle jedes Polyphons dessen Topologie eingesetzt. Die jeweils letzten Zustände einer Topologie werden dann um die Zustandsübergänge erweitert, die zu den Anfangszuständen der im Polyphongraph erlaubten Nachfolgepolyphone führen.

5.2.4 Erkennung

In der Erkennung werden mit Hilfe eines modifizierten Viterbi-Algorithmus Wortfolgehypothesen erzeugt, von denen die mit der höchsten berechneten Wahrscheinlichkeit als die endgültige ausgegeben wird. Die Modifikation des Viterbi-Algorithmus besteht im wesentlichen aus einer Aufteilung in mehrere Durchgänge. So werden im ersten Durchgang nur Bereiche des Suchraums bewertet und aussortiert. In einem zweiten Durchgang wird dann im vorsortierten Suchraum die eigentliche Hypothese gefunden. Der gesamte Vorgang der Hypothesenfindung wird im internen Sprachgebrauch als „Suche" bezeichnet. An dieser Stelle soll nicht weiter auf die Details der Suche eingegangen werden. Der interessierte Leser sei verwiesen auf die tiefergehende Literatur [WF96] [WCE+93a].

In einem der Suche nachgeschalteten Schritt kann die gefundene Hypothese unter der Annahme, daß sie eine sehr gute Approximation des tatsächlich Gesprochenen darstellt, verwendet werden, um die akustischen Parameter des Erkenners auf die gefundene Hypothese zu adaptieren. Ein erneuter Suchvorgang, der dann die adaptierten akustischen Parameter verwendet, produziert in der Regel etwas weniger fehlerhafte Hypothesen. Der Adaptionsschritt kann so auch mehrfach hintereinander iteriert werden, allerdings fällt schon in der zweiten oder dritten Iteration der Gewinn insignifikant klein aus.

Kapitel 6

Entwurf der Parameterräume

6.1 Motivation der Ballung

Abbildung 6.6 zeigt, wie groß die Zahl der kontextabhängigen Modelle werden kann. Man sieht leicht ein, daß bei einer Million oder mehr Modellen nicht jedem davon ein eigenes Codebuch zugestanden werden kann. Selbst wenn das Problem des enormen Speicherbedarfs vermutlich in der Zukunft gelöst wird, so bleibt immer noch das Problem der Trainierbarkeit. Von all den möglichen Modellen kommen viele in den zur Verfügung stehenden Trainingsdaten nur ein einziges Mal vor. Eine sinnvolle Schätzung der akustischen Parameter für solche Modelle ist nicht möglich. Dadurch, daß auch Modelle mit in die Liste aufgenommen werden, die aufgrund potentieller Aussprachevarianten einiger Wörter sowie der optionalen Sprechpause zwischen zwei Worten vorkommen könnten, enthält die Liste sogar eine Reihe von Modellen, die überhaupt nicht trainiert werden, weil in allen in Frage kommenden Fällen eine andere Variante ausgewählt wird. Bei einer Nachprüfung dieses Umstandes auf der Wall Street Journal Datenbasis ergab sich ein Wert von 38% überhaupt nicht verwendeter Modelle. Leider lassen sich diese 38% nicht von vornherein aus dem Training ausschließen, da erst durch das Training selbst, das heißt durch das Finden des Viterbi-Pfades festgestellt wird, welche alternativen Modelle verwendet werden.

Neben der Trainierbarkeit gibt es einen weiteren wichtigen Grund für das Ballen der kontextabhängigen Modelle, nämlich die Generalisierungsfähigkeit derselben. Je spezifischer ein Modell ist, umso besser kann es trainiert wer-

den, weil alle Trainingsdaten einander sehr ähneln. Andererseits besteht die Gefahr, daß ein sehr spezifisches Modell weniger gut auf ungesehene Daten paßt. Das Gegenteil, ein sehr grobes Modell, das mit vielen sehr verschiedenen Daten trainiert wird, fällt dann nicht besonders „scharf" aus, aber es paßt dafür wenigstens einigermaßen auf ungesehene, eventuell ein wenig andersartige zum selben Modell gehörende Daten, zumindest besser als ein auf wenigen Daten trainiertes spezifisches Modell. Ein Ziel der Ballung ist es also, einen möglichst optimalen Punkt auf der Linie zwischen der Modellgenauigkeit und der Generalisierungsfähigkeit zu finden. Für den Parameterraum stellt dieser Punkt die geeignete Menge an Ballung dar, bei der sowohl Modellgenauigkeit als auch die Generalisierungsfähigkeit in ausreichendem Maße erhalten bleiben.

6.2 Verschiedene Ballungsarten

Ballung kontextabhängiger Modelle kann auf verschiedene Arten geschehen. Sie unterscheiden sich nach der Einheit, die als Elementarmodell dient, und nach den von vornherein festgelegten Einschränkungen, die bestimmen, welche Elementarmodelle mit welchen zusammengelegt werden dürfen.

6.2.1 Generalisierte Triphone

Zu den ersten erfolgreichen Erkennern mit geballten kontextabhängigen Modellen zählte der Erkenner von Lee [Lee88]. Dieser Erkenner verwendete generalisierte Triphone. Als Elementarmodelle in der Ballung wurden Triphone verwendet. Da ein Triphon selbst aus mehreren Untereinheiten mit eigenen akustischen Modellen besteht, wurde für das Distanzmaß zwischen zwei Ballungsknoten die Summe der Distanzen der einander entsprechenden Untereinheiten benutzt.

Für die Distanz zwischen zwei akustischen Modellen wurde der Entropieanstieg, der durch das Zusammenlegen zweier Knoten entsteht, verwendet. Lee setzte ein agglomeratives Ballungsverfahren ein. Alle Triphone desselben Phonems konnten zusammengelegt werden. Dies war nur deshalb möglich, weil die Zahl der Triphone eines Phonems auf der damals eingesetzten Resource Management Datenbasis relativ überschaubar war. Sie lag um ca. eine Größenordnung unter der der Wall Street Journal Datenbasis. Da die

Berechnung aller Triphon-Distanzen mit der Zahl der Triphone quadratisch wächst, würde Lees Ballungsverfahren auf der für die Wall Street Journal Daten einen um zwei Größenordnungen höheren Aufwand bedeuten[1], was nicht praktikabel ist.

6.2.2 Senones (Generalisierte Subtriphone)

Senones wurden zum ersten Mal von Mei-Yuh Hwang [HH91] eingesetzt. Die Motivation für Senones bestand in der Annahme, daß komplette Triphone als zu ballende Einheiten zu grob sind. So kann man erwarten, daß ein in drei Segmente geteiltes Triphon **T** mit rechtem Kontext **R** und linkem Kontext **L1** mit dem Triphon **T** mit rechtem Kontext **R** und linkem Kontext **L2** zusammengelegt wird, weil die mittleren Segmente und die Endsegmente der beiden Triphone jeweils ähnlich sind. Die Anfangssegmente aber unterscheiden sich stärker, da die unterschiedlichen linken Kontexte einen stärkeren Einfluß auf die Anfangssegmente der Triphone haben. Wenn diese Triphone in dieselbe Klasse zusammengelegt werden, dann muß das akustische Modell für das Anfangssegment der Klasse auf relativ unhomogenen Trainingsdaten geschätzt werden. Wenn es nun möglich wäre, nur die mittleren Segmente und die Endsegmente der beiden Triphone in jeweils eine Klasse zu legen, dann könnten die akustischen Modelle für die Anfangssegmente getrennt modelliert werden.

Hwang berichtet in [HH91], durch Verwenden von Senones statt generalisierter Triphone eine Reduzierung der Fehlerrate von 4.7% auf 3.8% erzielt zu haben, was einer relativen Verbesserung um 20% entspricht.

6.2.3 Generalisierte Subpolyphone

Im JANUS-Erkenner werden generalisierte Subpolyphone als Klassen mit eigenen akustischen Modellen verwendet. Der Unterschied zu den Senones besteht nur in der größeren Kontextbreite. Während Senones stets die generalisierten Untereinheiten von Triphonen sind, reicht der Kontext bei JANUS bis in die angrenzenden Wörter.

[1] Die Zeit zur Ballung aller Resource Management Triphone betrug 1994 auf einer HP Apollo 735 ca. 7 CPU-Tage

6.3 Agglomerative Ballung

Verfahren der agglomerativen Kontextballung wurden in der kontinuierlichen Spracherkennung zum ersten Mal erfolgreich von Kai-Fu Lee eingesetzt [Lee88]. Dabei wurden mehrere komplette Triphone in eine Gruppe zusammengelegt. Lee verwendete einen rein semikontinuierlichen HMM-Spracherkenner mit drei Datenströmen (Cepstren, Delta-Cepstren und Energie). Da jedes Triphon mit drei HMM-Zuständen modelliert wurde, bestanden die Parameter einer zu ballenden Einheit s aus neun diskreten Wahrscheinlichkeitsverteilungen $\gamma_{s,1} \ldots \gamma_{s,9}$. Als Distanzmaß verwendete Lee die Summe des Entropieanstiegs, der sich durch das Zusammenlegen zweier Ballungsknoten q und r zu s ergibt:

$$D(q,r) = \sum_{d=1}^{9} (n_q \cdot H(\gamma_{q,d}) + n_r \cdot H(\gamma_{r,d}) - n_s \cdot H(\gamma_{s,d})) \qquad (6.1)$$

wobei n_i die Anzahl der Trainingsdaten des Knotens i ist, und $H(\gamma)$ die Entropie der Verteilung γ darstellt.

Lees Ballungsalgorithmus sah wie folgt aus:[2]

1. initialisiere jeden Ballungsknoten mit einem Triphon
2. berechne paarweise Distanzen zwischen allen Knoten
3. vereinige die beiden Knoten mit der kleinsten Distanz
4. (a) berechne für jedes Triphon den Informationsgewinn, der durch Versetzen desselben in einen anderen Knoten entsteht
   (b) falls durch Versetzen Gewinn erzielt werden kann, führe die Versetzung aus und gehe zu 4a
5. solange Endekriterium nicht erfüllt, gehe zu 2

[2] Er unterscheidet sich von dem Standardalgorithmus in [DH73] nur im zusätzlich eingeführten Schritt 4.

Der Schritt 4 ist nötig, um optimale Leistung zu erzielen. Wird er weggelassen, fällt die Erkennungsrate des resultierenden Erkenners signifikant ab. Es ist allerdings genau dieser Schritt, der den Algorithmus sehr aufwendig macht. Der Schritt 2 ist nur zu Beginn des Algorithmus aufwendig. Später müssen immer nur die noch nicht berechneten Distanzen berechnet werden, das heißt die Distanzen zwischen dem gerade neu erzeugten Ballungsknoten und den anderen Knoten. Im Schritt 4a muß für jedes einzelne Triphon, das nicht zum neu erzeugten Ballungsknoten gehört, getestet werden, ob es vorteilhaft wäre, es dorthin zu versetzen. Und für jedes Triphon des neuen Ballungsknotens muß geprüft werden, ob es vorteilhaft wäre, es in einen anderen Knoten zu versetzen. Die Anzahl der zu berechnenden Distanzen wächst also durch den Schritt 4 um ein Vielfaches an.

6.4 Divisive Ballung

Die agglomerative Ballung hat zwei wesentliche Nachteile. Zum einen steigt der Rechenaufwand für große Mengen akustischer Modelle sehr schnell ins Unerträgliche, da die Anzahl der möglichen Zusammenfassungen mit der Zahl der Modelle quadratisch wächst. Der zweite, schwerer wiegende Nachteil ist die suboptimale Modellierung ungesehener Kontexte. Da die durch agglomerative Ballung resultierende Zusammenfassung nur die im Training vorkommenden Kontexte verwendet, ist es auch nur für diese möglich, eine passende Klasse zuzuordnen. Kontexte, die nicht im Training vorkommen, müssen dann entweder irgendeiner mehr oder weniger willkürlich oder zufällig festgelegten Gruppe zugeordnet werden, oder sie werden zusammen mit anderen ungesehenen Kontexten mit einem groben (z.B. kontextunabhängigen) Modell modelliert. Dieser Nachteil wirkt sich insbesondere bei Erkennungsaufgaben aus, bei denen die Trainingsdaten und die zu erkennenden Daten verschiedene Vokabulare verwenden. Beim (D)ARPA Wall-Street-Journal Vergleichstest wird für die Erkennung ein Vokabular mit über 60 000 verschiedenen Wörtern verwendet, in den für das Training des Erkenners vorgesehenen Aufnahmen kommen aber nur gut 13 000 vor. Das heißt, die große Mehrzahl aller Wörter im Vokabular wurde im Training nie gesehen. Somit ergibt sich auch, daß viele Phonemkontexte, die erkannt werden müssen, nicht im Training vorkommen.

6.4.1 Entscheidungsbäume

Abbildung 6.1 zeigt einen Ausschnitt aus einem Beispiel, wie es in JANUS vorkommen könnte. Dargestellt sind drei Baumwurzelknoten. Für sämtliche hier beschriebenen Experimente wurde jedes Phonem mit drei HMM-Zuständen modelliert, die als Anfangs-, Mittel- und Endzustand bezeichnet werden. Auch wenn es prinzipiell möglich wäre, verschiedenartige, z.B. Anfangs- und Endzustände, zusammenzuballen, wurde von dieser Möglichkeit abgesehen, da der Ballungsalgorithmus einen quadratischen Aufwand hat und eine sehr große Rechenzeit (1-2 CPU-Wochen, SUN Ultra 2) schon für das Ballen von gleichartigen Zuständen benötigt. Aus dem gleichen Grund wurde auch darauf verzichtet, Zustände verschiedener Phoneme zusammenzuballen. Lediglich gleichartige Zustände desselben Phonems in verschiedenen Kontexten konnten durch Ballung mit demselben akustischen Atom modelliert werden.

Im Beispiel der Abbildung 6.1 stellen die Ovale nichtterminale Baumknoten dar, die eine Frage zum Kontext enthalten können und dann zwei Nachfolgeknoten haben, je einen für die positive und negative Antwort auf die Frage. Die abgerundeten Rechtecke stellen Codebücher dar, sie sind nicht Bestandteil des Entscheidungsbaumes, aber sie sind an den Stellen eingefügt, an denen sie durch Ballung entstanden sind. Über demselben Codebuch können mehrere Mixturgewichteverteilungen definiert sein. Im dargestellten Beispiel ist die mit **B-m(31)(1)** bezeichnete Mixturgewichteverteilung definiert über dem Codebuch **B-m(31)**. Das Codebuch wird für die Modellierung aller Mittelzustände des Phonems **B** verwendet, deren unmittelbarer linker Kontext kein Vokal und deren übernächster rechter Kontext ein Frikativ ist. Die Mixturgewichteverteilung wird für eine kleinere Menge von Kontexten verwendet, nämlich nur für die, für die zusätzlich gilt, daß der dritte Kontext links kein Stopp-Phonem ist.

Abbildung 6.2 zeigt zwei Schritte der Entstehung eines Entscheidungsbaumes zur divisiven Ballung von Kontexten am Beispiel des Phonems **B**. Die rechteckigen Knoten der Bäume stellen Blätter des Baumes dar und sind beschriftet mit dem Namen des akustischen Modells, für das sie stehen. An jedem Blatt hängt ein Behälter, in dem alle atomaren Modelle aufgeführt sind, die in dem Blatt zusammengeballt werden. Die Namen der atomaren Modelle sind in der in JANUS verwendeten Schreib-

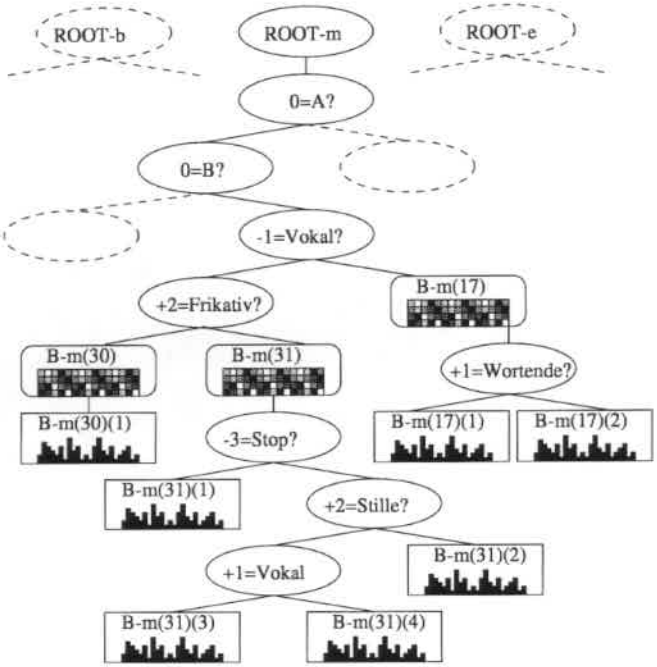


Abbildung 6.1: Ausschnitt aus einem Beispielbaum

weise **Phonem (linker Kontext | rechter Kontext)** angegeben. Die abgerundeten Baumknoten stellen Verzweigungen dar. Im ersten beispielhaft dargestellten Schritt wird die Frage gestellt, ob der Kontext +2, also das übernächste Phonem rechts vom zentralen Phonem, ein Vokal ist. Dies ist der Fall für zwei der vier vorhandene atomaren Modelle, die deshalb in einem neuen geballten Modell namens **B(1)** zusammengefaßt werden. Entsprechend ist **B(2)** das Modell, mit dem die anderen beiden Kontexte modelliert werden. Die Frage, ob der Kontext +2 ein Vokal ist, wurde aus einem vorher festgelegten Fragenkatalog als diejenige ausgewählt, bei deren Anwendung der Entropieverlust durch Aufteilen des Knotens in zwei Nachfolgeknoten

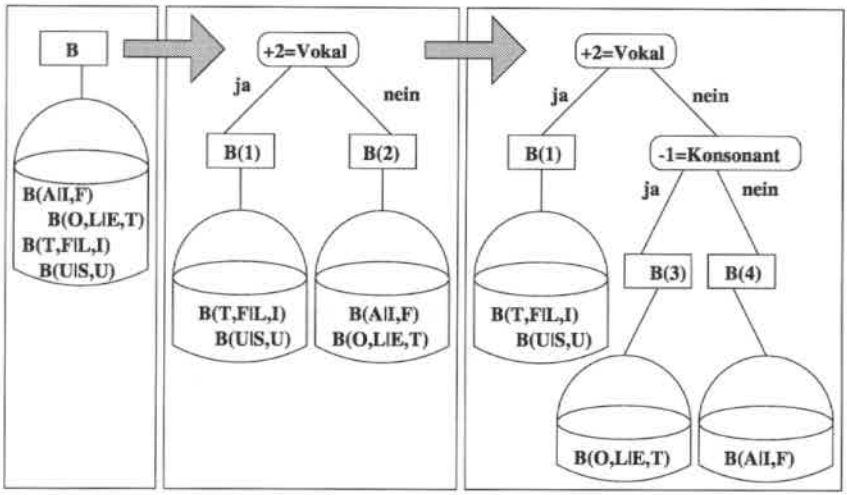


Abbildung 6.2: Entstehung eines Entscheidungsbaumes

am größten ausfällt. Im zweiten Schritt wird dann **B(2)** ersetzt durch **B(3)** und **B(4)**, die durch die Frage, ob das unmittelbare linke Nachbarphonem (Kontext -1) ein Konsonant ist, aufgetrennt werden.

Ein im Training ungesehenes Polyphon, z.B. **B(S|S,A)** wird dann entsprechend dem dargestellten Entscheidungsbaum mit dem akustischen Modell **B(3)** modelliert, weil die Frage $+2$ =**Vokal** mit „ja" und die darauffolgende Frage -1 =**Konsonant** mit „nein" beantwortet werden.

Der in der Abbildung 6.2 dargestellte Baum bezieht sich nur auf ganze Polyphone. In der Praxis werden jedoch Subpolyphone geballt, was einen wesentlich feineren Architekturentwurf erlaubt.

6.4.2 Erhalten der Trainierbarkeit

Um eine ausreichend gute Trainierbarkeit der durch das Ballen entstehenden Modelle zu gewährleisten wird im JANUS-Spracherkenner ein Minimalwert festgelegt. Beim divisiven Ballen werden dann keine Aufspaltungen

zugelassen, die einen Baumknoten entstehen lassen würden, der weniger Trainingsdaten erhielte als der festgelegte Minimalwert. Dadurch wird bei jedem Aufspaltungsschritt die Menge der möglichen Aufspaltungen eingeschränkt. Die Bestimmung des Minimalwertes erfolgt durch eine auf Erfahrung basierende Schätzung. Zur Orientierung für die Schätzung dient vor allem die geplante Größe der Codebücher. Der Minimalwert wird dann so festgelegt, daß für jeden zu trainierenden Referenzvektor im Schnitt eine Mindestmenge an Trainingsvektoren vorhanden ist. Bei allen hier beschriebenen Experimenten wurde ein Minimalwert von 1 000 verwendet, der sicherstellt, daß jedes Codebuch im System mit mindestens 1 000 Trainingsmustern, das heißt eine Sekunde, trainiert werden kann.

6.5 Verschiedene Distanzmaße

Im folgenden werden zwei Distanzmaße zur Ballung von Kontexten vorgestellt. Das erste, die Entropie-Distanz, hat zum Ziel, die Information im Parameterraum eines Erkenners zu optimieren. Das zweite, die Likelihood-Distanz, wurde vom Autor entwickelt und hat zum Ziel, die Wahrscheinlichkeit der Trainingsdaten zu maximieren.

6.5.1 Entropie-Distanz

Schon relativ früh [Lee88] wurden Entropie-Distanzmaße verwendet, um verschiedene Modelle zu ballen, die gegeben sind durch jeweils gleich große diskrete Wahrscheinlichkeitsverteilungen.

Im folgenden wird die Entropie-Distanz zweier Ballungsknoten definiert. Seien K_1 und K_2 zwei Ballungsknoten, definiert durch die Mixturgewichteverteilungen der einzelnen Elementarmodelle.

$$\begin{aligned} K_1 &= \{\gamma_{1,1} \dots \gamma_{1,m_1}\} \\ K_2 &= \{\gamma_{2,1} \dots \gamma_{2,m_2}\} \end{aligned}$$

K_1 bestehe also aus m_1 Elementarmodellen $\{e_{1,1} \dots e_{1,m_1}\}$ und K_2 aus m_2 Elementarmodellen $\{e_{2,1} \dots e_{2,m_2}\}$. Gemäß der Definition des Ballungs-

algorithmus sind K_1 und K_2 disjunkt.

$p_{i,j}$ bezeichne die a-priori Wahrscheinlichkeit für das Elementarmodell $e_{i,j}$. Dann sind die a-priori Wahrscheinlichkeiten für K_1 und K_2 gegeben durch:

$$p(K_1) = \sum_{l=1}^{m_1} p_{1,l}$$

$$p(K_2) = \sum_{l=1}^{m_2} p_{2,l}$$

Sei $K_0 = K_1 \cup K_2$ die Vereinigung der beiden Ballungsknoten. Sei außerdem:

$$\begin{aligned}
\gamma_1(k) &= \frac{1}{\sum_{l=1}^{m_1} p_{1,l}} \cdot \sum_{l=1}^{m_1} p_{1,l}\gamma_{1,l}(k) \\
\gamma_2(k) &= \frac{1}{\sum_{l=1}^{m_2} p_{2,l}} \cdot \sum_{l=1}^{m_2} P_{2,l}\gamma_{2,l}(k) \\
\gamma_0(k) &= \frac{p(K_1)\cdot\gamma_1(k) + p(K_2)\cdot\gamma_2(k)}{p(K_1)+p(K_2)}
\end{aligned}$$

Dann ist die Entropie-Distanz zwischen K_1 und K_2:

$$D = p(K_1)\cdot H_1 + p(K_2)\cdot H_2 - (p(K_1)+p(K_2))\cdot H_0, \tag{6.2}$$

wobei H_i die Entropie der Verteilung γ_i ist, also:

$$H_i = -\sum_{k=1}^{n} \gamma_i(k)\cdot\log\gamma_i(k)$$

Es ist offensichtlich, daß dieses Distanzmaß nicht geeignet ist, um automatisch festzustellen, wann es sinnvoll ist, einen Ballungsvorgang (egal ob divisiv oder agglomerativ) abzubrechen. Beim divisiven Ballen mit diesem Distanzmaß werden die bei jedem Aufteilen eines Ballungsknotens in zwei kleinere Knoten erzielten Entropieverluste kleiner, aber sie werden nie negativ und in der Regel auch nie Null, so daß mit jedem Schritt ein wenn auch kleiner, so doch meist vorhandener Gewinn erzielt wird. Um den Ballungsvorgang zu beenden, muß entweder vorgegeben werden, wieviele Ballungsknoten am Ende vorhanden sein sollen, oder es muß ein Mindestgewinn (das heißt eine Mindestdistanz) vorgegeben werden, der nicht unterschritten werden darf.

6.5.2 Likelihood-Distanz

Es sprechen zwei Gründe gegen eine Verwendung der Entropie als Distanzmaß für die Ballung. Zum einen die angesprochene Schwierigkeit festzustellen, wann ein guter Zeitpunkt zur Beendigung des Ballungsalgorithmus gekommen ist, und zum anderen, weil Informationsgewinn nicht direkt das Ziel ist, das der Motivation der Ballung entspricht. Die Ballung verfolgt zwei Ziele: Zum einen soll sie Klassen so erzeugen, daß diese einen geeigneten Kompromiß zwischen Trainierbarkeit und Generalisierungsfähigkeit erreichen, zum anderen sollen die in Frage kommenden Ballungen möglichst gut auf die vorhandenen Trainingsdaten passen, das heißt, die Gesamtwahrscheinlichkeit dieser Daten bei gegebenem akustischen Modell maximieren. Beide Ziele werden durch die Likelihood-Distanz mit Kreuzvalidierungsmenge verfolgt.

Bei den folgenden Ausführungen gehen wir davon aus, daß alle akustischen Elementarmodelle, die durch Ballung zusammengelegt werden können, dasselbe Codebuch benutzen (subphonetisch gekoppeltes semikontinuierliches HMM).

Sei $P = \{i_1 \dots i_T\}$ eine Menge von Indizes von Trainingsvektoren $\{x_{i_1} \dots x_{i_T}\}$, die mit Hilfe des Viterbi-Algorithmus dem gemeinsamen Codebuch zugeordnet wurden. P kann auch als Bezeichnung für den entsprechenden Baumknoten verwendet werden, der alle akustischen Elementarmodelle enthält, die das gemeinsame Codebuch benutzen. Nun bezeichne $N_{l,k}$ den Wert der k-ten Normalverteilung des gemeinsamen Codebuchs an der Stelle x_l. γ_P bezeichne die Mixturgewichteverteilung, wie sie gemäß des EM-Algorithmus auf allen Daten P geschätzt wird. Dann ist die Wahrscheinlichkeit (Likelihood) dafür, daß $x_{i_1} \dots x_{i_T}$ beobachtet wird, unter der Bedingung, daß das Modell bestehend aus N und γ vorliegt:[3]

$$L_P(P) = \prod_{f \in P} \sum_k \gamma_{P,k} \cdot N_{f,k} \tag{6.3}$$

Während des Ballungsvorganges muß irgendwann eine antizipierte Teilung des Knotens P in zwei Nachfolgeknoten Q und R bewertet werden und

[3]Der Index P bezieht sich auf die Parameter des Knotens P, die für die Wahrscheinlichkeitsberechnung verwendet werden, und das Argument P bezieht sich auf die Menge der Trainingsdaten, deren Beobachtungswahrscheinlichkeit berechnet wird.

die Bewertung mit der anderer möglicher Teilungen verglichen werden. Seien also Q und R auch die Bezeichnungen für zwei disjunkte Teilmengen von P, so daß $P = Q \cup R$. Nach der Aufteilung des Knotens P beträgt die bedingte Wahrscheinlichkeit für die Beobachtung von P

$$L_{QR}(P) = L_Q(Q) \cdot L_R(R) = (\prod_{f \in Q} \sum_k \gamma_{Q,k} \cdot N_{f,k}) \cdot (\prod_{f \in R} \sum_k \gamma_{R,k} \cdot N_{f,k}) \quad (6.4)$$

Offensichtlich ist $L_{QR}(P) \geq L_P(P)$, da γ_Q und γ_R besser auf ihren jeweiligen Teilmengen geschätzt werden können als eine gemeinsame Verteilung auf der Vereinigungsmenge.

Ohne Verwendung einer Kreuzvalidierungsmenge würde also auch bei der Likelihood-Distanz jede Aufteilung eines Baumknotens einen Gewinn bringen. Mit Verwendung einer Kreuzvalidierungsmenge sieht die Sache etwas anders aus. Seien nun die gesamten Trainingsdaten aufgeteilt in zwei disjunkte Teilmengen A und B, also $P = B \cup A$. γ_P^A bezeichne im Gegensatz zu γ_P die Mixturgewichteverteilung, wie sie gemäß des EM-Algorithmus geschätzt wird, wenn sie nur auf Daten aus A trainiert wird. Wir interessieren uns nun für den Anstieg der Wahrscheinlichkeit der Beobachtung von B beim Teilen von P in Q und R unter der Bedingung, daß als akustisches Modell N und γ^A vorliegen. Jetzt ist nicht mehr davon auszugehen, daß jede Aufteilung einen positiven Anstieg der Beobachtungswahrscheinlichkeit bewirkt, da durch viele Teilungen zwar die Daten aus A „auswendig" gelernt werden können, aber die dann geschätzten akustischen Modelle nicht mehr so gut auf die Kreuzvalidierungsdaten B passen.

Es bezeichne $L_{P,A}(B)$ die bedingte Beobachtungswahrscheinlichkeit von B, unter der Bedingung, daß das akustische Modell P auf den Daten A trainiert worden ist. Es gilt:

$$L_{P,A}(B) = \prod_{f \in B} \sum_k \gamma_{P,k}^A \cdot N_{f,k} \quad (6.5)$$

Nach einer Aufteilung von P in Q und R ergibt sich die Beobachtungswahrscheinlichkeit von B zu $L_{Q,A}(B) \cdot L_{R,A}(B)$ mit

$$L_{Q,A}(B) = \prod_{f \in B} \sum_k \gamma_{Q,k}^A \cdot N_{f,k} \quad \text{und} \quad (6.6)$$

$$L_{R,A}(B) = \prod_{f \in B} \sum_{k} \gamma_{R,k}^{A} \cdot N_{f,k} \tag{6.7}$$

Als Distanzmaß $D_{A \to B}(Q, R)$ für den Ballungsalgorithmus definiert man dementsprechend:

$$D_{A \to B}(Q, R) = \log \frac{L_{Q,A}(B) \cdot L_{R,A}(B)}{L_{P,A}(B)} \tag{6.8}$$

Der Wert von $D_{A \to B}(Q, R)$ ist positiv, wenn ein Gewinn erzielt werden kann. Er ist 0, wenn B nach der Aufteilung die gleiche Beobachtungswahrscheinlichkeit hat. Und er ist negativ, wenn durch die Aufteilung eine Verkleinerung der Beobachtungswahrscheinlichkeit eintritt.

Dadurch, daß die Trainingsdaten in eine Trainingsuntermenge A und eine Kreuzvalidierungsmenge B aufgeteilt werden, leidet die Qualität der Schätzung der Modellparameter. Diesem Problem begegnet man üblicherweise mit der „Leaving-One-Out" oder „Round-Robin" Methode. Dabei wird jeder Teil der Gesamttrainingsmenge sowohl zum Schätzen der Parameter als auch als Teil der Kreuzvalidierungsmenge verwendet. In der obigen Rechnung bedeutet dies, daß zur Berechnung des Distanzmaßes nicht nur $D_{A \to B}(Q, R)$ sondern auch $D_{B \to A}(Q, R)$ berücksichtigt wird, also die gleiche Sache nur mit vertauschten Rollen der Trainings- und Kreuzvalidierungsmenge.

Bei einer Aufteilung der Daten in zwei Untermengen, von denen jede einmal Trainings- und einmal Kreuzvalidierungsmenge ist, wurde in der vorliegenden Arbeit das folgende Distanzmaß verwendet:

$$D_{A,B}(Q, R) = \log\left(\frac{L_{Q,A}(B) \cdot L_{R,A}(B)}{L_{P,A}(B)} \cdot \frac{L_{Q,B}(A) \cdot L_{R,B}(A)}{L_{P,B}(A)}\right) \tag{6.9}$$

Bei einer Aufteilung der Daten in k Untermengen $C_1 \ldots C_k$ ergibt sich als Distanzmaß:

$$D_{C_1 \ldots C_k}(Q, R) = \log \prod_{i=1}^{k} \frac{L_{Q,P \setminus C_i}(C_i) \cdot L_{R,P \setminus C_i}(C_i)}{L_{P,P \setminus C_i}(C_i)} \tag{6.10}$$

6.5.3 Problematik des Likelihood-Maßes

Das Likelihood-Maß bringt einige auf den ersten Blick nicht offensichtliche Probleme mit sich. Ein nicht unbedeutendes Problem ist die Speicherung aller $N_{f,k}$. Beachtet man, daß der Grundbereich von f für die hier verwendete Wall Street Journal Datenbasis von 0 bis ca. 30 000 000 reicht und k die Werte 0 bis 48 annehmen kann, dann ergibt sich bei einer Darstellung von Fließkommazahlen mit 32 Bits ein Gesamtspeicherbedarf von ca. 5 bis 6 Gigabyte. Solche Speichermengen sind erst in den letzten Jahren zum Standard geworden.

Auch wenn heute diese Speichermengen auf Festplatten problemlos zu benutzen sind, so stellt die Rechnerarchitektur mit ihrem wesentlich kleineren Speicher einen Flaschenhals dar. Für die vorliegende Arbeit war die größte Speicherkapazität eines zur Verfügung stehenden Rechners ca. 1/2 Gigabyte, so daß nicht alle $N_{f,k}$ gleichzeitig im Speicher gehalten werden konnten.

Um die oben aufgeführten Probleme zu lösen, wurden die folgenden zwei Maßnahmen unternommen:

Selektives Speichern der Normalverteilungswerte

Es wurde ein auf Erfahrung basierend geschätzter Wert von $\epsilon = 10^{-3}$ gewählt. Alle $N_{f,k}$, für die $N_{f,k} < \epsilon \cdot \max_i N_{f,i}$ galt, wurden als 0 betrachtet. Diese Werte mußten auch nicht gespeichert werden. Dadurch war sichergestellt, daß alle Normalverteilungen, die mehr als ein Promille zur Summe aller Normalverteilungen einer Mixtur beitragen, auch in die späteren Berechnungen einbezogen wurden. Durch das nun nötige zusätzliche Abspeichern der Indizes der Normalverteilungen entstand zwar ein gewisser Mehraufwand, aber insgesamt konnte der benötigte Speicher von ca. 5 bis 6 Gigabyte auf ca. 1 Gigabyte gesenkt werden.

Nach Subpolyphonen getrennte Ballung

Nach der Definition des Ballungsalgorithmus gilt es, in jedem Ballungsschritt diejenige Klasse aufzutrennen, die unter allen Klassen den größten Gewinn erzielt. Da von vornherein festgelegt wird, daß kein Subpolyphon mit einem Subpolyphon eines anderen Phonems in einer Klasse liegen kann, darf man den Ballungsalgorithmus nach Subphonemen getrennt durchführen.

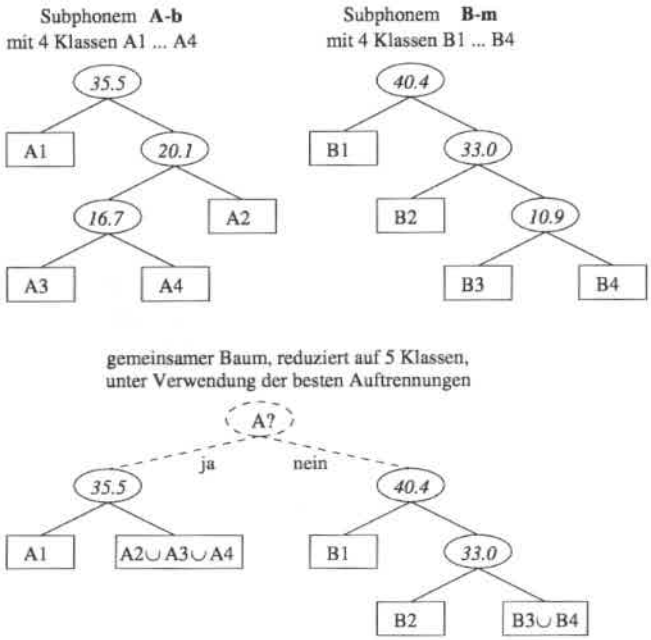


Abbildung 6.3: Nach Subpolyphonen getrennte Ballung

Dabei entstehen dann Teilbäume des gesamten Entscheidungsbaumes. Die einzelnen Ballungsvorgänge werden nicht abgebrochen, sondern so lange durchgeführt, wie Klassenaufteilungen möglich sind. Auf diese Art kann der sehr aufwendigen Ballungsvorgang auf getrennten Prozessoren parallel berechnet werden, ohne daß dabei eine Kommunikation zwischen den Prozessoren nötig wäre. Am Ende wird von allen Teilungsschritten nur eine bestimmte Anzahl der besten in den gesamten Entscheidungsbaum aufgenommen. Abbildung 6.3 veranschaulicht dies. Die Werte in den Baumknoten sind die Gewinne (Entropieverluste), die durch das Aufspalten der Konten entstehen.

In dem abgebildeten Beispiel wurde der Teilbaum für die Anfangssegmente des Phonems **A** auf einem Prozessor berechnet und der Teilbaum für die mittleren Segmente des Phonems **B** auf einem anderen Prozessor. Beide Prozessoren haben die Ballung so weit durchgeführt, bis jeweils vier Klassen entstanden waren. Dann wurde die Entscheidung getroffen, einen gemeinsamen Baum mit nur fünf Klassen zu erzeugen. Dazu wurden die gewinnbringendsten Klassenauftrennungen der beiden Teilbäume betrachtet. Am Ende fielen die zuvor erzeugten Klassen A2, A3 und A4 sowie die Klassen B3 und B4 jeweils in eine Klasse zusammen. Bei der Verwendung der Likelihood-Distanz mit Kreuzvalidierung entfällt die Entscheidung über die Zahl der am Ende erwünschten Klassen, und der gemeinsame Baum besteht stets aus der Vereinigung der kompletten Teilbäume.

6.5.4 Ein Spezialfall

Für den Fall, daß die Berechnung der Emissionswahrscheinlichkeiten nach der „top-1" Methode (siehe Abschnitt 2.5) durchgeführt und für das Distanzmaß keine Kreuzvalidierung verwendet wird, ergibt sich, daß die beiden Distanzmaße identisch sind, wie die folgende Rechnung zeigt.

Sei ein Modell definiert durch die Menge seiner Trainingsmuster F. Sei L_F die bedingte Wahrscheinlichkeit, F zu beobachten, bei gegebenen Modellparametern $\gamma_{F,k}, \Sigma_{F,k}$ und $\mu_{F,k}$, wobei $\mu_{F,k}$ der k-te Mittelwert des Codebuchs von F ist, und $\Sigma_{F,k}$ die zu $\mu_{F,k}$ gehörende Kovarianzmatrix. $\gamma_{F,k}$ ist das k-te Mixturgewicht von F. $N_{f,k}$ sei der Wert der Normalverteilung mit Mittelwert $\mu_{F,k}$ und Kovarianz $\Sigma_{F,k}$ an der Stelle f. Dann gilt (analog zu Gleichung 6.5):

$$L_F = \prod_{f \in F} \sum_{k=1}^{m} \gamma_{F,k} \cdot N_{f,k} \tag{6.11}$$

$$\log L_F = \sum_{f \in F} \log \sum_{k=1}^{m} (\gamma_{F,k} \cdot N_{f,k}) \approx \sum_{f \in F} \log(\max_k(\gamma_{F,k} \cdot N_{f,k})) \tag{6.12}$$

Die Gleichung 6.12 stellt die Approximation der korrekten Emissionswahrscheinlichkeit mit der „top-1" Methode dar. Sei nun $r(f) := \text{argmax}_k N_{f,k}$, also der Index der Normalverteilung mit dem größten Wert aus dem Codebuch. Wenn wir nun zusätzlich – wie in der top-1-Implementation üblich – annehmen, daß

$$\max_k(\gamma_{F,k} \cdot N_{f,k}) \overset{top-1}{\approx} \gamma_{F,\operatorname{argmax}_k N_{f,k}} \cdot \max_k N_{f,k} \tag{6.13}$$

$$= \gamma_{F,r(f)} \cdot N_{f,r(f)} \tag{6.14}$$

dann ergibt sich:

$$\log L_F \overset{top-1}{\approx} \sum_{f \in F} \log(\gamma_{F,r(f)} \cdot N_{f,r(f)}) \tag{6.15}$$

$$= \sum_{f \in F} \log(\gamma_{F,r(f)}) + \sum_{f \in F} \log N_{f,r(f)} \tag{6.16}$$

$$= \sum_{f \in F} \sum_{k=1}^{m} (\log \gamma_{F,k}) \delta_{k,r(f)} + \sum_{f \in F} \sum_{k=1}^{m} (\log N_{F,k}) \delta_{k,r(f)} \tag{6.17}$$

wobei hier δ das Kronecker-Delta darstellt. Nun gilt aber, daß gemäß der Definition des EM-Trainingsalgorithmus (siehe Abschnitt 2.5) die Mixturgewichte nach Gleichung 2.20 bestimmt werden:

$$\gamma_{F,k} = \frac{1}{|F|} \cdot \sum_{f \in F} \delta_{k,r(f)} \tag{6.18}$$

das heißt, das k-te Mixturgewicht ist bestimmt durch die relative Häufigkeit des Ereignisses, daß die k-te Normalverteilung die größte ist.

Definiert man nun

$$g_F := \sum_{f \in F} \sum_{k=1}^{m} (\log N_{F,k}) \delta_{k,r(f)} \tag{6.19}$$

so ist leicht zu erkennen, daß für disjunkte Mengen F_1, F_2 gilt $g_{F_1 \cup F_2} = g_{F_1} + g_{F_2}$. Somit ergibt sich aus Gleichung 6.17:

$$L'_F := \log L_F \overset{top-1}{\approx} \sum_{k=1}^{m} (\log \gamma_{F,k}) \cdot \sum_{f \in F} \delta_{k,r(f)} + g_F \tag{6.20}$$

$$= \sum_{k=1}^{m} (\log \gamma_{F,k}) \cdot \gamma_{F,k} \cdot |F| + g_F \tag{6.21}$$

$$= |F| \cdot H_F + g_F \tag{6.22}$$

Hier ist H_F die Entropie der auf F geschätzten Verteilung γ_F. Somit gilt für den Anstieg der top-1-Approximation des Logarithmus der bedingten Wahrscheinlichkeit von $F_1 \cup F_2$:

$$\begin{aligned} L'_{F_1 \cup F_2} - (L'_{F_1} + L'_{F_2}) &= |F_1 \cup F_2| \cdot H_{F_1 \cup F_2} + g_{F_1 \cup F_2} \\ &\quad - |F_1| \cdot H_{F_1} - g_{F_1} \\ &\quad - |F_2| \cdot H_{F_2} - g_{F_2} \\ &= |F_1 \cup F_2| \cdot H_{F_1 \cup F_2} - |F_1| \cdot H_{F_1} - |F_2| \cdot H_{F_2} \end{aligned} \tag{6.23}$$

Gleichung 6.23 ist genau der gewichtete Entropieanstieg beim Vereinigen der Mengen F_1 und F_2.

6.6 Experimente

Zur Evaluierung des Ballungsverfahrens mit Kreuzvalidierungsmenge und Likelihood-Distanz wurden mehrere Trainings- und Testläufe auf der Wall Street Journal Datenbasis durchgeführt. Zum Vergleich wurden drei verschieden große Parameterräume mit der Entropie-Distanz erzeugt. Die Erkenner hatten 2 000, 3 000 und 5 000 Codebücher. Nach einer Epoche Training lagen die Fehlerraten zwischen 11.6% und 14.0%.

Parallel dazu wurde ein Entscheidungsbaum mit dem Likelihood-Distanzmaß und zwei Trainingsteilmengen entsprechend Gleichung 6.10 erzeugt. Der Ballungsalgorithmus terminierte, nachdem 4585 Klassen erzeugt waren. Bei etwa der Hälfte aller subphonemspezifischen Teilbäume war die am besten bewertete Auftrennung für beide Distanzmaße gleich. Insgesamt war die Ähnlichkeit der einander entsprechenden Teilbäume in der „Nähe" der Baumwurzeln recht groß. Die Rangordnungen der möglichen Aufteilungen unterschieden sich immer mehr, je mehr man sich von den Wurzeln zu den Blättern bewegte (siehe Abbildung 6.4).

Bei allen Erkennern wurde stets verlangt, daß für jede Klasse mindestens 1 000 Trainingsbeispiele verfügbar sind. Das heißt, ein Aufteilen eines Baumknotens in zwei Nachfolger, von denen einer weniger als 1000

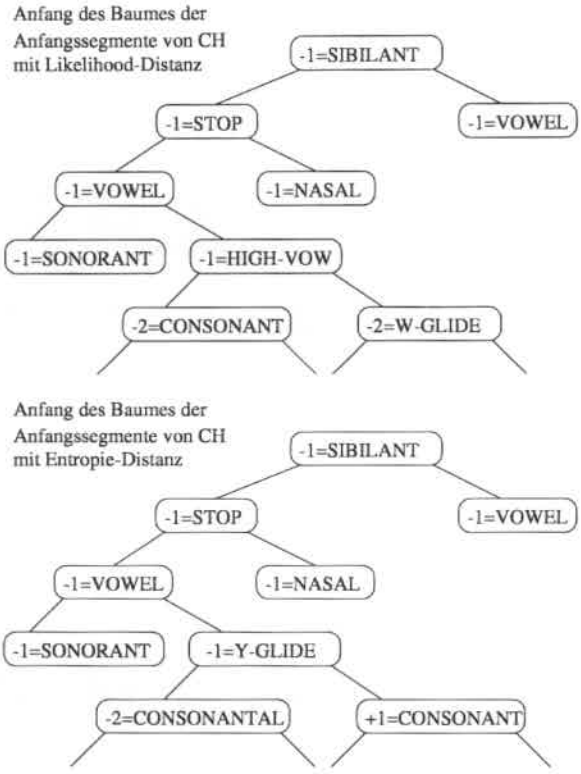


Abbildung 6.4: Ähnliche Anfänge der Entscheidungsbäume

Tabelle 6.1: Vergleich verschiedener Ballungsmethoden

Distanzmaß	Entropie			Likelihood
Anzahl Codebücher	2000	3000	5000	4585
Fehlerrate	14.0%	13.3%	11.6%	10.8%

Trainingsvektoren hätte, wurde nicht zugelassen. Dies war sowohl bei den mit Entropie-Distanz als auch bei den mit Likelihood-Distanz erzeugten Systemen der Fall. Diese Einschränkung erfüllt zwei Aufgaben. Zum einen stellt sie sicher, daß die entstehenden Klassen trainierbar bleiben und keine Klassen entstehen, für die so wenige Trainingsdaten vorhanden sind, daß ihre Parameter zu schlecht geschätzt werden. Zum anderen sorgt sie dafür, daß solche Aufteilungen nur erschwert zum Zug kommen, die aus einem Baumknoten, der eine Klasse mit vielen Elementarmodellen repräsentiert, lediglich eine ganz kleine Klasse mit einem oder zwei Elementarmodellen abtrennen. Ohne diese Einschränkung würden in der Tat bei beiden Distanzmaßen sehr oft einzelne Subpolyphone aus einer Klasse extrahiert, und die so entstehenden Bäume würden sehr „einseitig" aussehen. Das heißt sie wären „nein-lastig" mit vielen Fragen zu einzelnen Phonemen oder sehr kleinen Phonemklassen.

Der Erkenner, der mit der Likelihood-Distanz erzeugt wurde, erzielte eine Fehlerrate von 10.8% und war damit besser als alle mit Entropie-Distanz erzeugten Erkenner. Tabelle 6.1 faßt die Ergebnisse zusammen.

6.7 Die Suche nach einer guten Kontextbreite

In Kapitel 3 wurden verschiedene Spracheinheiten vorgestellt und erläutert, warum Phoneme die bevorzugte Einheit in der Spracherkennung sind. Bei der Modellierung kontextabhängiger Phoneme stellt sich die Frage nach einer sinnvollen Breite des zu verwendenden Kontextes. Hierbei ist zu erwarten, daß weit entfernte Kontexte die Aussprache eines Phonems weniger stark beeinflussen als nahe Kontexte. Nahe Kontexte wirken sich vor allem dadurch aus, daß die Mechanik des Artikulationsapparates nur kontinuierliche Bewegungen machen kann. Daher hängt die Akustik am Anfang eines Phonems vom vorhergegangenen Phonem ab, solange der Artikulationsapparat sich noch nicht vom Zustand des vorhergehenden

Phonems auf das aktuelle Phonem vollständig umgestellt hat. Ebenso hängt die Akustik am Ende eines Phonems vom nachfolgenden Phonem ab, weil der Artikulationsapparat sich schon darauf vorbereitet. Wie die im folgenden beschriebenen Untersuchungen zeigen, kommt den nachfolgenden Phonemen sogar eine größere Bedeutung zu als den vorhergehenden. Bis vor wenigen Jahren waren in der Spracherkennung Triphone die am häufigsten verwendeten kontextabhängigen Phoneme. Erst in letzter Zeit werden vermehrt breitere Kontexte eingesetzt. Im folgenden wird die Bedeutung der Kontextbreite für kontinuierliches Diktieren anhand der Wall Street Journal Datenbasis untersucht.

6.8 Statistische Untersuchungen

Bevor die in der vorliegenden Arbeit verwendeten Methoden zur Modellierung breiter Kontexte erläutert werden, werden einige Resultate von statistischen Untersuchungen berichtet.

6.8.1 Wortlängen

Aufgrund verschiedener technischer Beschränkungen durch den JANUS-Dekoder war es in der vorliegenden Arbeit nicht möglich, die Breite der modellierten Kontexte über das erste angrenzende Phonem des Folgewortes oder über das letzte Phonem des vorhergehenden Wortes auszudehnen. Für Phoneme, die in der Mitte eines Wortes vorkamen, wurden überhaupt keine Wortübergangskontexte verwendet. Das heißt, daß für viele kurze Worte, die nicht mehr Phoneme als die maximale verwendete Kontextbreite haben, quasi Ganzwortmodelle benutzt werden, zumindest solange die Parameter dieser Modelle nicht durch Ballung mit denen anderer Modelle gekoppelt werden.

Abbildung 6.5 zeigt die Verteilung der Wortlängen in Anzahl der Phoneme, so, wie sie im Aussprachelexikon vorkommen und gewichtet nach der Häufigkeit ihres Vorkommens in den gesamten Trainingsdaten der Wall Street Journal Datenbasis. Man sieht, daß zwar die häufigste Wortlänge im Aussprachelexikon sechs Phoneme ist, aber weil kürzere Wörter in der natürlichen Sprache wesentlich öfter vorkommen als längere, ist die häufigste

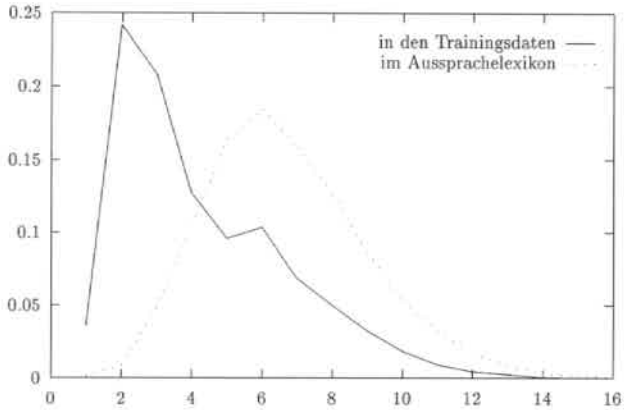


Abbildung 6.5: relativer Anteil verschiedener Wortlängen (in Phonemen)

tatsächlich auftauchende Wortlänge zwei Phoneme. Der kleine Artefakt, nach dem mehr Wörter mit sechs als mit fünf Phonemen gesprochen werden hängt damit zusammen, daß in der Wall Street Journal Datenbasis sehr oft Jahreszahlen, Personennamen und Dollarbeträge vorkommen, und die Worte „nineteen", „Mister" und „Dollars" werden alle standardmäßig mit sechs Phonemen modelliert. Außerdem taucht das Wort „period" beim Diktieren mit Interpunktion am Ende jedes Satzes auf, und auch dieses Wort wird mit sechs Phonemen modelliert.

Im Hinblick auf diese Verteilung von Wortlängen kann man leicht feststellen, daß bei einer verwendeten maximalen Kontextbreite von eins, das heißt bei der Verwendung von Triphonen, etwa 50% aller Wörter in den Trainingsdaten eigene Ganzwortmodelle erhalten, bei der Verwendung von Quintphonen (Kontextbreite zwei) sind es etwa 80%. Bei Septphonen (Kontextbreite drei) sind es gar 90%. Demnach ist zu erwarten, daß der Gewinn in der Erkennungsleistung, der durch Verbreitern des Kontextes erzielt wird, mit der Breite des Kontextes abnimmt.

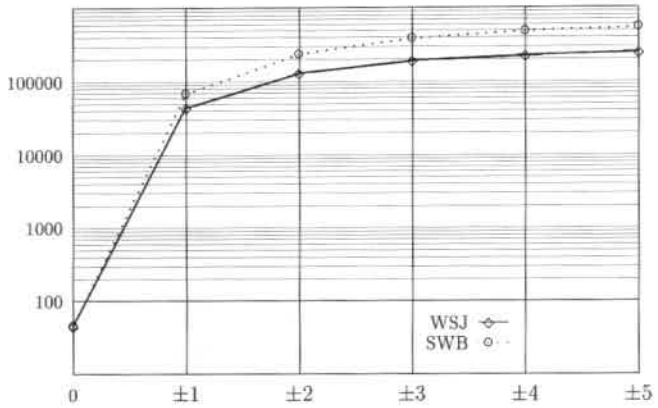


Abbildung 6.6: Anzahl Polyphone bei verschiedenen Kontextbreiten

6.8.2 Viele Polyphone

Man sieht leicht ein, daß mit wachsender Breite der maximalen modellierten Kontexte, auch die Anzahl der zu modellierenden Polyphone steigt. Abbildung 6.6 zeigt die Anzahl der verschiedenen Polyphone für zwei verschiedene Datenbasen, die Wall Street Journal Datenbasis als Vertreter der englischen gelesenen Sprache und die Switchboard Datenbasis als Vertreter der englischen spontanen Sprache (Telefonkonversation). Bei beiden Kurven wird davon ausgegangen, daß zur Unterscheidung von Polyphonen die Information über Wortgrenzen mit einfließt, das heißt, daß der gleiche Kontext am Anfang eines Wortes durch ein anderes Polyphon repräsentiert wird als wenn er mitten in einem Wort auftritt.

Man erkennt, daß ab einer Kontextbreite von drei die Anzahl der Polyphone in den Sättigungsbereich übergeht. Die etwa doppelt so große Zahl der Polyphone in der Switchboard Datenbasis kommt zum einen Teil daher, daß das Aussprachelexikon für diese Datenbasis mehr Aussprachevarianten

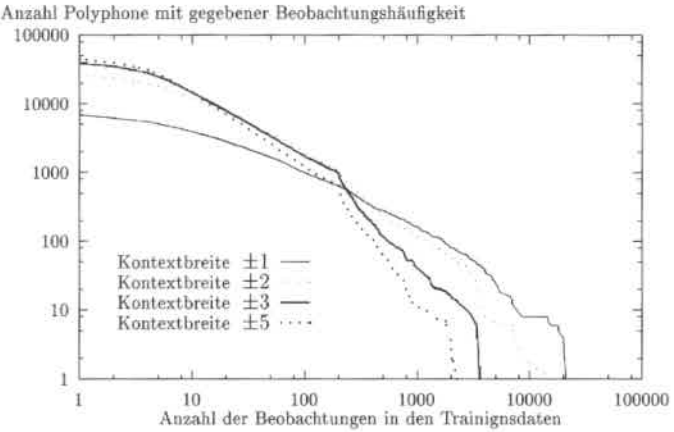


Abbildung 6.7: Verteilung der Polyphonhäufigkeiten

kennt, und zum anderen Teil daher, daß in der spontanen Sprache weniger grammatikalisch gesprochen wird, was an Wortübergängen zu einer größeren Anzahl möglicher Polyphone führt.

Abbildung 6.7 zeigt, wie oft welche Polyphonhäufigkeit vorkommt. Man sieht, daß bei einer maximalen Kontextbreite von ±1 knapp 7 000 Polyphone nur ein einziges Mal vorkommen, während es einige Polyphone gibt, die mehr als 20 000 mal auftauchen. Bei einer maximalen Kontextbreite von ±5 kommen fast 45 000 Polyphone nur einmal vor, während hier nur sehr wenige Polyphone öfter als 2 000 mal auftreten. Das zeigt, wie sehr das Problem der Trainierbarkeit mit der maximalen Kontextbreite wächst. Unter Berücksichtigung der Tatsache, daß schon bei einer Kontextbreite von ±3 circa 90% aller Wörter mit dem maximal möglichen Kontext vom Anfang des Wortes bis zum Ende des Wortes modelliert werden, sieht man leicht ein, daß bei noch breiteren Kontexten kaum noch signifikante Leistungssteigerungen zu erwarten sind.

6.9 Ballung kontextabhängiger Modelle

6.9.1 Erhalten der Trainierbarkeit

Je breiter der zu modellierende Kontext gewählt wird, desto mehr Elementarmodelle gibt es. Entsprechend groß fällt auch der Parameterraum vor der Ballung aus. Eine wie sonst übliche Verwendung voll kontinuierlicher HMMs mit 32-teiligen Gauß-Mixturen über einem 48-dimensionalen Merkmalsraum ist bei mehreren hunderttausend Modellen nicht praktikabel. Nach der Ballung allerdings spielt die Kontextbreite sowohl bei der Größe des Parameterraumes als auch bei der Rechengeschwindigkeit keine Rolle mehr. Die Trainierbarkeit wird lediglich durch die Anzahl der am Ende benutzten Ballungsklassen bestimmt.

6.9.2 Binäre oder ternäre Bäume

Beim Aufbau eines Satz-HMM (siehe Abschnitt 5.2.3) werden Kontexte nicht in unbegrenzter Breite über Wortgrenzen hinaus erlaubt. Insbesondere kann eine kombinatorische Explosion eintreten, wenn man mehrere kurze Worte hintereinander mit jeweils mehreren möglichen Aussprachevarianten und optionalen Stillen dazwischen mit unbeschränktem Kontext modellieren will. Deshalb werden im JANUS-Spracherkenner nur Phoneme an den Worträndern, also das erste und letzte Phonem eines Wortes, mit wortübergreifenden Kontexten modelliert. Dabei wird auch maximal das entsprechende Randphonem das angrenzenden Wortes berücksichtigt.

Durch die zwangsweise Einschränkung der Kontextbreite ergibt sich allerdings ein noch nicht angesprochenes Problem, nämlich das, daß bestimmte Fragen nicht beantwortet werden können. Wenn zum Beispiel eine Frage im Entscheidungsbaum lautet „ist das dritte Phonem nach rechts ein Vokal" und der gerade verwendete Kontext reicht nur zwei Phoneme nach rechts, dann kann keine Antwort auf die Frage gegeben werden.

Für dieses Problem bieten sich zwei Lösungen an: entweder werden alle nicht beantwortbaren Fragen beliebig, aber konsistent mit „ja" oder „nein" beantwortet, oder statt binärer Entscheidungsbäume mit je zwei möglichen Antworten werden ternäre Bäume mit drei möglichen Antworten „ja", „nein" und „unbekannt" verwendet.

Erste Experimente mit ternären Bäumen brachten unvorhergesehene Schwierigkeiten mit den verwendeten Distanzmaßen ans Licht. Prinzipiell lassen sich sowohl das Entropie- als auch das Likelihood-Distanzmaß dahingehend erweitern, daß nicht die Distanz zweier, sondern dreier Ballungsklassen berechnet werden (vgl. Gleichungen 6.2 und 6.10), nämlich:

$$\begin{aligned} D_E(Q,R,S) &= p(Q)\cdot H_Q + p(R)\cdot H_R + p(S)\cdot H_S \\ &- (p(Q)+p(R)+p(S))\cdot H_{Q\cup R\cup S} \end{aligned} \tag{6.24}$$

$$D_L(Q,R,S) = \log \prod_{i=1}^{k} \frac{L_{Q,P\setminus C_i}(C_i)\cdot L_{R,P\setminus C_i}(C_i)\cdot L_{S,P\setminus C_i}(C_i)}{L_{P,P\setminus C_i}(C_i)} \tag{6.25}$$

Aber dabei stellt sich heraus, daß Fragen, die eine gleichmäßige Aufteilung der Trainingsdaten auf alle drei Nachfolgeklassen bewirken, überproportional bevorzugt werden, während es eher wünschenswert ist, den Anteil der Trainingsdaten, die zur Antwort „unbekannt" zugeordnet werden, zu minimieren. Die Folge ist somit, daß Fragen über weit entfernte Kontexte schon relativ früh im Entscheidungsbaum vorkommen. Die Qualität solcher Entscheidungsbäume ist signifikant schlechter als die von binären Bäumen, bei denen jede Frage zu einem unbekannten Kontext mit „nein" beantwortet wird. Deshalb wurden in der vorliegenden Arbeit ausschließlich binäre Bäume entwickelt und eingesetzt.

6.10 Experimente

6.10.1 Auswahl der Fragen

Die Abbildungen 6.8 und 6.9 zeigen, mit welcher Häufigkeit Fragen zu bestimmten Kontexten gestellt werden. Am hervorstechendsten sind die Häufigkeiten der Kontexte -1 und +1, also der unmittelbaren Nachbarphoneme. Die Fragen zu weiter entfernt liegenden Kontexten kommen erst in größeren Tiefen der Entscheidungsbäume vor. Erst dort erhöht sich auch ihr relativer Anteil an der Gesamtmenge aller gestellten Fragen.

In Tabelle 6.2, die die Abbildung 6.9 noch einmal in Zahlen darstellt, kann man erkennen, daß dem rechten Kontext am Anfang des Ballungsprozesses eine größere Bedeutung zukommt als dem linken. Erst weiter unten

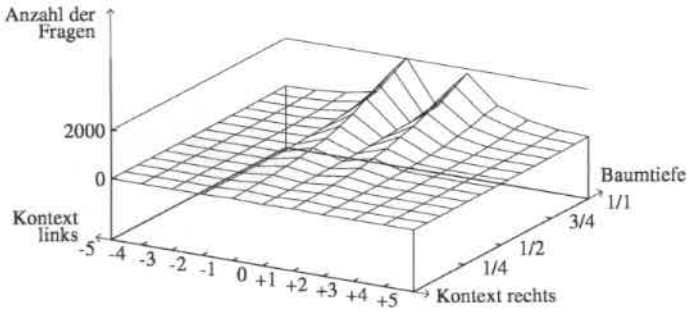


Abbildung 6.8: Häufigkeit der Kontextfragen (absolut)

Tabelle 6.2: Anteil der Kontextbreiten im Entscheidungsbaum

	Kontext										
Baumtiefe	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5
1/128	0.0	0.0	0.0	0.0	44.1	2.9	53.0	0.0	0.0	0.0	0.0
1/64	0.0	0.0	0.0	0.0	48.2	3.7	47.4	0.7	0.0	0.0	0.0
1/32	0.0	0.0	0.0	0.0	48.0	3.3	47.2	1.5	0.0	0.0	0.0
1/16	0.0	0.0	0.0	0.6	48.6	3.1	44.4	3.0	0.3	0.0	0.0
1/8	0.0	0.0	0.4	2.5	43.7	4.5	43.1	5.0	0.7	0.1	0.0
1/4	0.1	0.3	0.8	4.6	41.5	5.0	37.7	8.1	1.5	0.4	0.1
1/2	0.3	0.6	1.6	5.8	38.5	4.5	33.1	10.7	3.3	1.1	0.5
1/1	0.7	1.4	3.3	7.2	33.8	3.9	30.3	11.6	5.1	1.9	0.8

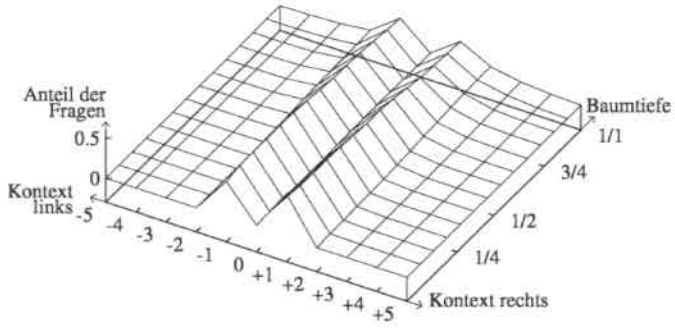


Abbildung 6.9: Häufigkeit der Kontextfragen (relativ)

im Entscheidungsbaum steigt der relative Anteil der Fragen zum linken Kontext. Daraus kann man schließen, daß die Akustik eines Phonems stärker von der Vorbereitung des Artikulationsapparates auf die zeitlich folgenden Phoneme (rechter Kontext) abhängt als von den Nachwirkungen der zuvor gesprochenen Phoneme (linker Kontext).

Interessante Erkenntnisse erhält man auch, wenn man die Verteilung der Fragen nach den Untersegmenten der Polyphone betrachtet. Abbildung 6.10 zeigt die Häufigkeiten der angefragten Kontexte für einen Erkenner mit 7 000 Modellen. Man sieht deutlich, daß die Endsegmente Fragen zum rechten Kontext bevorzugen, während die Anfangssegmente mehr Fragen zum linken Kontext stellen. Noch deutlicher wird das, wenn man dabei nur das obere[4] Drittel des Entscheidungsbaumes betrachtet. Dort dominieren die Fragen in die Richtung des Segmentes noch mehr. Die Wurzelfragen, das heißt die Fragen, die zu einem Subphonem als erste gestellt werden,

[4] wir bezeichnen mit „oben" den wurzelnahen Bereich eines Baumes, wegen der in der Informatik üblichen Darstellungsweise von Bäumen

Tabelle 6.3: Fragen in den Wurzelknoten

Kontext	-2	-1	0	+1	+2
Anfangssegmente	0	42	0	1	0
Mittensegmente	0	12	1	30	1
Endsegmente	0	1	0	42	0

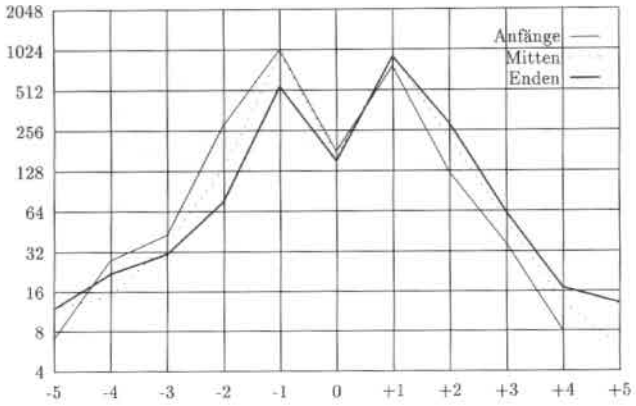


Abbildung 6.10: Häufigkeit der Kontextfragen für verschiedene Polyphonsegmente

verteilen sich wie in Tabelle 6.3 dargestellt.

Insgesamt sind von den 7000 Modellen 4875 Anfangssegmente, 4977 Mittelsegmente und 4159 Endsegmente. Insgesamt werden 3245 Fragen zu linken und 3458 Fragen zu rechten Kontexten gestellt. Die Tatsache, daß mehr Fragen zum rechten Kontext gestellt werden, obwohl die rechte Kontexte bevorzugenden Endsegmente am seltensten vorkommen, hebt noch einmal hervor, daß die Akustik eines Phonems von den darauffolgenden Phonemen stärker beeinflußt wird als von der vorhergegangenen.

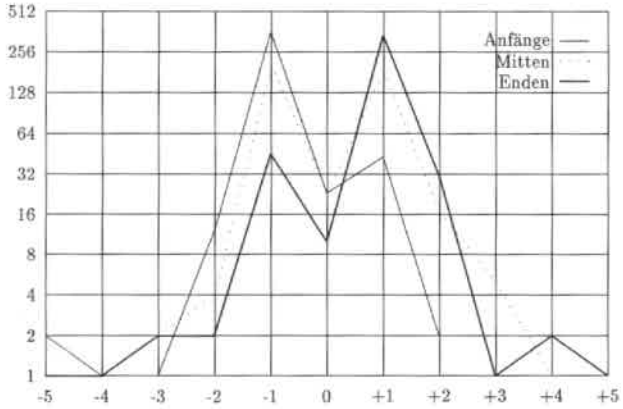


Abbildung 6.11: Häufigkeit der Kontextfragen für verschiedene Polyphonsegmente, beschränkt auf das obere Drittel des Entscheidungsbaumes

Tabelle 6.4: Fehler bei verschiedenen maximalen Kontextbreiten (WSJ-Testmenge 1994)

maximale Kontextbreite	±0	±1	±2	±3
Fehlerrate	35.5%	20.9%	20.2%	19.9%

6.10.2 Erkennungsleistung

In einem frühen Entwicklungsstadium des Wall Street Journal Erkenners wurden Experimente mit verschiedenen maximalen Kontextbreiten durchgeführt. Dabei ergaben sich die in Tabelle 6.4 zusammengefaßten Fehlerraten.

Während auf den englischen Wall Street Journal Aufnahmen eine wenn auch moderate Verbesserung durch Verbreitern des Kontextes beobachtet werden konnte, war dies für deutsche Aufnahmen, die im Rahmen des VERBMOBIL [WE92] Projektes gesammelt worden waren, nicht der Fall (Tabelle 6.5).

Tabelle 6.5: Verschiedene Kontextbreiten im Deutschen (VERBMOBIL-Testmenge 1996)

maximale Kontextbreite	±0	±1	±2	±3
Fehlerrate	31.7%	19.6%	19.2%	19.6%

6.11 Zusammenfassung

In diesem Kapitel wurde die Verwendung kontextabhängiger akustischer Modelle in JANUS vorgestellt. Die Auswirkungen verschiedener Kontextbreiten wurden untersucht und Experimente mit verschiedenen Teststichproben beschrieben. Dabei stellte sich heraus, daß durch Vergrößern der verwendeten Kontextbreite eine signifikante Fehlerreduktion erzielt werden kann. Die beobachteten Verbesserungen waren nach Testmenge unterschiedlich. Aufgrund der statistischen Analyse der Struktur der Phoneme und ihrer Kontexte wurde die Vermutung aufgestellt, daß der zu erwartende Gewinn mit der Breite des modellierten Kontextes abnimmt. Diese Vermutung wurde bestätigt.

Zwei Verfahren zur Entwicklung des Parameterraumes mittels divisiver Ballung wurden verglichen. Das bisher im JANUS-Erkenner verwendete Verfahren verwendet ein Distanzmaß, das die Aufteilung eines Ballungsknotens in zwei Nachfolgeknoten nach dem durch die Spaltung entstehenden Entropieverlust bewertet. Es hat also zum Ziel, den Informationsgehalt der Parameter des Systems zu maximieren. Im allgemeinen bedeutet dies aber nicht, daß die bedingte Wahrscheinlichkeit, die Trainingsdaten bei gegebenem akustischen Modell zu beobachten, auch maximiert wird. Deshalb wurde ein Verfahren entwickelt, das zum Ziel hat, die Beobachtungswahrscheinlichkeit der Trainingsdaten zu maximieren. Zur Implementierung dieses likelihood-basierten Verfahrens mußten Maßnahmen getroffen werden, um den Rechenzeit- und Speicherbedarf zu begrenzen. Die Möglichkeit, die Trainingsdaten in mehrere Untermengen aufzuteilen und diese als Kreuzvalidierungsmengen zu verwenden, erlaubt es, ein Haltekriterium für den Ballungsalgorithmus zu definieren, nämlich den Fall, daß keine Aufteilung eines Baumknotens in zwei Nachfolgeknoten möglich ist, durch die die Beobachtungswahrscheinlichkeit der Kreuzvalidierungsmenge erhöht werden kann. In Experimenten wurde festgestellt, daß dieses Haltekriterium zu einem ähnlich

großen Parameterraum führt, wie er durch manuelles mehrfaches Austesten verschiedener Größen gewählt würde. Zusätzlich konnte durch die Verwendung des likelihood-basierten Verfahrens die Fehlerrate von 11.6% auf 10.8% gesenkt werden. Der Gesamtaufwand für einen vollständigen Ballungsvorgang mit der Likelihood-Distanz ist in etwa gleich groß wie der Aufwand mit der Entropie-Distanz.

Kapitel 7

Kompaktifizierung des Parameterraumes

Im folgenden werden einige Arten der möglichen Kompaktifizierung des Parameterraumes vorgestellt. Dazu gehören die Verkleinerung der Codebücher durch Entfernen einzelner Referenzvektoren, die Radialisierung von Kovarianzmatrizen und die Kopplung derselben. In den dazu durchgeführten Experimenten hat sich vor allem die Radialisierung der Kovarianzmatrizen als sinnvoll erwiesen, da durch sie nicht nur eine erhebliche Zahl an Parametern eingespart werden, sondern auch die Erkennungsleistung gesteigert werden kann.

7.1 Methoden zur Größenbestimmung von Codebüchern

Während sehr wenige Vektoren je Codebuch sehr gut geschätzt werden können, dafür aber die Trainingsdaten zu grob modellieren, tritt bei zu vielen Vektoren der unerwünschte Effekt der Überanpassung auf. Verschiedene Methoden zur Bestimmung einer angemessenen Anzahl Vektoren je Codebuch wurden in der Spracherkennungsforschung eingesetzt. Sie unterscheiden sich im wesentlichen durch den Zeitpunkt, an dem die Entscheidung getroffen wird. Die Entscheidung über die Größe der Codebücher kann im voraus getroffen werden, bevor die Codebücher angelegt werden. Sie kann während des Trainings stattfinden, wobei bei Bedarf einzelne Vektoren zu den Co-

debücher hinzugefügt oder aus ihnen entfernt werden. Als dritte Variante kann man auch Codebücher mit relativ vielen Vektoren trainieren und am Ende unerwünschte Vektoren entfernen.

7.1.1 Bestimmung im voraus

Eine Bestimmung im voraus kann auf verschiedenen Informationen basieren. Naheliegend wäre, die Größe eines Codebuchs in Abhängigkeit der Anzahl der dafür zur Verfügung stehenden Trainingsdaten festzulegen, so daß die Codebücher, für die viele Trainingsdaten zur Verfügung stehen, größer gewählt werden.

In der Praxis ist aber die am häufigsten angewandte Methode die Festlegung aufgrund einer auf Erfahrung basierenden Schätzung. Die üblicherweise gewählten Codebuchgrößen liegen zwischen 10 und 50 Vektoren. Bei der Festlegung spielen Faktoren wie Rechenzeit, Speicherbedarf und Trainierbarkeit eine Rolle.

T. Kemp stellt in [Kem95] ein Verfahren vor zur Im-Voraus-Bestimmung der Codebuchgrößen anhand des durchschnittlichen Abstandes der Vektoren einer Kreuzvalidierungsmenge zu ihrem jeweils nächsten Referenzvektor. Dabei wird aus den Trainingsdaten für ein Codebuch eine Teilmenge als Kreuzvalidierungsmenge entfernt. Dann werden verschieden große Codebücher angelegt und für verschiedene Größen Referenzvektoren mit der k-Mittelwerte Methode berechnet. Auf den so entstehenden Codebüchern wird dann der durchschnittliche Abstand der Vektoren der Kreuzvalidierungsmenge zu ihren zugehörigen Referenzvektoren gemessen, jeweils gewichtet mit der absoluten Anzahl der zu einem Referenzvektor gehörenden Trainingsvektoren. In der Regel sinkt dieser gewichtete durchschnittliche Abstand mit steigender Anzahl Referenzvektoren. Ab einer gewissen Anzahl aber setzt der Effekt der Überanpassung ein, und das wachsende Codebuch fängt an, die Trainingsdaten „auswendig" zu lernen. Die Generalisierungsfähigkeit läßt nach und der durchschnittliche Abstand der Kreuzvalidierungsmenge nimmt wieder zu. Die Größe, bei der der gewichtete Abstand am kleinsten ist, wird dann als endgültige Codebuchgröße gewählt und das eigentliche Training beginnt danach. Kemp zeigte, daß mit diesem Verfahren signifikante Leistungssteigerungen von 33.1% Fehler auf 30.1% auf deutscher Spontansprache erzielt werden können. Gleichzeitig kann auf diese Art die Gesamtzahl aller Refe-

renzvektoren kleiner gewählt werden als diejenige des besten der Erkenner, bei denen alle Codebücher dieselbe Größe haben.

7.1.2 Iteratives Wachsen

Der Erkenner der Cambridge University in England verwendet ein iteratives Wachstumsverfahren [WOVY94]. Dabei werden alle Codebücher anfangs mit einem einzigen Vektor initialisiert. Dann wird im Training bestimmt, auf welche Art jeder Vektor am besten durch zwei ersetzt werden kann. Nachdem dies durchgeführt worden ist, wird die Leistung des Systems auf einer Kreuzvalidierungsmenge getestet. Diese Schritte werden iterativ so lange fortgeführt, bis die Erkennungsleistung nicht mehr zunimmt.

7.1.3 Entfernen unerwünschter Vektoren

In der vorliegenden Arbeit wurde eine zusätzliche Methode evaluiert, um Codebuchgrößen zu bestimmen. Hierbei geht es darum, aus einem fertig trainierten Erkenner diejenigen Referenzvektoren zu entfernen, die entweder für die Erkennung kaum von Bedeutung sind oder dieser sogar schaden könnten.

Motivation

Durch das Entfernen von Vektoren werden mehrere Ziele verfolgt:

- Erhöhung der Modellrobustheit:
  Weil modelluntypische Referenzvektoren, sogenannte Ausreißer, meist auf sehr wenigen Trainingsbeispielen geschätzt werden, werden diese als erste entfernt. Ausreißer, die typisch für ein anderes Codebuch sind, können der Erkennungsleistung erheblich schaden.
  Beinahe-Singularitäten (siehe auch Abschnitt 7.3) treten mit größerer Wahrscheinlichkeit bei schlecht trainierten Kovarianzmatrizen auf. Durch Entfernen der entsprechenden Vektoren werden die negativen Auswirkungen von Beinahe-Singularitäten verringert.

- Kompaktifizierung des Parameterraumes:
  Weniger Referenzvektoren können nicht nur robuster geschätzt werden, es wird auch Speicherplatz gespart.

Tabelle 7.1: Fehlerate (%) nach Entfernen von Referenzvektoren

Menge entfernter Referenzvektoren / Kriterium	0	0.625%	1.25%	2.5%	5%	10%	20%	40%
Trainingsdaten	8.8	8.8	8.7	8.7	8.6	8.8	9.7	13.1
Determinante	8.8	8.8	8.9	9.0	9.0	9.2	9.5	11.0
Varianzen	8.8	8.8	8.8	8.9	9.1	9.2	9.3	9.9

- Rechenzeiteinsparung:
  Wenn weniger Gauß-Werte zu berechnen sind, wird die Berechnung der Emissionswahrscheinlichkeiten beschleunigt.

Bestimmung der zu entfernenden Vektoren

- Anzahl der Trainingsdaten:
  Entferne alle Vektoren, deren Zahl n der Beispielvektoren aus den Trainingsdaten unterhalb eines Schwellwertes liegt.

- Beinahe-Singularitäten:
  Entferne alle Vektoren, deren Kovarianzmatrix eine Determinante hat, die unterhalb eines Schwellwertes liegt, oder entferne alle Vektoren, deren diagonale Kovarianzmatrix einen Varianzwert enthält, der unterhalb eines Schwellwertes liegt.

7.2 Experimente

Im Rahmen der vorliegenden Arbeit wurden die oben aufgeführten Methoden der Eliminierung von Referenzvektoren untersucht und miteinander verglichen. Tabelle 7.1 und Abbildung 7.1 fassen die Ergebnisse zusammen.

Man sieht, daß der Parameterraum durch Entfernen von Vektoren ein wenig verkleinert werden kann, ohne daß dadurch die Erkennungsleistung leidet. Bei geeigneter Verkleinerung stellt sich sogar eine kleine Fehlerreduktion ein. Ab einer Entfernung von ca. 10% aller Vektoren fängt die Fehlerrate an, deutlich zu steigen.

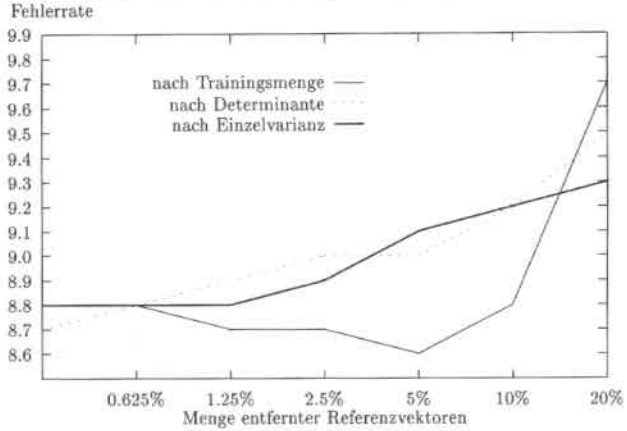


Abbildung 7.1: Fehlerrate nach Entfernen von Referenzvektoren

Die unterschiedlichen Kriterien für die Entfernung eines Referenzvektors unterscheiden sich in ihrer Wirkung nicht wesentlich. In früheren Experimenten wurde die Erfahrung gemacht, daß eine im voraus gewählte, um 20% kleinere Anzahl Referenzvektoren je Codebuch weniger starke Effekte auf die Erkennungsleistung hat als das nachträgliche Entfernen von 20% aller Vektoren. Tabelle 7.2 zeigt die Fehlerraten eines kontextunabhängigen Erkenners für verschiedene Codebuchgrößen. Der Erkenner, der mit 16 statt 32 Vektoren je Codebuch, also mit 50% weniger Parameter, trainiert wurde, hatte eine um nur ca. 4% erhöhte Fehlerrate, während das nachträgliche Entfernen von nur 20% der Vektoren die Fehlerrate um 6% bis 10% erhöhte. Daher ist davon auszugehen, daß der Schaden, der durch Entfernen von Vektoren entsteht, nicht auf die Verkleinerung des Parameterraumes zurückzuführen ist, sondern auf die Abwesenheit der Modellierung bestimmter wichtiger Sprachsignale, die bei einem EM-Training auch mit kleinerem Parameterraum erfaßt werden können.

Während des EM-Trainings werden die Referenzvektoren nur mit positiven Beispielen trainiert. Während der Erkennung aber wird die Emissionswahrscheinlichkeit auch für klassenfremde Sprachvektoren berechnet, so daß

Tabelle 7.2: Fehlerate bei verschiedene Codebuchgrößen

Anzahl Referenzvektoren je Codebuch	8	16	32	64	128
Fehlerrate	46.6%	41.2%	39.5%	36.8%	36.3%

die Verteilung der Häufigkeit, wie oft ein Referenzvektor der nächste Vektor zu einem Sprachvektor war, dann ganz anders ausfällt. In der Tat ist es sogar so, daß während der Erkennung sogar mehrheitlich Distanzen zu klassenfremden Vektoren berechnet werden. Entfernt man nun die Referenzvektoren nicht in Abhängigkeit von ihrer Trainingshäufigkeit, sondern von ihrer Testhäufigkeit, so ergibt sich ein deutlich günstigeres Bild. In Experimenten konnte festgestellt werden, daß dann gut die Hälfte aller Vektoren aus den Codebüchern entfernt werden kann, bevor die Erkennungsrate signifikant darunter leidet. Das heißt, daß die Größe der Trainingsdatenmenge eines Vektors nicht die beste Eigenschaft zur Bewertung seiner Wichtigkeit während des Testens ist.

7.3 Typen von Kovarianzmatrizen

Es gibt verschiedene Gründe anzunehmen, daß die Verwendung von normalen Kovarianzmatrizen für die Berechnung der HMM Emissionswahrscheinlichkeiten mit Mixturen von Normalverteilungen nicht die beste denkbare Lösung ist. Die Gründe dafür sind zum einen praktischer Natur. Die Anzahl der Parameter einer vollen Kovarianzmatrix für einen d-dimensionalen Merkmalsraum beträgt $d(d + 1)/2$, während die Mittelwertevektoren lediglich jeweils d Parameter haben. Dadurch ergibt sich nicht nur ein enormer Speicherbedarf, sondern auch das Problem der Trainierbarkeit. Viele Parameter benötigen auch viele Trainingsdaten, um ausreichend gut geschätzt werden zu können.

Ein anderer Nachteil voller Kovarianzmatrizen ist der erhöhte Rechenaufwand. Die Berechnung einer Normalverteilung mit einer Kovarianzmatrix, bei der nur die Diagonalelemente ungleich Null sind, kann wesentlich schneller durchgeführt werden, als wenn alle Matrixelement in die Rechnung mit einbezogen werden müssen.

Ein weiterer Grund, der für die Verwendung diagonaler Kovarianzmatrizen spricht, ist die Tatsache, daß viele Dimensionen des Merkmalsraumes in der Regel gar nicht nennenswert miteinander korreliert sind, insbesondere dann, wenn Vorverarbeitungsmethoden angewendet werden, die als Seiteneffekt die Dekorrelation des Merkmalsraumes haben, wie zum Beispiel die lineare Diskriminanzanalyse (LDA).

Schließlich bleibt noch zu erwähnen, daß zu wenige Trainingsdaten für volle Kovarianzmatrizen leicht dazu führen können, daß Beinahe-Singularitäten entstehen, das heißt, daß die Form der Gaußglocke einer Normalverteilung in einer bestimmten Richtung so spitz ist, daß die berechnete Wahrscheinlichkeit für Merkmalsvektoren, die in die Mitte der Gaußglocke fallen, extrem hoch ausfällt, während weiter entfernt liegenden Vektoren eine extrem niedrige Wahrscheinlichkeit zugewiesen wird. Wenn so eine beinahe-singuläre Kovarianzmatrix in dem Codebuch eines bestimmten Modells vorkommt, und ein Merkmalsvektor eines anderen Modells befindet sich „zufällig" innerhalb dieser Beinahe-Singularität, so wird ihm eine vermutlich wesentlich überhöhte Wahrscheinlichkeit zugewiesen, im ungünstigen Fall sogar eine wesentlich höhere als diejenige, die ihm vom Codebuch seines eigenen Modells zugewiesen würde.

Wenn man Viterbi-Pfade analysiert und sich diejenigen genauer ansieht, die eine verhältnismäßig kleine Gesamtwahrscheinlichkeit haben, dann findet man oft eine Pfadform, wie sie in Abbildung 7.2 dargestellt ist. Der durch ausgefüllte Punkte dargestellte Pfad sei der „korrekte", bestimmt zum Beispiel durch einen besseren Erkenner. Der von einem schlechteren Erkenner gefundene Pfad ist durch die nicht ausgefüllten Markierungen dargestellt. Der Grund für die Abweichung des schlechten Pfades liegt an dem mit dem Quadrat markierten Punkt. Man sieht, daß der Pfad davor sehr steil nach oben steigt, das heißt, er überspringt viele HMM Zustände in kurzer Zeit, um an dem markierten Punkt anzukommen. Danach bleibt er lange Zeit im erreichten Zustand, um schließlich wieder mit dem korrekten Pfad zu verschmelzen. In diesem Fall war die lokale Wahrscheinlichkeit des quadratisch markierten Punktes so groß, daß dieser unter allen Umständen in den Pfad mit aufgenommen werden mußte. Wie Überprüfungen in solchen Fällen ergeben haben, ist oft eine Beinahe-Singularität in einer Kovarianzmatrix des zugehörigen Modells dafür verantwortlich.

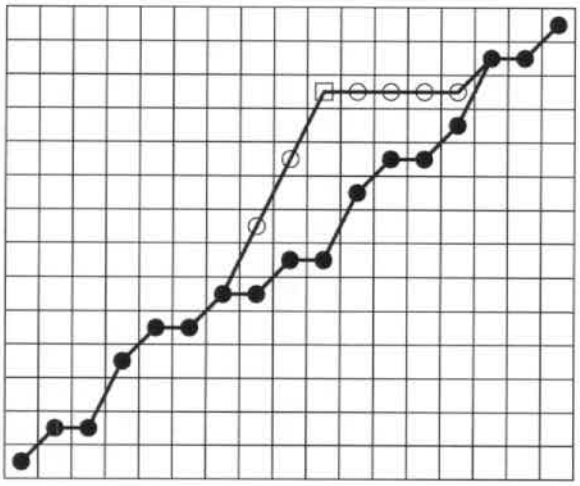


Abbildung 7.2: Ein typischer schlechter Viterbi-Pfad

Bei der Verwendung von diagonalen Kovarianzmatrizen sinkt die Wahrscheinlichkeit des Auftretens von Beinahe-Singularitäten für Modelle mit wenigen Trainingsdaten, weil alle kleinen Eigenwerte, deren Eigenvektoren nicht nahezu parallel zu einer Raumachse liegen, auf alle Achsen „verteilt" werden. So ist z.B. die volle Kovarianzmatrix der Menge $\{\binom{0}{0}, \binom{1}{1}\}$ singulär, die diagonale aber nicht.

Aber selbst diagonale Kovarianzmatrizen können mehr Parameter als nötig enthalten. Oft ist es möglich, ohne signifikante Leistungseinbußen statt voller oder diagonaler Kovarianzmatrizen einfach die Einheitsmatrix zu verwenden. Einen Kompromiß zwischen diagonalen Matrizen und der Einheitsmatrix stellen radiale Matrizen dar, deren Diagonalelemente alle denselben Wert r haben, wobei die Nichtdiagonalelemente alle Null sind.

Es liegt nun nahe, für verschiedene Codebuchvektoren auch verschiedene Kovarianzmatrixtypen zu verwenden. Da eine volle Matrix mehr Parameter hat, ist anzunehmen, daß sie – vorausgesetzt, sie kann hinreichend gut

geschätzt werden – die Daten besser modelliert als eine diagonale. Wenn also für einen Codebuchvektor sehr viele Trainingsdaten zur Verfügung stehen, dann kann es vorteilhaft sein, für diesen Vektor eine volle Matrix zu verwenden. Umgekehrt kann es sinnvoll sein, für einen Vektor, der nur sehr wenige Trainingsdaten hat, eine Matrix mit wenigen Parametern, z.B. eine radiale Matrix zu verwenden.

7.3.1 Vereinfachung von Kovarianztypen

Die Verwendung von diagonalen statt vollen bzw. radialen statt diagonalen Kovarianzmatrizen führt nicht nur zu einer Verkleinerung des Parameterraumes, sondern auch zu einer Beschleunigung der Berechnung der Emissionswahrscheinlichkeiten. Die Mahalanobis-Distanz zwischen x und μ mit einer vollen Kovarianzmatrix $\Sigma^{-1} = (1/\sigma_{i,j})$ berechnet sich:

$$\begin{aligned} M &= (x-\mu)^T \Sigma^{-1} (x-\mu) \\ &= \sum_{i=1}^{d} y_i \cdot \sum_{j=1}^{d} (y_j \cdot 1/\sigma_{i,j}) \qquad \text{wobei} \quad y_i = x_i - \mu_i \end{aligned}$$

Dabei müssen – vorausgesetzt, die Invertierung von Σ wurde bereits berechnet – $O(d^2)$ Multiplikationen und $O(d)$ Additionen und Subtraktionen durchgeführt werden.

Für eine diagonale Kovarianzmatrix $\Sigma^{-1} = \begin{pmatrix} 1/\sigma_1 & & \\ & \ddots & \\ & & 1/\sigma_d \end{pmatrix}$ ergibt sich:

$$\begin{aligned} M &= (x-\mu)^T \Sigma^{-1} (x-\mu) \\ &= \sum_{i=1}^{d} y_i^2 \cdot 1/\sigma_i \end{aligned}$$

Hier müssen nur $2d$ Multiplikationen und $O(d)$ Additionen und Subtraktionen gerechnet werden.

Bei einer radialen Kovarianzmatrix $\Sigma^{-1} = \begin{pmatrix} 1/\sigma & & \\ & \ddots & \\ & & 1/\sigma \end{pmatrix}$ ergibt sich:

$$\begin{aligned} M &= (x-\mu)^T \Sigma^{-1} (x-\mu) \\ &= 1/\sigma \cdot \sum_{i=1}^{d} y_i^2 \end{aligned}$$

also neben $O(d)$ Additionen und Subtraktionen, nur noch $d+1$ Multiplikationen.

Beim Umwandeln von Kovarianzmatrizen eines Typs in einen anderen wurde stets darauf geachtet, daß die resultierende Determinante gleich der ursprünglichen Determinanten ist.

Wenn eine diagonale Kovarianzmatrix $M_D = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{pmatrix}$ in eine radiale $M_R = \begin{pmatrix} \sigma & & \\ & \ddots & \\ & & \sigma \end{pmatrix}$ umgewandelt wird, dann wird σ definiert als $\sqrt[d]{\prod_i \sigma_i}$.

Dann gilt $|M_D| = |M_R|$. Veranschaulicht bedeutet das, daß der Ellipsoid, der die einfache Kovarianz um den Mittelwert darstellt, umgewandelt wird in eine Kugel mit gleichem Volumen.

7.3.2 Selektive Radialisierung

Motivation

Der Begriff Radialisierung bezeichne die Umwandlung von diagonalen Kovarianzmatrizen in radiale, entsprechend der obigen Definition. Ausgehend davon, daß Fließkommamultiplikationen den größten Teil des Aufwands zur Berechnung einer multivariaten Normalverteilung darstellen, lohnt es sich, deren Anzahl zu minimieren. Die Multiplikation mit den Diagonalwerten der inversen Kovarianzmatrizen stellen auch ca. ein Drittel aller Fließkommaoperationen. Das heißt, daß durch Radialisierung im günstigsten Fall eine Zeitersparnis von etwa 30% erwartet werden kann. In der Praxis fällt diese Zeitersparnis kleiner aus, da die heutigen Rechnerarchitekturen komplizierte Fließkommaprozessoren besitzen, die mehrere Operationen gleichzeitig bzw. im sogenannten „Pipeline-Modus" durchführen können, was bedeutet, daß durch Vermeidung von Rechenschritten vor allem eine geringere Auslastung der Fließkommaprozessoren, aber nur eine kleine Zeitersparnis erreicht wird.

Es lohnt sich aber auch aus anderen Gründen, so viele Kovarianzmatrizen wie möglich zu radialisieren. Ein Spracherkenner mit 5 000 Codebüchern zu je 32 48-dimensionalen Normalverteilungen benötigt für seine insgesamt 160 000 Mittelwertsvektoren 7.68 Millionen Parameter. Die gleiche Menge wird noch einmal für die zugehörigen diagonalen Kovarianzmatrizen benötigt. Wenn nun jede Matrix mit einem einzigen statt 48 Parametern darstellbar ist, dann reduziert sich der Gesamtparameterraum von 15.36 Millionen Parametern um 49% auf nur noch 7.84 Millionen.

Durch die Verwendung von radialen Kovarianzmatrizen geht natürlich Information verloren. Dafür kann aber angenommen werden, daß zumindest durch die Radialisierung der weniger gut geschätzten Kovarianzmatrizen die Generalisierungsfähigkeit des Erkenners steigt. Während der Informationsgehalt mit der Anzahl der radialisierten Matrizen sinkt, steigt die Generalisierungsfähigkeit. Diese beiden auf die Erkennungsleistung entgegengesetzt wirkenden Einflüsse könnten bei einer bestimmten Zahl von radialisierten Matrizen eine optimale Erkennungsleistung liefern, die sogar über der Leistung des unveränderten, nur diagonale Matrizen verwendenden Erkenners liegt.

Experimente

Ausgehend von einem Erkenner, dessen Fehlerrate bei 8.8% lag, wurden unterschiedliche Mengen an Kovarianzmatrizen radialisiert. Die Abbildung 7.3 zeigt die Fehlerraten für verschiedene Kompaktifizierungsgrade. Dabei wurden alle Kovarianzmatrizen der Codebücher mit den wenigsten Trainingsdaten radialisiert. Man sieht, daß der Fehler durch geeignete Wahl der zu radialisierenden Matrizen von 8.8% auf 8.1% reduziert werden kann. Das ist eine relative Fehlerreduktion von 8%. Eine Radialisierung aller Kovarianzmatrizen führt zu einer moderaten Erhöhung der Fehlerrate um etwa 1%, bei gleichzeitiger Nahezu-Halbierung des Parameterraumes und einer Einsparung von fast einem Drittel aller Fließkommaoperationen zur Berechnung der Emissionswahrscheinlichkeiten.

7.4 Kopplung von Kovarianzparametern

Neben der Verwendung von Kovarianzmatrizen mit wenigen Parametern bietet sich eine weitere Möglichkeit an, Parameter einzusparen, nämlich die

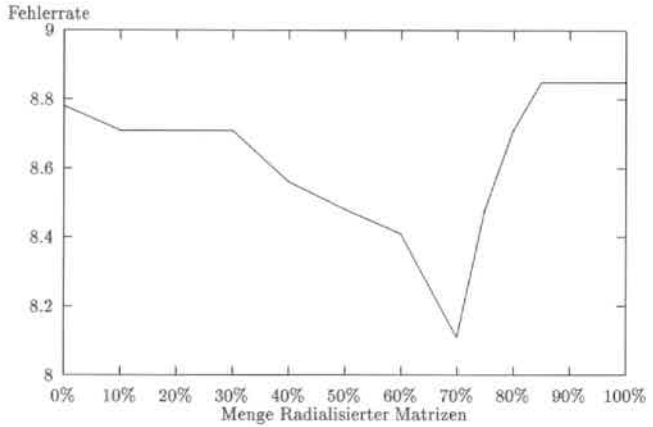


Abbildung 7.3: Fehlerrate nach Radialisierung

Kopplung von Kovarianzmatrizen verschiedener Vektoren eines Codebuchs. Auf den ersten Blick stellt sich natürlich die Frage, weshalb ein Zusammenhang zwischen den Kovarianzen verschiedener Vektoren bestehen soll. In der Praxis stellt sich aber heraus, daß es oft gar nicht nötig ist, wirklich für jeden Vektor eine eigene Matrix zu verwenden. Dadurch, daß mehrere Vektoren sich eine Matrix teilen, wird diese zwar nicht mehr so spezifisch ausfallen, ihre Determinante wird wachsen, aber dafür kann sie dann auch zuverlässiger geschätzt werden und kann eine bessere Generalisierungsfähigkeit besitzen.

Im folgenden werden Experimente vorgestellt, bei denen verschiedene Grade der Kopplung bis hin zur Nahezuhalbierung aller akustischen Parameter ausgewertet wurden.

Distanzmaß

Um zu entscheiden, ob eine gemeinsame Modellierung zweier Kovarianzmatrizen erfolgversprechend ist, wird ein Distanzmaß benötigt, so daß Matrizen, die sehr verschieden sind, seltener und einander sehr ähnliche Matrizen öfter zusammengeballt werden. Die hier beschriebenen Ballungsexperimente wurden alle mit rein diagonalen Kovarianzmatrizen durchgeführt. Als Distanz-

Tabelle 7.3: Fehlerraten bei Kovarianzballung

weniger Matrizen	0	0.625%	1.25%	2.5%	5%	10%	20%
Fehlerate	8.8%	8.9%	8.9%	8.9%	9.7%	11.2%	20.2%

maße wurde die einfache euklidische Distanz verwendet. Sie ist einfach die euklidische Distanz der als Vektor betrachteten Diagonalelemente.

7.4.1 Ballung der Kovarianzmatrizen

Zur Ballung der Kovarianzmatrizen wurde der folgende einfache agglomerative Algorithmus verwendet:

1. bestimme für jedes Codebuch s alle Distanzen $D(s,i,j)$ zwischen den Kovarianzmatrizen $\Sigma_s(i)$ und $\Sigma_s(j)$
2. für die n kleinsten Distanzen:
   kopple die zugehörigen Kovarianzmatrizen

Dabei wurde die Wirkung des Wertes n auf die Erkennungsleistung ausgewertet. Da bei der akustischen Modellierung mit Mixturen von multivariaten Normalverteilungen der Parameterraum im wesentlichen aus den Mittelwertsvektoren und den Kovarianzmatrizen besteht, nehmen diagonale Kovarianzmatrizen also etwa die Hälfte der gesamten Parameter ein. Damit wird die maximal erreichbare Einsparung an Parametern festgelegt.

Man sieht aus Tabelle 7.3, daß durch Ballung von Kovarianzmatrizen mit euklidischer Distanz keine sinnvolle Kompaktifizierung des Parameterraumes möglich ist. Schon ab einer Beseitigung von nur 5% der Matrizen stellt sich eine nicht mehr tolerable Steigerung der Fehlerrate ein. An dieser Stelle bleibt die Frage offen, ob durch ein anderes Distanzmaß, oder ein spezielles Nachtrainieren des reduzierten Systems ein besseres Ergebnis erreichbar ist.

7.5 Zusammenfassung

In diesem Kapitel wurde gezeigt, daß es möglich ist, einen bereits trainierten Erkenner mit großerm Parameterraum so zu verkleinern, daß ca. die Hälfte

aller Parameter eingespart werden, ohne daß dies zu Lasten der Erkennungsleistung geht.

Weniger erfolgreich verliefen Experimente, die zum Ziel hatten, die Anzahl der Referenzvektoren zu reduzieren. Keines der drei untersuchten Auswahlkriterien für zu entfernende Vektoren führte zu einer signifikanten Verkleinerung des Parameterraumes ohne eine gleichzeitige deutliche Erhöhung der Fehlerrate.

Erfolgreicher war Idee der Radialisierung der Kovarianzmatrizen. Durch die Radialisierung bleibt die Determinante und somit das Volumen des durch die Kovarianzmatrix beschriebenen Ellipsoiden unverändert. Allerdings genügt ein einziger Parameter je Kovarianzmatrix, und die Zeit zur Berechnung der HMM-Emissionswahrscheinlichkeiten kann zusätzlich reduziert werden.

Durch die Radialisierung von 70% aller Kovarianzmatrizen konnte die Fehlerrate von 8.8% auf 8.1% gesenkt werden und der Parameterraum um ca. 35% verkleinert werden.

Kapitel 8

Entwicklung des JANUS-Diktiersystems

8.1 Diktieren mit großem Vokabular

8.1.1 Diktieren als spezielle Sprechart

In der Spracherkennung wird immer wieder festgestellt, daß die Fehlerrate beim Wechsel der Sprechweise oder der Aufnahmebedingungen dramatisch ansteigt. Es ist leicht einzusehen, daß ein Erkenner, der mit Sprache trainiert wurde, die fehlerfrei in einer geräuscharmen Umgebung diktiert wurde (sog. „clean speech"), auch am besten auf dieser Art Sprache funktioniert. Wenn derselbe Erkenner mit Telefonaufnahmen oder mit Aufnahmen von spontaner Sprache konfrontiert wird, dann steigt die Fehlerrate nicht selten um ein Vielfaches [JSD+96]. Seit einigen Jahren wird in der Spracherkennung vermehrt auf dem Gebiet der Robustheit gegenüber Wechseln der Aufnahmeart geforscht [Wei96]. Bei den öffentlichen internationalen (D)ARPA Evaluationen ist man von der Auswertung reiner Diktiersysteme übergegangen zur Spontansprache über Telefon und zur Erkennung kompletter Rundfunksendungen. Dieser Übergang geschah leider schon zu einer Zeit, zu der bekannt war, daß die Fehlerrate der maschinellen Spracherkennung selbst auf diktierter Sprache immer noch um eine Größenordnung über der von Menschen lag.

Es gibt zwei allgemein anerkannte Herangehensweisen an das Problem der unterschiedlichen Spracharten. Die eine besteht darin, spezialisierte

Erkenner für jede einzelne Sprachart oder eine Klasse von Spracharten zu entwickeln und während der Erkennung zu entscheiden, um was für eine Sprachart es sich handelt. Die andere versucht, einen generischen Erkenner auf die gerade vorliegende Sprachart zu adaptieren. Beide Methoden lassen sich auch kombinieren. Trotz der Bemühungen, mit einem System verschiedene Spracharten zu beherrschen, gilt aber immer noch die Feststellung, daß ein spezialisierter Erkenner auf seinem eigenen Gebiet in der Regel deutlich besser arbeitet als ein adaptierter generischer Erkenner. Wenn man bedenkt, daß die Fehlerraten der spezialisierten Erkenner immer noch weit entfernt sind von dem, was auf lange Sicht angestrebt wird, dann erkennt man die Bedeutung der Forschung auf dem Gebiet der Diktiersysteme. Es ist auch in vielen Fällen davon auszugehen, daß Erkenntnisse, die zur Verbesserung von Diktiersystemen führen, auch der Verbesserung der Erkenner für andere Spracharten dienen können.

8.1.2 Unbeschränktes Vokabular

Wenn hier die Rede von unbeschränktem Vokabular ist, dann bedeutet das, daß keine Aussage über die Größe des Vokabulars der zu erkennenden Sprache getroffen wird. Praktisch ist das erkennbare Vokabular allein schon implementationstechnisch begrenzt. In der vorliegenden Arbeit war es nicht in vertretbarer Zeit möglich, mehr als 2^{16} Wörter im Vokabular zu verwenden. Ein derartig großes Vokabular deckt aber ca. 99.7% der Wörter der Testmenge ab, so daß nur ca. 0.3% aller zu erkennenden Wörter der DARPA Evaluationsmenge von 1994 nicht erkannt werden kann. Verfahren zum Erkennen von Wörtern außerhalb des Erkennervokabulars werden zwar entwickelt [SWW93] [KJ96], haben sich aber noch nicht etabliert.

8.2 Ältere Arbeiten des Autors

In diesem Abschnitt werden Arbeiten des Autors auf dem Gebiet der Spracherkennung angesprochen, die der Entwicklung des neuesten Diktiersystems vorausgingen. Die Ergebnisse dieser Arbeiten sind nicht Bestandteil des Diktiersystems, haben aber dennoch wertvolle Informationen geliefert.

8.2.1 Alternative Emissionswahrscheinlichkeiten

Ende der achtziger und Anfang der neunziger Jahre hatten die HMM-Erkenner, deren Emissionswahrscheinlichkeiten mit Gauß-Mischverteilungen berechnet werden, noch nicht diese dominante Stellung wie heute. Verschiedene andere Ansätze [KBC88] [TW90], mit denen auch der Autor gearbeitet hat, wurden verfolgt und im JANUS-Erkenner eingesetzt [ST92] [Sch91]. Mit der Einführung der regelmäßigen DARPA-Evaluationen und der dadurch beeinflußten Finanzierung der Forschung wurden viele Forschungsanstalten gezwungen, auf den Zug mit der erfolgversprechendsten Methode aufzuspringen. Verschiedene alternative Spracherkennungsmethoden wie Expertensysteme [Kla77], selbstorgansierende Karten [Koh90] oder neuronale Netze [TWPS91] wurden wegen ihrer damaligen geringeren Leistungsfähigkeit nicht weiter verfolgt. Von den 15 Teilnehmern an der ARPA Evaluation vom November 1994 benutzten 13 Teilnehmer Hidden-Markov Modelle mit Berechnung der Emissionswahrscheinlichkeiten durch Gauß-Mischverteilungen, ein Teilnehmer berechnete diese mit Laplace-Mischverteilungen [ADNS94], und nur ein einziger Teilnehmer von der Cambridge University in England verwendete rekurrente neuronale Netze (hybrider Erkenner) [HCR+95]. Dieser hybride Erkenner profitierte von der Segmentierung der Trainingsdaten, die vom auch aus Cambridge stammenden Evaluationssieger (HTK) [WGPV96] berechnet wurden. Trotz der hervorragenden Erkennungsleistung des hybriden Erkenners bei dieser und bei vorangegangenen DARPA-Evaluationen lieferten stets reine HMM-Erkenner die besten Ergebnisse.

Linked Predictive Neural Networks (LPNNs)

Die erste JANUS-Version verwendete LPNNs [TWPS91] zur Berechnung der Emissionswahrscheinlichkeiten. Ein LPNN ist ein mehrschichtiges Perzeptron, das als Eingabe zwei Sprachvektoren x_{t-1} und x_{t+1} zu den Zeitpunkten $t-1$ und $t+1$ als Eingabe erhält und in seiner Ausgabe den Vektor x_t vorhersagen muß. Die gewichtete Distanz der Vorhersage zu x_t wurde dann als Approximation für $-\log p(x|s)$ betrachtet und im Viterbi-Algorithmus statt dieser verwendet. Mit dieser Methode wurden bis Anfang 1992 die meisten JANUS-Experimente durchgeführt und vom Autor der erste deutschsprache JANUS-Erkenner entwickelt. Dieser Erkenner erzielte auf sprecherabhängiger kontinuierlicher Sprache mit einer niedrigen Perplexität von ca. 7 eine Fehlerrate von etwa 7%. Das Erkennungsvokabular bestand aus 400 Wörtern.

Die Fehlerrate des LPNN-Erkenners auf sprecherunabhängiger Sprache war trotz Trainings auf derselben um ein vielfaches höher. Ebenso anfällig zeigten sich die LPNNs auf eine Erhöhung der Perplexität. Bei einer Perplexität von 111 stieg die Fehlerrate des sprecherabhängigen Systems auf ca. 40%.

Learning Vector Quantization (LVQ)

Während eines einjährigen Forschungsaufenthaltes an der Carnegie Mellon University entwickelte der Autor einen auf LVQ [KBC88] basierenden Erkenner, der wesentlich bessere Leistungen auf sprecherunabhängiger und perplexerer Sprache erzielte. LVQ ist ein Verfahren zur optimierten Schätzung von Referenzvektoren. Dabei wird für jeden Trainingsvektor x bestimmt, welcher Referenzvektor y ihm korrekterweise zugeordnet wird. Dieser Referenzvektor wird dann im Raum so verschoben, daß er näher an x zu liegen kommt. Das LVQ-2 Verfahren ist eine Verfeinerung des LVQ Verfahrens. Das Training mit LVQ-2 funktioniert wie folgt:

Der Parameterraum des Erkenners besteht aus je einem Codebuch von Referenzvektoren für jede Klasse. Eine Klasse entspricht einem akustischen Modell (Subphonem). Ein Trainingsvektor x gehöre zur Klasse C_1 (entweder manuell oder mittels Viterbi-Algorithmus zugeordnet). Von allen Referenzvektoren des Systems ist der Abstand zu x für den Vektor y_2 aus dem Codebuch der Klasse C_2 am kleinsten, insbesondere ist er kleiner als der Abstand zum Vektor y_1, der von allen Vektoren des Codebuchs von C_1 der x am nächsten liegende ist. Wenn nun zusätzlich die Bedingung

$$||x - \frac{y_1 + y_2}{2}|| < w \tag{8.1}$$

erfüllt ist, also der Vektor x innerhalb eines Fensters der Breite $2w$ um die Mitte zwischen y_1 und y_2 liegt, dann werden y_1 und y_2 aktualisiert gemäß:

$$y_1 \rightarrow y_1 - \alpha(t)(x - y_1) \quad \text{und} \tag{8.2}$$
$$y_2 \rightarrow y_2 + \alpha(t)(x - y_2) \tag{8.3}$$

Das heißt, y_1 wird um das $\alpha(t)$-fache des Abstands zu x in die Richtung von x bewegt. Entsprechend wird y_2 von x weg bewegt. Der Wert von $\alpha(t)$ ändert sich mit der Zeit gemäß:

$$\alpha(t) = \alpha(0) \cdot (1 - \frac{t}{T}) + \alpha_f \tag{8.4}$$

Damit wird am Anfang des Trainings zum Zeitpunkt $t = 0$ die größte Schrittweite $\alpha(0) + \alpha_f$ gemacht, und am Ende $t = T$ die kleinste Schrittweite α_f, dazwischen nimmt sie linear ab. Durch die Abnahme der Schrittweite wird ein „Abkühlungsverfahren" (Simulated Annealing) verwendet, das dafür sorgt, daß die Systemparameter sich gegen Ende des Trainings stabilisieren.

Dieser LVQ-Erkenner wurde vom Autor um die Fähigkeit der Modellierung kontextabhängiger Modelle erweitert. Die Fehlerrate auf sprecherunabhängiger Sprache war schließlich mit 6.8% bei einer Perplexität von 7 vergleichbar gut wie der LPNN-Erkenner auf sprecherabhängiger Sprache. Die Fehlerrate des LVQ-Erkenners auf sprecherabhängiger Sprache lag bei ca. 1%.

8.2.2 Zustandsabhängige Stromgewichte

JANUS erlaubt die Verwendung mehrerer Datenströme. In der Regel sind verschiedene Datenströme Berechnungsquellen für Emissionswahrscheinlichkeiten, die auf verschiedenen Vorverarbeitungen basieren. Aber auch verschiedene Berechnungsarten (Gauß-Mischverteilungen, Neuronale Netze) können in verschiedenen Datenströmen angelegt sein. Wenn verschiedene Datenströme verwendet werden, dann wird die Gesamtemissionswahrscheinlichkeit durch das exponentiell gewichtete Produkt der einzelnen Emissionswahrscheinlichkeiten definiert:

$$p(x|s) = \prod_{i=1}^{m} \left(\sum_{k=1}^{n_s} \gamma_{s,i}(k) \cdot N(x_i, \mu_{s,i,k}, \Sigma_{s,i,k}) \right)^{\alpha_i} \tag{8.5}$$

In Gleichung 8.5 wurde der Merkmalsvektor x_i mit dem Datenstrom indiziert, da die verschiedenen Datenströme über verschiedenen Merkmalsräumen definiert sein können.

Die exponentielle Gewichtung kann notwendig sein, wenn die Varianzen der berechneten Emissionswahrscheinlichkeiten der einzelnen Datenströme wesentlich voneinander abweichen. Durch die Gewichtung können solche

Abweichungen zum Teil kompensiert werden. Auch der HTK-Spracherkenner [YW93] der Universität Cambridge, der bei den letzten DARPA Evaluationen für Diktiersysteme die besten Ergebnisse liefert, bietet die Möglichkeit der Verwendung mehrerer Datenströme mit eigenem Gewicht an.

Abgesehen von der Verwendung verschiedener Berechnungsarten stellt sich prinzipiell die Frage, warum es von Vorteil ist, mehrere Datenströme einzusetzen, und nicht statt dessen einen Datenstrom zu verwenden, dessen Merkmalsraum aus der Vereinigung der verschiedenen Merkmalsräume besteht. Eine Zusammenlegung verschiedener Merkmalsräume hat den Vorteil, daß Korrelationen zwischen bestimmten Dimensionen der verschiedenen Räume ausgenutzt werden können. Selbst wenn die Kovarianzmatrizen der Gauß-Mischverteilungen diagonal sind, so fließt doch Korrelationsinformation schon durch die Quantisierung in eine diskrete Anzahl von Mittelwertsvektoren ein. Bei einer Aufteilung in mehrere Datenströme kann die Korrelation über die Grenzen der Datenströme nicht mehr genutzt werden. Dafür wird aber, insbesondere bei semikontinuierlichen HMM-Varianten, der Parameterraum durch Einführen zusätzlicher Datenströme und somit zusätzlicher Mixturgewichte signifikant vergrößert und kann mehr Information aufnehmen. Ein weiterer Vorteil mehrerer und dafür niedriger dimensionaler Merkmalsräume besteht darin, daß die kleineren Räume durch die Systemparameter „dichter" abgedeckt werden, die Mahalanobis Distanzen werden kleiner und dominieren die Mixturgewichte nicht mehr so stark. Für eine optimale Entwurfsentscheidung müßte also klar sein, ob der Vorteil durch Ausnutzung von Korrelationen zwischen verschiedenen Merkmalsteilräumen oder der Vorteil der Erweiterung des Parameterraumes überwiegt.

Es gibt allerdings noch einen weiteren Vorteil der Verwendung mehrerer Datenströme, nämlich die vom Autor in [RW94] veröffentlichte Möglichkeit der Schätzung zustandsabhängiger Stromgewichte. Dabei werden die einzelnen Emissionswahrscheinlichkeiten der Datenströme nicht mit α_i, sondern mit $\alpha_i(s)$ in Abhängigkeit vom aktuellen HMM-Zustand gewichtet. Diese Idee basiert auf der Erwartung, daß für die Bewertung bestimmter HMM-Zustände unterschiedliche Bereiche des Merkmalsraumes unterschiedlich wichtig sind. In [RW94] wird beispielsweise festgestellt, daß bei einer Aufteilung des Merkmalsraumes in einen, der nur Spektralkoeffizienten enthält, und einen, der deren zeitliche Ableitung (Delta-Koeffizienten) enthält, die Deltakoeffizienten zur Bewertung des letzten von je drei HMM-Zuständen

eines Phonems signifikant besser geeignet sind.

Die Einstellung der Stromgewichte $\alpha_i(s)$ wird wie folgt durchgeführt: Zu Beginn des Trainings werden alle $\alpha_i(s)$ unabhängig von i oder s auf denselben Wert $1/m$ gesetzt, so daß die Randbedingung

$$\sum_{i=1}^{m} \alpha_i(s) = 1 \quad \forall s \qquad \text{und} \qquad 0 \leq \alpha_i(s) \leq 1 \quad \forall i, s \tag{8.6}$$

erfüllt ist. Diese Randbedingung muß auch während des Trainings aufrechterhalten werden, damit das resultierende exponentiell gewichtete Produkt nicht beliebige Werte annehmen kann. Denn das Ziel der Optimierung der Stromgewichte ist ja die Maximierung der Wahrscheinlichkeit der Beobachtung der Trainingsdaten. Wenn als Gesamtemissionswahrscheinlichkeit das gewichtete Produkt der Einzelwahrscheinlichkeiten benutzt wird, dann könnte dieser Wert beliebig groß oder beliebig klein werden.

Die Beobachtungswahrscheinlichkeit der Trainingsdaten $x_1 \ldots x_T$ berechnet sich für einen gegebenen Viterbipfad $l_1 \ldots l_T$, der jedem x_t einen HMM-Zustand s_{l_t} zuordnet, als:

$$p(x_1 \ldots x_T | s_{l_1} \ldots s_{l_T}) = \prod_{t=1}^{T} p(x_t | s_{l_t}) =$$
$$\prod_{t=1}^{T} \prod_{i=1}^{m} \left(\sum_k \gamma_{s_{l_t},i}(k) \cdot N(x_i, \mu_{s_{l_t},i,k}, \Sigma_{s_{l_t},i,k}) \right)^{\alpha_i(s_{l_t})}$$

Bei einem reinen Maximum Likelihood Training würde es genügen, jetzt die $\alpha_i(s)$ so zu verändern, daß $p(x_1 \ldots x_T | s_{l_1} \ldots s_{l_T})$ vergrößert wird. In [RW94] wird jedoch ein diskriminativer Ansatz verfolgt, der zum Ziel hat, $p(x_t|s_{l_t})$ zu vergrößern, und zudem $p(x_t|s_q)$ zu verkleinern, wobei $s_q = \text{argmax}_s\, p(x_t|s)$ der Zustand ist, der die größte Emissionswahrscheinlichkeit zum Zeitpunkt t hat. Falls $s_q = s_{l_t}$ ist, findet kein Training statt, da der korrekte Zustand auch die höchste Wahrscheinlichkeit hat. Eine Veränderung der Stromgewichte ist in diesem Fall nicht zweckmäßig.

Die Regel zur Optimierung von $\alpha_i(s)$ besteht darin, $\alpha_i(s_{l_t})$ für alle i um eine Schrittweite ϵ in die Richtung von $\frac{\mathrm{d}p(x_t|s_{l_t})}{\mathrm{d}\alpha_i(s_{l_t})}$ eingeschränkt auf den

Tabelle 8.1: Verbesserung durch zustandsabhängige Stromgewichte

Datenbasis	ursprüngliche Fehlerrate	trainierte Stromgewichte	relative Verbesserung
CR	6.8%	3.6%	47%
RM	9.7%	7.8%	20%

Unterraum $\sum_i \alpha_i(s_{l_t}) = 1$ zu bewegen. Entsprechend werden alle $\alpha_i(s_q)$ in die andere Richtung ihres entsprechenden Gradienten bewegt.

Zwei Sprachdatenbasen, die „Conference Registration Task" (CR) und die „Resource Management Task" (RM) wurden verwendet, um den Effekt des Trainings von zustandsabhängigen Stromgewichten zu evaluieren. Dabei wurden zwei Datenströme (Spektren und Delta-Spektren) berechnet. Die Verbesserungen in der Fehlerrate gegenüber dem Ausgangssystem, bei dem alle $\alpha_1(s) = 0.7$ und alle $\alpha_2(s) = 0.3$ waren, zeigt Tabelle 8.1. Die Werte 0.7 und 0.3 wurden durch Testen verschiedener Werte als die besten zustandsunabhängigen Stromgewichte gefunden.

8.3 Komponenten des Erkenners

8.3.1 Die Signalvorverarbeitung

Das Sprachsignal liegt unmittelbar nach der Aufnahme so vor, wie es der Analog-nach-Digital-Wandler liefert. Ein typischer Wandler tastet das Signal mit 16 kHz ab und verwendet dabei eine Auflösung von 16 Bit, so wurden auch alle Aufnahmen der Wall Street Journal Datenbasis gemacht. Für Aufnahmen von Telefonsprache wird oft auch mit 8 kHz abgetastet und eine Auflösung von nur 8 Bit verwendet, weil dort die technischen Umstände (Bandbegrenzung, Rauschen) eine höhere Aufnahmequalität nicht erfordern. Für „saubere" Sprache, die in einer geräuscharmen Umgebung mit einem guten Mikrophon aufgenommen wird, lohnt sich die höhere Auflösung und schnellere Abtastung aber doch. So wie das Signal nach der Aufnahme vorliegt, ist es als Eingabemerkmal für einen HMM Spracherkenner aber völlig unbrauchbar, da es zum einen sehr viel unnötige Information enthält und zum anderen die eindimensionale Darstellungsform zur Modellierung mit Mixturen von Normalverteilungen ungeeignet ist. Kein funktionierender

Spracherkenner der Welt verwendet die rohen Aufnahmen, ohne sie vorher auf die eine oder andere Art zu verarbeiten. In der Regel wird dabei das Signal aus dem Zeitbereich in den Frequenzbereich mittels einer diskreten Fourier-Transformation transformiert.

An dieser Stelle sollen nicht die verschiedenen Signalverarbeitungsverfahren diskutiert werden. Lediglich die im JANUS-Diktiersystem verwendete Vorverarbeitung wird vorgestellt. In den letzten Jahren konzentrierte sich die Forschung auf dem Gebiet der Signalanalyse vor allem auf die Robustheit, also die Verarbeitung von Aufnahmen unter widrigen Bedingungen. Insgesamt aber verwenden fast alle Spracherkenner das gleiche Grundprinzip der Transformation des Signals aus dem Zeitbereich in den Frequenzbereich, wo dann nach unterschiedlichen Zwischenschritten alle drei bis 20 Millisekunden ein Merkmalsvektor extrahiert wird [Gon95] [You96].

Versatz-Korrektur

Da viele Analog-Digital-Wandler im Ruhezustand nicht der Wert Null liefern, sondern einen davon teilweise erheblich abweichenden Wert, kann der nullte Fourier-Koeffizient in der folgenden Fourier-Analyse etwas verfälscht werden. Deshalb wird in JANUS der Mittelwert aller Abtastwerte über eine Aufnahme von allen Abtastwerten subtrahiert. Damit wird der Versatz des Wandlers korrigiert.

Fourier-Analyse

Mit einer 256-Punkte diskreten Fourier-Transformation werden 129 Spektralkoeffizienten berechnet. Das dabei verwendete Hamming-Fenster wird um jeweils 10 Millisekunden, das heißt 160 Abtastpunkte, verschoben, so daß je Sekunde 100 Vektoren zu je 129 Spektralkoeffizienten erzeugt werden. Danach folgt das sogenannte „Mel-Scaling" [DM90], bei dem die 129 Koeffizienten auf 16 heruntergerechnet werden, so daß dabei jeweils wenige niedrigfrequente Spektralkoeffizienten oder viele hochfrequente zu einem Mel-Scale-Koeffizienten gemittelt werden. Schließlich wird noch jeder der entstehenden 16 Koeffizienten ersetzt durch seinen Logarithmus.

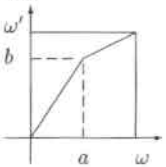
Mittelwertsubtraktion

Die Fourier-Analyse liefert also alle 10 Millisekunden einen 16-koeffizientigen Vektor. Alle Vektoren einer Äußerung werden nun gemittelt, und der gefundene Mittelwertsvektor von allen Vektoren subtrahiert. Dies wird getan, um langfristige Frequenzüberlagerungen, die durch den Aufnahmekanal oder durch andere Störsignale auftreten können, aus dem Signal herauszumitteln. Nach diesem Schritt wird die möglicherweise von Aufnahme zu Aufnahme schwankende Lautstärke so ausgeglichen, daß die Varianzen der einzelnen Koeffizienten auf einen konstanten Wert normiert werden.

Vokaltraktnormierung

Bei der Vokaltraktnormierung [ZW97] handelt es sich um eine Transformation des Spektralraumes, mit dem versucht wird, die Auswirkungen unterschiedlich langer Vokaltrakte zu kompensieren. Üblicherweise haben Frauen einen kürzeren Vokaltrakt als Männer, was dazu führt, daß die Grundfrequenzen der Sprache von Frauen höher liegen als bei Männern. Wenn ein Codebuch mit Referenzvektoren für ein bestimmtes akustisches Modell trainiert werden soll, dann bedeutet das für einen geschlechtsunabhängigen Erkenner, daß sowohl Referenzen für lange als auch für kurze Vokaltrakte im Codebuch enthalten sein müssen. Besser wäre es, wenn für das gleiche akustische Phänomen nur eine Referenz gelernt werden muß, diese kann dann sowohl lange als auch kurze Vokaltrakte modellieren und kann, da sie mit mehr Daten trainiert werden kann, robuster geschätzt werden. Die Vokaltraktnormierung im JANUS-Erkenner besteht in einer Abbildung des Energiespektrums $s(\omega)$ auf $s'(\omega) = s(\omega')$ mit

$$\omega' = \begin{cases} \frac{b\omega}{a} & \text{falls} \quad \omega \leq a \\ \frac{(1-b)\omega}{1-a} + \frac{b-a}{1-a} & \text{falls} \quad \omega > a \end{cases}$$



Der Parameter a wird einmalig und willkürlich für alle Sprecher gleich eingestellt. Der Parameter $b = b(\text{Sprecher})$ wird für jeden Sprecher so eingestellt, daß die Summe der akkumulierten Viterbi-Pfadwahrscheinlichkeiten über alle Viterbipfade des Sprechers maximiert wird. In der Regel ist es notwendig, die Einstellung von b mehrfach zu wiederholen, da die akku-

mulierte Viterbi-Pfadwahrscheinlichkeit von dem b abhängt, das während der Berechnung des Viterbi-Pfades verwendet wurde. Nach mehreren Wiederholungen konvergieren die Werte von b und können dann für den Rest der Erkennerentwicklung unverändert bleiben. Ein Wert von $b > a$ bedeutet, daß die Frequenzen des Sprechers für eine optimale Normierung angehoben werden müssen, und wenn $b < a$ ist, dann werden die Frequenzen abgesenkt. Die gefundenen Einstellungen ergeben deutlich unterschiedliche Durchsnittswerte für Frauen ($b < a$) und Männer ($b > a$).

Lineare Diskriminanzanalyse, LDA

Jeweils sieben aufeinanderfolgende Vektoren werden aneinandergefügt und bilden so einen 112-dimensionalen Vektor, der dann mit einer LDA-Matrix multipliziert wird. Die LDA-Matrix ist eine Matrix, die den Wert $|W| \cdot |T|^{-1}$ minimiert, wobei W die arithmetisch gemittelte Streumatrix aller zuvor definierten Klassen ist, und T die Streumatrix aller Trainingsvektoren ist. Das Ziel der linearen Diskriminanzanalyse ist es also, die durchschnittliche Streuung innerhalb einer Klasse möglichst klein werden zu lassen, während die Gesamtstreuung groß gehalten wird. Dadurch rücken alle Vektoren einer Klasse näher zusammen und die einzelnen Klassen entfernen sich voneinander. Die Klassen werden stets durch die im System zu trainierenden Codebücher definiert, jeweils eine Klasse je Codebuch. Ein Seiteneffekt der LDA ist, daß die Menge der Mustervektoren einer Klasse dekorreliert wird.

Dimensionalitätsreduktion

Da die LDA-Matrix eine 112×112 Matrix ist, liefert die Multiplikation mit ihr 112 Koeffizienten. Dabei sind die Koeffizienten so sortiert, daß der erste Koeffizient die größte Varianz hat und der letzte die kleinste, mit der Folge, daß bei Distanzberechnungen zwischen zwei Vektoren die Koeffizienten niedriger Ordnung stärker ins Gewicht fallen. Das heißt, daß die Koeffizienten höherer Ordnung eher unwichtig sind. Außerdem tragen sie dazu bei, den Parameterraum des akustischen Modells aufzublähen. Deshalb wird nach der LDA eine Dimensionalitätsreduktion durchgeführt, bei der nur die ersten 48 Koeffizienten übernommen werden, alle anderen werden ignoriert.

In einer frühen Entwicklungsphase wurde die Fehlerrate des kontextunabhängigen Erkenners mit verschiedener Anzahl von LDA Koeffizienten

Tabelle 8.2: Auswirkung der Dimensionalitätsreduktion

Anzahl LDA Koeffizienten	8	16	24	32	48	64
gemessene Fehlerrate	55.5	40.1	37.8	35.5	35.2	35.2

getestet. Die Codebücher hatten dabei alle jeweils 32 Referenzvektoren. Tabelle 8.2 faßt die Ergebnisse zusammen.

Man sieht, daß schon bei 32 Koeffizienten eine Erkennungsrate erzielt wird, die durch Verwenden von noch mehr Koeffizienten nicht mehr signifikant gesteigert werden kann. Der Autor hat sich nach diesem Experiment dazu entschlossen, für die folgende Experimente und für den Rest der Entwicklung des Diktiersystems 48 Koeffizienten zu verwenden.

8.3.2 Das Aussprachelexikon

Das im JANUS-Diktiersystem verwendete Aussprachelexikon basiert auf dem von der Carnegie Mellon University (CMU) in Pittsburgh, USA, öffentlich zur Verfügung gestellten Aussprachelexikon. Es enthält alle 60 000 Wörter, die im Vokabular des JANUS-Diktiersystems verwendet werden. Während des Trainings wurde das Aussprachelexikon erweitert um alle Aussprachevarianten aus dem LIMSI Aussprachelexikon. Letzteres ist erhältlich vom LIMSI-Institut in Paris.
Der dabei verwendete Phonemsatz bestand aus den 43 Phonemen, die in Tabelle 8.3 mit Beispielen aufgeführt sind, und je einem Modell für die Stille und für allgemeine nichtsprachliche Geräusche.

Das Aussprachelexikon der CMU ist auf das amerikanische Englisch abgestimmt und ist für britisches Englisch weniger gut geeignet. Die Einträge entsprechen den am häufigsten vorkommenden Aussprachen. So ist als Standardaussprache für das Wort „February" die Phonemfolge **F EH B AX W EH R IY** eingetragen. Die viel seltener vorkommende korrekte Aussprache **F EH B R UW EH R IY** ist allerdings auch als mögliche Aussprachevariante eingetragen.

Phonemmarkierer

Der JANUS-Spracherkenner ermöglicht es, beliebige Markierer für jedes Phonem zu verwenden. Markierer sind binäre Eigenschaften eines Phonems, wie

Tabelle 8.3: Der verwendete Phonemsatz

AA	bottle, car	AE	bank, can	AH	but, duck
AO	four, walk	AW	cow, loud	AX	dual, idol
AXR	fire, ivory	AY	buy, bite	B	but, cab
CH	virtue, church	D	do, bad	DH	the, father
DX	butter, city	EH	get, end	ER	wirk, earth
EY	say, baker	F	fit, gift	G	get, big
HH	head, behave	IH	in, begin	IX	begin, magic
IY	see, meal	JH	jet, digit	K	cat, week
L	let, ball	M	man, sum	N	man, net
NG	sing, think	OW	cold, show	OY	toy, join
P	cap, put	R	run, car	S	set, bus
SH	shop, ratio	T	top, cut	TH	path, thin
UH	book, input	UW	shoe, boot	V	verb, over
W	one, swim	Y	yes, few	Z	does, zoo
ZH	usual, garage				

zum Beispiel die Betonung. Ein Phonem kann betont sein oder auch nicht. Andere Markierer können die Position eines Phonems innerhalb eines Wortes beschreiben, wie zum Beispiel ein Markierer für den Anfang eines Wortes, oder ein Markierer für das letzte Phonem eines Wortes. Auch andere Positionsmarkierer, die Silbengrenzen beschreiben, sind möglich. Verschiedene Spracherkenner verwenden Wortgrenzen- und Silbengrenzenmarkierungen [Hon92] [ST95]. Für das hier beschriebene Diktiersystem wurde lediglich ein Markierer „**WB**" verwendet, der Wortrandphoneme markiert. Das heißt, daß in jedem Wort jeweils das erste und letzte Phonem mit **WB** markiert sind, alle anderen Phoneme haben keine Markierung.

8.3.3 Der Fragenkatalog

Bei der Erzeugung des Entscheidungsbaumes wurde ein Satz von 76 Phonemmengen und einer Markierermenge verwendet. Eine Elementarfrage ist stets die Frage nach der Zugehörigkeit eines Phonems zu einer bestimmten Phonemmenge, oder die Frage nach der Zugehörigkeit eines Markierers zu einer bestimmten Markierermenge. Eine Frage kann sich auf beliebige Kontexte beziehen. Da das hier beschriebene Diktiersystem eine maximale

Kontextbreite von plus-minus zwei verwendet, ergeben sich somit $5 \cdot 76 = 380$ Elementarfragen über Phonemmengen. Zu jeder Frage, die sich auf eine Phonemmenge bezieht, wurde zusätzlich die Frage aufgenommen, ob der befragte Kontext den Wortrandmarkierer besitzt. Zu diesen 760 Fragen kommen noch acht weitere Fragen, die nur Bezug nehmen auf die Markierer im Kontext.

Tabelle 8.3.3 zeigt, welche Phonemmengen verwendet wurden. Die Liste der verwendeten Phonemmengen basiert auf den in [Hon92] [Ode92] vorgestellten.

8.3.4 Erkennung

Zeitsynchrone Mehrpaß-Suche

Für die Erkennung wird neben dem akustischen Modell zusätzlich noch ein Sprachmodell benötigt, und es muß ein Vokabular festgelegt werden. Diese wurden von der DARPA vorgegeben und waren nicht Bestandteil der hier berichteten Forschung. Während der Erkennung wird ein HMM aufgebaut, das für jedes Wort aus dem Erkennungsvokabular eine entsprechende Zustandsfolge enthält und vom letzten Zustand jedes Wortes in den ersten Zustand jedes Wortes einen zusätzlichen Übergang enthält, dessen Wahrscheinlichkeit vom Sprachmodell bestimmt wird. In diesem großen gerichteten Graphen wird dann eine zeitsynchrone Suche durchgeführt, das heißt, es werden mögliche Pfade durch den Graphen hypothetisiert, und die während des Aufbaus der Hypothesen jeweils letzten Zustände sind alle demselben Zeitpunkt zugeordnet.

Unter der **P0**-Bedingung (siehe Abschnitt 4.2) war es erlaubt, ein beliebiges Sprachmodell zu verwenden. Da Sprachmodellierung nicht in den Bereich der akustischen Modellierung fällt, wurden in der vorliegenden Arbeit keine Experimente in dieser Hinsicht durchgeführt. Erwähnt sei aber an dieser Stelle, daß der Evaluationssieger vom November 1994, HTK von der Cambridge University, ein Viergramm Sprachmodell verwendete, wodurch die Fehlerrate gegenüber einem Trigramm Sprachmodell, wie es in JANUS verwendet wird, um relativ 4% gesenkt werden konnte. Ein Viergramm Sprachmodell stand dem Autor für seine Experimente nicht zur Verfügung. Die Erstellung von guten Sprachmodellen ist selbst eine Unterdisziplin der Spracherkennungsfor-

Tabelle 8.4: Die verwendeten Phonemmengen

DEL-REL	CH JH	LATERAL	L
APICAL	T D N DX	HIGH-CONS	K G NG W Y
BACK-CONS	K G NG W	LABIALIZED	R W ER AXR
BILABIAL	P B M W	LABIODENTAL	F V
LABIAL	P B M W F V	INTERDENTAL	TH DH
ALVEOPALATAL	SH ZH CH JH	RETROFLEX	R ER AXR
PALATAL	Y	VELAR	K G NG W
ASPIRATED	HH	PLOSIVE	P B T D K G
FLAP	DX	NASAL	M N NG
AFFRICATE	CH JH	APPROXIMANT	R L Y W
LAB-PL	P B	ALV-PL	T D
VEL-PL	K G	VLS-PL	P T K
VCD-PL	B D G	LAB-FR	F V
DNT-FR	TH DH	ALV-FR	SH ZH
VLS-FR	F TH SH	VCD-FR	V DH ZH
MID-VOW	EH AH AX	LOW-VOW	AA AE AO
FRONT-VOW	IY IH EH AE	CENTRAL-VOW	AH AX IX
BACK-VOW	AA AO UH UW	TENSE-VOW	IY UW AE
ROUND-VOW	AO UH UW	REDUCED-VOW	IX AX
REDUCED-CON	AXR	REDUCED	IX AX AXR
LH-DIP	AY AW	MH-DIP	OY OW EY
BF-DIP	AY OY AW OW	Y-DIP	AY OY EY
W-DIP	AW OW	ROUND-DIP	OY AW OW
LIQUID-GLIDE	L R W Y	W-GLIDE	UW AW OW W
LIQUID	L R	LW	L W
Y-GLIDE	IY AY EY OY Y	LQGL-BACK	L R W
SIBILANT	S SH Z ZH CH JH	ALVEOLAR-RIDGE	T D N S Z L DX
HIGH-VOW	IY IH UH UW IX	LAX-VOW	IH AA EH AH UH
NOISES	GARBAGE	SILENCES	SIL
CONSONANT	P B F V TH DH T D S Z SH ZH CH JH K G HH M N NG R Y W L ER DX AXR		
CONSONANTAL	P B F V TH DH T D S Z SH ZH CH JH K G HH M N NG DX		
OBSTRUENT	P B F V TH DH T D S Z SH ZH CH JH K G		
SONORANT	M N NG R Y W L ER AXR DX		
SYLLABIC	AY OY EY IY AW OW EH IH AO AE AA AH UW UH IX AX ER AXR		
VOWEL	AY OY EY IY AW OW EH IH AO AE AA AH UW UH IX AX		
DIPHTHONG	AY OY EY AW OW		
CARDVOWEL	IY IH EH AE AA AH AO UH UW IX AX		
VOICED	B D G JH V DH Z ZH M N NG W R Y L ER AY OY EY IY AW OW EH IH AO AE AA AH UW UH DX AXR IX AX		
UNVOICED	P F TH T S SH CH K		
CONTINUANT	F TH S SH V DH Z ZH W R Y L ER		
ANTERIOR	P T B D F TH S SH V DH Z ZH M N W Y L DX		
CORONAL	T D CH JH TH S SH DH Z ZH N L R DX		
STRIDENT	CH JH F S SH V Z ZH		
ALVEOLAR	T D N S Z L SH ZH CH JH DX		
STOP	P B T D K G M N NG		
FRICATIVE	F V TH DH S Z SH ZH HH		
ROUND	AO OW UH UW OY AW OW		

schung und hätte den Rahmen dieser Arbeit gesprengt. Außerdem erlaubte die Implementation der Suche keine weiterreichenden Sprachmodelle als Trigramme.

Die Suche erfolgt in mehreren Durchläufen, die an dieser Stelle nicht im Detail erläutert werden sollen.

In der Regel werden während der Entwicklung eines Erkenners auf einer Kreuzvalidierungsmenge Tests durchgeführt, um festzustellen, daß keine Probleme aufgetreten sind, und sich die Erkennungsrate verbessert.

Eine Suche ist allerdings nicht möglich in der Phase vor der Ballung, da die Suche davon profitiert, daß verschiedene HMM-Zustände die gleichen akustischen Modelle verwenden, deren Emissionswahrscheinlichkeiten so nur einmal je Zeiteinheit berechnet werden müssen, und die keinen eigenen Platz im Rechnerspeicher benötigen. Wenn die Suche mit einer nicht geballten Akustik mit mehreren hunderttausend einzelnen Modellen arbeiten muß, dann steigt der Speicher und Rechenaufwand so stark, daß die Kapazität heutiger Arbeitsplatzrechner bei weitem übertroffen wird.

Sprecher- und Signaladaption

Zur Erzielung der besten Ergebnisse wurde eine Adaption der Systemparameter an das Signal der gerade zu erkennenden Äußerung durchgeführt. Die Adaption durch Maximum Likelihood lineare Regression wurde in [LW94] [LW95] in die Spracherkennung eingeführt. Dabei werden die Mittelwerte der Gauß-Mischverteilungen so durch eine affine Abbildung transformiert, daß sie auf das vorliegende Signal optimal passen.

Im Detail heißt das, daß die Emissionswahrscheinlichkeit nicht mehr wie

$$p(x|s) = \sum_{k=1}^{n_s} \gamma_s(k) \cdot N(x, \mu_{s,k}, \Sigma_{s,k}) \tag{8.7}$$

berechnet wird, sondern:

$$p(x|s) = \sum_{k=1}^{n_s} \gamma_s(k) \cdot N(x, A_s\mu_{s,k} + b_s, \Sigma_{s,k}) \tag{8.8}$$

Dabei bezeichnet A_s eine Rotationsmatrix und b_s einen Translationsvektor. In [LW95] wird von einer marginalen Verbesserung berichtet, wenn zusätzlich eine Transformation für die Kovarianzmatrizen berechnet wird. In der vorliegenden Arbeit wurde wegen der geringen Gewinnerwartung und des Mehraufwandes darauf verzichtet. In der Regel wird nicht für jeden Zustand s eine eigene Transformation berechnet, sondern mehrere Zustände teilen sich die gleiche Transformation. Wenn nur wenige Adaptionsdaten vorliegen, das heißt, wenn die gesprochene Äußerung kurz ist, dann wird sogar nur eine einzige Transformation für alle Zustände verwendet.

Die Berechnung der Transformation geschieht nach folgenden Schema:

- führe Erkennung durch, erhalte Hypothese H_1
- berechne $H1$-Viterbipfad $S = (s_{i_1}, s_{i_2}, \ldots s_{i_T})$
- sei $\mu \rightarrow \mu' = A\mu + b$ eine affine Abbildung,
  dann ist der Anteil eines Datenstroms
  an der Wahrscheinlichkeit von H_1
  nach der Adaption: $\prod_{t=1}^{T} p_{A,b}(x_t|s_{i_t}) =$
  $$\prod_{t=1}^{T} \sum_{k=1}^{M} c_{i_t,k} N(x_t, \mu', \Sigma_{i_t,k})$$
- bestimme $(\bar{A}, \bar{b}) = \underset{A,b}{\operatorname{argmax}} \prod_{t=1}^{T} p_{A,b}(x_t|s_{i_t})$
  $\Rightarrow$ lineares Gleichungssystem
- ersetze jeden Codebuchvektor μ durch $\bar{A}\mu + \bar{b}$
- führe Erkennung durch, erhalte Hypothese H_2

Tabelle 8.5: Fehlerraten des Basissystems vom November 1994 auf verschiedenen Entwicklungsmengen

Testmenge	'92	'92	'93	'94	Evaluation '94
Vokabular	5 000	20 000	20 000	20 000	20 000
Fehlerrate	9.4%	13.7%	31.2%	25.2%	22.8%

8.4 Das Evaluationssystem von 1994

Der JANUS-Erkenner, der an der DARPA Evaluation im November 1994 teilnahm, verwendete 2885 verschiedene geballte Subtriphone. Die Akustik wurde mit voll kontinuierlichen HMMs modelliert. Jedes der Subtriphone hatte eine eigene Mixturgewichteverteilung über einem eigenen Codebuch bestehend aus je 16 16-dimensionalen Referenzvektoren mit diagonalen Kovarianzmatrizen.

Die Signalvorverarbeitung bestand aus einer 256-Punkte diskreten Fourier-Analyse alle 10 ms. 16 Mel-Spektralkoeffizienten sowie deren erste und zweite Ableitungen (Delta- und Delta-Delta-Koeffizienten) wurden zu einem 48-dimensionalen Vektor zusammengefaßt, der mittels einer linearen Diskriminanzanalyse auf 16 Dimensionen reduziert wurde.

Die 2885 akustischen Modelle wurden mittels divisiver Ballung mit der Entropiedistanz ermittelt. Dabei wurden nur die Daten der kleinen WSJ0 Datenbasis verwendet. Die restlichen Trainingsdaten aus dem WSJ1 Korpus wurden lediglich verwendet, um die akustischen Parameter mit dem EM-Algorithmus zu optimieren. Der Entscheidungsbaum wurde nicht mehr verändert.

Durch die fehlende Spektralmittelwertsubtraktion war dieser Erkenner sehr anfällig gegenüber Wechseln der Testumgebung, wie Tabelle 8.5 zeigt. Die Testdaten von 1992 wurden in der gleichen Zeitspanne und unter ähnlichen Aufnahmebedingungen wie die Mehrzahl der Trainingsaufnahmen gesammelt. Bei den späteren Testdaten waren die Charakteristiken der Aufnahmeumgebung weniger ähnlich. Außerdem war der verwendete Wortschatz und damit auch die Menge der Triphone durch das Diktieren der dann aktuellen Nachrichten mit den Trainingsdaten weniger kompatibel als bei den 1992er Testdaten.

8.5 Der Erkenner vom Frühjahr 1997

In diesem Abschnitt wird die Entwicklung des Erkenners, der die besten Erkennungsraten im Frühjahr 1997 erzielte, entsprechend den oben beschriebenen Entwicklungsphasen dargestellt.

8.5.1 Initialisierung (Bootstrapping)

Gegebene Voraussetzungen

Für die Initialisierung des Erkenners vom Frühjahr 1997 wurden die abgespeicherten Viterbipfade, die mit dem Erkenner von 1994 berechnet worden waren, verwendet. Neben den Wall Street Journal Sprachaufnahmen war auch ein Sprachmodell und ein Aussprachelexikon vorgegeben.

Definition der kontextunabhängigen Architektur

Im kontextunabhängigen System wurde jedes der 43 Phoneme mit drei Zuständen modelliert, von denen jedem ein eigenes Codebuch mit 32 48-dimensionalen Referenzvektoren zugeordnet wurde. Das allgemeine Stillemodell und das allgemeine Geräuschmodell wurden jeweils mit einem Zustand modelliert. Insgesamt hatte das System also 131 Codebücher.

Die HMM-Topologien für Phoneme (Abbildung 8.1) bestanden aus drei aufeinanderfolgenden Zuständen mit jeweils drei Übergängen. Jedem der Übergänge wurde eine Wahrscheinlichkeit von 1/3 zugewiesen und später nie mehr verändert. Diese Topologie erlaubt das Überspringen einzelner Zustände. Jeder der drei Zustände wurde durch ein unterschiedliches akustisches Modell modelliert.

Die Topologie für Stille (Abbildung 8.2) wurde so gewählt, daß jede erkannte Stille mindestens fünf Zeiteinheiten (50 ms) dauert. Wenn kürzere Stillezeiten erlaubt werden, dann führt dies oft dazu, daß an Wortübergängen, an denen eine optionale Stille erkannt werden darf, eine solche erkannt wird, obwohl keine Stille vorhanden ist. Durch das Einfügen einer Stille wird aber der Kontext der angrenzenden Phoneme verändert. So ist der linke Kontext des Phonems **W** von **WORLD** im Satz **HELLO WORLD** das Phonem **OW** von **HELLO**. Wenn das akustische Modell von **W** mit linkem Kontext **OW** zufällig ungünstig ist, und das akustische

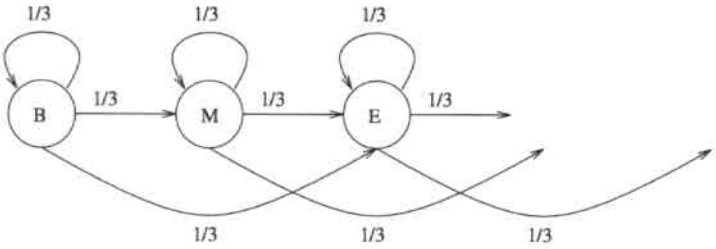


Abbildung 8.1: HMM-Topologie für Phoneme

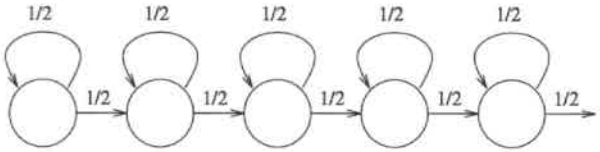


Abbildung 8.2: HMM-Topologie für Stille

Modell von **W** mit linkem Kontext **SIL** (Stille) eine bessere Emissionswahrscheinlichkeit liefert, dann würde der Viterbipfad das falsche Modell wählen. Das führt im Training zu einer Verschiebung der Trainingsdaten von den richtigen zu den falschen Modellen. Damit die durch die optionale Stille eingeführte Freiheit nicht zum Nachteil wird, wurde eine Mindestlänge von fünf Zeiteinheiten für Stille gefordert. Während die geringe Wahrscheinlichkeit eines einzelnen Stillezustandes noch durch die größere Wahrscheinlichkeit des falschen Kontextmodells ausgeglichen werden kann, ist dies bei fünf aufeinanderfolgenden Stillezuständen kaum noch möglich.

In einem Vergleichsexperiment, bei dem einmal Stillen mit einer minimalen Länge von einem Zustand und einmal mit mindestens fünf Zuständen verwendet wurden, sank die Fehlerrate insignifikant von 9.30% mit einem Zustand auf 9.15% mit fünf Zuständen. Dabei wurden akustische Modelle verwendet, die mit einem Ein-Zustand-Stillemodell trainiert wurden. Nach diesem Experiment wurden auch im Training nur noch Fünf-Zustand-Stillemodelle verwendet. Die Auswirkungen auf die Fehlerrate wurden aber nicht gesondert evaluiert.

Berechnen einer Linearen Diskriminanzanalyse

Das Berechnen einer Transformationsmatrix zur Linearen Diskriminanzanalyse (LDA) ist immer der erste Schritt, da diese Bestandteil der Signalvorverarbeitung ist. Zwei Dinge gilt es bei der Verwendung einer LDA festzulegen, zum einen den Urmerkmalsraum und zum anderen das Ausmaß der Dimensionalitätsreduktion im Bildmerkmalsraum. Letzteres wird in der Regel willkürlich auf einen Wert zwischen 2^4 und 2^6 festgelegt. Untersuchungen haben ergeben, daß eine hohe Dimensionalität des Bildmerkmalsraumes normalerweise keine Nachteile mit sich bringt, abgesehen von dem erhöhten Speicherbedarf und Rechenbedarf. Ein negativer Einfluß auf die Erkennungsrate stellt sich erst bei zu klein dimensionierten Merkmalsräumen ein.

Bei der Wahl des Urmerkmalsraumes sollte man beachten, daß möglichst viel Information enthalten ist. Dabei haben Experimente ergeben, daß, obwohl die LDA eine lineare Abbildung ist, die zum Beispiel aus einer zeitlichen Folge von Spektralkoeffizienten auch deren zeitliche Ableitung „berechnen" könnte, es durchaus sinnvoll sein kann, diese explizit vorzuberechnen und damit den Urmerkmalsraum zu erweitern. Im allgemeinen aber ist die Bedeutung der Form des Urmerkmalsraumes weniger wichtig, wenn er nur ausreichend groß ist. In der vorliegenden Arbeit wurden zur Transformation eines Sprachvektors mit 16 Spektralkoeffizienten der Vektor selbst sowie seine drei zeitlich vorhergehenden und drei nachfolgenden Vektoren zu einem insgesamt 112-dimensionalen Vektor aneinandergehängt. Nach der Transformation wurden dann nur die 48 Dimensionen verwendet, für die die LDA die größten Eigenwerte gefunden hatte.

Sortieren der Datenbasis

Für das Berechnen initialer Codebücher mit dem k-Mittelwerte Algorithmus ist es sinnvoll, die gesamten Trainingsvektoren so anzuordnen, daß alle Vektoren, die zum selben Codebuch gehören, kompakt liegen (z.B. in einer Datei). Dies ist deshalb nötig, da das k-Mittelwerte Verfahren iterativ arbeitet. Die Menge aller Trainingsdaten (WSJ0 und WSJ1 sind ca. 10 Gigabyte) kann nicht auf einmal im Rechnerspeicher gehalten werden. Ein sequentielles Einlesen aller Daten in jeder Iteration des k-Mittelwerte Algorithmus benötigt sehr viel Zeit. Wenn für jedes Codebuch die Trainingsdaten, die zur Initiali-

sierung benötigt werden, auf einmal eingeladen werden können, dann können mehrere Iterationen durchgeführt werden, ohne daß Sprachdaten nachgeladen werden müßen. Im JANUS Erkenner werden deshalb vor der Ausführung des k-Mittelwerte Algorithmus alle Trainingsdaten auf den Massenspeichern entsprechend sortiert. Für das hier beschriebene Diktiersystem bedeutet eine Vorsortierung der Trainingsdaten eine Zeitersparnis von ca. 10 CPU-Tagen auf einen CPU-Tag.

Initiale Codebücher mit dem k-Mittelwerte Algorithmus

Der k-Mittelwerte oder auch „Basic-ISODATA" Algorithmus kann als ein Spezialfall des „Neural-Gas" [MBS93] Algorithmus angesehen werden. Beiden Verfahren liegt zugrunde, daß aus einer Menge von m Mustervektoren ein Codebuch aus n Referenzvektoren berechnet werden soll, so daß der durchschnittliche Abstand jedes Mustervektors zu seinem nächsten Referenzvektor minimiert wird. Bei Neural-Gas wird im Gegensatz zum reinen k-Mittelwerte Algorithmus ein Trainingsvektor nicht nur dem nächsten Referenzvektor zugeordnet, sondern mit einer kleineren Wahrscheinlichkeit auch allen anderen Referenzvektoren. Dabei werden alle Referenzvektoren entsprechend ihrer Abstände geordnet. Das Gewicht, mit dem der Trainingsvektor dann in die Schätzung eines Referenzvektors eingeht, fällt exponentiell ab mit der Position x des Referenzvektors in der nach Abstand geordneten Liste. Die Steilheit t der Exponentialfunktion $e^{-x/t}$ kann als Temperatur betrachtet werden, so daß für $t \to \infty$ alle Referenzvektoren gleich geschätzt werden und für $t \to 0$ der reine k-Mittelwerte Algorithmus approximiert wird. In der vorliegenden Arbeit wurde für t ein initialer Wert von 1.0 gewählt. Dieser Wert wurde dann mit jeder Iteration des Algorithmus durch den Faktor 100 dividiert (abgekühlt), so daß schon in der dritten von zehn durchgeführten Iterationen das durch die Temperatur eingeführte Rauschen so gut wie eliminiert war.

Da der k-Mittelwerte Algorithmus nicht nur initiale Referenzvektoren berechnet, sondern auch eine Zuordnung aller Trainingsvektoren zu den Referenzvektoren trifft, kann aus dieser Information auch die a-priori Wahrscheinlichkeit dafür, daß ein Referenzvektor der nächste zu einem Trainingsvektor ist, berechnet werden. Diese Werte können dann als initiale Werte für die Mixturgewichteverteilung des Codebuchs verwendet werden. Obwohl auch eine entsprechende Initialisierung der Kovarianzmatrizen möglich ist, wurde

in der vorliegenden Arbeit der Einfachheit halber darauf verzichtet.

8.5.2 Übergang zur Kontextabhängigkeit

Der Übergang von einem kontextunabhängigen System zu einem kontextabhängigen besteht aus dem Definieren der kontextabhängigen Modelle, dem Trainieren einer eigenen Mixturgewichteverteilung für jedes Modell, dem divisiven Ballen, dem Einführen eines eigenen Codebuches für jede entstandene Klasse und dem Initialisieren und Trainieren der dann vorhandenen Parameter.

Erzeugen von Polyphonlisten

Bevor ein kontextabhängiger Erkenner trainiert werden kann, muß erst die Menge der zu trainierenden kontextabhängigen Modelle festgelegt werden. Dazu werden Satz-HMMs für jede Äußerung der Trainingsdaten berechnet (siehe Abschnitt 5.2.3). Aus diesen Satz-HMMs werden dann alle möglicherweise vorkommenden Polyphone extrahiert. Wenn für die maximale Kontextbreite der Rand des aktuellen Wortes verwendet würde, dann ließen sich die vorkommenden Polyphone direkt aus dem Aussprachelexikon extrahieren. Tatsächlich aber wurden in der vorliegenden Arbeit wortübergangsabhängige Kontexte verwendet und außerdem optionale Stilleperioden zwischen zwei aufeinanderfolgenden Worten erlaubt. So können wesentlich mehr Polyphone entstehen. Im Falle einer maximalen Kontextbreite von jeweils drei Phonemen nach links und nach rechts (Septphone) wird die Menge aller Polyphone durch Berücksichtigen der Wortübergänge von etwa 170 000 auf über 570 000 erhöht.

Trainieren kontextabhängiger Mixturgewichte

Nachdem alle Polyphone der Wall Street Journal Trainingsdaten gesammelt wurden, wurde für jedes Subpolyphon eine eigene Mixturgewichteverteilung eingeführt und jeweils mit der Gleichverteilung initialisiert.

Mit zwei Epochen Viterbi-Training wurden dann kontextabhängige Mixturgewichte geschätzt. Viele Verteilungen wurden auf sehr wenigen Trainingsbeispielen trainiert. Abbildung 8.3 zeigt, wie viele der insgesamt über 700 000 Modelle eine bestimmte Menge an Trainingsdaten hatten. Es

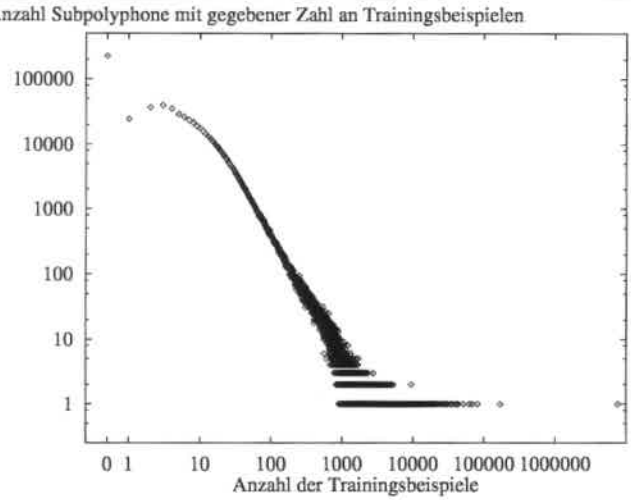


Abbildung 8.3: Häufigkeit der Größen der Trainingsmengen

fällt auf, daß über 200 000 Modelle gar kein Training erhalten haben. Das kommt daher, daß die HMM-Satztopologie, die im Viterbi-Algorithmus verwendet wird, zwischen zwei aufeinanderfolgenden Worten eine optionale Stille erlaubt. Dadurch entstehen für jedes Phonem am Rande eines Wortes mindestens zwei mögliche Kontexte (bei einphonemigen Wörtern sogar vier). Während der Forward-Backward Algorithmus allen verschiedenen Kontexten eine positive Wahrscheinlichkeit zuordnen kann, liefert der Viterbi-Algorithmus nur für einen Kontext die Wahrscheinlichkeit 1.0, die anderen Kontexte gehen leer aus, und werden nicht trainiert.

Eine Fehlerrate des Erkenners kann zu diesem Zeitpunkt nicht bestimmt werden, da die enorme Menge verschiedener akustischer Modelle den Zeitaufwand für einen Dekodiervorgang unerträglich hoch macht (ca. 10^5 bis 10^6-fache Echtzeit).

Ballung der kontextabhängigen Modelle

Wenn die vielen kontextabhängigen Elementarmodelle trainiert sind, können sie geballt werden. Dazu wurde ein Entscheidungsbaum initialisiert, der zunächst genau ein Blatt für jedes Subphonem hatte. Das Stillemodell und das Geräuschmodell wurden nicht kontextabhängig modelliert und somit nicht in den Ballungsprozeß einbezogen. Das heißt, für jedes der 129 Subphonemcodebücher existierte ein Blatt, dem alle entsprechenden kontextabhängigen Subpolyphone zugeordnet wurden. Nach dem im Abschnitt 6.5.3 beschriebenen Verfahren wurden die über 700 000 Subpolyphone zu 4 000 Klassen zusammengeballt.

Einführen kontextabhängiger Codebücher

Nach der Ballung der Subpolyphone wurde für jede entstandene Klasse ein eigenes Codebuch eingeführt. Während in der Phase vor der Kontextballung der allergrößte Teil des akustischen Parameterraumes in den Mixturgewichten lag, waren nach der Einführung der kontextabhängigen Codebücher die meisten Parameter in den Referenzvektoren und ihren Kovarianzmatrizen.

Erneute Initialisierung

Eine mögliche Art der Fortführung der Entwicklung des Erkenners wäre gewesen, die Parameter jedes kontextabhängigen Codebuchs mit den Parametern des entsprechenden kontextunabhängigen Codebuchs zu initialisieren. Durch Viterbi-Training würden dann die zunächst für verschiedene Kontexte gleichen Parameter divergieren. Aus zwei Gründen wurde aber ein anderer Weg gewählt:

Zum einen bestünde die Gefahr, daß einige Referenzvektoren in den Codebüchern bleiben, die für den entsprechenden Kontext relativ unpassend sind. Das Vorhandensein von „Ausreißern" in den Codebüchern würde somit gefördert werden.

Zum anderen bietet es sich in dieser Entwicklungsphase an, eine neue lineare Diskriminanzanalyse durchzuführen, wobei diesmal jedes kontextabhängige Codebuch als eine eigene LDA-Klasse behandelt wird. Da die bisher verwendete LDA-Matrix eine Transformation so durchführt, daß

alle Trainingsvektoren eines Subphonems unabhängig von ihrem Kontext einander näher rücken, wird so eine Diskriminierung verschiedener Kontexte erschwert.

Deshalb wurden im hier beschriebenen Erkenner alle Codebücher komplett neu initialisiert. Zuerst wurde eine neue LDA-Matrix berechnet. Dann wurden allen Codebüchern mit dem k-Mittelwerte Algorithmus neue Referenzvektoren zugeordnet.

Training des kontextabhängigen Erkenners

Das EM-Training der jetzt initialisierten Parameter erfolgte zunächst unter Verwendung der in einer früheren Phase abgespeicherten Viterbi-Pfade. Nachdem sich die Erkennungsleistung durch das EM-Training signifikant verbessert hatte, wurden neue Viterbi-Pfade berechnet, und abgespeichert, damit sie in späteren Phasen wieder verwendet werden konnten.

Insgesamt wurden fünf EM-Trainingsepochen durchgeführt. Diese Zahl wurde von vorn herein festgelegt als ein Wert, der aus der Erfahrung heraus sinnvoll erschien. Eine Kreuzvalidierungsmenge zum Abbrechen des Trainings wurde nicht benutzt, da die Erfahrung zeigte, daß ein Overfitting durch zu viele EM-Trainingsepochen nicht zu erwarten ist, und nach ca. 3 bis 5 Epochen die Kurve der Erkennungsleistung in einen gesättigten Bereich kommt.

8.5.3 Die iterative Trainingsmethode

Wenn hier die Rede von der iterativen Trainingsmethode ist, dann ist damit nicht gemeint, daß die Parameter des Systems mehrmals hintereinander mit EM-Training optimiert werden, sondern daß die hier vorgestellten Trainingsschritte immer wieder wiederholt werden. Das Training eines Erkenners besteht aus verschiedenen Schritten, die nicht nur die Parameter, sondern auch die Struktur des Parameterraums verändern. Zuerst wird ein kontextunabhängiger Erkenner mit kleinem Parameterraum trainiert, dann wird ein semikontinuierlicher kontextabhängiger Erkenner gebaut, dieser wird dann mittels Kontextballung zu einem voll kontinuierlichen Erkenner mit großem Parameterraum. Bei jedem dieser Schritte werden mehrere

Teilschritte durchgeführt, wie zum Beispiel die Berechnung einer neuen LDA Matrix, die Neuinitialisierung der Codebücher mit dem k-Mittelwerte Algorithmus und auch die eventuell über mehrere Epochen gehende EM-Optimierung.

Der Vorteil dieser Methode ist, daß Trainingsinformation von Systemen mit großem Parameterraum, die in früheren Entwicklungsphasen trainiert wurden, in der aktuellen Entwicklungsphase schon für das Trainieren eines Systems mit kleinem Parameterraum einfließen kann. Da in der Regel die Systeme mit vielen Parametern nicht nur bessere Erkennungsleistung haben, sondern auch bessere Viterbi-Pfade berechnen, können diese guten Pfade abgespeichert werden und für das erneute Trainieren eines Systems mit kleinem Parameterraum verwendet werden. Das kleinere und schlechtere System wäre gar nicht in der Lage, mit seinen wenigen Parametern vergleichbar gute Pfade zu finden. Wenn aber das kleine System dadurch verbessert ist, ist auch die Ausgangssituation für das darauf aufbauende größere System besser.

8.5.4 Geschlechtsabhängige Modellierung

Nachdem ein vollständiger Entwicklungsdurchgang vollendet war, erzielte der Erkenner eine Fehlerrate von 10.7% bei Verwendung der in Abschnitt 8.3.4 beschriebenen Adaption. Mit diesem Erkenner wurden neue Viterbi-Pfade berechnet und als Grundlage für die Entwicklung eines geschlechtsabhängigen Erkenners verwendet. Bei dem geschlechtsabhängigen Erkenner handelt es sich eigentlich um zwei voneinander völlig unabhängige Erkenner, von denen jeder nur auf den Trainingsdaten eines Geschlechts trainiert wurde. Die Wall Street Journal Datenbasis enthält 141 weibliche und 143 männliche Sprecher mit annähernd gleich viel diktierten Sätzen. Die geschlechtsabhängigen Erkenner erhielten jeweils eigene Entscheidungsbäume, in denen die ca. 570 000 Subpolyphone je Geschlecht zu je 3 000 zusammengeballt wurden. Vor der eigentlichen Erkennung mußte nun eine Geschlechtsidentifikation durchgeführt werden. Es stellte sich heraus, daß die Wahrscheinlichkeit der erkannten Hypothese eines Erkenners bei einer Aufnahme von einer Person des entsprechenden Geschlechts stets (bis auf einen Fall) besser war als die Wahrscheinlichkeit der erkannten Hypothese des Erkenners für das falsche Geschlecht. In diesem einen Fall waren die erkannten Hypothesen beider Erkenner zudem identisch. Nachdem

für jedes Geschlecht ein kompletter Entwicklungsdurchgang vollendet war, erzielte das geschlechtsabhängige Erkennerpaar eine Fehlerrate von 9.3% (9.30% für Frauen und 9.34% für Männer).

In einem erneuten kompletten Entwicklungsdurchgang, bei dem dieses Mal 5 000 akustische Modelle erzeugt wurden, konnte die Erkennungsrate auf 8.8% gesenkt werden. Dieses System diente als Basissystem für die Experimente zur Parameterraumkompaktifizierung in Kapitel 7. Eine Adaption der Parameter, wie in Abschnitt 8.3.4 beschrieben, konnte die Fehlerate auf 7.7% senken. Dabei wurden zur Berechnung der Transformation für die Adaption nur die Sprachvektoren des gerade zu erkennenden Satzes einbezogen. Im Gegensatz zu den Berichten der Cambridge University [WGPV96] konnte eine inkrementelle Adaption, bei der auch alle vorausgegangenen Sätze desselben Sprechern berücksichtigt werden, die Fehlerrate nicht weiter senken. Der Grund dafür liegt wahrscheinlich in der Verwendung eines Konfidenzmaßes im JANUS Erkenner. Der Erkenner der Cambridge University verwendete kein Konfidenzmaß. Das Konfidenzmaß bewertet die vermutete Korrektheit jedes Wortes der Hypothese anhand der Stabilität des Wortes gegenüber leichten Modifikationen der Parameter q und z aus der Gleichung 2.22. Dieses Maß ist sehr stark korreliert mit der tatsächlichen Korrektheit. Für die Berechnung der Adaptionstransformation wurden in dieser Arbeit nur diejenigen Teile der Hypothese benutzt, die eine Mindeststabilität hatten.

8.5.5 Fehleranalyse

Bei der Entwicklung eines Spracherkenners empfiehlt es sich, regelmäßig eine Fehleranalyse durchzuführen, bei der alle Fehler, die der Erkenner gemacht hat, klassifiziert werden. Aus den Häufigkeiten der Vorkommen bestimmter Fehlerklassen kann man dann entscheiden, gegen welche Fehlerquellen ein konzentrierteres Vorgehen sinnvoll ist. In der Regel werden solche Fehlerklassen nicht disjunkt ausfallen.

Irreparable Fehler

In der WSJ-Evaluation im November 1994 macht der JANUS-Spracherkenner auch Fehler, die nicht durch eine noch so gute akustische Modellierung

repariert werden können. Dazu gehören einige falsche Transkriptionen. An diesen Stellen erkennt der Erkenner korrekt das, was gesprochen wurde, dennoch wird ein Fehler gerechnet, weil das Gesprochene nicht mit dem Transkribierten übereinstimmt (z.B. gesprochen „**THERE'S**", transkribiert „**THERE IS**)".

Zu den irreparablen Fehlern gehören auch Ambiguitäten, bei denen verschiedene Varianten sich akustisch nicht unterscheiden, sogenannte Homophone. In der Regel betreffen Homophone nur einzelne Wörter, die die gleiche Aussprache haben (z.B. **IT'S** und **ITS**). Hin und wieder aber beobachtet man auch Homophone über mehrere Wörter hinweg (z.B. „**USE IS**" und „**USES**"). Das Zusammensetzen und die Rechtschreibung von Wörtern ist im amerikanischen Englisch nicht eindeutig festgelegt, so daß bei einer Erkennung von „**VIDEO TAPE**" statt „**VIDEOTAPE**" zwei volle Fehler berechnet werden, ebenso wird die Erkennung von „**GOVERNMENT**" bei einer „**GOVERMENT**" lautenden Transkription als Fehler gezählt. Schließlich gibt es noch Aussprachevarianten bestimmter Wörter, die denen anderer Wörter gleich sind. So sprechen viele Amerikaner die Wörter „**WRITING**" und „**RIDING**" gleich aus, so daß in Fällen, in denen es der Sprachkontext erlaubt, die akustische Modellierung keines der Wörter bevorzugen kann, und die Auswahl allein vom Sprachmodell bestimmt wird. Das Aussprachelexikon, das für die Entwicklung verwendet wurde, enthält 70 747 Einträge mit insgesamt nur 63 829 verschiedenen Aussprachen, was bedeutet, daß 6 918 Einträge, also etwa 10%, Homophone sind (die meisten davon sind Genitiv-**S** mit Plural-**S** Vertauschungen). Da im Wall Street Journal sehr viel über Firmen und Personen berichtet wird, tauchen auch sehr viele Namen auf, die zum Teil identische Aussprachen haben oder zumindest ermöglichen. Das häufigste Homophon hat die Aussprache „**R AY T S**" und paßt auf die zehn Wörter **REITS**, **REITZ**, **RIGHT'S**, **RIGHTS'**, **RIGHTS**, **RITE'S**, **RITES**, **WRIGHT'S**, **WRIGHTS**, und **WRITES**.

Insgesamt summieren sich die irreparablen Fehler zu einer minimal erreichbaren Fehlerrate von 2.1%. Weniger Fehler sind bei einem gegebenen Sprachmodell und Aussprachelexikon durch Optimierung der akustischen Parameter nicht möglich. In diesem Kontext könnte man eine Fehlerreduktion von 5% auf 4% als eine relative Verbesserung um ca. 1/3 statt 1/5 ansehen.

Einfügungen und Auslassungen

Aus der Definition der Fehlerrate ergibt sich eine Klassifizierung der Fehler in drei Klassen: Vertauschungen, Einfügungen und Auslassungen. In der Regel wird die Gesamtfehlerrate dominiert durch die Vertauschungsfehler, während die Mengen der Auslassungen und Einfügungen eher klein ausfallen. Besonders in der Anfangsphase der Entwicklung eines Erkenners kann es aber vorkommen, daß die Zahl der Einfügungen oder Auslassungen überdurchschnittlich groß ist. Wenn sich die Größen dieser beiden Fehlerklassen wesentlich unterscheiden, dann läßt sich das Ungleichgewicht meist darauf zurückführen, daß die Wortübergangsstrafe (Variable q aus Gleichung 2.22) in der Suche suboptimal eingestellt ist. Durch Vergrößern von q kann man die Zahl der Einfügungen verringern, und durch Verkleinern von q reduziert man die Zahl der Auslassungen. Der optimale Wert von q wird gemeinsam mit der Gewichtung des Sprachmodells z mittels einer Kreuzvalidierungsmenge optimiert und dann unverändert bei eigentlichen Test verwendet.

Funktionswörter und inhalttragende Wörter

Ein sehr großer Anteil der Erkennungsfehler fällt auf die sogenannten Funktionswörter. Das sind die meisten Artikel, Präpositionen und Konjunktionen. Sie stellen eine häufige Fehlerquelle dar, nicht nur weil sie oft gesprochen werden, sondern auch weil sie verhältnismäßig kurz, sprich leicht verwechselbar sind, und weil sie sehr oft schlecht artikuliert werden. Gerade im amerikanischen Englisch werden die meisten Artikel und Konjunktionen sehr undeutlich gesprochen und verschmelzen oft mit den angrenzenden Worten („Rock'n'Roll").

Die zehn am häufigsten gemachten Fehler sind in Tabelle 8.7 aufgeführt. Die dritte Spalte gibt den Anteil eines Fehlers unter allen Fehlern an. Der Artikel **THE** ist an ca. 24% aller Fehler beteiligt. Die 6 Wörter **THE**, **A**, **AND**, **OF**, **THAT** und **IN** sind an ca. 45% aller Fehler beteiligt.

Tabelle 8.6 zeigt, an wievielen Fehlern einzelne Funktionsworte beteiligt sind. Die Zahlen sind als „ungefähre" Werte anzusehen und schwanken leicht von Erkenner zu Erkenner. Man beachte beim Betrachten der Tabelle, daß an einem Fehler mehrere Wörter beteiligt sein können.

Früher war eine Herangehensweise an das Problem die Verwendung von

Tabelle 8.6: Beteiligung an Fehlern

Wort	beteiligt an Fehlern
THE	24%
A	15%
THAT	13%
IN	12%
AND	11%
BUT	11%
OF	10%

Tabelle 8.7: häufigste Fehler

gesprochen	erkannt	Anteil (ca.)
AND	*nichts*	5%
nichts	**THE**	3%
THE	**THAT**	3%
THE	*nichts*	3%
nichts	**AND**	2%
A	*nichts*	2%
THEIR	**THE**	1%
AT	**AND**	1%
A	**OF**	1%
A	**THE**	1%

Ganzwortmodellen für die häufigsten Funktionswörter, damit zum einen für diese Wörter implizit breitere Kontexte verwendet werden und zum anderen keine Kontaminierung der Trainingsdaten mit den entsprechenden Phonemen aus den inhaltstragenden Wörtern vorkommt. Da in der vorliegenden Arbeit aber breite Kontexte verwendet werden, werden die meisten Funktionsworte, insbesondere die kurzen, sowieso vollständig kontextabhängig mit maximalem Kontext modelliert. Ein signifikanter Gewinn in der Erkennungsgenauigkeit ist also durch Ganzwortmodellierung nicht zu erwarten.

Als erfolgversprechenderer Ansatz bieten sich spezielle Sprachmodelle an, die die Funktionswörter zusammen mit den angrenzenden Wörtern zu sogenannten „Clustern" zusammenfassen.

Signalqualität

5% aller Fehler sind auf schlechte Signalqualität oder artikulatorische Geräusche wie Atmen oder Schmatzen, die die gesprochene Sprache überlagern, zurückzuführen. In machen Fällen ist die Qualität der Aufnahme so schlecht, daß selbst Menschen Schwierigkeiten haben, zu verstehen, was gesagt wurde.

Das amerikanische „AND"

In den Wall Street Journal Daten kommen sehr viele Zahlen vor. Bei großen Zahlen wird das Wort **AND** (zum Beispiel in „one hundred *and* two") sehr schlecht ausgesprochen und hört sich an wie ein einfaches **N** wie in „Rock'n'Roll". Knapp 4% aller Fehler sind Auslassungen von **AND** in Zahlen. Die Idee, eine Aussprachevariante ins Aussprachelexikon aufzunehmen, die auch diese Aussprache abdeckt, führt leider zu vielen Einfügungsfehlern, da das sehr kurze Wort, das aus der Sicht des Sprachmodells an sehr vielen Stellen auftreten kann, öfter fälschlicherweise eingefügt wird, als es vorher Auslassungen gab. Mit der folgenden Lösung gelang es dem Autor, 93% (14 von 15) aller **AND** Auslassungen in Zahlen zu beheben. In das Aussprachelexikon wurden für die Wörter **HUNDRED** und **THOUSAND** die Aussprachevarianten **HH AH N D R AX D N** bzw. **TH AW Z AX N D N** eingefügt. Wenn dann in der erkannten Hypothese eine der zusätzlich eingefügten Varianten verwendet wurden, dann wurde in der Hypothese an der entsprechenden Stelle das Wort **AND** nachträglich eingefügt. In nur einem Fall wurde durch diese Methode ein vorher nicht gemachter Fehler mehr gemacht.

8.6 Evaluierung und Bewertung

Der Erkenner wurde zu verschiedenen Zeitpunkten der Entwicklung evaluiert. Oft wurden mehrere Veränderungen gleichzeitig vorgenommen. Dieses Vorgehen ist zwingend, da so der benötigte Zeitbedarf für das Trainieren und Testen der Erkenner auf ein tolerables Maß beschränkt werden konnte. Aus demselben Grund war es nicht möglich, alle Kombinationen der eingesetzten

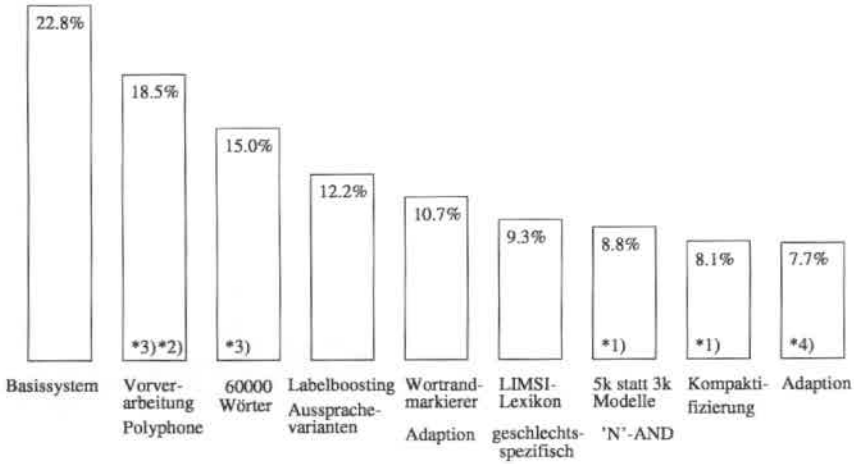


Abbildung 8.4: Fehlerraten nach verschiedenen Phasen der Entwicklung

Algorithmen zu evaluieren. Abbildung 8.4[1] faßt die Evaluierungen nach verschiedenen Verbesserungen zusammen.

Ausgangspunkt war der Erkenner vom November 1994 mit einer Fehlerrate von 22.8%. Von diesem Erkenner wurden lediglich die vorberechneten Viterbi-Pfade verwendet, um einen neuen Erkenner zu trainieren, der eine etwas veränderte Signalvorverarbeitung hatte und Quintphone statt Triphone verwendete. Teil der verbesserten Vorverarbeitung war auch die Verwendung einer Mittelwertsubtraktion. Der so trainierte Erkenner erzielte eine Fehlerrate von 18.5%.

[1]Die mit *1) gekennzeichneten Ergebnisse wurden auf einer repräsentativen Untermenge, die etwa einem Drittel der gesamten Testmenge entsprach und alle Testsprecher enthielt, gemessen. Das mit *2) gekennzeichnete Ergebnis enthält ca 0.8% Fehler, die aufgrund eines Programmfehlers im Fehlermeßprogramm fälschlicherweise aufgenommen wurden. Die mit *3) gekennzeichneten Ergebnisse enthalten ca 0.9% Fehler, die von DARPA nach der Evaluation von 1994 nachträglich nicht mehr als Fehler gewertet wurden, weil tatsächlich die Referenztranskriptionen fehlerhaft waren. Das mit *4) gekennzeichnete Ergebnis wurde ohne Kompaktifizierung erzielt.

Durch die Verwendung eines Vokabulars mit 60 000 Wörtern statt der bis dahin verwendeten 20 000 Wörter konnte der Anteil der nicht abgedeckten Wörter aus der Testmenge von 2.3% auf 0.3% gesenkt werden. Da jedes nicht abgedeckte Wort mindestens einen Fehler verursacht, der wegen der größeren Reichweite des statistischen Sprachmodells oft Folgefehler nach sich zieht, konnte die Fehlerrate um mehr als 2 Prozentpunkte auf 15.0% gesenkt werden. In diesem Schritt wurde auch ein Programmfehler aus dem Fehlerberechnungsprogramm entfernt, der alle korrekt erkannten Worte mit einem Apostroph fälschlicherweise als Fehler wertete. Die dadurch beobachtete Reduktion der Fehlerrate um absolut 0.8% ist in der Reduktion auf 15% Fehler mit eingerechnet.

Die erneute Berechnung von Viterbi-Pfaden unter Zuhilfenahme einer Adaption bei gleichzeitiger Verwendung von Varianten im Aussprachelexikon reduzierte die Fehlerrate auf 12.2%.

Danach wurden alle Phoneme am Anfang und am Ende eines Wortes mit einem Wortrandmarkierer versehen und ein neuer Entscheidungsbaum berechnet. Beim Test dieses Erkenners wurde eine Adaption verwendet. Die erzielte Fehlerrate betrug vor der Adaption 11.5%, danach 10.7%.

Dann wurde das Aussprachelexikon, das bisher das unveränderte CMU-Lexikon war, erweitert um die Aussprachevarianten aus dem LIMSI-Lexikon. Zwei völlig unabhängige Erkenner wurden auf den nach Geschlechtern getrennten Trainingsdaten trainiert. Die Fehlerrate des Gesamtsystems wurde auf 9.3% ohne Adaption reduziert.

Eine Vergrößerung des Parameterraumes durch Verwenden von 5 000 statt 3 000 Codebüchern und die Einführung der **AND**-Varianten der Wörter **HUNDRED** und **THOUSAND** senkte die Fehlerrate auf 8.8%. Etwa die Hälfte der Fehlerreduktion wurde durch die **AND**-Varianten erzielt.

Das am Ende beste Ergebnis wurde durch den Einsatz einer Adaption der Systemparameter auf den zu erkennenden Satz erzielt. Die Fehlerrate konnte um etwa 12% auf schließlich 7.7% reduziert werden.

Die Bedeutung dieses Ergebnisses veranschaulicht die Abbildung 8.5.

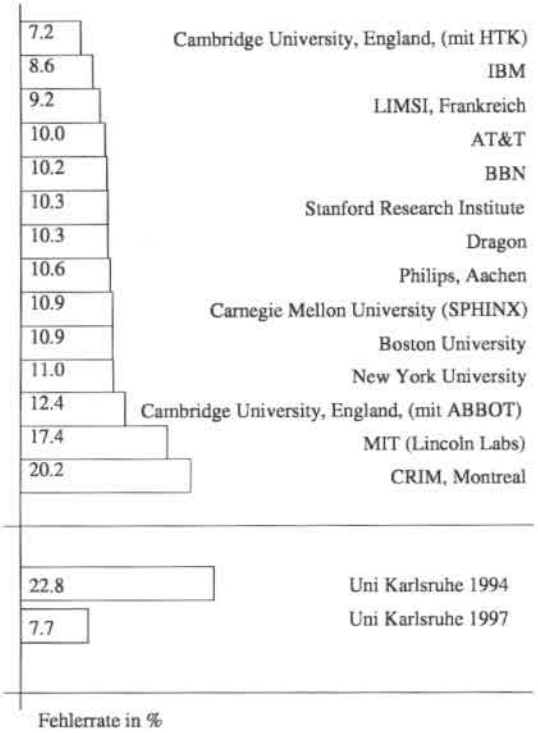


Abbildung 8.5: Die Erkennungsleistung des JANUS-Erkenners im Vergleich mit den Ergebnissen der ARPA Evaluation vom November 1994

Nach der Evaluation im November 1994 wurden von der ARPA keine weiteren Evaluationen für Diktieren auf sauberer Sprache durchgeführt. Die meisten Teilnehmer fokussierten ihre Forschung danach auf „unsaubere" Sprache wie Telefonaufnahmen von spontanen Dialogen oder reale Nachrichtensendungen. Dabei wurden die Hauptaugenmerke statt auf die Gestaltung der Parameterräume auf die Adaption der vorhandenen Erkenner an die neue Problematik gelegt wie die automatische Segmentierung der Aufnahmen und die Kompensation von Störungen wie Hintergrundgeräusche, Telefonleitungsprobleme oder gleichzeitiges Sprechen mehrerer Personen. Im Frühjahr 1997 lag die beste Erkennungsrate des HTK-Erkenners der Universität Cambridge auf der Testmenge von November 1994 immer noch bei 7.2% [You97]. Der JANUS-Erkenner unterscheidet sich im wesentlichen von HTK darin, daß er keine Viergramm Sprachmodelle verwendet, keinen Gewinn aus einer inkrementellen Adaption erhält und eine andere Art der Parameterraumgestaltung hat (siehe Abschnitt 7.1.2). Weitere Unterschiede bestehen in der eingesetzten Signalvorverarbeitung, der Art des iterativen Trainierens (Abschnitt 8.5.3) und der Implementierung der Adaption und der Suche. Insgesamt kann man sagen, daß die beiden Erkenner deutlich verschieden sind. Vor allem der Karlsruher Erkenner wird in der Zukunft von den noch nicht ausgenutzten Optimierungsverfahren profitieren können.

Kapitel 9

Zusammenfassung

Erkennungsleistung

In den ersten vier Kapiteln wurden die Grundlagen der Erkennung sprecherunabhängiger kontinuierlicher Sprache eingeführt sowie die Wall Street Journal Datenbasis als international vergleichbare Bewertungsumgebung und der JANUS-Erkenner als Entwicklungswerkzeug vorgestellt. Die Wall Street Journal Datenbasis wurde über viele Jahre von zahlreichen Forschungsinstituten als Vergleichstest für Diktiersysteme eingesetzt und wurde bis heute nicht durch einen neuen Testdatensatz ersetzt. Daher sind die hier vorgestellten Ergebnisse relevant und mit denen anderer Forscher vergleichbar.

Durch die hier beschriebene Arbeit wurde die Fehlerrate des JANUS-Erkenners gedrittelt und befand sich schließlich bei unter 8%. Dieses Ergebnis gehört heute zu den besten weltweit, die auf der verwendeten Testmenge erzielt wurden. Zum Erreichen dieser guten Erkennungsleistung wurden verschiedene Verfahren eingesetzt, die im Kapitel 8 beschrieben wurden. Viele Verfahren hängen synergetisch voneinander ab und entwickeln ihre volle Leistung erst im Zusammenarbeiten mit anderen Methoden. Andererseits sind die isoliert betrachteten Verbesserungen nicht additiv, so daß der Einsatz mehrerer Optimierungen weniger Gewinn bringt als die Summe der Einzelgewinne. Bei den heute verwendeten sehr großen Datenbasen und Parameterräumen ist es rein praktisch nicht möglich, alle Kombinationen von Optimierungen zu evaluieren. Die dafür benötigten Rechenzeiten zwingen den Sprachforschern ein Vorgehen auf, sich auf

diejenigen Methoden zu konzentrieren, die den meisten Erfolg versprechen.

Die wichtigsten Verbesserungen, die die Fehlerrate von über 22% auf unter 8% senkten, waren:

- der Einsatz von Polyphonen statt Triphonen
- die Vergrößerung des Vokabulars von 20 000 auf 60 000 Wörter
- eine robustere Signalvorverarbeitung mit spektraler Mittelwertsubtraktion
- die vermehrte Verwendung von Aussprachevarianten aus zwei Aussprachelexika
- das Berechnen von Viterbi-Pfaden mit vorausgegangener Adaption
- die Verwendung von Wortrandmarkierern in den Entscheidungsbäumen
- das Trainieren geschlechtsspezifischer akustischer Modelle
- die Verwendung von 5 000 statt 3 000 Codebüchern
- die explizite Modellierung des Wortes AND innerhalb von Zahlen
- die Adaption der Systemparameter an die Aufnahme durch lineare Regression

Architekturentwurf

In Kapitel 6 wurden Methoden zum Entwurf des Parameterraumes der akustischen Modellierung vorgestellt und die Bedeutung der Kontextbreite wurde untersucht. Es wurde festgestellt, daß das bisher in JANUS und von anderen Forschern verwendete Ballungsverfahren mit der Theorie des Expectation-Maximization (EM) Algorithmus zum Trainieren von Hidden Markov Modellen inkonsistent ist, da das optimierte Kriterium der Informationsgehalt (Entropiedistanz) der Systemparameter war und nicht die Gesamtwahrscheinlichkeit (Likelihood-Distanz) der Trainingsdaten bei gegebenem akustischen Modell.

Daher wurde vom Autor ein neuartiges Ballungsverfahren entwickelt, das es ermöglicht, die akustischen Modelle in einem Entscheidungsbaum anzuordnen und dabei die Gesamtwahrscheinlichkeit der Trainingsdaten zu maximieren. Um eine Verwendung dieses Verfahrens praktikabel zu machen, mußten Maßnahmen getroffen werden, die den enormen Bedarf an Rechenzeit und Speicher reduzierten. Ein Vergleich der Fehlerraten bei Verwendung einer Likelihood-Distanz gegenüber der Verwendung einer Entropie-Distanz ergab eine relative Fehlerreduktion um ca. 7%.

Das Likelihood-Distanzmaß erlaubt eine einfache Erweiterung dahingehend, daß nicht die Gesamtwahrscheinlichkeit aller Trainingsdaten, sondern nur die einer Kreuzvalidierungsmenge berechnet wird. Dadurch ist es möglich, den Effekt der Überanpassung zu vermeiden. Bei der divisiven Ballung durch Wachsenlassen eines Entscheidungsbaumes wird in jedem Schritt, bei dem ein Knoten in zwei Nachfolgeknoten aufgespalten wird, sowohl der Informationsgehalt des Parameterraumes erhöht als auch die Wahrscheinlichkeit der zum Schätzen der Parameter verwendeten Trainingsdaten. Auf einer Kreuzvalidierungsmenge stellt sich allerdings ab einer bestimmten Modellierungsfeinheit keine Wahrscheinlichkeitssteigerung mehr ein. Diese Tatsache kann als Abbruchkriterium für das Wachsen des Entscheidungsbaumes herangezogen werden. Ein derartiger Automatismus ist von Vorteil gegenüber der sonst üblichen Methode, viele verschiedene Parameterraumgrößen auszuprobieren und sich für diejenige zu entscheiden, die in einem Test am wenigsten Fehler produziert.

Ein Vertreter dieser „üblichen" Methode ist der HTK-Erkenner der Cambridge University in seinem Verfahren zur Bestimmung der Größe seiner Codebücher. Der Erkenner verwendet zwar ein Kriterium zum Maximieren der Wahrscheinlichkeit der Trainingsdaten, aber die letztendliche Entscheidung über die Größe der Codebücher ebenso wie über den geeigneten Zeitpunkt zum Abbruch des Ballungsvorganges obliegt dem Entwickler. Für diese beiden Aufgaben bietet sich dem HTK-Erkenner an, mit Hilfe von Kreuzvalidierungsmengen den Entscheidungsprozeß zu automatisieren.

Die meisten Erkenner, die an den ARPA Evaluationen teilgenommen haben, verwenden eine Art hierarchische Ballung der kontextabhängigen Modelle. Die üblicherweise verwendeten Kriterien und Distanzmaße dabei könnten erweitert werden mit der Verwendung einer Kreuzvalidierung. Dadurch könnte der Entwurfsprozeß für den Parameterraum automatisiert werden, und die Erkennungsraten sollten zumindest stabil bleiben.

Parameterraumkompaktifizierung

In Kapitel 7 wurden verschiedene Methoden zur Kompaktifizierung des akustischen Parameterraumes untersucht und verglichen. Ein Interesse an einer Kompaktifizierung des Parameterraumes besteht nicht nur, weil dadurch der praktische Einsatz eines Erkenners im Alltag erleichtert wird, sondern auch, weil kleinere Parameterräume robuster geschätzt werden und zu einer Erhöhung der Erkennungsgeschwindigkeit beitragen können.

Es wurden Methoden untersucht, nach verschiedenen Kriterien einzelne Referenzvektoren aus den Codebüchern zu entfernen, eine Kopplung von Kovarianzmatrizen durchzuführen und durch Radialisierung von Kovarianzmatrizen Parameter einzusparen. Am besten erwies sich dabei die Radialisierung von Kovarianzmatrizen, bei denen jede Kovarianzmatrix durch eine Diagonalmatrix gleicher Determinante ersetzt wird, bei der aber alle Diagonalelemente denselben Wert haben. Bei einer Radialisierung aller Matrizen im Erkenner kann damit der Parameterraum nahezu halbiert werden, wobei nur leichte Steigerungen in der Fehlerrate in Kauf zu nehmen sind. Bei einer Radialisierung von ca. zwei Dritteln aller Matrizen kann sogar eine Verbesserung der Erkennungsrate um etwa relativ 8% erzielt werden. Der Vorteil der Verwendung radialer Kovarianzmatrizen besteht nicht nur darin, daß Parameter eingespart werden, sondern auch in der Verringerung des Auftretens der schädlichen Beinahe-Singularitäten.

Die Probleme, die volle Kovarianzmatrizen mit sich bringen, werden in den meisten Spracherkennern dadurch umgangen, daß diagonale Matrizen verwendet werden. Dadurch werden weniger Parameter gebraucht, die besser geschätzt werden können und einen Geschwindigkeitsvorteil bei der Berechnung der Emissionswahrscheinlichkeiten mit sich bringen. Einen Schritt weiter in diese Richtung gehen radiale Kovarianzmatrizen. Die Verwendung radialer Matrizen ist aus der Literatur bekannt im Zusammenhang mit Aktivierungen von Neuronen in neuronalen Netzen (Radial Basis Functions [Pow85]).

Das Prinzip der Prameterraumkompaktifizierung durch radiale Kovarianzmatrizen läßt sich auf alle Spracherkenner anwenden, die Gauß-Mischverteilungen verwenden. Inwiefern sich die Auswirkungen, insbesondere die Gewinne in der Erkennungsleistung, die in der vorliegenden Arbeit beobachtet wurden, auch auf andere Erkenner mit anderen Architekturen

übertragen lassen, muß erst noch untersucht werden. Prinzipiell ist jedoch festzuhalten, daß die Radialisierung auch auf die Erkennung anderer als diktierter Sprache anwendbar ist, weil man auch dort beobachten kann, daß Beinahe-Singularitäten oft zu Fehlern führen.

Ausblick

Die Forschung in der Spracherkennung ist sehr stark zielorientiert und weniger neugierorientiert. Eine neue Idee wird in der Regel mehr an ihrer Erfolgsaussicht als an ihrer Einfachheit oder Erfindungshöhe gemessen. Dies und das Risiko, durch Umsteigen auf völlig neue Paradigmen bei Vergleichen mit anderen Spracherkennern zunächst schlecht abzuschneiden, lenken die Entwicklung in der Spracherkennung in den letzten Jahren dahin, daß immer neue Verfahren untersucht werden, die einen konkreten Aspekt der HMM-Erkenner betreffen, und diesen zu optimieren versuchen. Wegen der Schnellebigkeit der Spracherkennung und der relativ hohen Anforderungen an die Rechnerkapazitäten ist es meist nicht möglich, alle Kombinationen von Optimierungsmethoden einzusetzen und auszuwerten. Noch bevor alle interessanten Fragen zu eingesetzten Algorithmen geklärt werden können, bieten sich neue Algorithmen und neue Optimierungsmöglichkeiten an. Auch wenn dies aus Forschersicht bedauerlich ist, so hat diese erfolgs- und praxisbezogene Art der Forschung in den letzten Jahren die Spracherkennung sehr weit vorangebracht. Für die Zukunft ist zu erwarten, daß weiterhin auf diese Art geforscht wird. So werden neben den Versuchen, sich Fragen zu widmen, die in der vorliegenden Arbeit unbeantwortet blieben, gleichzeitig neue Ansätze verfolgt werden.

Für die Ballung von kontextabhängigen Modellen existieren zahlreiche Ideen. Die interessantesten laufen darauf hinaus, einzelnen HMM-Zuständen keine konkreten akustischen Modelle zuzuordnen, sondern je nach Zustand des Erkenners eine Mischung verschiedener Modelle. Was die Kompaktifizierung der Parameter angeht, muß festgestellt werden, daß Erfahrung immer wieder gezeigt hat, daß durch die Verwendung größerer Parametermengen bei gleichzeitigem Vorhandensein größerer Mengen an Trainingsdaten die Erkennungsleistung gesteigert werden kann. Das heißt, auch wenn mit der Zeit die Kapazitäten der Rechner in den Forschungslabors steigen werden, so wird dennoch weiterhin ein Interesse daran bestehen, Parameterräume möglichst kompakt zu gestalten. Zur Verbesserung der Erkennungsraten

liegen auch reichlich Ideen vor. Viele gehen in die Richtung der Adaption, oder mit anderen Worten in Richtung auf das Trainieren mit den zu erkennenden Aufnahmen. Auch hybride Ansätze, bei denen neben Gauß-Mischverteilungen auch Neuronale Netze verwendet werden [FFW97], werden an Bedeutung gewinnen.

Diktieren als eine spezielle Form der Spracherkennung wird in den nächsten Jahren tendenziell aus der universitären Forschung in die industrielle verlagert, weil zum einen die Fördergelder immer mehr für die Erkennung spontaner Sprache unter widrigen Bedingungen gegeben werden, und zum anderen die Entwicklung einsetzbarer Produkte zum Diktieren im Interesse der Industrie ist.

Die in dieser Arbeit gewonnenen Erkenntnisse lassen sich übertragen auf die Erkennung nicht diktierter Sprache wie spontane Telefongespräche oder Bedienung von Geräten zum Beispiel im fahrenden Kraftfahrzeug, solange dabei die Erkennung kontinuierlicher Sprache mit nicht beschränktem Vokabular gewünscht wird. Bei kleinen beschränkten Vokabularen oder bei der Erkennung isolierter Wörter oder Kommandos wird man weiterhin wie bisher weniger allgemeine und mehr auf die konkrete Anwendung spezialisierte Lösungen suchen. Die Problematik bei Nicht-Diktiersystemen liegt im Gegensatz zu den Diktiersystemen weniger in der Gestaltung der Parameterräume, sondern vielmehr in der Signalvorverarbeitung und der Anpassung der Erkenner an die „neuen" Aufnahmebedingungen. Die Problematik des Entwurfs des Parameterraums wird aber eine Rolle spielen, wenn es darum geht, daß Nicht-Experten für konkrete Anwendungen einen Spracherkenner bauen wollen. So gibt es heute viele Tätigkeiten im Umgang mit Maschinen und Computeranwendungen oder Arbeiten, bei denen die Hände entweder nicht frei sind oder keine Hilfsmittel wie Tastaturen vorhanden sind, die alle ergonomischer gestaltet oder effizienter erledigt werden könnten, wenn eine automatische Spracherkennung eingesetzt würde. Da es auf absehbare Zeit keine Universalerkenner geben wird, die in jeder Umgebung und unter allen Bedingungen zufriedenstellend arbeiten, wird es nötig sein, daß Techniker, die keine Ausbildung in der Spracherkennung haben, selbständig für die von ihnen gewünschte Anwendung einen Spracherkenner bauen. Für solche Aufgaben ist eine automatische Gestaltung des Parameterraumes wünschenswert.

Literaturverzeichnis

[ADNS94] **X. Aubert, C. Dugast, H. Ney und V. Steinbiss**. Large Vocabulary Continuous Speech Recognition of Wall Street Journal Corpus. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 129–132. IEEE, April 1994, Adelaide, Australia.

[BE67] **L. E. Baum und J. A. Egon**. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology. *Bulletin of the American Meteorological Society*, 73:360–363, 1967.

[BS68] **L. E. Baum und G. R. Sell**. Growth Function for Transformations on Manifolds. *Pac. J. Math.*, 27(2):211–227, 1968.

[DBB52] **K. Davis, R. Biddulph und S. Balashek**. Automatic Recognition of Spoken Digits. *Journal of the Acoustic Society of America (JASA)*, 24:637–642, 1952.

[DGDP96] **N. Deshmukh, R. Ganapathiraju, R. J. Duncan und J. Picone**. Human Speech Recognition Performance on the 1995 HUB-3 Corpus. In *DARPA Speech Recognition Workshop*, S. 129–134. Morgan Kaufmann, February 1996, Arden House, New York.

[DH73] **R. O. Duda und P. E. Hart**. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Signapore, 1973.

[DLR77] **A. P. Dempster, N. M. Laird und D. B. Rubin**. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[DM90] **S. B. Davis und P. Mermelstein**. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In A. Waibel und K.-F. Lee, Herausgeber, *Readings in Speech Recognition*, S. 65–74. Morgan Kaufmann, 1990.

[EP95] **W. J. Ebel und J. Picone**. Human Speech Recognition Performance on the 1994 CSR Spoke 10 Corpus. In *DARPA Speech Recognition Workshop*, S. 53–59. Morgan Kaufmann, January 1995, Austin, Texas.

[FF59] **J. Forgie und C. Forgie**. Results Obtained from a Vowel Recognition Computer Program. *Journal of the Acoustic Society of America (JASA)*, 31(11):1480–1489, 1959.

[FFW97] **J. Fritsch, M. Finke und A. Waibel**. Context-Dependent Hybrid HME/HMM Speech Recognition using Polyphone Clustering Decision Trees. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 3, S. 1759–1762. IEEE, April 1997, München.

[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. IEEE, April 1997, München.

[Fin96] **M. Finke**. Mode Dependent Pronunciation Modeling in LVCSR. In *Proceedings of the Summer Workshop at the Center for Language and Speech Processing*, (keine Seitennumerierung), 1996, Johns Hopkins University, Baltimore.

[For73] **G. D. Forney, Jr.** The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[FR96] **J. Fritsch und I. Rogina**. The Bucket Box Intersection Algorithm for Fast Approximative Evaluation of Diagonal Mixture Gaussians. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, May 1996, Atlanta, USA.

[FR97] **M. Finke und I. Rogina**. Wide Context Acoustic Modeling in Read vs. Spontaneous Speech. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 3, S. 1743–1746. IEEE, April 1997, München.

[FRSA95] **J. Fritsch, I. Rogina, T. Sloboda und W. A**. Speeding Up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Itersection Algorithm. In *EUROSPEECH95*, S. 1091–1094, Madrid, September 1995.

[Fry59] **D. Fry**. Theoretical Aspects of Mechanical Speech Recognition. *Journal of the Britisch Institute for Radio Engineering*, 19:211–219, 1959.

[FZ96] **M. Finke und T. Zeppenfeld**. LVCSR Switchboard April 1996 Evaluation Report. In *Proceedings of the Large Vocabulary Conversational Speech Recognition Hub 5 Workshop*, (keine Seitennumerierung), April 1996, Maritime Institute of Technology, Linthicum Heights, Maryland.

[Gon95] **Y. Gong**. Speech recognition in noisy environments: A survey. *Speech Communication*, 16:261–291, 1995.

[GSK+95] **P. Geutner, B. Suhm, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczyna und A. Waibel**. Integrating Different Learning Approaches into a Multilingual Spoken Language Translation System. In *JCAI-Workshop on New Approaches to Learning for Natural Language Processing*, August 1995, Montreal, Canada.

[HAJ90] **X. Huang, Y. Ariki und M. D. Jack**. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[HCR+95] **M. M. Hochberg, G. Cook, S. Renals, A. Robinson und R. Schechtman**. The 1994 ABBOT Hybrid Connectionist-HMM Large Vocabulary Recognition System. In *ARPA Workshop on Spoken Language Systems Technology*, S. 170–175. Morgan Kaufmann, January 1995, Austin, Texas.

[HH91] **M.-Y. Hwang und X. Huang**. Shared-Distribution Hidden Markov Models for Speech Recognition. Technical Report CMU-CS-91-124, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, April 1991.

[HJ89] **X. Huang und M. Jack**. Semi-continuous hidden Markov models for speech signals. *Computer, Speech and Language*, 3:239–251, 1989.

[Hon92] **H.-W. Hon**. Vocabulary-Independent Speech Recognition: The VOCIND System. Technical Report CMU-CS-92-108, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, March 1992.

[Ita75] **F. Itakura**. Minimum Prediction Residual Principle Applied to Speech Recognition. In *IEEE Transactions on Acoustic, Speech, Signal Processing*, Band 23, S. 67–72, 1975.

[Jel90] **F. Jelinek**. Self-Organized Language Modeling for Speech Recognition. In A. Waibel und K.-F. Lee, Herausgeber, *Readings in Speech Recognition*, S. 450–503. Morgan Kaufmann, 1990.

[JSD+96] **U. Jain, M. A. Siegler, S.-J. Doh, E. Gouvea, J. Huerta, P. J. Moreno, B. Raj und R. M. Stern**. Recognition of Continuous Broadcast News with Multiple Unknown Speakers and Environments. In *DARPA Speech Recognition Workshop*, S. 61–66. Morgan Kaufmann, February 1996.

[KBC88] **T. Kohonen, G. Barna und R. Chrisley**. Statistical Pattern Recognition with Neural Networks: Benchmarking Studies. Technical Report SF-02150, Helsinki University of Technology, Espoo, Finland, 1988.

[Kem95] **T. Kemp**. Data-Driven Codebook Adaptation in Phonetically Tied SCHMMs. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 377–479. IEEE, May 1995, Detroit, USA.

[KJ96] **T. Kemp und A. Jusek**. Modeling Unknown Words in Spontaneous Speech. In *Proc. IEEE International Conference on*

Acoustics, Speech, and Signal Processing, Band 1, S. 533–536. IEEE, May 1996, Atlanta, USA.

[Kla77] **D. Klatt**. Review of the ARPA Speech Understanding Project. *Journal of the Acoustic Society of America (JASA)*, 62:1345–1366, 1977. (Siehe auch [WL90], S. 554-575).

[Koh90] **T. Kohonen**. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[KRR96] **D. Kershaw, A. Robinson und S. Renals**. The 1995 Abbot Hybrid Connectionist-HMM Large-Vocabulary Recognition System. In *ARPA Workshop on Spoken Language Technology*. Morgan Kaufmann, 1996.

[KS73] **D. Klatt und K. Stevens**. On the Automatic Recognition of Continuous Speech: Implications from a Spectrogram Reading Experiment. *IEEE Transactions on Audio Electroacoustics*, 21:210–217, 1973.

[Lee88] **K.-F. Lee**. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*. CMU-CS-88-148, Carnegie Mellon University, Pittsburgh, PA, April 1988.

[LW94] **C. Leggetter und P. Woodland**. Speaker Adaptation of Continuous Density HMMs Using Linear Regression. In *Proceedings of the International Conference on Speech and Language Processing*, Band 2, S. 451–454, 1994, Yokohama.

[LW95] **C. Leggetter und P. Woodland**. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, 9:171–185, 1995.

[LWL+97] **A. Lavie, A. Waibel, L. Levin, M. Finke, D. Gates, M. Gavalda, T. Zeppenfeld und Z. Puming**. JANUS-III: Speech-to-Speech Translation in Multiple Languages. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 99–102. IEEE, April 1997, München.

[MBS93] **T. Martinez, S. Berkovic und K. Schulten**. Neural Gas Network for Vector Quantization and its Applications to Time Series Prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, Jul 1993.

[MOM+96] **T. Matsuoka, K. Ohtsuki, T. Mori, S. Furui und K. Shirai**. Large-Vocabulary Continuous-Speech Recognition Using a Japanese Business Newspaper. In *DARPA Speech Recognition Workshop*, S. 137–142. Morgan Kaufmann, February 1996.

[OAM+92] **L. Osterholtz, C. Augustine, A. McNair, I. Rogina, H. Saito, T. Sloboda, J. Tebelskis, A. Waibel und M. Woszczyna**. Testing Generality in JANUS: A Multi-Lingual Speech Translation System. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1992.

[OB56] **H. Olson und H. Belar**. Phonetic Typewriter. *Journal of the Acoustic Society of America (JASA)*, 28(6):1072–1081, 1956.

[Ode92] **J. J. Odell**. The Use of Decision Trees with Context Sensitive Phoneme Modelling. Diplomarbeit, Department of Engineering, Cambridge University, Cambridge, UK, August 1992.

[PFGP96] **D. S. Pallett, J. G. Fiscus, J. S. Garofolo und M. A. Przybocki**. 1995 HUB-4 Dry Run Broadcast Materials Benchmark Tests. In *DARPA Speech Recognition Workshop*, S. 12–25. Morgan Kaufmann, February 1996.

[PGF+95] **D. S. Pallett, F. J. G., W. M. Fisher, J. S. Garofolo, B. A. Lund, A. Marin und M. A. Przybocki**. 1994 Benchmark Tests for the ARPA Spoken Language Programm. In *ARPA Workshop on Spoken Language Systems Technology*, S. 5–36. Morgan Kaufmann, January 1995, Austin, Texas.

[Pow85] **M. J. D. Powell**. Radial basis functions for multivariate interpolation: A review. Technical Report DAMPT 1985/NA12, Department of Applied Mathematics and Theoretical Physics, Cambridge University, 1985.

[Rab89] **L. Rabiner**. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*. IEEE, 1989. (Auch in [WL90], S. 267–296).

[RLRW79] **L. Rabiner, S. Levinson, A. Rosenberg und J. Wilpon**. Speaker-Independent Recognition of Isolated Words Using Clustering Techniques. In *IEEE Transactions on Acoustic, Speech, Signal Processing*, Band 27, S. 336–349, 1979.

[Rog97] **I. Rogina**. Automatic Architecture Design by Likelihood-Based Context Clustering with Crossvalidation. In *Eurospeech*, September 1997, Rhodos.

[RW94] **I. Rogina und W. Waibel**. Learning State-Dependent Stream Weights for Multi-Codebook HMM Speech Recognition Systems. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 217–220. IEEE, April 1994, Adelaide, Australia.

[RW95] **I. Rogina und A. Waibel**. The JANUS Speech Recognizer. In *ARPA Workshop on Spoken Language Systems Technology*, S. 166–169. Morgan Kaufmann, January 1995, Austin, Texas.

[Sch91] **O. Schmidbauer**. An LVQ based Reference Model for Speaker-Independent and Adaptive Speech Recognition. Technical report, Siemens AG, ZFE IS KOM 3, Otto-Hahn-Ring 6, 8000 München 83, Germany, September 1991.

[SGK+95] **B. Suhm, P. Geutner, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczyna und A. Waibel**. JANUS: Towards Multilingual Spoken Language Translation. In *ARPA Workshop on Spoken Language Systems Technology*, S. 221–226. Morgan Kaufmann, January 1995, Austin, Texas.

[SR95] **T. Schultz und I. Rogina**. Acoustic and Language Modeling of Human and Nonhuman Noises for Human-To-Human Spontaneous Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 1, S. 293–296. IEEE, May 1995, Detroit, USA.

[SRW95] **T. Schultz, I. Rogina und A. Waibel**. Experiments with LVCSR-based Language Identification. In *Proceedings of the Speech Research Symposium SRS XV*, S. 89–94, June 1995, Baltimore, USA.

[SRW96] **T. Schultz, I. Rogina und A. Waibel**. LVCSR-based Language Identification. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 871–874. IEEE, May 1996, Atlanta, USA.

[ST92] **O. Schmidbauer und J. Tebelskis**. An LVQ Based Reference Model for Speaker-Adaptive Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, March 1992. IEEE.

[ST95] **E. Schukat-Talamazzini**. *Automatische Spracherkennung*. Vieweg, Braunschweig, Germany, 1995.

[SWW93] **B. Suhm, M. Woszczyna und A. Waibel**. Detection and Transcription of New Words. In *EUROSPEECH93*, Berlin, September 1993.

[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*. IEEE, April 1990.

[TWPS91] **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*. IEEE, May 1991.

[Vit67] **A. J. Viterbi**. Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[WAWB+94] **M. Woszczyna, N. Aoki-Waibel, F. Buø, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose, T. Schultz, B. Suhm, M. Tomita und A. Waibel**. JANUS94: Towards Spontaneous

Speech Translation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* IEEE, April 1994, Adelaide, Australia.

[WCE+93a] **M. Woszczyna, N. Coccaro, A. Eisele, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose, T. Sloboda, M. Tomita, J. Tsutsumi, N. Aoki-Waibel, A. Waibel und W. Ward**. Recent Advances in JANUS: A Speech Translation System. In *ARPA Workshop on Human Language Technology Systems.* Morgan Kaufmann, March 1993.

[WCE+93b] **M. Woszczyna, N. Coccaro, A. Eisele, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose, T. Sloboda, M. Tomita, J. Tsutsumi, N. Aoki-Waibel, A. Waibel und W. Ward**. Recent Advances in JANUS: A Speech Translation System. In *ARPA Workshop on Human Language Technology Systems.* Morgan Kaufmann, March 1993.

[WE92] **W. Wahlster und J. Egelkamp**. Wissenschaftliche Ziele und Netzpläne für das VERBMOBIL Projekt. DFKI Saarbrücken, April 1992.

[Wei96] **C. Weinstein**. ARPA Speech Research: New Directions. In *DARPA Speech Recognition Workshop*, S. 154–154. Morgan Kaufmann, February 1996.

[Wel95] **B. Welch**. *Practical Rogramming in Tcl and Tk.* Prentice Hall, 1995.

[WF96] **M. Woszczyna und M. Finke**. Minimizing Search Errors due to Delayed Bigrams in Real-Time Speech Recognition Systems. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, S. 137–140. IEEE, May 1996.

[WFG+96] **A. Waibel, M. Finke, D. Gates, M. Gavalda, T. Kemp, A. Lavie, M. Maier, L. Mayfield, A. McNair, I. Rogina, K. Shima, T. Sloboda, M. Woszczyna, P. Zhan und T. Zeppenfeld**. JANUS II - Advances in Spontaneous Speech Translation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* IEEE, May 1996, Atlanta, USA.

[WGPV96] **P. Woodland, M. Gales, D. Pye und V. Valtchev**. The HTK Large Vocabulary Recognition System for the 1995 ARPA H3 Task. In *DARPA Speech Recognition Workshop*, S. 99–104. Morgan Kaufmann, February 1996.

[WL90] **A. Waibel und K. Lee**. *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, CA., 1990.

[WOVY94] **P. Woodland, J. Odell, V. Valtchev und S. Young**. The HTK Large Vocabulary Recognition System: An Overview. In *ARPA Spoken Language Technology Workshop*, (keine Seitennumerierung), March 1994, Princeton, New Jersey.

[You96] **S. Young**. Large Vocabulary Continuous Speech Recognition: a Review. Technical report, Cambridge University Engineering Department, Trumpington Street, Cambridge, CB21PZ, UK, April 1996.

[You97] **S. Young**. Persönliches Gespräch auf der *International Conference on Acoustic, Speech, and Signal Processing* in München, April 1997.

[YW93] **S. Young und P. Woodland**. *HTK: Hidden Markov Model Toolkit*. Entropic Cambridge Research Laboratory, Compass House, 80-82 Newmarket Road, Cambridge CB5 8DZ, England, February 1993.

[ZFR+97] **T. Zeppenfeld, M. Finke, K. Ries, M. Westphal und A. Waibel**. Recognition of Conversational Telephone Speech using the Janus Speech Engine. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 3, S. 1815–1818. IEEE, April 1997, München.

[ZW97] **P. Zhan und M. Westphal**. Speaker Normalization Based on Frequency Warping. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Band 2, S. 1039–1042. IEEE, April 1997, München.