Semikontinuierliche Hidden-Markov-Modelle mit mehreren Kodebüchern

Diplomarbeit im Fach Informatik

vorgelegt
von

Christian Hacker

Geboren am 08.09.1975 in Amberg

Angefertigt am

Lehrstuhl für Mustererkennung (Informatik 5)
Institut für Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg.

Betreuer: Dipl.Inf. Georg Stemmer, Dr.-Ing. Elmar Nöth

Beginn der Arbeit: 07.03.2002

Abgabe der Arbeit: 09.09.2002

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien des Lehrstuhls für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Erlangen, den 2. Oktober 2002

Übersicht

Zur Zeit der Anfertigung der vorliegenden Arbeit wird am LME ein Spracherkenner verwendet, der auf semikontinuierlichen Hidden-Markov Modellen basiert. Ein einziges Kodebuch, in dem die Klassen durch Normalverteilungen approximiert werden, wird von allen Zuständen geteilt. Lediglich die Gewichtungen der Normalverteilungsdichten sind zustandsspezifisch. Bei einem kontinuierlichen HMM dagegen liegt für jeden Zustand ein separates Kodebuch vor.

In der vorliegenden Arbeit wird die Implementierung so erweitert, dass man sich einen Schritt in Richtung kontinuierliches HMM bewegt. Es werden nun verschiedene Kodebücher für unterschiedliche Merkmaltypen trainiert, z.B. separat für statische und dynamische Merkmale. In einem weiteren Ansatz werden Kodebüchern für verschiedene Lautoberklassen bereitgestellt.

Kontinuierliche HMM gelten bei ausreichend Trainingsmaterial als besser; von der vorliegenden Arbeit verspricht man sich einen Schritt in diese Richtung zu machen und gleichzeitig die Anzahl der zu schätzenden Parameter nur maßvoll zu vergrößern. Evaluiert wird der Ansatz mit der EVAR-Stichprobe. Ferner wird ein VERBMOBIL-Erkenner trainiert und die Robustheit gegenüber Hall untersucht.

Kurzfassung

In der Diplomarbeit werden Hidden-Markov Modelle mit mehreren Kodebüchern in das Spracherkennungssystem des LME integriert. Kodebücher werden für unterschiedliche Merkmaltypen sowie für unterschiedliche Lautoberklassen berechnet.

Ein Hidden-Markov Modell (HMM) ist ein stochastisches Modell eines Wortes oder einer Wortuntereinheit. Es besteht aus einer Menge von Zuständen, Übergangswahrscheinlichkeiten und Ausgabeverteilungen. In jedem Zustand erfolgt eine beobachtbare Ausgabe, der genaue Weg durch das HMM bleibt jedoch verborgen.

Aus dem Sprachsignal einer Äußerung wird eine Folge von Merkmalvektoren berechnet. Im Referenzsystem werden 12 statische und 12 dynamische Merkmale berechnet. Bei diskreten HMM werden die Merkmalvektoren zunächst quantisiert. Ein Kodebuch unterteilt den Merkmalraum in Klassengebiete, das durch Normalverteilungen modelliert wird. Der Vektorquantisierer bildet jeden Merkmalvektor auf eine Klassennummer ab. Aus dem Sprachsignal erhalten wir so eine Folge von diskreten Werten. Diese entspricht der Ausgabe des diskreten HMM.

Den Informationsverlust, der durch die Vektorquantisierung entsteht, vermeidet man beim kontinuierlichen HMM (CHMM). Nun werden von den HMM-Zuständen Merkmalvektoren ausgegeben. Die Ausgabeverteilungen sind zustandsspezifisch. Jeder Zustand hat also ein eigenes Kodebuch. Mit dem CHMM erreicht man bessere Erkennungsraten als mit dem diskreten HMM, wenn ausreichend Trainingsmaterial zur Verfügung steht.

Auch vom semikontinuierlichen HMM (SCHMM) werden Merkmalvektoren ausgegeben. Es steht aber nur ein einziges Kodebuch zur Verfügung, das von allen Zuständen geteilt wird. Die einzelnen Dichten des Kodebuches werden mit zustandsspezifischen Gewichten multipliziert. Bei wenig Trainingsmaterial werden beim SCHMM bessere Erkennungsraten erzielt; mit hinreichend vielen Daten ist das kontinuierliche HMM aber besser.

Im ersten Teil dieser Diplomarbeit wird der Spracherkenner, der auf SCHMM basiert, so erweitert, dass verschiedene Kodebücher für unterschiedliche Typen von Merkmalen (z.B. statische und dynamische Merkmale) verwendet werden. Wenn ein Zustand eine Ausgabe erzeugt, müssen alle Kodebücher gleichzeitig berücksichtigt werden. Es wird angenommen, dass die Merkmaltypen voneinander stochastisch unabhängig sind. Im zweiten Teil der Arbeit wird von jedem HMM-Zustand nur ein Kodebuch betrachtet. Nun verwenden Zustände verschiedener Lautoberklassen unterschiedliche Kodebücher. Durch beide Ansätze können die zustandsspezifischen Ausgabeverteilungen flexibler modelliert werden.

Die Untersuchungen erfolgen mit einer Teilmenge der EVAR-Stichprobe mit einer Gesamtlänge von 8 Stunden Sprache. In den Experimenten zur Hallrobustheit werden die

VERBMOBIL-Stichprobe und Daten aus dem Müdigkeitsexperiment verwendet. Jeder Erkenner wird mit dem ISADORA-System trainiert und anschließend mit dem LRBEAM-Erkenner getestet. Im Referenzsystem mit 256 Kodebuchklassen werden 74.4 % WA erzielt.

In den Untersuchungen mit mehreren Kodebüchern für unterschiedliche Merkmaltypen hat sich das heuristische Vorgehen bewährt, die Kodebücher mit sogenannten Kodebuchexponenten α zu gewichten ($0 \leq \alpha \leq 1$). Das wichtigere Kodebuch erhält einen höheren Exponenten. Zunächst trainieren wir einen Erkenner mit 2×128 Klassen für statische Merkmale und Ableitungen ohne Gewichte ($\alpha = 1$) und optimieren die Kodebuchexponenten zum Testen auf der Validierungsstichprobe. Auf der Teststichprobe erreichen wir so eine Wortakkuratheit von 76.9 %. Nun wird das System erneut mit den optimalen Kodebuchexponenten trainiert. So kann die Erkennungsrate auf 77.9 % WA gesteigert werden. Weitere Experimente werden mit verschiedener Anzahl an Kodebuchklassen durchgeführt, sowie mit separaten Kodebüchern für Energie, zweite Ableitungen und erste Ableitungen aus verschiedenen Zeitauflösungen.

Um die Robustheit dieses Ansatzes mit zwei Kodebüchern gegenüber Hall zu testen, benötigen wir zwei Teststichproben. Eine wurde mit einem Nahbesprechungsmikrophon aufgenommen, die andere mit einem verrauschten Raummikrophon. Mit dem Nahbesprechungsmikrophon ist der 2-Kodebuch-Ansatz wieder besser. Auf den verrauschten Aufnahmen konnten die Erkennungsrate aber nicht gesteigert werden.

In den Experimenten mit Kodebüchern für verschiedene Lautoberklassen werden 47 Kodebücher mit je 25 Klassen trainiert. Zunächst werden die Kodebücher zu einem einzigen zusammengefasst. Wir erhalten 1175 Kodebuchklassen und erzielen 77.6 % WA. Nach der Neuschätzung verschlechtert sich die Erkennung, was wohl daran liegt, dass zu wenig Trainingsdaten für das riesige Kodebuch verwendet werden. Nun werden die 47 Kodebücher getrennt verwendet. Die Erkennungsraten brechen stark ein. Die Ursache dafür liegt wohl an der festen Klassenzahl pro Kodebuch; so ist das System sehr unflexibel. In einem rein datengetriebenen Optimierungsverfahren konnte nachgewiesen werden, dass für Zustände verschiedener Oberklassen eine sehr unterschiedliche Anzahl von Kodebuchklassen favorisiert wird. Dies erfordert weitere Experimente in Zukunft.

Zusammenfassen lässt sich sagen, dass die Erweiterung der Erkenners für mehrere Kodebücher eine deutliche Reduzierung der Fehlerraten um 12 % relativ mit sich bringt.

Summary

In the diploma thesis a multiple codebook approach for Hidden-Markov models is integrated in the speech recognition system at the LME. Codebooks are computed for different kinds of features and for different clusters of similar acoustic events.

A Hidden-Markov model (HMM) is model of a word or a subunit of a word. It consists of a set of states, transition probabilities and output probabilities. Observable outputs are produced by the states. The path through the HMM is hidden.

From a speech signal of an utterance a sequence of feature vectors is computed. In the baseline system 12 static features and 12 derivatives are used. In the case of a discrete HMM feature vectors are quantized at first. A codebook partitions feature space into several classes, which are modeled by Gaussian density functions. The vector quantizer assigns to each feature vector a classnumber. In this way we obtain a sequence of discrete values, which corresponds to the output of the discrete HMM.

We can avoid the loss of information caused by the vector quantization, if we use the continuous HMM (CHMM). Now the outputs of the HMM states are feature vectors. The observation is created by state dependent probability density functions. Each state has its own codebook. With the CHMM the word recognition rates are improved in comparision with the discrete HMM, if enough training data is available.

The outputs on semi-continuous HMM (SCHMM) are feature vectors again, but only one codebook is used. This codebook is shared by all states. The densities of the codebooks are multiplied by state dependent weights. With a small number of training data SCHMM achieve better performance. If enough data is available, the CHMM in better.

In the first part of the diploma thesis the recognizer, which is based on SCHMM, is upgraded for multiple codebooks for different kinds of features (e.g. static and dynamic features). If an output is produced by the HMM state, all codebooks are taken into account. We assume, that different kinds of features are stochastically independent. In the second part for each states only one codebooks is used. Now states with different subphonetic labels have different codebooks. Both approaches provide higher flexibility in order to model state dependent probability density functions.

Acoustic models are trained on a part of the EVAR data set with a total amount of 8 hours speech. In some tests for the robustness of the recognizer against reverberation a part of the VERBMOBIL data set is used as well as data from the fatigue experiment. The speech recognizer is trained with the ISADORA-System and tested with the LRBEAM-recognizer. In the baseline system with 256 classes we achieve a word recognition rate of 74.4 %.

In experiments with multiple codebooks for different kinds of features performance is improved by the following heuristic approach: the codebooks are weighted by so called codebook exponents α ($0 \leq \alpha \leq 1$). More important codebooks are weighted by higher exponents. Firstly, we train some system with 2×128 codebook classes for static features and their derivatives without weighting the codebooks ($\alpha = 0$). We optimize the exponents for the test on the validation data set. After that we achieve an accuracy of 76.9 % on the test data set. Now the system is trained again with the optimal codebook exponents. Thus we can increase the word recognition rate to 77.9 %. Further experiments are conducted with several numbers of codebook classes as well as with separate codebooks for the energy, the second order derivative and the first order derivatives in different time resolutions.

In order to test the robustness of the recognizer against reverberation we use two test data sets. One is recorded with a close talk microphone to avoid reverberation, the other with a space microphon. Again, we achieve with the two codebook approach better performance on the first test set. But with reverberated data the recognition rate could not be increased.

In experiments with codebooks for different clusters of similar acoustic events 47 codebooks with 25 classes each are computed. First we merge the codebooks to one single codebook. The word recognition rate on this system with 1175 classes rises on 77.6 %. After codebook reestimation accuracy gets worse, because not sufficient training data is used for the estimation of this huge number of parameters. If we use 47 codebooks separately, the recognition rate drops steeply. The reason seems to be, that each codebook has exact 25 classes. This causes the system to be very inflexible. In a data driven approach can be verified, that states with different subphonetic labels prefer a very unequal numbers of codebook densities. Further experiments have to be done in future.

Summing up, by the upgrade of the speech recognition system with a multiple codebook approach the word error rate is reduced by 12 % relative.

Inhaltsverzeichnis

1 Einleitung **13**

1.1 Mensch-Maschine-Kommunikation in Dialogsystemen 13
1.2 Automatische Spracherkennung 14
1.3 Hidden-Markov Modelle 16
1.4 Zielsetzung der Arbeit und Beitrag zur Forschung 17
1.5 Aufbau der Diplomarbeit 20

2 HMM basierte Spracherkennung **21**

2.1 Klassifikation von Mustern 21
2.1.1 Optimale Entscheidung 22
2.1.2 Schätzen von Normalverteilungen 23
2.1.3 Der EM-Algorithmus 25
2.2 Vorgehensweise in der Spracherkennung 26
2.2.1 Merkmalberechnung 26
2.2.2 Vektorquantisierung 28
2.3 Hidden-Markov Modelle 29
2.3.1 Formalisierung 30
2.3.2 Diskrete HMM 31
2.3.3 Kontinuierliche HMM 34
2.3.4 Semikontinuierliche HMM 36

3 HMM mit mehreren Kodebüchern **39**

3.1 Getrennte Behandlung verschiedener Merkmaltypen 39
3.1.1 Vorgehensweise 40
3.1.2 Literaturüberblick 43
3.2 Getrennte Behandlung verschiedener Lautoberklassen 45

3.2.1 Vorgehensweise . . . 45
3.2.2 Literaturüberblick . . . 48

4 Versuchsaufbau und Referenzsystem 51
4.1 Die Stichprobe . . . 51
4.2 Bewertung . . . 52
4.3 Training und Test . . . 53
4.3.1 Training mit ISADORA . . . 53
4.3.2 Test mit LRBEAM-Erkenner . . . 56
4.4 Untersuchung des Referenzsystems . . . 57

5 Kodebücher für verschiedene Merkmaltypen 61
5.1 Vorgehensweise . . . 61
5.2 Experimente mit zwei Kodebüchern . . . 64
5.2.1 Training ohne Gewichtung der Kodebücher . . . 64
5.2.2 Variation der Klassenzahl für die Kodebücher . . . 69
5.2.3 Training mit Gewichtung der Kodebücher . . . 73
5.2.4 Zweites Training mit optimalen Gewichten . . . 74
5.3 Experimente mit mehreren Kodebüchern . . . 76
5.3.1 Separates Kodebuch für Energiemerkmale . . . 76
5.3.2 Separates Kodebuch für zweite Ableitungen . . . 79
5.3.3 Ableitungen in verschiedenen Auflösungen . . . 81
5.4 Untersuchungen zur Robustheit gegen Hall . . . 85
5.5 Zusammenfassung . . . 87

6 Kodebücher für verschiedene Lautoberklassen 89
6.1 Vorgehensweise . . . 89
6.2 Experimente . . . 92
6.2.1 Experimente mit einem phonetischen Mischungsverteilungskodebuch . . . 92
6.2.2 Der Multi-Kodebuch Ansatz . . . 93

7 Ausblick 97

8 Zusammenfassung 101

A Berechnung von $\zeta_t(j,k)$ 105

Verzeichnis der Bilder **107**

Verzeichnis der Tabellen **109**

Literaturverzeichnis **113**

Kapitel 1

Einleitung

1.1 Mensch-Maschine-Kommunikation in Dialogsystemen

Maus oder Tastatur in der einen Richtung und Bildschirm oder Display in der anderen sind bei weitem nicht die einzigen Möglichkeiten zur Mensch-Maschine-Kommunikation. Schneller zur Hand ist in vielen Situationen ein Telefon. Warum also nicht einfach den Rechner anrufen, der gerade weiß, wann der nächste Bus fährt, welcher Film im Kino läuft oder wie die Aktien stehen?

Sprache ist also ebenfalls ein Mittel zur Mensch-Maschine-Kommunikation. Nur eine von vielen Anwendungen der Spracherkennung sind Dialogsysteme. Am Lehrstuhl für Mustererkennung (LME) der Universität Erlangen-Nürnberg wurde das automatische Dialogsystem EVAR (**E**rkennen,**V**erstehen,**A**ntworten,**R**ückfragen) entwickelt, ein Zugfahrplan-Auskunft-System [Gal98]. Von Sympalog, einer Ausgründung des LME, wurde der Dialogmanager von EVAR völlig neu implementiert und aufgaben- sowie sprachunabhängig gestaltet. Mehrere verschiedene Dialogsysteme wurden daraus entwickelt, eines davon ist FRÄNKI, das **fränk**ische **Ki**noauskunftsystem.

FRÄNKI[1] gibt Auskunft über das aktuelle Kinoprogramm der ca. 60 Kinos im Großraum Nürnberg. Die Anfrage erfolgt per Telefon in deutscher Sprache. Das Vokabular umfasst etwa 1500 Wortformen, die für solch begrenzte Domänen ausreichen. Das System zeichnet sich unter anderem durch die flexible Dialogform aus. Der Benutzer kann seine Anfrage frei formulieren, während andere verbreitete Systeme oft ja/nein-Antworten erzwingen. Auch kann der Anrufer die Reihenfolge der Teilziele (z.B. Kinoname, Uhrzeit, Filmtitel) frei wählen und mehrere Informationen in einem Satz geben. Folgende Beispieldialoge sind möglich:

[1] Kinoauskunft FRÄNKI: 09131/610016

System:	Hallo hier ist FRÄNKI, wie kann ich ihnen helfen?
Benutzer:	Hallo, ich möchte in Erlangen ins Kino gehen.
System:	Sie möchten in Erlangen ins Kino?
	Um wieviel Uhr möchten sie ins Kino?
Benutzer:	So um halb acht.
System:	Sie möchten gegen 19:30 Uhr ins Kino?
Benutzer:	Ja.
System:	Ich nenne Ihnen nun die in Frage kommenden Filmtitel ...

System:	Hallo hier ist FRÄNKI, wie kann ich ihnen helfen?
Benutzer:	Hallo, ich möchte heute gegen 19:30 Uhr in Erlangen ins Kino Manhattan.
System:	Sie möchten heute einen Film im Kino Manhattan gegen 19:30 Uhr in Erlangen sehen?
Benutzer:	Ja.
System:	Ich nenne Ihnen nun die in Frage kommenden Filmtitel ...

Aktueller Stand von Dialogsystemen und zukünftige Herausforderungen für die Forschung werden in [Nöt01] beschrieben. Herzstück eines jeden Dialogsystems ist der Spracherkenner. Einen kurzen Überblick über die Vorgehensweise in der Spracherkennung gibt der nächste Abschnitt.

1.2 Automatische Spracherkennung

Das Verstehen von Sprache ist für uns Menschen eine alltägliche und selbstverständliche Fähigkeit, jedoch erst nach einer hochkomplexen neurophysiologischen und kognitiven Verarbeitung des Sprachschalls möglich. Schwierigkeiten bei der Spracherkennung sind u.a. die folgenden [ST95, S. 8ff]:

- Unterschiedliche akustische Realisierungen einer Spracheinheit. Ursachen sind Störquellen bei der Aufnahme, Eigenschaften des Aufnahmekanals, Sprechereigenschaften (Geschlecht, Anatomie des Vokaltraktes, Gesundheitszustand), Sprechweise (Tempo, Emotionen), Dialekt, Soziolekt oder kontextabhängige Aussprachevariationen.
- Detektion von Wortgrenzen. In kontinuierlich gesprochener Sprache werden Wörter in der Regel nicht durch Pausen getrennt und oft sogar verschliffen (z.B. Koartikulation: *'hast du'* wird zu *'hasdu'*).

- Spontan gesprochene, grammatikalisch falsche Sprache

- Ambiguitäten, das heißt Mehrdeutigkeiten die von Menschen mit Hilfe von Kontext und Weltwissen aufgelöst werden (z.B. *'Stau-becken'* und *'Staub-ecken'*, *'Rad'* und *'Rat'* oder *'das Tonband, das Nixon vernichtete'*).

Die automatische Klassifikation von Sprache erfolgt auf verschiedenen Ebenen. Zunächst wird das kontinuierliche Signal in einer Vorverarbeitungsphase aufbereitet und unter Beachtung des Abtasttheorems digitalisiert. Zur Vorverarbeitung zählen auch Normierung und Rauschunterdrückung, um das Signal von aufnahmebedingten Verzerrungen zu bereinigen. Anschließend wird es im Rahmen der *Kurzzeitanalyse* in kleine Zeitfenster von etwa 16 ms zerlegt. Aus diesen Fenstern werden Merkmale berechnet. Ziel ist es, so die Datenmenge zu reduzieren und trennscharfe Informationen hervorzuheben.

Ein relativ einfaches Problem ist die Klassifikation von Lauten. Ein Laut oder Phon ist eine Realisierung eines Phonems, der kleinsten Einheit, die zur Bedeutungsunterscheidung in einer Sprache beiträgt. Man unterscheidet etwa 40 - 60 Phoneme und etwa 200 Allophone; die Klasseneinteilung in automatischen Erkennungssystemen wird oft noch feiner gewählt (z.B. 256 Klassen). Jedes Kurzzeitanalysefenster wird nun einzeln aufgrund seiner spektralen Eigenschaften und evtl. unter zusätzlicher Einbeziehung des Kontextes klassifiziert. Übliche Merkmale, die sich dabei bewährt haben, sind die Energie, Mel-Cepstrum-Koeffizienten und deren Ableitungen. Die Abbildung von Merkmalvektoren auf Klassensymbole heißt *Vektorquantisierung* (siehe Kapitel 2.2.2). Danach ist das Sprachsignal segmentiert und es liegt eine Folge von Klassennamen bzw. Lautsymbolen vor.

Diese Lautfolge einzelnen Wörtern zuzuordnen, ist insofern ein nichttriviales Problem, als dass in freier Sprache gesprochene Wörter nichtlinear verzerrt werden (manche Silben werden gedehnt, andere gestaucht) bzw. oft verschliffen artikuliert werden. Die Abbildung von Lautketten auf Wörter erfolgt durch dynamische Programmierung oder *Hidden-Markov Modelle*. Letztere Vorgehensweise wird im nächsten Abschnitt kurz erläutert wird und im anschließenden Kapitel (Abschnitt 2.3) ausführlich beschrieben.

Die grammatische Modellierung von Wortfolgen übernimmt schließlich das linguistische Modell, bei dem z.B. diskrete stochastische *n-Gramm-Grammatiken* eingesetzt werden [ST95, S. 199ff].

In allen Ebenen erfolgt die Verarbeitung rein statistisch, das heißt, es muss eine Vielzahl von Parametern anhand einer Trainingsstichprobe geschätzt oder gelernt werden. Schätzalgorithmen findet man ebenfalls in Kapitel 2.

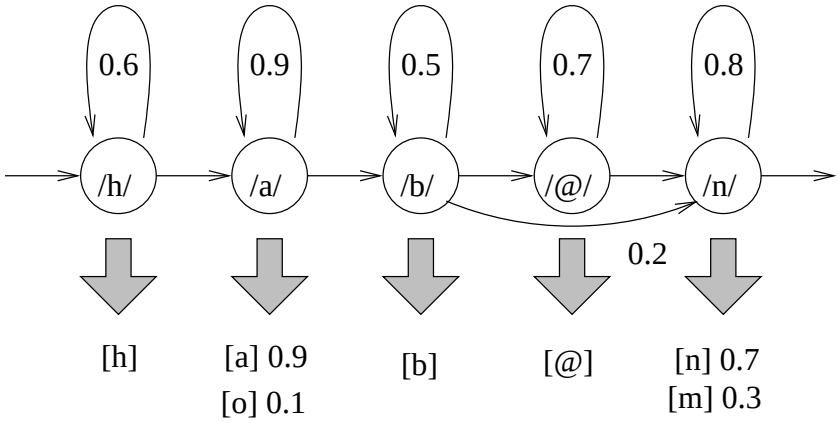


Bild 1.1: Statistisches Modell der Sprachproduktion für das Wort *'haben'*. Aus [ST95, S.126]

1.3 Hidden-Markov Modelle

Schon seit 1975 werden Hidden-Markov Modelle (HMM) in der Spracherkennung eingesetzt [Bak75] [Jel76], und haben sich Ende der achtziger Jahre als Standardverfahren durchgesetzt.

Hidden-Markov Modelle sind stochastische Automaten, die für die Worterkennung verwendet werden. Ein HMM ist dabei ein Modell für ein bestimmtes Wort oder eine Wortuntereinheit. Als Motivation wird in Abbildung 1.1 ein HMM für das Wort *'haben'* gezeigt. Jeder Kreis stellt einen Zustand dar, der einen Laut repräsentiert. Der Automat wird von links nach rechts durchlaufen. Man startet im Zustand /h/ und verweilt dort mit Wahrscheinlichkeit 0.6 oder wechselt mit Wahrscheinlichkeit 0.4 in den benachbarten Zustand /a/.

Durch diese erste Stufe des stochastischen Prozesses werden unterschiedliche zeitliche nichtlineare Verzerrungen modelliert [ST95, S.125f]. Ist ein Laut in mehreren aufeinanderfolgenden Kurzzeitanalysefenstern des Sprachsignals beobachtbar, so verweilt man im HMM-Zustand, der diesen Laut repräsentiert, ebenso viele Zeittakte. Neben diesen zeitlichen Stauchungen und Streckungen des Wortes sind auch Auslöschungen einzelner Laute möglich. Zustand /@/ wird mit Wahrscheinlichkeit 0.2 übersprungen.

Die zweite Stufe modelliert die Ausgabe in einem Zustand. Das /a/ aus dem zweiten Zustand kann beispielsweise mit Wahrscheinlichkeit 0.9 durch ein [a] aber auch durch ein [o] realisiert werden. Im Endeffekt erhält man eine Sequenz von Lautsymbolen, die vom Hidden-Markov Modell ausgegeben werden; welche Zustandsfolge im Inneren des Automaten aber genau durchlaufen wird, bleibt verborgen, was mit 'hidden' ausgedrückt wird[2].

[2]Das anschauliche Modell aus Bild 1.1 ist so einfach, dass man von der Ausgabesequenz eindeutig auf die Zustandsfolge schließen kann. Dies ist dann nicht mehr der Fall, wenn man z.B. einen weiteren Zustand, der [a] ausgibt, an dritter Stelle von links einfügt.

Ist nun die Lautsequenz eines unbekannten Wortes gegeben, wird für jedes HMM die Wahrscheinlichkeit berechnet, dass es eben diese Sequenz erzeugt. Jedes HMM steht für ein Wort oder eine Wortuntereinheit. Das HMM, das mit größter Wahrscheinlichkeit die unbekannte Lautfolge produziert, ist schließlich das Modell des Wortes, nach dem klassifiziert wird. Eine Formalisierung von Hidden-Markov Modellen erfolgt in Kapitel 2.3.

Die Ausgabe der HMM ist eine Folge von Lautsymbolen. Aus dem kontinuierlichen Sprachsignal wird jedoch eine Folge von Merkmalvektoren berechnet. Der entscheidende Zwischenschritt ist die Vektorquantisierung. Der Merkmalraum wird in Klassengebiete aufgeteilt (Kodebuch) und jeder Vektor seinem Klassennamen zugeordnet. Die Vorgehensweise beim *diskreten HMM* ist nun:

- Merkmalberechnung (Signal → Folge von Merkmalvektoren)
- Vektorquantisierung (Merkmalvektor → Lautsymbol)
- Klassifikation (Man sucht das HMM, das am wahrscheinlichsten die Lautfolge erzeugt)

Das *kontinuierliche HMM* dagegen gibt statt Lautsymbolen Merkmalvektoren aus:

- Merkmalberechnung (Signal → Folge von Merkmalvektoren)
- Klassifikation (Man sucht das HMM, das am wahrscheinlichsten die Folge von Merkmalvektoren erzeugt)

Dazu muss jedem Zustand eine Ausgabeverteilung vorliegen, um die Vektoren zu erzeugen. Werden diese Verteilungsdichten durch Überlagerung von K Normalverteilungen erzeugt, so lässt sich die Dichtefunktion als Kodebuch mit K Klassen interpretieren. Jeder Zustand besitzt hier also sein eigenes Kodebuch. Beim *semikontinuierlichen HMM* dagegen teilen sich alle Zustände ein einziges Kodebuch, jedoch werden die K Normalverteilungen von jedem Zustand individuell gewichtet. Ziel dieser Arbeit ist ein Schritt vom semikontinuierlichen hin zum kontinuierlichen HMM.

1.4 Zielsetzung der Arbeit und Beitrag zur Forschung

Diskrete HMM haben den Vorteil, dass nur wenige statistische Parameter in der Trainingsphase geschätzt werden müssen. Durch die Vektorquantisierung geht aber viel Information verloren.

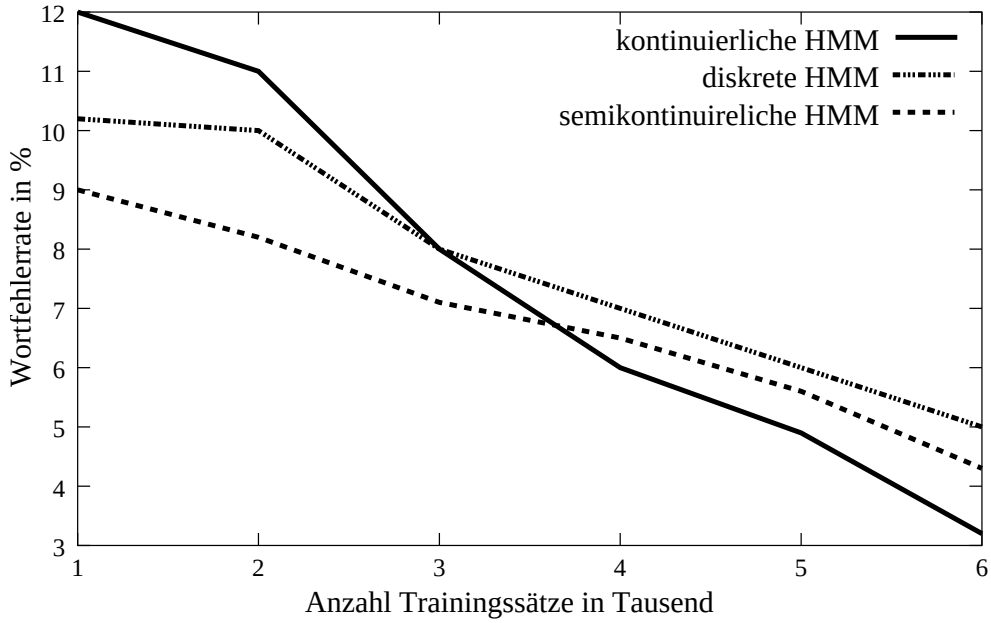


Bild 1.2: Wortfehlerrate verschiedener HMM-Typen in Abhängigkeit von der Größe der Stichprobe. Aus [Ace01, S.440]

Beim kontinuierlichen HMM vermeidet man diesen Informationsverlust; es sind jedoch wesentlich mehr Parameter zu schätzen. Ein Kompromiss sind die semikontinuierlichen HMM wie sie am Lehrstuhl für Mustererkennung (LME) verwendet werden.

In [Ace01, S.439 ff] wird beschrieben, wie sich die Fehlerraten der verschiedenen HMM-Typen in Abhängigkeit von der Größe der Trainingsstichprobe verhalten (Abbildung 1.2). Mit semikontinuierlichen HMM werden durchwegs geringere Fehlerraten erzielt als mit diskreten HMM. Die kontinuierlichen HMM sind bei wenig Trainingsmaterial am schlechtesten. Steht dagegen eine ausreichend große Stichprobe zur Verfügung, so sind die kontinuierlichen HMM am besten und die diskreten am schlechtesten. Die semikontinuierlichen HMM nehmen dann den mittleren Platz ein. Neben einer genügend großen Stichprobe werden für die kontinuierlichen HMM auch viele Kodebuchdichten gefordert; der Trainingsaufwand für diese Vielzahl von statistischen Parametern ist sehr hoch.

In der vorliegenden Arbeit wurden semikontinuierliche HMM mit mehreren Kodebüchern implementiert, ein Schritt vom semikontinuierlichen HMM hin zum kontinuierlichen. Man verspricht sich dadurch höhere Erkennungsraten bzw. niedrigere Fehlerraten zu erzielen und gleichzeitig die Anzahl der zu schätzenden Parameter nur maßvoll zu vergrößern.

Die Diplomarbeit lässt sich in zwei Gebiete unterteilen. In einem Teil werden Kodebücher für verschiedene Merkmaltypen trainiert, also beispielsweise separat für dynamische und für sta-

tische Merkmale. Die Kodebücher werden hier von allen Zuständen geteilt. Erzeugt ein Zustand als Ausgabe einen Merkmalvektor, so müssen alle Kodebücher gleichzeitig berücksichtigt werden. Dieser Ansatz ist laut [Ace01, S.439f] äußerst erfolgreich: Es wird eine Verringerung der Fehlerrate um über 10 % genannt. Die Dimension der Normalverteilungsdichten ist nun niedriger, da nur Teile des Merkmalvektors erzeugt werden; die Anzahl an Dichten insgesamt in allen Kodebüchern zusammen ist dafür höher. Da jeder Zustand durch mehr Gewichtungen von mehr Normalverteilungsdichten nun mehr Möglichkeiten zur individuellen Gestaltung des gemeinsamen Kodebuchs hat, lässt sich dieser Ansatz als Schritt Richtung kontinuierliches HMM interpretieren.

Im zweiten Teilgebiet der Arbeit werden mehrere Kodebücher für verschiedene Lautoberklassen bereitgestellt. Jeder Zustand berücksichtigt also nur ein Kodebuch; Zustände derselben Oberklasse teilen sich ein Kodebuch. Dies ist ein direkter Schritt Richtung kontinuierliche HMM. Wieder verspricht man sich bessere Erkennungsraten, wenn ausreichend Trainingsmaterial zum Schätzen der nun vergrößerten Gesamtzahl von Parametern bereitsteht.

Ferner wird die Robustheit des verbesserten Spracherkenners mit mehreren Kodebüchern gegenüber Hall getestet. Dazu wird ein Erkenner mit der VERBMOBIL-Stichprobe trainiert. Im Rahmen des Müdigkeitsexperimentes [Had02] wurden verschiedene Verbmobiltexte neu aufgezeichnet und zwar sowohl mit einem Nahbesprechungsmikrophon als auch mit verschiedenen Raummikrofonen. Letztere Aufnahmen sind verhallt. Mit diesen Aufzeichnungen wird getestet, ob der neue Erkenner mit mehreren Kodebüchern robuster gegenüber Hall ist. Ursprünglich sollte die Robustheit gegenüber Störgeräuschen allgemein getestet werden. Dazu wird im Rahmen der EMBASSI-Projektes eine neue Stichprobe erstellt, die zum Abschluß der Diplomarbeit allerdings noch nicht vorlag. Die Aufnahme der Referenzdialoge geschieht dort einmal unter Simulierung des Lombard-Effekts und einmal in einem Raum mit Störgeräuschen (z.B. Musik, Gespräche ...).

Verschiedene Experimente werden auch mit den in der Studienarbeit [Hac01] gefundenen verbesserten Merkmalen durchgeführt. Dabei werden neben den 12 statischen Merkmalen die Ableitungen für verschiedene Zeitauflösungen berechnet. In der Studienarbeit wurde diese Vielzahl von Ableitungen anschließend mittels Karhunen-Loève-Transformation auf 12 dynamische Merkmale reduziert. Anstelle der Reduktion wird die direkte Verwendung im Multi-Kodebuch-Ansatz untersucht.

Einen Überblick über den Aufbau der vorliegenden Arbeit gibt der nächste Abschnitt.

1.5 Aufbau der Diplomarbeit

Die vorliegende Diplomarbeit ist wie folgt gegliedert: In *Kapitel 2* findet man wichtige theoretische Grundlagen der HMM-basierten Spracherkennung. Es werden allgemeine Grundlagen aus der Mustererkennung sowie das spezielle Vorgehen in der Spracherkennung beschrieben und anschließend verschiedene Typen von Hidden-Markov Modellen erläutert und verglichen. Die Anwendung der HMM bei der Klassifikation und das Training der statistischen Parameter werden erklärt.

In *Kapitel 3* werden die verschiedenen Ansätze motiviert, die Spracherkennung durch Modifikation der HMM-Architektur bzw. der Trainingsverfahren zu verbessern. Dort wird auch ein Literaturüberblick gegeben

Eine technische Beschreibung der Experimente gibt *Kapitel 4*. Es werden die Stichproben, Trainings- und Testverfahren, sowie die verwendete Software kurz vorgestellt. Auch die Berechnung der Erkennungsraten wird dort angegeben, sowie die Ergebnisse der Untersuchungen des Referenzsystems. Mit diesen Ergebnisses werden die Experimente in den nachfolgenden Kapiteln verglichen.

Der erste Schwerpunkt der Diplomarbeit ist, mehrere Kodebücher für verschiedene Merkmalsarten zu berechnen. Diese Kodebücher werden von allen HMM-Zuständen geteilt und gleichzeitig berücksichtigt. Die Beschreibung der Experimente und Diskussion der Ergebnisse findet man in *Kapitel 5*. Anschließend wird die Robustheit dieses Ansatzes gegenüber Hall untersucht.

Kapitel 6 befasst sich schließlich mit dem zweiten Schwerpunkt der vorliegenden Arbeit. Nun teilen sich jeweils Zustände verschiedener Lautoberklassen ein Kodebuch. Wieder werden mehrere Kodebücher trainiert, jeder Zustand berücksichtigt aber nur genau eines. Auch dazu werden verschiedene Experimente beschrieben und verglichen.

Nach einem Ausblick in *Kapitel 7* schließt die Arbeit mit der Zusammenfassung (*Kapitel 8*).

Kapitel 2

HMM basierte Spracherkennung

In diesem Kapitel werden theoretische Grundlagen aus der Spracherkennung diskutiert. In einem ersten Abschnitt werden allgemeine Gesetze aus der Mustererkennung zusammengestellt. Es folgt ein Abschnitt, der das Vorgehen speziell in der Spracherkennung erläutert. Zuletzt wird die Klassifikation mit Hidden-Markov Modellen in Einzelheiten erklärt.

2.1 Klassifikation von Mustern

Ziel in der Mustererkennung ist es, einem Muster $\boldsymbol{f}^r(\boldsymbol{x})$ eine Klasse Ω_κ zuzuordnen. Die Klassen Ω_κ sind paarweise disjunkt und bilden zusammen den Problemkreis Ω. Die einzelnen Schritte eines Klassifikationssystems sind in Abbildung 2.1 veranschaulicht. Ein Muster $\boldsymbol{f}$ wird zunächst aufgenommen (z.B. Sprache mit einem Mikrophon) und danach gegebenenfalls vorverarbeitet, damit man in den folgenden Schritten schneller zu besseren Ergebnissen kommt. Danach werden d charakteristische Merkmale berechnet und in einem Merkmalvektor $\boldsymbol{c}$ gespeichert. Im anschlie-

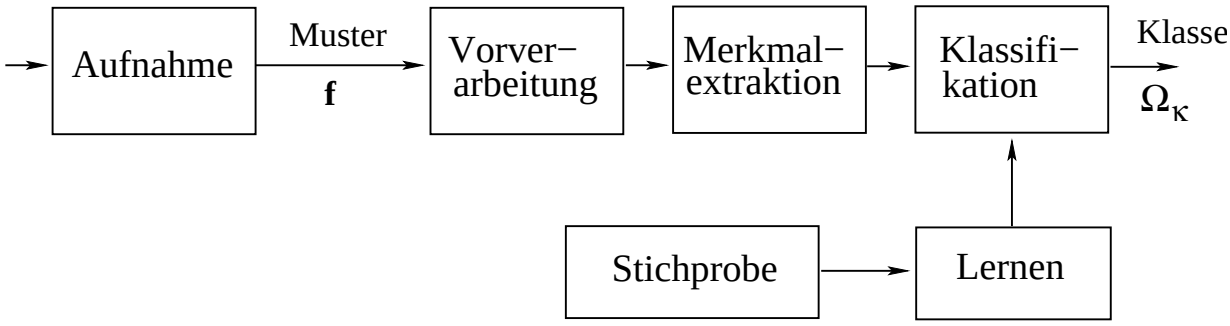


Bild 2.1: Struktur eines Klassifikationssystems. Nach: [Nie83, S.14]

ßenden Klassifikationsschritt werden diese auf eine Klasse Ω_κ abgebildet. Für diese Klassifikation braucht man Informationen, welche Bereiche im Merkmalraum welcher Klasse zugeordnet werden sollen. Diese gewinnt man aus einer Stichprobe von Mustern in einer Trainings- oder Lernphase. Gegebenenfalls kann eine Eingabe auch zurückgewiesen werden, indem man sich für die spezielle Klasse Ω_0 entscheidet [Nie83, S.5, S.14f].

2.1.1 Optimale Entscheidung

Für den Zweck der Klassifikation leiten wir nun eine Entscheidungsregel her [Nie83, S.164ff] [ST95, S.76ff]. Die *a priori Wahrscheinlichkeit* für die Klasse Ω_κ ist $P(\Omega_\kappa) = p_\kappa$ (mit $\sum_\kappa p_\kappa = 1$) und die *a posteriori Wahrscheinlichkeit* $P(\Omega_\kappa|\boldsymbol{c})$. Die Entscheidungsregel

$$\delta(\Omega_\kappa|\boldsymbol{c}) \quad \text{mit} \quad \sum_\kappa \delta(\Omega_\kappa|\boldsymbol{c}) = 1 \quad \text{für alle} \quad \boldsymbol{c} \in \mathbb{R}^d \tag{2.1}$$

ordnet einem Merkmalvektor eine Klasse mit einer bestimmten Wahrscheinlichkeit zu. Sei die Verwechslungswahrscheinlichkeit $P(\Omega_\lambda|\Omega_\kappa)$ die Wahrscheinlichkeit, dass man sich für Klasse Ω_λ entscheidet, obwohl Klasse Ω_κ vorliegt, und seien $r_{\lambda\kappa}$ die Kosten für eine solche Verwechslung. Dann ist das Risiko $V(\delta)$ einer Entscheidungsregel wie folgt definiert:

$$V(\delta) = \sum_{\kappa\neq 0} \sum_\lambda p_\kappa P(\Omega_\lambda|\Omega_\kappa) r_{\lambda\kappa} \tag{2.2}$$

Der optimale Klassifikator ist nun diejenige Entscheidungsregel, die das Risiko minimiert. Nach einigen Überlegungen ergeben sich für das Risiko die Umformungen

$$\begin{aligned} V(\delta) &= \sum_\lambda \sum_{\kappa\neq 0} p_\kappa \int_{\mathbb{R}^d} P(\boldsymbol{c}|\Omega_\kappa)\delta(\Omega_\lambda|\boldsymbol{c})d\boldsymbol{c} \quad r_{\lambda\kappa} \\ &= \int_{\mathbb{R}^d} \sum_\lambda \left[\sum_{\kappa\neq 0} r_{\lambda\kappa} p_\kappa P(\boldsymbol{c}|\Omega_\kappa)\right] \delta(\Omega_\lambda|\boldsymbol{c})d\boldsymbol{c} \\ &= \int_{\mathbb{R}^d} \sum_\lambda u_\lambda(\boldsymbol{c})\delta(\Omega_\lambda|\boldsymbol{c})d\boldsymbol{c} \end{aligned} \tag{2.3}$$

mit der Prüfgröße

$$u_\lambda(\boldsymbol{c}) = \sum_{\kappa\neq 0} r_{\lambda\kappa} p_\kappa P(\boldsymbol{c}|\Omega_\kappa), \quad \lambda = 0, 1, 2, ... \tag{2.4}$$

Zur optimalen Klassifikation unter Minimierung des Risikos berechnet man also die Prüfgrößen $u_\lambda(\boldsymbol{c})$ für alle λ, und entscheidet sich für diejenige Klasse, deren Prüfgröße minimal ist.

Im Spezialfall der erzwungenen Entscheidung [Nie83, S.172f] entfällt die Klasse Ω_0. Verwendet man zusätzlich die einfache (0,1)-Kostenfunktion mit $r_{\kappa\kappa} = 0$ und $r_{\lambda\kappa} = 1$ für $\lambda \neq \kappa$ so vereinfacht sich die Prüfgrößenberechnung zu

$$u_\lambda(\boldsymbol{c}) = \sum_{\kappa\neq 0, \kappa\neq\lambda} p_\kappa P(\boldsymbol{c}|\Omega_\kappa), \quad \lambda = 1, 2, 3.... \tag{2.5}$$

Diese Summe wird minimal, wenn der größte Summand fehlt. Es gilt nun für den optimalen Klassifikator

$$\begin{aligned} \delta(\Omega_\kappa|\boldsymbol{c}) &= 1 \quad , \quad \text{falls} \quad \bar{u}_\lambda(\boldsymbol{c}) = p_\kappa P(\boldsymbol{c}|\Omega_\kappa) = \max_\lambda p_\lambda P(\boldsymbol{c}|\Omega_\lambda) \\ \delta(\Omega_\kappa|\boldsymbol{c}) &= 0 \quad , \quad \text{sonst.} \end{aligned} \tag{2.6}$$

Dies ist der *Bayes-Klassifikator*, bei dem die a posteriori Wahrscheinlichkeit

$$P(\Omega_\lambda|\boldsymbol{c}) = \frac{p_\lambda P(\boldsymbol{c}|\Omega_\lambda)}{P(\boldsymbol{c})} \tag{2.7}$$

maximiert wird. Jeder andere gute Klassifikator approximiert den Bayes-Klassifikator.

2.1.2 Schätzen von Normalverteilungen

Die optimale Entscheidungsregel wird nun durch vereinfachende Annahmen approximiert. Der statistische Klassifikator schätzt die klassenbedingten Verteilungsdichten $P(\boldsymbol{c}|\Omega_\kappa)$. Wir betrachten hier den Spezialfall des *Normalverteilungsklassifikators* [Nie83, S.161ff, S.175ff] [ST95, S.79f] und nehmen klassenweise normalverteilte Merkmale an:

$$P(\boldsymbol{c}|\Omega_\kappa) = |2\pi\boldsymbol{K}_\kappa|^{-1/2} \exp\left(-\frac{(\boldsymbol{c}-\boldsymbol{\mu}_\kappa)^T \boldsymbol{K}_\kappa^{-1}(\boldsymbol{c}-\boldsymbol{\mu}_\kappa)}{2}\right) \tag{2.8}$$

$\boldsymbol{K}_\kappa$ und $\boldsymbol{\mu}_\kappa$ sind dabei Kovarianzmatrix und Mittelwertvektor der Dichte von Klasse Ω_κ. $\hat{\boldsymbol{K}}_\kappa$ und $\hat{\boldsymbol{\mu}}_\kappa$ bezeichnen Schätzwerte von $\boldsymbol{K}_\kappa$ und $\boldsymbol{\mu}_\kappa$, $\hat{p}_\kappa$ ist Schätzwert der a priori Wahrscheinlichkeit der Klasse Ω_κ. Ist die Stichprobe beschriftet und die Klassenzugehörigkeit der Merkmalvektoren bekannt ($\boldsymbol{c}_\kappa$ gehört der Klasse Ω_κ an), so sind die Maximum-Likelihood-Schätzwerte

$$\begin{aligned}
\hat{p}_\kappa &= \frac{N_\kappa}{\sum_{\kappa=1}^{K} N_\kappa} \\
\hat{\boldsymbol{\mu}}_\kappa &= \frac{1}{N_\kappa}\sum_{j=1}^{N_\kappa} \boldsymbol{c}_\kappa^{(j)} \\
\hat{\boldsymbol{K}}_\kappa &= \frac{1}{N_\kappa}\sum_{j=1}^{N_\kappa} (\boldsymbol{c}_\kappa^{(j)} - \hat{\boldsymbol{\mu}}_\kappa)(\boldsymbol{c}_\kappa^{(j)} - \hat{\boldsymbol{\mu}}_\kappa)^T
\end{aligned} \tag{2.9}$$

N_κ ist die Anzahl der Merkmalvektoren aus Klasse Ω_κ bei insgesamt K Klassen. Zusammen mit der Gleichung 2.8 ergibt sich sofort die Prüfgrößen wie in Gleichung 2.6. Man vereinfacht die Formel durch Transformation mit $-2\ln$ und unternimmt im Zielraum eine Minimumsuche statt der bisherigen Maximumsuche. Eine weitere Vereinfachung ist das Weglassen der Wurzel, was eine Optimumsuche nicht verfälscht. Es gilt

$$\begin{aligned}
\bar{u}_\lambda(\boldsymbol{c}) &= \max_\lambda p_\lambda P(\boldsymbol{c}|\Omega_\lambda) \\
&= \min_\lambda -2\ln(p_\lambda P(\boldsymbol{c}|\Omega_\lambda)) \\
&= \min_\lambda -2\ln p_\lambda + \ln|2\pi\boldsymbol{K}_\lambda| + \boldsymbol{\mu}_\lambda^T \boldsymbol{K}_\lambda^{-1}\boldsymbol{\mu}_\lambda - 2\boldsymbol{x}^T\boldsymbol{K}_\lambda^{-1}\boldsymbol{\mu}_\lambda + \boldsymbol{x}^T\boldsymbol{K}_\lambda^{-1}\boldsymbol{x}.
\end{aligned} \tag{2.10}$$

Einfacher lässt sich dieser Term über ein Skalarprodukt der Dimension

$$(d^2+d)/2 + d + 1 \tag{2.11}$$

berechnen, wobei d die Dimension des Merkmalvektors $\boldsymbol{c}$ ist. Für jedes $\boldsymbol{c}$ werden K Skalarprodukte berechnet und eine Entscheidung für diejenige Klasse getroffen, deren Prüfgröße minimal ist.

Sind $\boldsymbol{a}$ alle zu schätzenden Parameter der Normalverteilung, so wird bei der *Maximum-Likelihood-Schätzung* die Wahrscheinlichkeit der Stichprobe ω maximiert

$$P(\omega|\hat{\boldsymbol{a}}) = \max_{\boldsymbol{a}} P(\omega|\boldsymbol{a}). \tag{2.12}$$

Durch die Bayes-Schätzung wird hingegen die *a posteriori Wahrscheinlichkeit* des Parametervektors

$$P(\hat{\boldsymbol{a}}|\omega) = \max_{\boldsymbol{a}} P(\boldsymbol{a}|\omega) \tag{2.13}$$

maximiert.

In den Formeln 2.9 ist die Klassenzugehörigkeit der Merkmalvektoren $\boldsymbol{c}$ bekannt. Das Schätzen wird in diesem Zusammenhang auch als überwachtes Lernen bezeichnet. Unüberwachtes Lernen wird im folgenden Abschnitt diskutiert.

2.1.3 Der EM-Algorithmus

Meist ist unbekannt, welcher Klasse die Merkmalvektoren $\boldsymbol{c}$ angehören; die Stichprobe ist nicht beschriftet. Das Lernen der Klassenzugehörigkeit kann dann unüberwacht erfolgen. Wir beschränken uns an dieser Stelle auf das Schätzen von statistischen Parametern bei der Identifikation von Gaußschen Mischungsverteilungen, also von p_κ, $\boldsymbol{\mu}_\kappa$ und $\boldsymbol{K}_\kappa$. [Nie83, S.148ff].

Ein iteratives Verfahren zur Bestimmung dieser unbekannten Parameter ist der *EM-Algorithmus* (*E*stimation *M*aximization) [Dem77]. Er findet immer dann Anwendung, wenn ein Zufallsprozess von zwei Zufallsvariablen abhängt, wobei nur eine beobachtbar ist und die andere verborgen bleibt. Diese verborgene Zufallsvariable ist hier die Klassenzugehörigkeit der Merkmalvektoren. Das Vorgehen ist das folgende:

1. Bestimme Startwerte der unbekannten Parameter
   (z.B. aus einer kleinen klassifizierten Stichprobe oder zufällig).

2. Estimation: Klassifiziere die Stichprobe anhand der Verteilungen mit den aktuellen Parametern (hier: $\hat{p}_\kappa$, $\hat{\boldsymbol{\mu}}_\kappa$ und $\hat{\boldsymbol{K}}_\kappa$). So wird die verborgene Zufallsvariable (hier: die Klassenzugehörigkeit) geschätzt.

3. Maximization: Berechne neue Schätzwerte der Parameter mit Hilfe der Schätzwerte der verborgenen Zufallsvariable.

4. Wenn keine Änderungen oder anderes Abbruchkriterium: Ende
   Sonst: wiederhole ab Schritt 2.

Werden im Schritt 2 die a posteriori Wahrscheinlichkeiten $P(\Omega_\kappa|\boldsymbol{c})$ berechnet, spricht man von vager Entscheidungsüberwachung; bei harter Entscheidungsüberwachung klassifiziert man dagegen nach genau einer Klasse. Die zu Gleichung 2.9 analogen Maximum-Likelihood Schätzwerte der Mischungsverteilungsparameter sind im unüberwachten Fall mit vager Entscheidungsüberwachung

$$
\begin{aligned}
\hat{p}_\kappa &= \frac{1}{N}\sum_{j=1}^{N}\hat{P}(\Omega_\kappa|\boldsymbol{c}^{(j)}) \\
\hat{\boldsymbol{\mu}}_\kappa &= \frac{1}{N\hat{p}_\kappa}\sum_{j=1}^{N}\hat{P}(\Omega_\kappa|\boldsymbol{c}^{(j)})\boldsymbol{c}^{(j)} \\
\hat{\boldsymbol{K}}_\kappa &= \frac{1}{N\hat{p}_\kappa}\sum_{j=1}^{N}\hat{P}(\Omega_\kappa|\boldsymbol{c}^{(j)})(\boldsymbol{c}_\kappa^{(j)}-\hat{\boldsymbol{\mu}}_\kappa)(\boldsymbol{c}_\kappa^{(j)}-\hat{\boldsymbol{\mu}}_\kappa)^T
\end{aligned}
\tag{2.14}
$$

N sei hier die Gesamtzahl der Stichprobenelemente. Nach diesen Formeln werden im Schritt 3 des EM-Algorithmus die Parameter geschätzt.

Nachdem bisher allgemeine Grundlagen zur Klassifikation von Mustern dargestellt wurden, werden im nächsten Abschnitt spezielle Vorgehensweisen in der Spracherkennung erläutert.

2.2 Vorgehensweise in der Spracherkennung

In diesem Abschnitt betrachten wir die Klassifikation von Sprache auf Lautebene. Aus dem Sprachsignal werden wie nachfolgend erläutert Merkmalvektoren berechnet. Der daran anschließende Abschnitt gibt Aufschluss über die Vorgehensweise, wie ein Merkmalvektor einem Lautsymbol zugeordnet wird (Vektorquantisierung).

2.2.1 Merkmalberechnung

Das kontinuierliche Sprachsignal wird in einer Vorverarbeitungsphase diskretisiert, d.h. in zeitlicher Richtung unter Beachtung des Abtasttheorems abgetastet und anschließend quantisiert. Man erhält eine zeitliche Folge von Abtastwerten f_n ($n = 0, 1, 2, ...$) aus der Merkmale extrahiert werden ([Hac01, S.17ff].

Bei der Kurzzeitanalyse [ST95, S.48ff] werden die Merkmale nicht aus der gesamten Sprachsequenz berechnet sondern kleine, evtl. auch überlappende Fenster betrachtet. Aus jedem dieser Kurzzeitanalysefenster werden mehrere Merkmale berechnet und in einem Merkmalvektor zusammengefasst. Man erhält eine Folge von Merkmalvektoren c_τ ($\tau = 0, 1, 2, ...$). Merkmalvektoren, die derselben akustischen Einheit angehören, sollen im Raum einen kompakten Bereich

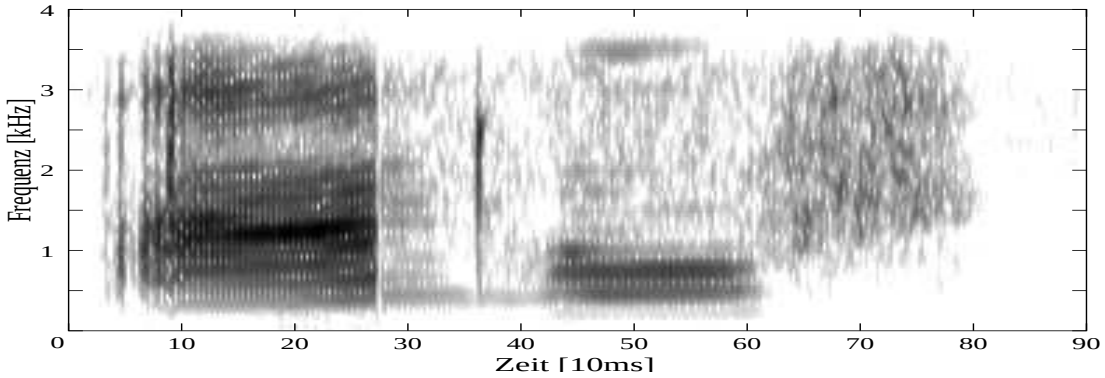


Bild 2.2: Spektrogramm des Wortes “Bahnhof”

einnehmen und möglichst getrennt von den Merkmalen anderer Klassen liegen [Nie83, S.11].

Die statischen Merkmale geben Aufschluss über die spektrale Zusammensetzung der Sprache innerhalb einzelner Kurzzeitanalysefenster. In der Spracherkennung übliche Merkmale sind z.B. cepstrale Merkmale, die im folgenden genauer betrachtet werden [ST95, S.53ff, S.58ff].

Durch die Fourier-Transformation FT eines Zeitsignals erhält man sein Spektrum. Die Kurzzeitanalysefenster des Signals werden einzeln transformiert, da man Änderungen der spektralen Eigenschaften von Laut zu Laut beobachten will. Seien $f_{n,m}$ ($n = 0, 1, ..., N-1$) die Abtastwerte im Fenster τ, das an der Stelle m beginnt und N Abtastwerte umfasst, dann berechnet sich die Diskrete Fourier Transformation (DFT) nach der Formel

$$S_{k\tau} = \sum_{n=0}^{N-1} f_{n,m} \cdot e^{-\frac{2\pi i n k}{N}}; \quad k = 0, ..., N-1. \tag{2.15}$$

Die Spektrumkoeffizienten werden anschließend logarithmiert, um die Lautheits-Empfindung des Menschen zu modellieren. Die gewonnene zeitliche Folge von Spektren lässt sich in Spektrogrammen darstellen (Abbildung 2.2).

Ein wichtiges Merkmal ist die Kurzzeitenergie. Diese erhält man, indem man die Spekrumkoeffizienten addiert

$$E_\tau = \frac{1}{N} \sum_{k=1}^{N-1} |S_{k\tau}|^2 \tag{2.16}$$

Aus dem Spektrum wird das Bandspektrum berechnet, indem man Energiewerte für bestimmte Frequenzbereiche berechnet. Diese Bereiche werden durch Multiplikation mit Bandpass-Filtern extrahiert. Liegen die Filter auf der Mel-Skala äquidistant, erhält man ein Mel-Spektrum. So wird der menschlichen Tonhöhenempfindung Rechnung getragen.

Das Spektrum einer Funktion f wird normalerweise nicht selbst zur Berechnung von Merkmalen verwendet. Stattdessen berechnet man das Cepstrum:

$$C = FT^{-1}\left(\log|FT(f)|\right) \tag{2.17}$$

In der Praxis berechnet man die inverse DFT oder die Kosinustransformation der Ordnung L. Letztere ist für das Fenster τ

$$C_{k\tau} = \sum_{l=1}^{L} \log|S_{l\tau}| \cdot \cos\left(\frac{k \cdot (2l-1)\pi}{2L}\right). \tag{2.18}$$

So werden die Spektrum-Koeffizienten dekorreliert. Durch Kosinustransformation des Mel-Spektrums erhält man das Mel-Cepstrum. Die Koeffizienten $k > 1$ werden oft als MFCCs (Mel-frequency cepstral coefficients) bezeichnet. Zusammen mit der Gesamtenergie $k = 0$ erhält man einen in der Spracherkennung üblichen Satz von statischen Merkmalen.

Beim Klassifizieren kleiner Sprachausschnitte sind zudem dynamische Merkmale [ST95, S.68ff] wichtig, die den zeitlichen Kontext der statischen Merkmale berücksichtigen. So sind beispielsweise Plosive durch ihr abruptes Zeitverhalten charakterisiert. Übliche Merkmale sind die Ableitung des zeitlichen Verlaufs der einzelnen Merkmalkomponenten, oft approximiert durch Differenzen oder Regressionsgeraden.

Wir erhalten also Vektoren aus statischen und dynamischen Merkmalen. Mit einem Vektorquantisierer werden diese wie im folgenden Abschnitt beschrieben beispielsweise einem Lautsymbol zugeordnet.

2.2.2 Vektorquantisierung

Die Merkmalvektoren $\boldsymbol{c} \in \mathbb{R}^d$ werden in einem nächsten Schritt in skalare Werte quantisiert. Diese Sprachsegmentierung erfolgt mit einem Vektorquantisierer [Nie90, S.148f] [ST95, S.97ff]. Er induziert einen Klassifikator, der zunächst den Merkmalvektor einer Klasse Ω_κ zuteilt. Ausgabe des Vektorquantisierers ist ein Prototyp, der repräsentativ für die jeweilige Klasse steht. Ähnliche Muster werden also im Sinne einer Datenreduktion zusammengefasst.

Die wichtige Information zur Vektorquantisierung steht in einem Kodebuch, welches den $\mathbb{R}^d$ in Klassen partitioniert. Die Anzahl der Klassen K ist üblicherweise größer als die der unterscheidbaren Laute. Da die Klassenzugehörigkeit eines Merkmalvektors i.A. unbekannt ist, erfolgt das Lernen der Klassengebiete unüberwacht (vgl. Abschnitt 2.1.3).

Ein Kodebuch gilt als optimal, wenn der verursachte Quantisierungsfehler ϵ bezüglich eines

Abstandsmaßes $||.||$ minimal ist. Sei V_κ das Klassengebiet, das durch den Prototyp b_κ repräsentiert wird, dann gilt

$$\epsilon = \sum_{\kappa=1}^{K} \int_{\boldsymbol{c}\in V_\kappa} ||\boldsymbol{c} - \boldsymbol{b_\kappa}||^2 P(\boldsymbol{c}) d\boldsymbol{c}. \tag{2.19}$$

Zur Quantisierung eindimensionaler Messwerte bietet die PCM eine geschlossene Form. Eine solche gibt es im Mehrdimensionalen nicht. Das Kodebuch wird mit dem LBG-Algorithmus (Linde, Buzo, Gray) geschätzt [Lin80].

1. Wähle initiale Prototypvektoren für jede Klasse
2. Klassifiziere die Merkmalvektoren und berechne ϵ
3. Wenn es keine Änderungen gibt oder der relative Fehler klein genug ist: Ende
4. Berechne neue Prototypvektoren und wiederhole ab Schritt 2

Da die Stichprobe endlich ist, gibt es nur endlich viele mögliche Kodebücher und man gerät nach endlichen Schritten entweder in einen Fixpunkt oder in einen Zyklus von Vektorquantisierern mit konstantem Verzerrungswert. Der LBG-Algorithmus ist ein Spezialfall des EM-Algorithmus (siehe Abschnitt 2.1.3)

Jeder Merkmalvektor wird bei der Vektorquantisierung einem Prototypvektoren zugeordnet. Geben wir jedem dieser Prototypen einen Namen, etwa den des repräsentierten Lautes, erhalten wir aus der Folge von Merkmalvektoren eine Symbolkette. Mit der Klassifikation ganzer Wörter befasst sich der folgende Abschnitt.

2.3 Hidden-Markov Modelle

Erschwerend bei der Klassifikation ganzer Wörter ist, dass Sprache zeitlich nichtlinearen Verzerrungen unterliegt. Vergleicht man also zwei Realisierungen desselben Wortes durch verschiedene Sprachaufnahmen, erkennt man, dass häufig z.B eine Silbe stark verkürzt ist, während die andere gedehnt wird. Oft werden auch Laute ganz ausgelassen, wie z.B. die Realisierung von *'e'* in *'haben'*.

Hidden-Markov Modelle (HMM) sind stochastische Wortmodelle die den soeben beschriebenen Umständen Rechnung tragen. Ein HMM ist Modell für ein bestimmtes Wort oder eine Wortuntereinheit. Zeitliche Stauchung und Streckung sowie Auslöschungen einzelner Laute können,

wie in Abbildung 1.1 für das Wort *'haben'* gezeigt, modelliert werden. Die Zustände des Automaten werden von links nach rechts durchlaufen. Übergangswahrscheinlichkeiten ermöglichen ein Verweilen in einem Zustand oder auch das Überspringen von Zuständen. Welche Zustände aber genau durchlaufen werden, bleibt verborgen, was mit 'hidden' ausgedrückt wird. Beobachtbar ist nur die Sequenz von Lautsymbolen, die von den Zuständen ausgegeben werden; eine bestimmte Lautfolge kann im allgemeinen durch verschiedene Zustandsfolgen erzeugt werden.

Ist nun die Lautsequenz eines unbekannten Wortes gegeben, sucht man dasjenige HMM, das mit größter Wahrscheinlichkeit die unbekannte Lautfolge produziert. Das Wort oder die Wortuntereinheit, für die das gefundene HMM steht, ist Lösung des Klassifikationsproblems.

In den nachfolgenden Abschnitten werden zunächst die HMM formalisiert und allgemein erläutert und danach die Spezialfälle diskrete, kontinuierliche und semikontinuierliche HMM beschrieben, sowie Trainingsalgorithmen angegeben.

2.3.1 Formalisierung

In diesem Abschnitt wird die oben anschaulich beschriebene Idee der Hidden-Markov Modelle formalisiert. Zunächst wird der Begriff Markovkette beschrieben.

Definition 1 *Ein zeitdiskreter stochastischer Prozess, der die Markov-Eigenschaft besitzt, heißt Markovkette.*

Ein zeitdiskreter stochastischer Prozess ist ein Zufallsprozess, der zu diskreten Zeitpunkten $t_1 < t_2 < t_3 < ...$ eintrit. Genauer wird an dieser Stelle nicht darauf eingegangen, sondern auf [, ???] verwiesen. Die *Markov-Eigenschaft* beschreibt die "Gedächtnislosigkeit" eines Zufallsprozess, d.h. die zukünftige Entwicklung eines Prozesses hängt nur vom gegenwärtigen Zustand ab und nicht von dessen Vergangenheit. Anders ausgedrückt heißt dies, dass Zukunft und Vergangenheit eines Prozesses stochastisch unabhängig sind. Sei G nun ein Ereignis aus der Gegenwart, V eines aus der Vergangenheit und Z eines aus der Zukunft, so gilt

$$P(Z|G,V) = P(Z|G) \tag{2.20}$$

und folglich die Unabhängigkeitsbedingung

$$P(Z,V|G) = P(Z|V,G)P(V|G) = P(Z|G)P(V|G) \tag{2.21}$$

Eine Markovkette ist nun gegeben durch

- eine Menge Q von *Zustandssymbolen* $Q = \{s_1, s_2, ..., s_I\}$. (Der Zustand, in dem man sich zum diskreten Zeitpunkt t_n befindet wird durch die Zufallsvariable $q_n \in Q$ beschrieben),
- die *Anfangswahrscheinlichkeiten* $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_I)^T$ mit $\pi_i = P(q_1 = s_i)$ und
- *Übergangswahrscheinlichkeiten* a_{ij}, die in einer Matrix $\boldsymbol{A}$ zusammengefasst sind und die Wahrscheinlichkeit eines Zustandsüberganges $s_i \rightarrow s_j$ angeben.

Die Übergangswahrscheinlichkeiten

$$a_{ij} = P(q_{n+1} = s_j | q_n = s_i) \tag{2.22}$$

erfüllen die Stochastizitätsbedingungen $a_{ij} > 0$ und $\sum_j a_{ij} = 1$.

In der Spracherkennung werden die Markovketten noch mit eine zweite Stufe des stochastischen Prozesses überlagert. In jedem Zustand wird ein Symbol ausgegeben. Die Folge von Ausgabesymbolen kann beobachtet werden, die Zustandsfolge bleibt verborgen. Zusätzlich definiert man also

- eine Menge K von *Ausgabesymbolen* $O = \{v_1, v_2, ..., v_K\}$ (Beobachtbar ist eine Folge $\boldsymbol{o} = o_1 o_2, ..., o_T$ mit $o_i \in O, i = i, 2, ..., T$) und
- *Ausgabewahrscheinlichkeiten* b_{jk} die in einer Matrix $\boldsymbol{B}$ gespeichert sind.

Es gilt

$$b_{jk} = b_j(v_k) = P(o_t = v_k | q_t = s_j) \tag{2.23}$$

mit $b_{jk} > 0$ und $\sum_k b_{jk} = 1$.

Ein Hidden-Markov Modell ist dann als Tripel definiert

$$HMM = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) \tag{2.24}$$

In der Praxis betrachtet man nur spezielle Teilmengen aller HMM, insbesondere das lineare HMM mit $\boldsymbol{\pi} = (1, 0, 0, ..., 0)^T$ und $a_{ij} = 0$ für $j - i < 0$ oder $j - i > 1$ Ein Beispiel zeigt Abbildung 2.3.

2.3.2 Diskrete HMM

Ein diskretes HMM (DHMM) wie in Abbildung 2.3 ist durch seine diskreten Ausgabesymbole $O = v_1, v_2, ..., v_K$ gekennzeichnet, die im Zustand s_i mit Wahrscheinlichkeiten $b_{i1}, b_{i2}..., b_{iK}$ ausgegeben werden.

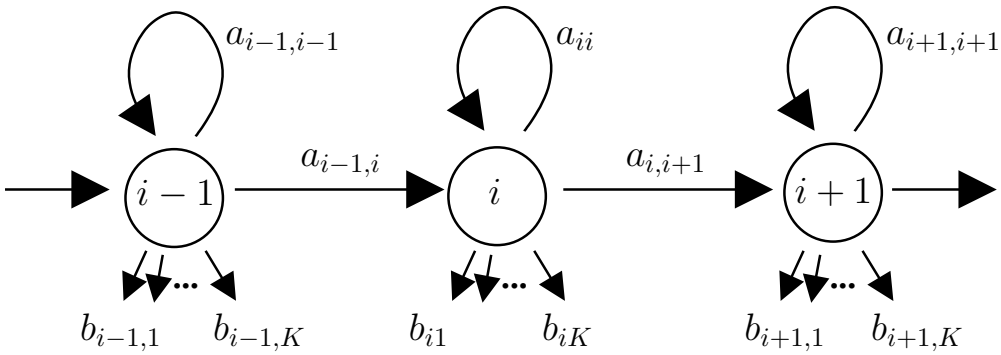


Bild 2.3: Diskretes Hidden-Markov Modell

Zuvor werden die Merkmalvektoren des Sprachsignals aus Abschnitt 2.2.1 in eben diese Symbole v_i mit einem Vektorquantisierer (Abschnitt 2.2.2) quantisiert. Man erhält eine Symbolfolge $\boldsymbol{o} = o_1 o_2 ... o_T$. Bei der Klassifikation fällt die Entscheidung für dasjenige Wort, dessen HMM am wahrscheinlichsten diese Symbolkette erzeugt. Dazu muss für jedes HMM die Wahrscheinlichkeit $P(\boldsymbol{o}|HMM)$ berechnet werden. Dies geschieht mit dem Vorwärtsalgorithmus.

Der Vorwärtsalgorithmus

Eine effiziente Methode zur Berechnung der Wahrscheinlichkeit $P(\boldsymbol{o}|HMM)$ für eine Folge von Beobachtungen $\boldsymbol{o}$ der Länge T bietet der Vorwärtsalgorithmus [Nie90, S.171f] [ST95, S.130f]. Dazu führen wir die *Vorwärtsvariable*

$$\alpha_t(j) = P(o_1 o_2 ... o_t, q_t = s_j | HMM) \tag{2.25}$$

ein. Im analogen Rückwärtsalgorithmus benötigt man sie *Rückwärtsvariable*

$$\beta_t(j) = P(o_{t+1} o_{t+2} ... o_T | q_t = s_j, HMM). \tag{2.26}$$

Die Anzahl der Zustände im HMM sei wieder I. Initialisiert wird die Vorwärtsvariable nun mit $\alpha_1(j) = \pi_j b_j(o_1)$. Weitere Werte werden rekursiv berechnet:

$$\alpha_t(j) = \left(\sum_{i=1}^{I} \alpha_{t-1}(i) a_{ij} \right) b_j(o_t), \quad t \leq T j = 1, 2, ..., I \tag{2.27}$$

Man betrachtet also Pfade über alle Zustände i zur Zeit $t-1$ zum Zustand j zur Zeit t. Durch Summieren über alle Endzustände erhält man schließlich

$$P(\boldsymbol{o}|HMM) = \sum_{j=1}^{I} \alpha_T(j). \tag{2.28}$$

Die Wahrscheinlichkeit, dass die Beobachtungssequenz so erzeugt wurde, dass Zustand s_j zum Zeitpunkt t durchlaufen wird ist

$$P(\boldsymbol{o}, q_t = s_j|HMM) = \alpha_t(j)\beta_t(j). \tag{2.29}$$

Ist der Zustand zur Zeit t nicht von Belang, ergibt sich

$$P(\boldsymbol{o}|HMM) = \sum_{j=1}^{I} \alpha_t(j)\beta_t(j). \tag{2.30}$$

Die Berechnung mit dem Vorwärtsalgorithmus ist sehr effizient. Ist T die Länge der Beobachtungssequenz und I die Anzahl der Zustände, so beträgt die Komplexität $O(I^2 \times T)$. Die nötigen Parameter π_i, a_{ij} und b_{jk} werden in einer Trainingsphase geschätzt.

Baum-Welch Training

Die unbekannten Parameter eines HMM werden unüberwacht mit dem *Baum-Welch Algorithmus* [ST95, S.136f], einer Form des EM-Algorithmus (siehe Abschnitt 2.1.3), gelernt. Zunächst führen wir zwei neue Bezeichnungen ein. Dabei sei $\boldsymbol{o} = o_1 o_2 ... o_T$ wieder die Beobachtungssequenz, diesmal aus der Trainingsstichprobe. Die a-posteriori Wahrscheinlichkeit für Zustand i zur Zeit t sei

$$\gamma_t(i) = P(q_t = s_i|\boldsymbol{o}, HMM) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}. \tag{2.31}$$

Die a-posteriori Wahrscheinlichkeit für den Übergang $s_i \rightarrow s_j$ zur Zeit t sei

$$\begin{aligned} \xi_t(i,j) &= P(q_t = s_i, q_{t+1} = s_j|\boldsymbol{o}, HMM) \\ &= \frac{P(q_t = s_i, q_{t+1} = s_j, \boldsymbol{o}|HMM)}{P(\boldsymbol{o}|HMM)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(i)}{\sum_j \alpha_t(j)\beta_t(j)} \end{aligned} \tag{2.32}$$

Die *Baum-Welch Formeln* für die a-posteriori Schätzwerte von π_i, a_{ij} und b_{jk} sind

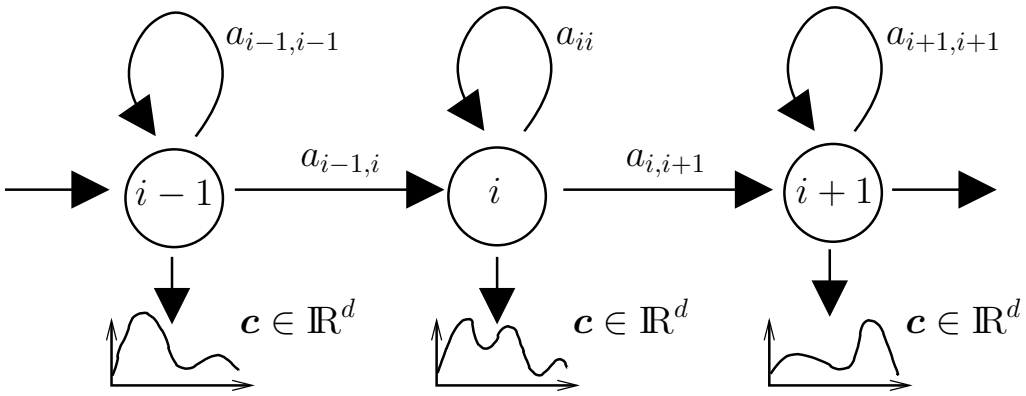


Bild 2.4: Kontinuierliches Hidden-Markov Modell

$$\begin{aligned} \hat{\pi}_i &= \gamma_1(i) \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ \hat{b}_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j) \chi_{[o_t = v_k]}}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{2.33}$$

Die charakteristische Funktion χ_A ist genau dann 1 wenn die Aussage A wahr ist, sonst Null.

Mit dem Baum-Welch Training werden die unbekannten Parameter der HMM geschätzt. Zur Klassifikation einer Beobachtungssequenz werden mit dem Vorwärtsalgorithmus für alle HMM die Wahrscheinlichkeiten $P(\boldsymbol{o}|HMM)$ berechnet. Da zuvor allerdings in einem Zwischenschritt die Merkmalvektoren zu einer Symbolfolge $\boldsymbol{o}$ quantisiert werden, entsteht ein bedeutender Informationsverlust. Diesen versucht man z.B. mit kontinuierlichen HMM aufzufangen.

2.3.3 Kontinuierliche HMM

Bei kontinuierlichen Hidden-Markov Modellen (CHMM) [ST95, S.140ff] werden in jedem Zustand die Merkmalvektoren $\boldsymbol{c} \in \mathbb{R}^d$ statt der diskreten Symbole ausgegeben. So spart man sich den Zwischenschritt der Vektorquantisierung und unnötigen Informationsverlust. Allerdings müssen nun für jeden Zustand nicht nur die K diskreten b_{ik} geschätzt werden sondern d-dimensionale Verteilungsdichten. Dieser Ansatz ist also wesentlich aufwändiger. Ein kontinuierliches HMM zeigt Abbildung 2.4

Die beliebigen Ausgabedichten in jedem Zustand modelliert man üblicherweise durch eine Mischungsverteilung aus K Normalverteilungen. Diese zustandsabhängigen Dichtefunktionen

lassen sich auch als Kodebuch mit K Klassen interpretieren. Für jeden der I Zustände steht also ein eigenes Kodebuch bereit und es müssen $K \times I$ Mittelwertvektoren, Kovarianzmatrizen und Gewichte berechnet werden. Die Ausgabeverteilung im Zustand j ist dann

$$b_j(\boldsymbol{c}) = \sum_{k=1}^{K} c_{jk} g_{jk}(\boldsymbol{c}). \tag{2.34}$$

Dabei ist g_{jk} die k-te Gausskomponente mit Mittelwert $\boldsymbol{\mu}_{jk}$ und Kovarianzmatrix $\boldsymbol{K}_{jk}$, die in der Mischungsverteilung mit c_{jk} gewichtet ist. Analog zu Gleichung 2.23 gilt

$$c_{jk} = P(k_t = k | q_t = s_j) \tag{2.35}$$

Die Zufallsvariable k_t bezeichnet hier die Nummer der Mischungsverteilungskomponente zur Zeit t. Die HMM-Parameter und die Kodebuchparameter werden gemeinsam trainiert und so die Wahrscheinlichkeit $P(\boldsymbol{c}_1\boldsymbol{c}_2...\boldsymbol{c}_T|HMM)$ optimiert. Die Folge von Merkmalvektoren fassen wir in der Matrix $\boldsymbol{C}$ zusammen und bezeichnen nun die a posteriori Wahrscheinlichkeit zur Selektion von Komponente k im Zustand j zur Zeit t mit

$$\begin{aligned} \zeta_t(j,k) &= P(q_t = s_j, k_t = k | \boldsymbol{C}, HMM) \\ &= \frac{P(q_t = s_j, k_t = k, \boldsymbol{C}|HMM)}{P(\boldsymbol{C}|HMM)} \end{aligned} \tag{2.36}$$

Der Zähler ist $\sum_{i=1}^{I} \alpha_{t-1}(i) a_{ij} c_{jk} g_{jk}(\boldsymbol{c}_t) \beta_t(j)$ für $t > 1$ und $\sum_{i=1}^{N} \pi_j c_{jk} g_{jk}(\boldsymbol{c}_1) \beta_1(j)$ für $t = 1$. Die *Baum-Welch Formeln* für die HMM-Parameter sind dann

$$\begin{aligned} \hat{\pi}_i &= \gamma_1(i) \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ \hat{c}_{jk} &= \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_t \gamma_t(j)} \\ \hat{\boldsymbol{\mu}}_{jk} &= \frac{\sum_{t=1}^{T} \zeta_t(j,k) \boldsymbol{x}_t}{\sum_{t=1}^{T} \zeta_t(j,k)} \\ \hat{\boldsymbol{K}}_{jk} &= \frac{\sum_{t=1}^{T} \zeta_t(j,k) \boldsymbol{x}_t \boldsymbol{x}_t^T - \hat{\boldsymbol{\mu}}_{jk} \hat{\boldsymbol{\mu}}_{jk}^T}{\sum_{t=1}^{T} \zeta_t(j,k)} \end{aligned} \tag{2.37}$$

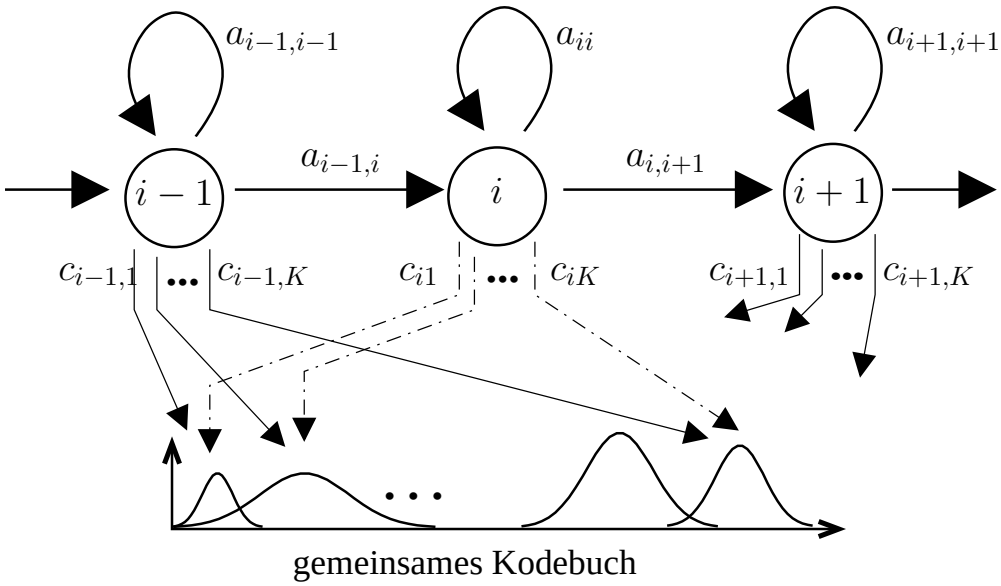


Bild 2.5: Semikontinuierliches Hidden-Markov Modell

Ein Kompromiss zwischen den einfachen diskreten und den aufwändigen aber genaueren kontinuierlichen HMM bietet das semikontinuierliche HMM.

2.3.4 Semikontinuierliche HMM

Um den Informationsverlust, der bei einer vorgeschalteten Vektorquantisierung erfolgt, zu vermeiden, werden beim semikontinuierliche HMM (SCHMM) [ST95, S.144f] wie im kontinuierlichen Fall Kodebuch- und HMM-Parameter gemeinsam trainiert. Die Anzahl der zu schätzenden Parameter wird aber gegenüber dem kontinuierlichen HMM reduziert, indem man nur ein gemeinsames Kodebuch für alle Zustände bereitstellt (*tied mixture model*). Durch das gemeinsame Training des Kodebuches mit den HMM-Parametern erhält man nicht mehr ein solches, das optimal im Sinne eines geringstmöglichen Informationsverlustes ist, sondern es ist optimal im Sinne einer maximalen Wahrscheinlichkeit $P(\boldsymbol{C}|HMM)$. Ein SCHMM zeigt Abbildung 2.5).

Da die Kodebuchdichten g_k nicht mehr vom Zustand abhängen verändert sich Gleichung 2.34 für die Ausgabeverteilung im Zustand j zu

$$b_j(\boldsymbol{c}) = \sum_{k=1}^{K} c_{jk} g_k(\boldsymbol{c}). \tag{2.38}$$

Die Gewichte c_{jk} sind demnach weiterhin zustandsspezifisch, so dass abgeänderte Kodebücher für jeden Zustand modelliert werden. Die *Baum-Welch Formeln* lauten nun

$$
\begin{aligned}
\hat{\pi}_i &= \gamma_1(i) \\
\hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
\hat{c}_{jk} &= \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_{t} \gamma_t(j)} \\
\hat{\boldsymbol{\mu}}_k &= \frac{\sum_{t=1}^{T} \sum_{j=1}^{I} \zeta_t(j,k) \boldsymbol{x}_t}{\sum_{t=1}^{T} \sum_{j=1}^{I} \zeta_t(j,k)} \\
\hat{\boldsymbol{K}}_k &= \frac{\sum_{t=1}^{T} \sum_{j=1}^{I} \zeta_t(j,k) \boldsymbol{x}_t \boldsymbol{x}_t^T - \hat{\boldsymbol{\mu}}_{jk} \hat{\boldsymbol{\mu}}_{jk}^T}{\sum_{t=1}^{T} \sum_{j=1}^{I} \zeta_t(j,k)}
\end{aligned}
\tag{2.39}
$$

In [Sch95, S.38] wird noch eine andere Berechnungsweise für $\zeta_t(j,k)$ (Gleichung 2.36) angegeben, so wie sie auch in der Software am Lehrstuhl für Mustererkennung (LME) implementiert ist:

$$
\begin{aligned}
\zeta_t(j,k) &= P(q_t = s_j, k_t = k | \boldsymbol{C}, HMM) \\
&= P(k_t = k | q_t = s_j, \boldsymbol{C}, HMM) P(q_t = s_j | \boldsymbol{C}, HMM) \\
&= \frac{g_k(\boldsymbol{c}_t) b_{jk}}{\sum_{k=1}^{K} g_k(\boldsymbol{c}_t) b_{jk}} \cdot \gamma_t(j)
\end{aligned}
\tag{2.40}
$$

Den genauen Rechenweg findet man in Anhang A.

Das Vorgehen im *ISADORA*-System (siehe Kapitel 4), das am LME verwendet wird ist diese: Die Kodebuchparameter werden nur etwa alle zwanzig Iterationen des EM-Algorithmus neu geschätzt. Die anschließende Vektorquantisierung erfolgt weich, d.h. es werden die Wahrscheinlichkeiten $P(\boldsymbol{c}_t | k_t = k)$ für alle Klassen berechnet und nicht etwa eine Entscheidung für eine einzelne Klasse getroffen. Es gilt

$$
g_k(\boldsymbol{c}_t) = P(\boldsymbol{c}_t | k_t = k). \tag{2.41}
$$

Eine Beschleunigung ist dabei die Einführung eine Schwellwertes $\theta \in (0, 1]$. Es werden dann die Prüfgrößen $-2 \cdot lnP(\boldsymbol{c}_t | k_t = k)$ (vgl. Abschnitt 2.1.2) für alle $\boldsymbol{c}_t$ und k berechnet. In weiteren Berechnungen betrachtet man nur noch diejenigen Klassen, deren Bewertung um höchstens $-2 \cdot ln\theta$ schlechter ist als die bestbewertete Klasse. Bei großem θ werden also wegen der Logarithmierung weniger Klassen berücksichtigt. Die Einführung dieses Schwellwertes erfolgte im Zusammenhang mit der sequentiellen Klassifikation [ST95, S.112f] [ST93a], die in

der vorliegenden Arbeit keine Rolle spielt. In anderen Erkennungssystemen werden oft auch nur die m besten Klassen berücksichtigt (z.B. $m = 2$ oder $m = 5$) [Hua89, S.243].

Ziel dieser Arbeit ist es, den Übergang vom semikontinuierlichen hin zum kontinuierlichen HMM zu implementieren. Dabei stehen mehrere Kodebücher bereit, die von allen bzw. von Gruppen von Zuständen verwendet werden. Im Kapitel 5 werden für verschiedene Merkmalsarten (z.B. statische und dynamische Merkmale) eigene Kodebücher trainiert, die von allen HMM-Zuständen geteilt werden. In Kapitel 6 verwenden Zustände derselben Lautoberklasse ein gemeinsames Kodebuch. Das folgende Kapitel gibt einen Literaturüberblick.

Kapitel 3

HMM mit mehreren Kodebüchern

Im Rahmen der vorliegenden Diplomarbeit werden semikontinuierliche Hidden-Markov Modelle (SCHMM) mit mehreren Kodebüchern implementiert. Dadurch soll ein Schritt in Richtung kontinuierliche HMM gegangen werden; bei genügend großer Trainingsstichprobe gelten diese als besser [Ace01, S.440 ff] (siehe Abbildung 1.2). Die Anzahl der zu schätzenden Parameter wird dabei nur maßvoll vergrößert. Betrachtet werden diese beiden Teilgebiete: Einmal werden Kodebücher für verschiedene Merkmaltypen (also beispielsweise für statische und dynamische Merkmale) separat trainiert. Die Kodebücher werden von allen Zuständen geteilt. Im zweiten Teil werden für die Lautoberklassen unterschiedliche Kodebücher bereitgestellt. Jeder Zustand gehört zu genau einem Kodebuch.

In diesem Kapitel wird die Vorgehensweise erläutert und es werden diverse Veröffentlichungen zu diesen und ähnlichen Teilgebieten zitiert und zusammengefasst. Die meisten zitierten Artikel wurden zu Beginn der 90er Jahre veröffentlicht, als sich die semikontinuierlichen HMM in der Spracherkennung durchsetzten. Der erneute Einstieg in dieses Forschungsgebiet im Rahmen der Diplomarbeit dient dazu, bewährte Ansätze mit mehreren Kodebüchern in das Spracherkennungssystem des LME zu integrieren und zugleich weitere neue Aspekte dank schnellerer Rechner systematisch zu untersuchen. In den folgenden beiden Abschnitten werden die zwei oben genannten Teilgebiete diskutiert.

3.1 Getrennte Behandlung verschiedener Merkmaltypen

Im folgenden werden verschiedene Typen vom Merkmalen separaten Kodebüchern zugeordnet. Dazu werden semikontinuierliche HMM (SCHMM) mit mehreren Kodebüchern trainiert. Dieser Multi-Kodebuch-Ansatz ist ein Schritt in Richtung kontinuierliche HMM; man verspricht sich

höhere Erkennungsraten. Zunächst wird die genaue Vorgehensweise erläutert und anschließend ein Literaturüberblick gegeben.

3.1.1 Vorgehensweise

Aus dem Sprachsignal werden wie in Abschnitt 2.2.1 beschrieben verschiedene Typen von Merkmalen berechnet. In einer ersten groben Unterteilung werden statische und dynamische Merkmale unterschieden. Erstere sind z.B. Energie und Mel-Cepstrum Koeffizienten, zweitere deren Ableitung. Zunächst teilen wir also den Merkmalvektor $\boldsymbol{c} \in \mathbb{R}^d$ aus $d/2$ statischen Merkmalen und ebenso vielen Ableitungen in die beiden Teilvektoren $\boldsymbol{c}_{stat}$ und $\boldsymbol{c}_{dyn}$ auf.

Für den gewöhnlichen Ansatz aus Kapitel 2 benötigt man ein Kodebuch mit K Klassen. Es wird durch eine Mischungsverteilung aus K Komponenten, jeweils d-dimensionale Normalverteilungen, approximiert. Die k-te Gausskomponente repräsentiert die Verteilung $g_k(\boldsymbol{c}) = P(\boldsymbol{c}|k)$. Beim SCHMM werden alle Normalverteilungen im Zustand j mit zustandsspezifischen Faktoren c_{jk} gewichtet (vgl. Kapitel 2.3.4). Da die Normalverteilungen überlappen, kann der Merkmalvektor $\boldsymbol{c}$ von mehreren Klassen erzeugt werden. Er wird also insgesamt mit der Wahrscheinlichkeitsdichte

$$b_j(\boldsymbol{c}) = \sum_{k=1}^{K} c_{jk} g_k(\boldsymbol{c}). \tag{3.1}$$

erzeugt.

Der Ansatz mit zwei Kodebüchern dagegen erfordert nur $(d/2)$-dimensionale Normalverteilungskomponenten, dafür aber doppelt soviele. Seien nun die Klassen im Kodebuch für die statischen Merkmale mit $k_{stat} \in \{1, 2, ..., K_1\}$ und die Klassen im Kodebuch für dynamische Merkmale mit $k_{dyn} \in \{K_1 + 1, K + 2, ..., K_2\}$ bezeichnet. Ein statischer Merkmalvektor wird von der Kodebuchklasse k_{stat} mit der Dichte $P(\boldsymbol{c}_{stat}|k_{stat})$ erzeugt, die dazugehörigen Ableitungen von einer Klasse k_{dyn} mit der Dichte $P(\boldsymbol{c}_{dyn}|k_{dyn})$. Unter der vereinfachenden Annahme, dass statische Merkmale und ihre dazugehörigen Ableitungen statistisch unabhängig sind, lässt sich die Dichtefunktion für der gesamten Merkmalvektor $\boldsymbol{c}$ folgendermaßen berechnen:

$$\begin{aligned} b_j(\boldsymbol{c}) &= b_j(\boldsymbol{c}_{stat}, \boldsymbol{c}_{dyn}) = b_j(\boldsymbol{c}_{stat}) \cdot b_j(\boldsymbol{c}_{dyn}) = \\ &= \left(\sum_{k_{stat}=1}^{K_1} c_{jk_{stat}} \cdot P(\boldsymbol{c}_{stat}|k_{stat}) \right) \cdot \left(\sum_{k_{dyn}=K_1+1}^{K_2} c_{jk_{dyn}} \cdot P(\boldsymbol{c}_{dyn}|k_{dyn}) \right) \end{aligned} \tag{3.2}$$

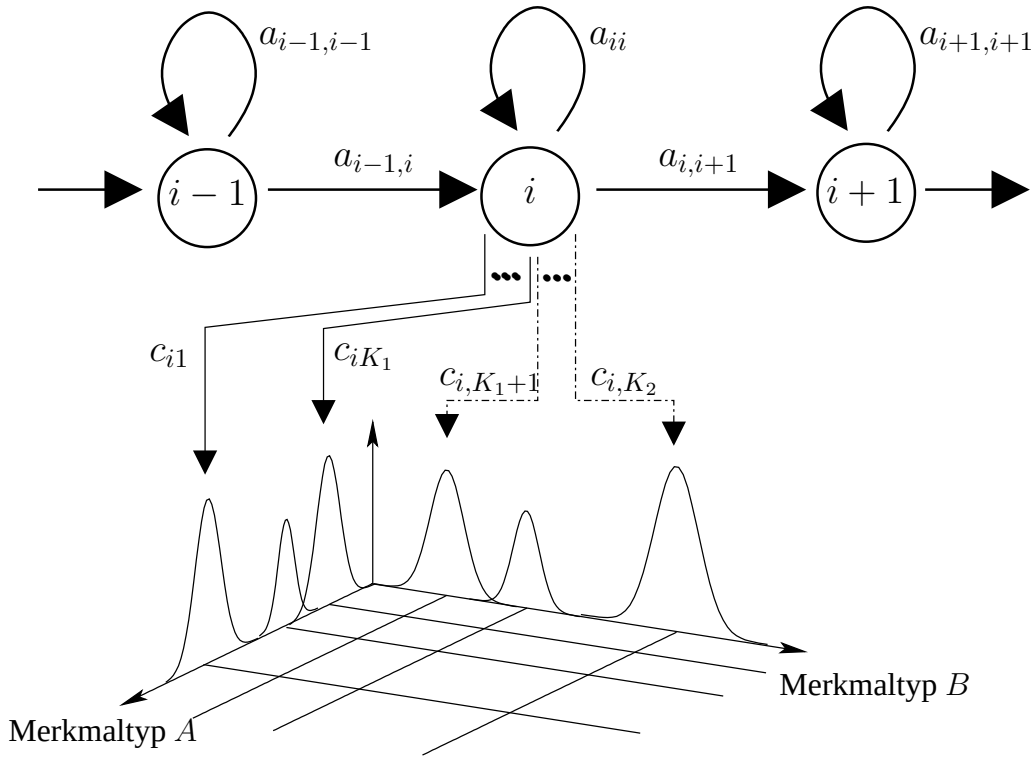


Bild 3.1: Semikontinuierliches Hidden-Markov Modell mit zwei orthogonalen Kodebüchern

Veranschaulicht wird dieses Vorgehen in Abbildung 3.1: Von jedem Zustand werden zur Berechnung der Ausgabewahrscheinlichkeit zwei Kodebücher herangezogen, eines mit K_1 Klassen für den Merkmaltyp A und ein zweites mit K_2-K_1 Klassen für Merkmaltyp B. Da die Merkmale A und B statistisch unabhängig sind, sind die beiden in Abbildung 3.1 eindimensionalen Merkmalräume orthogonal zueinander eingezeichnet. Über jedem Merkmalraum ist die Verteilung, die durch das jeweilige Kodebuch gegeben ist, sichtbar; nach oben ist die Wahrscheinlichkeit abgetragen. Wählt man nun einen festen Wert für A, so kann der davon unabhängige Wert B beliebig sein. Für jede Möglichkeit "A *und zugleich* B" findet man einen Punkt in der schraffierten Ebene. Die Wahrscheinlichkeit für dieses Ereignis "A *und zugleich* B" ergibt sich durch Multiplikation der beiden Wahrscheinlichkeiten für A und B. Man kann sich das Produkt der beiden eindimensionalen Kodebücher als zweidimensionale Mischungsverteilung über der schraffierten Ebene vorstellen; man erhält $K_1 \cdot (K_2 - K_1)$ Gaussglocken über der Ebene. Aus zwei linear unabhängigen Kodebüchern mit insgesamt K_2 eindimensionalen Normalverteilungen erhält man also ein zweidimensionales Kodebuch mit weitaus mehr Klassen. Allerdings können diese nur eingeschränkt von den einzelnen Zuständen modelliert werden, da nur K_2 verschiedene Gewichte zur Verfügung stehen. In dieser Arbeit wird untersucht, inwieweit mit diesem Vorgehen

Kodebücher flexibler gestaltet werden können und die Wortakkuratheit der Erkenners verbessert werden kann.

Eine weitere Einschränkung bei einem Kodebuch, das aus zwei unabhängigen Kodebüchern modelliert wird, ist die Tatsache, dass die stochastische Unabhängigkeit Unkorreliertheit impliziert. Die Kovarianzmatrix einer zehndimensionalen Normalverteilungskomponente eines Kodebuches, das aus zwei fünfdimensionalen Kodebüchern aufgespannt wird, hat also die Form

$$\boldsymbol{K} = \begin{pmatrix} * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \end{pmatrix}. \tag{3.3}$$

Im Single-Kodebuch-Ansatz ist die gesamte Matrix voll besetzt ($*$); der Speicherbedarf halbiert sich hier also, wenn die Korrelation zwischen den Merkmaltypen verloren geht.

Nun vergleichen wir die Rechenkomplexität des Single-Kodebuch-Ansatzes mit K Klassen mit dem Multi-Kodebuch-Ansatz und insgesamt ebenfalls $K_2 = K$ Klassen. Die Anzahl der Produkte bei der Vektorquantisierung (Gleichung 2.11) verringert sich im Ansatz mit mehreren Kodebüchern und niedrigerer Vektordimension stark. In Gleichung 3.2 dagegen erkennt man einen geringen Mehraufwand gegenüber Gleichung 3.1 (eine Multiplikation statt einer Addition im Multi-Kodebuch-Ansatz). Die Kodebuchneuschätzung wiederum vereinfacht sich aufgrund der niedrigeren Merkmaldimension der Teilvektoren (Formeln siehe Gleichungen 2.40). Auch die Berechnung von $\zeta_t(j,k)$, der a posteriori Wahrscheinlichkeit zur Selektion der Kodebuchklasse $k_t = k$ im Zustand $q_t = s_j$ (siehe Gleichung 2.41 bzw. Anhang A), wird einfacher. Es gilt nun für das Kodebuch der statischen Merkmale ($k \leq K_1$) mit $g_k(\boldsymbol{c}_{stat,t}) = P(\boldsymbol{c}_{stat}|k_t = k)$:

$$\begin{aligned}\zeta_t(j,k) &= P(q_t = s_j, k_t = k|\boldsymbol{C}, HMM) \\ &= P(q_t = s_j|\boldsymbol{C}, HMM) \cdot P(k_t = k|q_t = s_j, \boldsymbol{C}, HMM) \qquad (3.4) \\ &= P(q_t = s_j|\boldsymbol{C}, HMM) \cdot \frac{g_k(\boldsymbol{c}_{stat,t}) b_{jk}}{\sum_{\kappa=1}^{K_1} g_k(\boldsymbol{c}_{stat,t}) b_{jk}} \qquad (3.5)\end{aligned}$$

$\boldsymbol{C}$ ist die Folge von Merkmalvektoren und t der aktuelle Zeitpunkt.

Der Multi-Kodebuch-Ansatz lässt sich natürlich auf beliebig viele Kodebücher verallgemeinern. Es gilt im Fall mit N Kodebüchern und jeweils K_n Klassen

$$b_j(\boldsymbol{c}) = \prod_{n=1}^{N} \sum_{k=1}^{K_n} c_{jk} g_k(\boldsymbol{c}_{Teil\ n}). \qquad (3.6)$$

In der Literatur findet man ähnliche Verfahren, die im nächsten Abschnitt zusammengefasst sind.

3.1.2 Literaturüberblick

Der Ansatz, mehrere Kodebücher für unterschiedliche Typen von Merkmalvektoren zu verwenden ist in de Literatur weit verbreitet. Es wird auch häufig von unabhängigen Datenströmen (“*streams*”) gesprochen.

In [ST95, S.324 f] werden Untersuchungen zur *faktorisierten Vektorquantisierung* unternommen. Dabei wird die statistische Unabhängigkeit der cepstralen Merkmale und deren Ableitungen angenommen. Durch die getrennte Vektorquantisierung erhofft man sich Zeitersparnis und eine feinerer Aufteilung des Merkmalraumes. Haben die beiden Kodebücher K_1 bzw. $K_2 - K_1$ Merkmale, so ergeben sich $K_1 \cdot (K_2 - K_1)$ Klassengebiete. Die Versuchsergebnisse zeigen, dass die Cepstrummerkmale alleine wesentlich besser als die Ableitungsmerkmale sind. Mit zwei Kodebüchern ergibt sich zwar eine Verbesserung gegenüber den statischen Merkmalen alleine, jedoch ist das Experiment mit *einem* gemeinsamen Kodebuch deutlich besser.

Im Lehrbuch [Ace01, S.439 ff] wird beschrieben, dass die Vorgehensweise mit mehreren Kodebüchern für verschiedene Merkmaltypen zur signifikanten Verminderung der Wortfehlerrate um über 10 % (relativ) führt. Dabei bezieht sich der Autor auf das Spracherkennungssystem SPHINX. Wiederum werden die verschiedene Merkmalsarten, wie Energie, cepstrale Merkmale, die ersten Ableitungen und die zweiten Ableitungen als unabhängig angenommen. Der Ansatz wird dort kurz formalisiert. Die Wahrscheinlichkeit, dass im Zustand j der komplette Merkmal-

vektor $\boldsymbol{c}$ ausgegeben wird, wird wie in Gleichung 3.6 angegeben berechnet. Auch eine Formel zur Kodebuchneuschätzung wird äquivalent zu Gleichung 3.5 aufgeführt.

Eine Erweiterung des Ansatzes mit mehreren Kodebüchern ist eine zusätzliche Gewichtung der einzelnen Datenströme $n = 1, 2, ..., N$ mit Gewichten α_n (Kodebuchexponenten)

$$b_j(\boldsymbol{c}) = \prod_{n=1}^{N} b_j(\boldsymbol{c}_{Teil\ n})^{\alpha_n}. \tag{3.7}$$

Für die logarithmierte Wahrscheinlichkeiten gilt

$$-2 \ln b_j(\boldsymbol{c}) = -2 \sum_{n=1}^{N} \alpha_n \ln b_j(\boldsymbol{c}_{Teil\ n}). \tag{3.8}$$

Im Artikel [Rog94] wird dieser Ansatz beschrieben und es werden die Einschränkungen $0 \leq \alpha_n \leq 1$ und $\sum_n \alpha_n = 1$ gefordert. In einem Referenzsystem werden die Gewichte für statische Merkmale mit 0.55 und für dynamische Merkmale mit 0.45 initialisiert. Nun werden in [Rog94] individuelle Gewichte $\alpha_n(i)$ für verschiedene HMM-Zustände i trainiert, da verschiedene Laute wohl die einzelnen Kodebücher in unterschiedlicher Weise begünstigen. So konnte die Wortfehlerrate um 20 % (relativ) reduziert werden.

Im Spracherkennungssystem SPHINX, das an der Carnegie Mellon University in Pittsburgh, USA, entwickelt wurde, ist auch ein Multi-Kodebuch-Ansatz integriert. In Sphinx-II [Hua93] werden drei Kodebücher verwendet: eines für 12 statische Merkmale, eines für 12 Differenzen über 40 ms und ein drittes mit Energie und deren Ableitung. Später wird das System mit einem vierten Kodebuch mit Differenzen zweiter Ordnung ergänzt; die zweite Ableitung der Energie wird vom dann dreidimensionalen Energiekodebuch erzeugt. Die Kodebuchparameter werden gemeinsam mit den HMM-Parametern neu geschätzt, nur die Kovarianzmatrix der Energie bleibt konstant. Erste Untersuchungen mit der Implementation des semikontinuierliche HMM im SPHINX-System werden in [Hua90] beschrieben. Es werden mehrere Kodebücher für verschiedene Merkmaltypen benutzt.

Die Untersuchungen in [Hua91] beziehen sich auch auf das SPHINX-System. Die dynamischen Merkmale (Differenzen der statischen Merkmale) werden hier über verschiedene Zeitauflösungen berechnet. Zunächst werden Differenzen über 40 ms und über 80 ms Zeitfenster von separaten Kodebüchern quantisiert. Die Erkennungsraten können aber nicht signifikant verbessert werden, da die Unabhänggigkeitsannahme zwischen verschiedenen Merkmaltypen nun verletzt ist. Werden die beiden verschiedenen Differenzmerkmale aber gemeinsam von einem Kodebuch erzeugt, kann die Erkennungsrate verbessert werden. Die Kovarianzmatrizen sind in

diesen Untersuchungen Diagonalmatrizen.

In [Ste02] werden in jedem Zustand parallel mit einem Phonem-Erkenner und einem Wort-Erkenner die Ausgabewahrscheinlichkeiten berechnet. Die beiden Kodebücher werden mit Kodebuchexponenten gewichtet.

In vielen weiteren Veröffentlichungen stößt man auf den Hinweis, dass verschiedene Kodebücher für verschiedene Datenströme verwendet werden, ohne dass speziell Untersuchungen in diesem Gebiet unternommen werden (z.B. [Wil97]).

Nachdem in diesem Abschnitt verschiedene Kodebücher für unterschiedliche Merkmaltypen diskutiert wurden, werden im folgenden Abschnitt Kodebücher für verschiedene Lautoberklassen betrachtet.

3.2 Getrennte Behandlung verschiedener Lautoberklassen

Eine weitere Möglichkeit, durch Bereitstellung mehrerer Kodebücher einen Schritt vom semikontinuierlichen HMM hin zum kontinuierlichen HMM zu machen und gleichzeitig die Anzahl der zu schätzenden Parameter nur maßvoll zu erhöhen, wird in diesem Kapitel diskutiert. Es werden verschiedene Kodebücher für verschiedene Lautoberklassen bereitgestellt. Wieder verspricht man sich durch wenig Mehraufwand geringere Fehlerraten in der Spracherkennung. Zunächst wird der Ansatz allgemein erläutert und dann verschiedene ähnliche Arbeiten aus der Literatur vorgestellt.

3.2.1 Vorgehensweise

Semikontinuierliche HMM (SCHMM) werden auch als *tied mixture models* bezeichnet. Von allen Zuständen wird dasselbe Kodebuch geteilt. Als Kodebuch bezeichnen wir eine Gaußsche Mischungsverteilung aus $\bar{K}$ Normalverteilungen ($\bar{K}$ Klassen). Von der Verteilungsfunktion wird nun wieder – anders als im letzten Abschnitt – der gesamte Merkmalvektor erzeugt. Betrachten wir nun ein HMM mit N Kodebüchern, wobei das erste Kodebuch K_1 Klassen besitzt und die anderen Kodebücher $K_n - K_{n-1}$ Klassen ($n \in \{2, 3, ..., N\}$). Wir können nun alle Kodebücher zu einem einzigen mit $K = K_N$ Klassen zusammenfassen. Will man danach wieder ein bestimmtes Kodebuch betrachten, so muss man z.B. in eine Tabelle nachschlagen, welche Kodebuchkomponenten benötigt werden.

Generell gibt es zwei Extrema, wie die einzelnen Komponenten des Kodebuches an die verschiedenen HMM-Zustände gebunden werden können (“*tying*”): Beim SCHMM werden alle

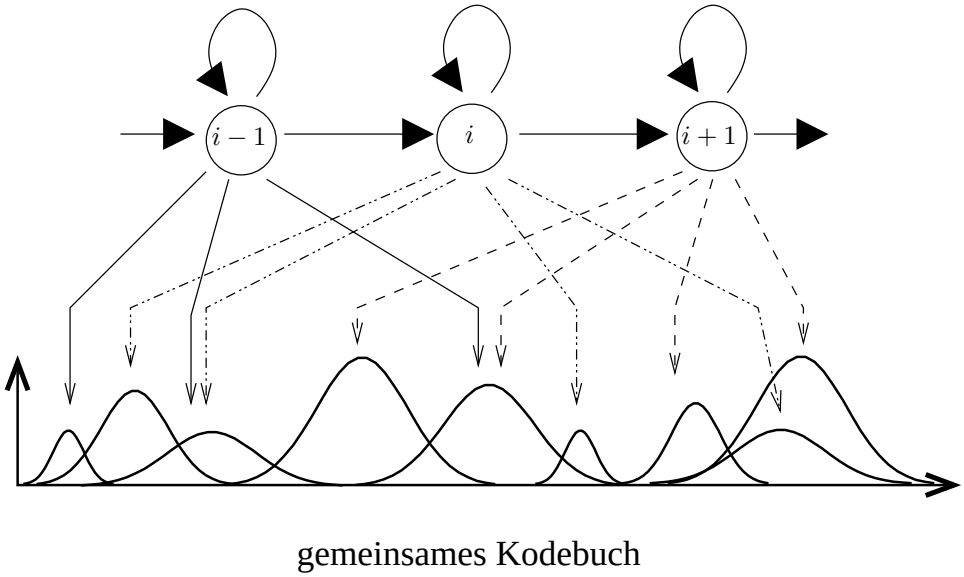


Bild 3.2: Beliebige Bindungen zwischen Zuständen und Kodebuchklassen bei HMM mit mehreren Kodebüchern

Komponenten von allen Zuständen geteilt und beim kontinuierlichen HMM ist jede Komponente nur an genau einen Zustand gebunden. Dazwischen sind aber auch beliebige Bindungen möglich. Jeder Zustand benötigt nun eine beliebige Teilmenge der Mischungskomponenten für sein individuelles Kodebuch. Skizziert ist dieser Ansatz in Abbildung 3.2. Es kann auftreten, dass zwei Mischungskomponenten übereinander liegen und bis auf Skalierung identisch sind. Dann liegt Redundanz vor, denn die Skalierungen werden durch zustandsspezifische Gewichte c_{jk} (vgl. Abbildung 2.5 Abschnitt 2.3.4) vorgenommen. In diesem Fall können die beiden Komponenten verschmolzen werden. Andere Komponenten hingegen sind an sehr viele Zustände gebunden und können geteilt werden, um die Verteilungsfunktion feiner zu modellieren. Hinweise zur Umsetzung dieser Überlegungen werden im anschließenden Literaturabschnitt gegeben.

In dieser Diplomarbeit wird untersucht, ob die Wortakkuratheit des Spracherkenners erhöht werden kann, wenn man unterschiedliche Kodebücher für verschiedene Lautoberklassen implementiert. Jede Mischungsverteilungskomponente wird also nur von Zuständen einer bestimmten Lautoberklasse geteilt. Die Kodebücher unterschiedlicher Lautoberklassen sind disjunkt, mögliche Redundanz wird also nicht geprüft. Da Merkmalvektoren derselben Lautoberklasse im Merkmalraum nahe beieinanderliegen und von Merkmalvektoren anderer Klassen möglichst getrennt sind, werden aber auch die Mittelwerte der Normalverteilungen desselben Kodebuches innerhalb eines stark begrenzten Bereiches liegen. In Abbildung 3.3 werden drei Kodebücher für drei verschiedene Lautoberklassen im zweidimensionalen Merkmalraum veranschaulicht.

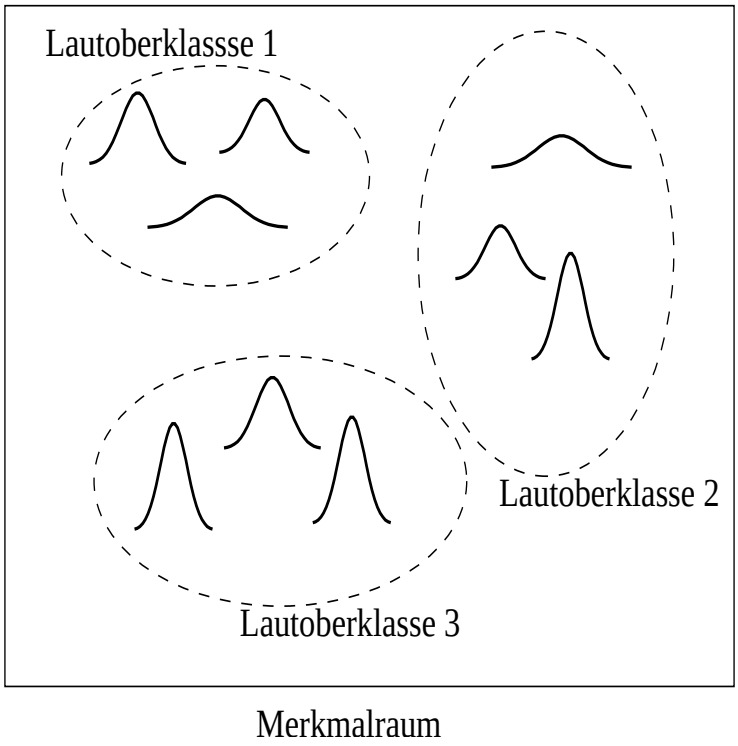


Bild 3.3: Drei Kodebücher für verschiedene Lautoberklassen mit je drei Normalverteilungskomponenten

Um initiale Kodebücher für die einzelnen Oberklassen zu berechnen, benötigt man eine etikettierte Stichprobe, d.h. für jedes Kurzzeitanalysefenster muss die Klassenzugehörigkeit bekannt sein. In die Dichteberechnung für ein Kodebuch gehen nur die Merkmalvektoren der jeweiligen Oberklasse ein. Die Beschriftung erfolgt datengetrieben: Zunächst wird ein Erkenner wie in Abschnitt 4.3 beschrieben trainiert. Dabei werden so genannte *hlb-Dateien* erzeugt (siehe Abschnitt 4.3.1). Sie beinhalten die wahrscheinlichste Zustandsfolge im HMM für eine bestimmte Trainingsäußerung. Die Anzahl der durchlaufenen Zustände ist stets identisch mit der Länge der Äußerung in Kurzzeitanalysefenstern, was wiederum der Anzahl der Merkmalvektoren entspricht. So erhält man für jeden Vektor seine geschätzte Klassenzugehörigkeit. Dies geschieht im ISADORA-System (siehe Abschnitt 4.3) mit dem Programm `laminate`.

Das Vorgehen in diesem Abschnitt entspricht einer deutlichen Erhöhung der Anzahl von Kodebuchklassen und gleichzeitigem Nullsetzen einer Vielzahl von Klassengewichten c_{jk}. Durch zusätzliches Einfließen von Wissen über die Verteilung einzelner Lautoberklassen im Merkmalraum erhofft man sich bessere Erkennungsraten.

Ein ähnliches Verfahren wurde am LME schon untersucht [ST95]. Damit beginnt der Literaturüberblick im nachfolgenden Abschnitt.

3.2.2 Literaturüberblick

In der Literatur gibt es zahlreiche Ansätze, separate Kodebücher für bestimmte Gruppen von Zuständen zu berechnen. Oft werden einzelne Komponenten der Kodebücher wieder verschmolzen.

In [ST95, S.322f] werden Experimente mit einem phonetischen Mischungsverteilungskodebuch beschrieben. Dazu werden für verschieden Laute mittels einer etikettierten Stichprobe eigene Kodebücher mit drei, fünf oder zehn Klassen berechnet. Die Etikettierung erfolgt datengetrieben wie im letzten Abschnitt beschrieben. Die Kodebücher werden zu einem großen Kodebuch vereinigt. Ähnliche Normalverteilungskomponenten der verschiedenen Kodebücher werden verschmolzen, um die Gesamtzahl der Dichten zu reduzieren. So kann die Erkennungsrate gegenüber einem Kodebuch, das mit dem LBG-Algorithmus erzeugt wurde, deutlich verbessert werden. Diese Vorgehensweise ist auch im ISADORA-System (siehe Abschnitt 4.3) implementiert. Allen HMM-Zuständen stehen bei diesem Ansatz sämtliche Kodebuchklassen zur Verfügung.

Das *tied mixture model* wird in [Wil97] verallgemeinert. Beim SCHMM werden nach jeder Kodebuchneuschätzung im Zustand j Bindungen zu Kodebuchkomponenten mit kleinem c_{jk} (vgl. Abbildung 2.5) entfernt. Dabei bleibt die Summe der nicht berücksichtigten c_{jk} unter einem Schwellwert. Alle anderen Gewichte c_{jk} müssen wieder normalisiert werden. Anschließend werden Normalverteilungskomponenten, die an sehr viele Zuständen gebunden sind, in zwei Klassen geteilt. Dazu werden Mittelwertvektor und Kovarianzmatrix dieser Klasse separat für jeden Zustand neu berechnet. Die einzelnen Normalverteilungen werden dann wieder durch Mittelung so zu zwei Gruppen zusammengefasst, dass die Trainingssequenz mit maximaler Wahrscheinlichkeit erzeugt wird. Ein resultierendes HMM mit 400 Kodebuchkomponenten und durchschnittlich 20 Bindungen pro Zustand ist besser und schneller als das Vergleichssystem, ein SCHMM mit 200 Kodebuchklassen (und 200 Bindungen pro Zustand).

Ein anderer Ansatz wird in [Dig96] beschrieben. Dort werden Teilmengen von HMM-Zuständen mit gemeinsamen Mischungskomponenten gesucht. Eine Menge von Normalverteilungen, die von einer solchen Zustandsmenge geteilt wird, bezeichnet man als *Genon*. Zum Festlegen dieser *Genone* wird ein *Clustering - and - Splitting*- Algorithmus beschrieben. Das Zusammenfassen von Zuständen zu Clustern erfolgt, wenn die Zunahme der Entropie minimal ist.

Im Artikel [You92] werden in Experimenten verschiedene HMM-Parameter ganz allgemein von bestimmten Komponenten geteilt, z.B. die Kovarianzmatrizen der Dichten oder die Mittelwertvektoren.

Experimente und Ergebnisse zu den Verfahren aus diesem Kapitel findet man weiter unten. Zuvor folgt nun noch ein Kapitel zur Beschreibung des Versuchsaufbaus und des Referenzsystems.

Kapitel 4

Versuchsaufbau und Referenzsystem

4.1 Die Stichprobe

Zur Durchführung der Untersuchungen, die in diesem und den folgenden Kapiteln vorgestellt werden, stehen zwei Stichproben mit Sprachdaten zur Verfügung: Primär wird die EVAR-Stichprobe herangezogen; ausgewählte Experimente werden zusätzlich mit einer weiteren Stichprobe durchgeführt, um den Einfluss von Störgeräuschen zu untersuchen.

Bei der EVAR-Stichprobe handelt es sich um Anfragen verschiedener Personen in kontinuierlicher, frei gesprochener Sprache an das Zugfahrplan-Auskunft-System am Lehrstuhl für Mustererkennung (LME). EVAR bedeutet “Erkennen, Verstehen, Antworten, Rückfragen” und ist z.B. in [Gal98] beschrieben. Die gesamte Stichprobe umfasst 20678 Äußerungen, gemischt aus Mikrophon- und Telefonaufnahmen. Bereits in der Studienarbeit [Hac01] wurden alle Mikrophonaufnahmen bandpaßgefiltert, so dass sie nun keine nennenswerten Frequenzanteile über 4000 Hz mehr enthalten. Aus allen diesen Äußerungen in Telefonqualität wurde zufällig eine Teilmenge von insgesamt 7438 Sätzen ausgewählt, um das Training und Testen zu beschleunigen. Diese wurde in drei disjunkte Stichproben zerlegt, eine Trainingsstichprobe bestehend aus 4999, eine Validierungsstichprobe aus 441, und eine Teststichprobe aus 1998 Äußerungen[1]. Insgesamt entspricht dies etwa acht Stunden gesprochener Sprache; fast alle vorkommenden Wörter sind Deutsch. Diese Stichprobe wird im folgenden mit ERDIAL_4999 bezeichnet.

Die zweite Stichprobe ist VERBMOBIL_TINY. Dazu wurden aus der VERBMOBIL-Stichprobe[2] in der Arbeit [Had02, S.41] 1880 Dateien ausgewählt, die insgesamt etwa vier Stun-

[1] Es hat keinerlei Bedeutung, dass die Größe der Stichproben nicht etwa 5000 bzw. 2000 beträgt (Durch einen Fehler zu Beginn der Studienarbeit sind drei gefilterte Dateien verloren gegangen).

[2] http://verbmobil.dfki.de

den Sprache enthalten. Die Dateien beinhalten Gespräche zur Terminvereinbarung; die Äußerungen sind deutlich länger als jene Anfragen zur Zugauskunft in der EVAR-Stichprobe. Die Trainingsstichprobe von VERBMOBIL_TINY besteht aus 940 Dateien, die Validierungsstichprobe aus 94 und die Teststichprobe aus 846 Dateien. Es wurden die VERBMOBIL Daten-CDs Nr.1 und Nr.2 herangezogen.

In den Experimenten mit VERBMOBIL_TINY wird das *Sprachmodell* des Erkenners mit einer größeren Menge von etwa 29.5 Stunden Sprache trainiert. Diese größere Teilmenge der Verbmobil-Stichprobe ist in [Had02, S.40f] beschrieben.

Zum Testen verwenden wir jedoch nicht die Testmenge aus VERBMOBIL_TINY. In der Müdigkeitsstichprobe[3] [Had02, S.19ff] wurden in Aufnahme 6 bis 9 Texte von der VERBMOBIL Daten-CD Nr.1 gesprochen. Wir betrachten hier die Aufnahme 6 und 7, mit einer Länge von etwa 8000 Wörtern. Die Aufnahme erfolgte sowohl mit einem Nahbesprechungsmikrophon als auch mit verschiedenen Raummikrophonen. Die Aufzeichnungen der Raummikrophone sind verhallt. Wir betrachten in dieser Arbeit nur das Mikrophon Nummer 7, das direkt dem Sprecher gegenüber angebracht war[Had02, S.23]. Im nächsten Kapitel werden verschiedene Erkenner mit VERBMOBIL_TINY trainiert und anschließend separat auf den beiden Stichproben mit und ohne Hall getestet. So kann untersucht werden, inwiefern die in der Diplomarbeit entwickelten Verfahren, gegenüber Störeinflüssen wie Hall robust sind und ob die Ergebnisse auch in einem weiteren Anwendungsgebiet gültig sind.

4.2 Bewertung

Bevor im nächsten Abschnitt die Vorgehensweise zum Testen eines Erkenners erläutert wird, wird nun zunächst ein Gütemaß für die Bewertung der erkannten Wortfolge eingeführt [Rie94, S.115f]. Dazu wird jene Wortkette mit der für die Testdaten vorliegenden Verschriftung verglichen. Man unterscheidet drei mögliche Abweichungen der erkannten Wortfolge von der tatsächlich gesprochenen:

- Ein nicht gesprochenes Wort wurde fälschlicherweise eingefügt. w_{ins} bezeichnet die Anzahl solcher Einfügungen (engl.: *insertions*).
- Ein gesprochenes Wort wurde ausgelassen. Die Anzahl solcher Auslöschungen (engl.: *deletions*) wird mit w_{del} bezeichnet.

[3] Die Probanden waren eine ganze Nacht lang wach. Es wurden Stichproben in verschiedenen Müdigkeitszuständen aufgenommen.

- Ein gesprochenes Wort wurde falsch erkannt, d.h. durch ein falsches substituiert. w_{subs} ist die Anzahl der Substitutionen.

Es können also zwei verschieden lange Wortketten zum Vergleich vorliegen. Sei nun w_{ges} die Gesamtzahl der gesprochenen Wörter. Dann definiert man die Wortakkuratheit

$$WA = \left(1 - \frac{w_{subs} + w_{ins} + w_{del}}{w_{ges}}\right) \cdot 100\% \tag{4.1}$$

Der Levensthein-Abstand [Lev66] gibt die minimale Anzahl von Einfügungen, Auslöschungen und Substitutionen an, die benötigt werden, um die gesprochene Wortkette auf die erkannte abzubilden. Man beachte, dass WA in sehr schlechten Fällen auch negativ werden kann. Das Komplement zur Wortakkuratheit ist die Wortfehlerrate. Bei der Wortkorrektheit bleiben die Einfügungen unberücksichtigt:

$$WC = \left(1 - \frac{w_{subs} + w_{del}}{w_{ges}}\right) \cdot 100\% \tag{4.2}$$

Ein Spracherkenner wird zunächst mit der Trainingsstichprobe trainiert und anschließend die Wortakkuratheit auf der Teststichprobe berechnet.

4.3 Training und Test

Nachdem ein Erkenner oft tagelang trainiert wurde[4] , kann er in Echtzeit getestet werden. Das Training erfolgt mit dem HMM-basierten Mustererkennungssystem ISADORA und wird im folgenden beschrieben. Das Testen mit dem LR_BEAM-Erkenner wird anschließend erläutert.

4.3.1 Training mit ISADORA

In diesem Abschnitt wird das Training mit dem Mustererkennungssystem ISADORA, das am LME entwickelt wurde, erläutert. Eine Beschreibung dieses Systems findet man in [ST95, S.271ff.] bzw in [ST93b]. Eine schematische Darstellung zeigt Abbildung 4.1. Für die Stichprobe liegt eine Verschriftung aller Sprachdateien vor, sowie die phonetische Transkription der 2640 Wörter des verwendeten Wortschatzes. Alle Laute und Nonverbalien (Geräusche wie z.B. Husten) sind wiederum in ihre subphonemischen Untereinheiten zerteilt.

[4]Gearbeitet wurde an Intel Pentium-III Rechnern mit 600 bzw. 700 MHz Prozessoren. Das gesamte Training eines Erkenners dauert etwa 4 Tage, wenn die Rechner tagsüber zugleich von Studenten zum Arbeiten genutzt werden. Auch bei Benutzung mehrerer Rechner war also die Gesamtzahl der durchführbaren Experimente sehr eingeschränkt.

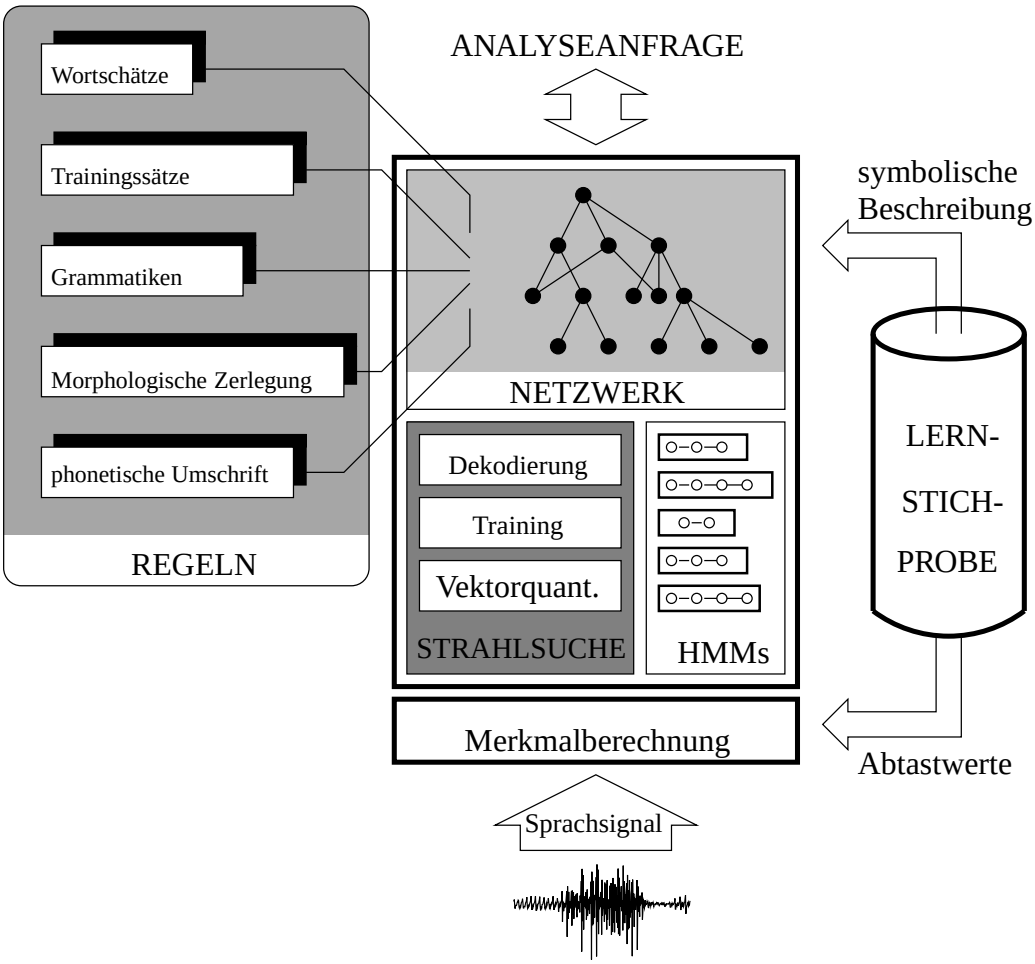


Bild 4.1: Architektur des ISADORA-Systems, aus [ST95, S.279]

Das ISADORA-System basiert auf *rekursiven Markov Modellen* (RMM), speziellen HMM, deren Zustände auch komplex sein dürfen und wieder aus HMM bestehen. Es entsteht ein Netzwerk, in dem die Wörter und die phonetischen Einheiten unterhalb der Wortebene hintereinander oder parallel für Alternativen und Spezialisierungen geschaltet werden können.

Im folgenden wird beschrieben, welche Programme beim Training der Reihe nach aufgerufen werden. Dann wird das Vorgehen beim Training, wie es in der Diplomarbeit erfolgt, erklärt.

Aus den Sprachsignalen der Trainings- und Validierungsstichprobe werden zunächst Merkmalvektoren berechnet. Mit dem Programm `fex3_1m` aus der Studienarbeit [Hac01] werden als statische Merkmale die Kurzzeitenergie sowie 11 Mel-Cepstrum-Merkmale berechnet und als dynamische Merkmale deren 12 erste Ableitungen. Die Merkmalberechnung erfolgt wie in Abschnitt 2.2.1 beschrieben. Die Breite des Kurzzeitanalysefensters beträgt 10 ms. Die 12 Ableitungen werden aus einem Zeitfenster von 50 ms mit Hilfe der Regressionsgeraden approximiert;

in einzelnen Experimenten in Kapitel 5 werden aus verschiedenen Zeitfenstern je 12 Ableitungen berechnet.

Das Programm `vqd` berechnet ein initiales Kodebuch *erdial.cch.raw*; mit `buildnvk` wird daraus ein Normalverteilungsklassifikator *erdial.cch* gebaut. Danach werden die Merkmalvektoren mit `gausseq` quantisiert. Wie bereits im Abschnitt 2.3.4 beschrieben führt man einen Schwellwert θ ein. Dann werden für jeden Merkmalvektor nur diejenigen Klassen betrachtet, deren Wahrscheinlichkeit $P(\boldsymbol{c}_t|k_t = k)$ um höchstens $-2 \cdot ln\theta$ schlechter ist als die bestbewertete Klasse. Bei großem θ werden also weniger Klassen berücksichtigt; alle berücksichtigten Klassen samt ihrer Bewertung werden in einer *svq-Datei* gespeichert.

Der `initializer` mit Option `-u` dient zur uniformen Initialisierung des HMM-Netzwerkes *erdial.apn*. Die Gewichtungen c_{jk} aus Abbildung 2.5 werden bei K Klassen alle auf $1/K$ gesetzt. Nun können mit dem `trainer` die *hlb-Dateien* erzeugt werden, die für jede Äußerung der Stichprobe die wahrscheinlichste Zustandsfolge angeben.

Nach einer erneuten vollständigen Initialisierung des HMM-Netzwerk mit dem `initializer` werden die HMM-Parameter mit dem `trainer` trainiert. Das Kodebuch bleibt konstant. Dazu wird der Baum-Welch-Algorithmus aus Kapitel 2 herangezogen. Trainiert wird 10 Iterationen lang; Protokolliert wird nach jeder Iteration die Bewertung

$$b = \frac{-\ln P(\boldsymbol{C}_{val}|HMM)}{T}. \tag{4.3}$$

Die Wahrscheinlichkeit $P(\boldsymbol{C}_{val}|HMM)$ der Folge von Merkmalvektoren $\boldsymbol{C}_{val}$ aus der Validierungsstichprobe sollte dabei möglichst groß und damit der Gesamtterm möglichst klein werden. T ist die Länge der Äußerung.

Nun wiederholen sich die folgenden Schritte.

1. Neuschätzen des Kodebuches mit dem `trainer` und anschließende Vektorquantisierung (Berechnung der *svq-Dateien*).

2. Initialisierung des HMM-Netzwerks (`initializer`).

3. 10 Iterationen Baum-Welch-Training der HMM-Parameter mit dem `trainer` und Berechnung von b

4. Berechnung der wahrscheinlichsten Zustandsfolge (*hlb-Dateien*) mit dem `trainer`.

5. Initialisierung des HMM-Netzwerks mit *hlb*-Daten (`initializer`).

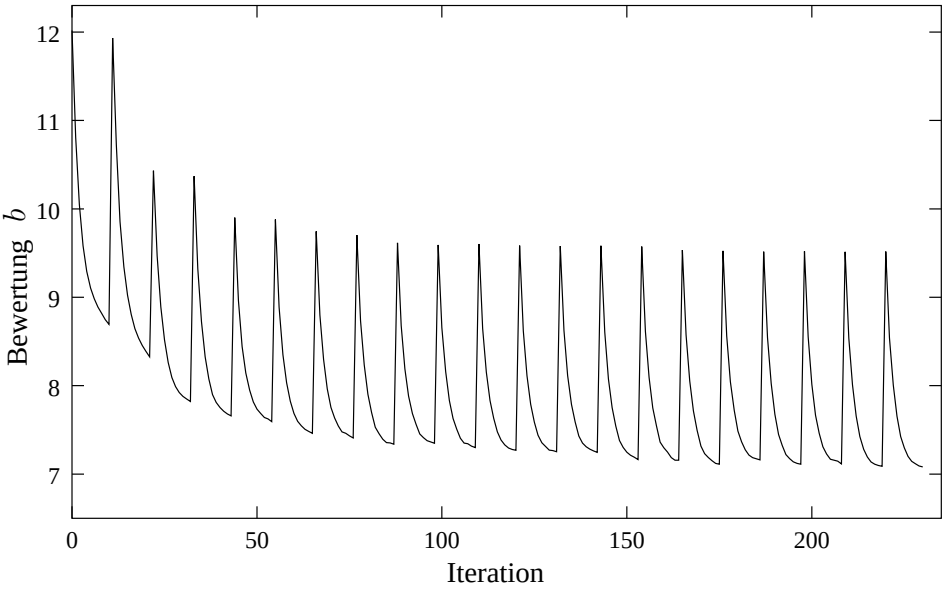


Bild 4.2: Konvergenz der Bewertung b beim Training

6. 10 Iterationen Baum-Welch-Training der HMM-Parameter mit dem `trainer` und Berechnung von b

Trainiert wird nun, bis die Bewertung b auf der Validierungsstichprobe konvergiert. Dazu wird die Schleife mit Punkten 1-6 in den Experimenten 10 mal wiederholt. Den zeitlichen Verlauf der Bewertung b zeigt Abbildung 4.2. Die lokalen Maxima resultieren aus der Neuinitialisierung nach jeweils zehn Baum-Welch-Iterationen. Deutlich zu sehen ist die Konvergenz der relativen Minima. Die Abbildung zeigt das Training des Referenzsystems, mit dem die Experimente aus der folgenden Kapiteln verglichen werden.

Der Wert, gegen den b konvergiert, sagt jedoch noch nichts über die Güte des Erkenners aus. Erst die Wortakkuratheit, die auf der Teststichprobe berechnet wird, gibt darüber Auskunft.

4.3.2 Test mit LRBEAM-Erkenner

Nach dem Training eines Spracherkenners wird dieser mit einer zu den Trainings- und Validierungsdaten disjunkten Teststichprobe getestet. Dies geschieht mit dem Programm `lr_beam`. Zusätzlich zu den im Training berechneten Daten wie dem HMM-Netzwerk *erdial.apn* und dem Kodebuch *erdial.cch* wird nun ein Sprachmodell verwendet. Ausgegeben werden u.a. die erkannten Sätze in Textform. Diese vergleicht das Skript `Evaluate.pl` mit der Verschriftung der Teststichprobe und berechnet daraus die Wortakkuratheit.

	θ_{test}	WA in %
Validierungsstichprobe:	0.00000001	75.6
	0.00001	75.7
	0.0005	75.8
	0.01	76.1
	0.1	78.3
	0.2	78.7
	0.5	78.0
	0.7	78.1
	0.9	77.4
Teststichprobe	0.0005	73.5
	0.2	**74.4**

Tabelle 4.1: Variation von θ_{test} und resultierende Wortakkuratheit beim Referenzsystem mit 256 Klassen

Das Ergebnis des Referenzsystems, dessen Training Abbildung 4.2 zeigt, beträgt 73.5% Wortakkuratheit bzw. 79.6% Wortkorrektheit. Genauere Untersuchungen des Referenzsystems findet man im nächsten Abschnitt.

4.4 Untersuchung des Referenzsystems

Im folgenden wird das Referenzsystem untersucht und geprüft, inwiefern sich einzelne Parametereinstellungen auf die Erkennungsraten auswirken. Dazu werden verschiedene Erkenner trainiert, die auf semikontinuierlichen HMM mit nur einem Kodebuch basieren.

Die Parameter, die untersucht werden, sind die Anzahl der Kodebuchklassen, sowie der Schwellwert θ (vgl. Abschnitt 2.3.4). Bei der Vektorquantisierung werden nur diejenigen Klassen betrachtet, deren Bewertung um höchstens $-2 \cdot ln\theta$ von der bestbewerteten Klasse abweicht. Bei großem θ werden also wegen der Logarithmierung weniger Klassen berücksichtigt. Man unterscheidet zwischen θ_{train} in der Trainingsphase und θ_{test} in der Testphase. Die Auswirkung von θ auf die Wortakkuratheit des Erkenners wird deshalb untersucht, da in Experimenten mit verschiedenen Kodebüchern wohl verschiedene Schwellwerte verwendet werden müssen. Die durchschnittliche Anzahl von berücksichtigten Klassen soll eine sinnvolle Größe haben.

Üblicherweise werden 12 statische Merkmale und 12 dynamische Merkmale berechnet und ein Kodebuch aus 256 24-dimensionalen Normalverteilungen geschätzt. Der Schwellwert beim Training beträgt $\theta_{train} = 0.01$ und der beim Testen $\theta_{test} = 0.0005$. Eine Veränderung von θ_{train} wirkt sich kaum erkennbar auf die Wortakkuratheit aus, wohl aber das Variieren von θ_{test}. Dazu

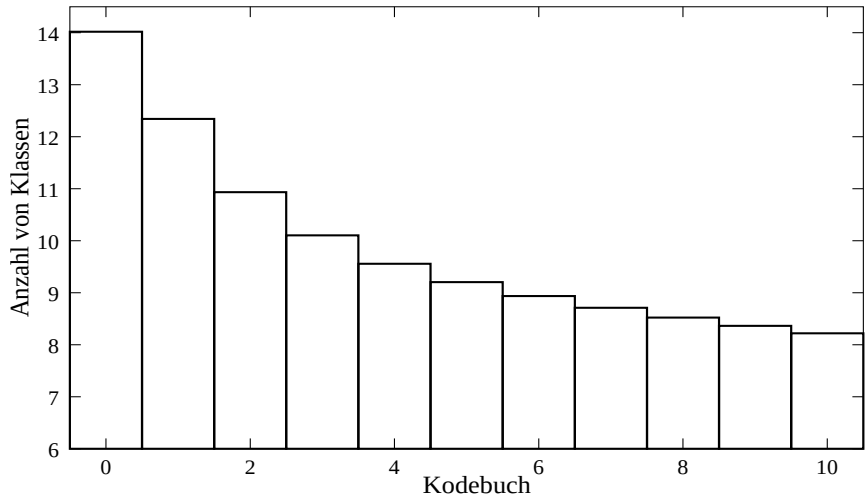


Bild 4.3: Durchschnittliche Anzahl von Klassen mit Bewertung unter einer Schwelle mit initialem Kodebuch (0) und den zehn Neuschätzungen im Referenzsystem

werden nach der Trainingsphase mit dem `lr_beam`-Erkenner auf der Validierungsstichprobe verschiedenen Werten von θ_{test} getestet (siehe Tabelle 4.1). Als optimaler Wert wird $\theta_{test} = 0.2$ gefunden. Mit dieser Einstellung wird auf der Teststichprobe eine Wortakkuratheit von 74.41 % erzielt, ein Prozentpunkt mehr als mit der Standardeinstellung $\theta_{test} = 0.0005$. Da bei größerem θ weniger Klassen berücksichtigt werden, gewinnen wir durch die Wahl von $\theta_{test} = 0.2$ zudem einen zeitlichen Vorteil. Auf der Validierungsstichprobe erhalten wir einen Echtzeitfaktor von $EFF = 1.1$ für $\theta_{test} = 0.0005$ sowie $EZF = 0.9$ für $\theta_{test} = 0.2$.

Der Erkenner mit diesen Einstellungen dient von nun ab in dieser Arbeit als Referenzsystem. Ziel aller weiteren Experimente ist es, eine Wortakkuratheit von mehr als 74.4 % zu erzielen. Dieses Referenzsystem ist, da in der Zwischenzeit einige weitere Optimierungen des Erkenners eingeführt wurden, besser als jenes in der Studienarbeit [Hac01]. Der Schwellwert beim Training $\theta_{train} = 0.01$ ist so gewählt, dass im Durchschnitt 14 Klassen gleichzeitig betrachtet werden. Bei einem verbesserten neugeschätzten Kodebuch werden schließlich bei gleichem θ_{train} nur noch 8 Klassen betrachtet (vgl. Abbildung 4.3).

Nun wird ein Erkenner mit 512 Klassen trainiert, um zu untersuchen, ob das alleinige Erhöhen der Klassenzahl bei HMM mit einem Kodebuch zu Verbesserungen führt. Wieder wird auf der Validierungsstichprobe ein optimaler Schwellwert θ_{test} gesucht (siehe Tabelle 4.2). Dieser beträgt auch hier 0.2. Die erzielte Wortakkuratheit auf der Teststichprobe ist mit $74.5\ \%$ nicht verbessert worden. Der deutliche Mehraufwand beim Training lohnt sich also nicht.

Von Interesse für die nachfolgenden Untersuchungen ist schließlich die Erkennungsleistung, wenn man entweder nur die 12 statischen oder nur die 12 dynamischen Merkmale berücksich-

	θ_{test}	WA in %
Validierungsstichprobe:	0.00000001	77.4
	0.00001	77.5
	0.0005	77.4
	0.01	76.7
	0.1	78.0
	0.2	78.2
	0.5	77.9
	0.7	77.2
	0.9	76.0
Teststichprobe	0.2	**74.5**

Tabelle 4.2: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 512 Klassen

	θ_{test}	WA in %
Validierungsstichprobe:	0.00000001	67.2
	0.0005	67.2
	0.01	67.3
	0.1	67.7
	0.2	67.4
	0.5	67.6
	0.7	66.5
Teststichprobe	0.1	**63.7**

Tabelle 4.3: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 256 Klassen, der nur die 12 *statischen* Merkmale verwendet

tigt. Dazu werden 2 Erkenner trainiert, deren Kodebücher aus jeweils 256 12-dimensionalen Normalverteilungsdichten modelliert sind. Ein optimales θ_{test} liegt jeweils bei 0.1. Im Experiment mit dynamischen Merkmalen wurde ein geringfügig besseres bei 0.7 gefunden. Die Wortakkuratheit auf der Teststichprobe ist erwartungsgemäß niedriger als bei den Experimenten mit 24-dimensionalen Merkmalvektoren. Sie liegt für die statischen Merkmale bei 63.7 % und für die dynamischen Merkmale bei 65.6 % (siehe Tabellen 4.3 und 4.3). Eigentlich hätte man erwartet, dass die statischen Merkmale besser abschneiden (vgl. [ST95, S.324f], was bei der herangezogenen Stichprobe aber anscheinend nicht zutrifft. Sprache in Telefonqualität ist verrauschter; die dynamischen Merkmale sind gegen Störeinflüsse weniger anfällig. Ferner ist dies das einzige Experiment, in dem die Ergebnisse auf der Teststichprobe besser sind, als die auf der Validierungsstichprobe. Freilich ist aber die Teststichprobe aufgrund ihrer Größe aussagekräftiger.

Bei gleichem $\theta_{train} = 0.01$ wie oben werden bei diesen Experimenten nach der zehnten Neuschätzung des Kodebuches immer noch 20 alternative Klassen betrachtet; beim initialen Ko-

	θ_{test}	WA in %
Validierungsstichprobe:	0.00000001	63.9
	0.0005	64.1
	0.01	64.3
	0.1	65.3
	0.2	65.0
	0.5	64.9
	0.7	65.5
	0.9	64.5
Teststichprobe	0.1	65.6
	0.7	**65.6**

Tabelle 4.4: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 256 Klassen, der nur die 12 *dynamischen* Merkmale verwendet

debuch sind es weitaus mehr. Grund dafür ist wohl, dass Merkmale eines bestimmten Typs weniger scharf trennbar sind, als nach Hinzunahme weiterer anderer Merkmaltypen. Ein größeres θ_{train} kann das Training hier sicherlich beschleunigen ohne sichtbare Einbrüche in den Erkennungsraten zu verursachen.

Nachdem in diesem Kapitel die Vorgehensweise beim Trainieren und Testen von Spracherkennern mit dem ISADORA-System erläutert wurde und die aktuellen Erkennungsraten untersucht wurden, folgen nun die Kapitel zu den Experimenten mit mehreren Kodebüchern.

Kapitel 5

Kodebücher für verschiedene Merkmaltypen

Eine Teilaufgabe der Diplomarbeit ist es, den Spracherkenner so zu erweitern, dass mit mehreren Kodebüchern für verschiedene Typen von Merkmalvektoren gearbeitet wird. In Abschnitt 3.1 wurde die Idee bereits ausführlich motiviert und ein Literaturüberblick gegeben.

Im folgenden wird zunächst die genaue Vorgehensweise beschrieben. Danach werden in zwei Kapiteln getrennt zuerst Ergebnisse von Experimenten mit zwei Kodebüchern für statische und dynamische Merkmale diskutiert, und anschließend Ansätze mit mehr als zwei Kodebüchern untersucht. Das dritte Kodebuch dient einmal zum Quantisieren der Energiemerkmale, einmal für die zweite Ableitung der statischen Merkmale und zuletzt für Ableitungen in unterschiedlichen Zeitauflösungen. Alle Experimente werden mit der Stichprobe ERDIAL_4999 durchgeführt. Es folgt ein Abschnitt mit Experimenten auf der VERBMOBIL_TINY Stichprobe, in denen die Robustheit gegenüber Hall untersucht wird. Am Ende des Kapitels wird eine Zusammenfassung wichtiger Experimente gegeben.

5.1 Vorgehensweise

In diesem Kapitel sind Experimente mit verschiedenen Kodebüchern für unterschiedliche Merkmaltypen zusammengefasst. Wie bereits in Abschnitt 3.1 diskutiert, ist dieser Ansatz ein Schritt vom semikontinuierlichen HMM hin zum kontinuierlichen HMM. Man erwartet also bessere Erkennungsraten.

Es wird die statistische Unabhängigkeit der Teilvektoren $\boldsymbol{c}_{stat}$ und $\boldsymbol{c}_{dyn}$ des Merkmalvektors $\boldsymbol{c}$ aus statischen und dynamischen Merkmalen angenommen. Die Wahrscheinlichkeit $b_j(\boldsymbol{c})$, dass

$\boldsymbol{c}$ im HMM-Zustand j ausgegeben wird, errechnet sich als Produkt der Wahrscheinlichkeiten, dass der statische und der dynamische Teilvektor getrennt ausgegeben werden.

$$b_j(\boldsymbol{c}) = b_j(\boldsymbol{c}_{stat}, \boldsymbol{c}_{dyn}) = b_j(\boldsymbol{c}_{stat}) \cdot b_j(\boldsymbol{c}_{dyn}) \tag{5.1}$$

Werden die einzelnen Kodebücher mit Kodebuchexponenten α_{stat} und α_{dyn} gewichtet, so ergibt sich aus den Gleichungen 3.2 und 3.7 als Ausgabewahrscheinlichkeit des Vektors $\boldsymbol{c}$

$$b_j(\boldsymbol{c}) = \left(\sum_{k_{stat}=1}^{K_1} c_{jk_{stat}} \cdot P(\boldsymbol{c}_{stat}|k_{stat}) \right)^{\alpha_{stat}} \cdot \left(\sum_{k_{dyn}=K_1+1}^{K_2} c_{jk_{dyn}} \cdot P(\boldsymbol{c}_{dyn}|k_{dyn}) \right)^{\alpha_{dyn}} \tag{5.2}$$

Dabei seien die Klassen im Kodebuch für die statischen Merkmale mit $k_{stat} \in \{1, 2, ..., K_1\}$ und die Klassen im Kodebuch für dynamische Merkmale mit $k_{dyn} \in \{K_1 + 1, K + 2, ..., K_2\}$ bezeichnet. Ein statischer Merkmalvektor wird von der Kodebuchklasse k_{stat} mit der Dichte $P(\boldsymbol{c}_{stat}|k_{stat})$ erzeugt, die dazugehörigen Ableitungen von einer Klasse k_{dyn} mit $P(\boldsymbol{c}_{dyn}|k_{dyn})$. Der Ansatz lässt sich unschwer für N Kodebücher verallgemeinern:

$$b_j(\boldsymbol{c}) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K_n} c_{jk} P(\boldsymbol{c}_{Teil\ n}|k) \right)^{\alpha_n} = \prod_{n=1}^{N} P(\boldsymbol{c}_{Teil\ n}|j)^{\alpha_n} \tag{5.3}$$

Das Vorgehen mit Kodebuchexponenten ist rein heuristisch. Werden alle Kodebuchexponenten auf Eins gesetzt, so ergibt sich der mathematisch korrekte ungewichtete Fall. Weist man einem Exponenten den Wert Null zu, so wird der betroffene Vektorteil ganz vernachlässigt. Der Faktor im Produkt wird dann konstant Eins und das Kodebuch dadurch “abgeschaltet”. In [Rog94] werden die Einschränkung $0 \leq \alpha_n \leq 1$ und $\sum_n \alpha_n = 1$ gefordert, da es sich sonst bei $b_j(\boldsymbol{c})$ nicht um Wahrscheinlichkeiten handle. Dieses Auffassung wird in der vorliegenden Arbeit jedoch als nicht relevant erachtet. Das Vorgehen, die Dichtefunktionen zu verzerren, ist ohnehin rein heuristisch und nicht mathematisch fundiert. Durch Exponenten kleiner Eins werden die Wahrscheinlichkeitswerte angehoben und zwar umso stärker, je kleiner die α_i werden. Durch Kodebuchexponenten größer eins werden die Wahrscheinlichkeitswerte vermindert. Beispiele der Funktionen $P(\boldsymbol{c}_{Teil\ n}|j)^{\alpha}$ sind in Abbildung 5.1 gezeichnet.

Es ist zu beachten, dass beim Produkt mehrerer Wahrscheinlichkeiten dasjenige Kodebuch, das für das unwichtigste gehalten wird, mit dem kleinsten Exponenten α gewichtet wird. Beim Kodebuchexponenten $\alpha = 0.3$ beispielsweise werden die stark unterschiedlichen Wahrscheinlichkeitswerte 0.4 und 0.8 auf die Werte 0.76 und 0.94 abgebildet und liegen anschließend deut-

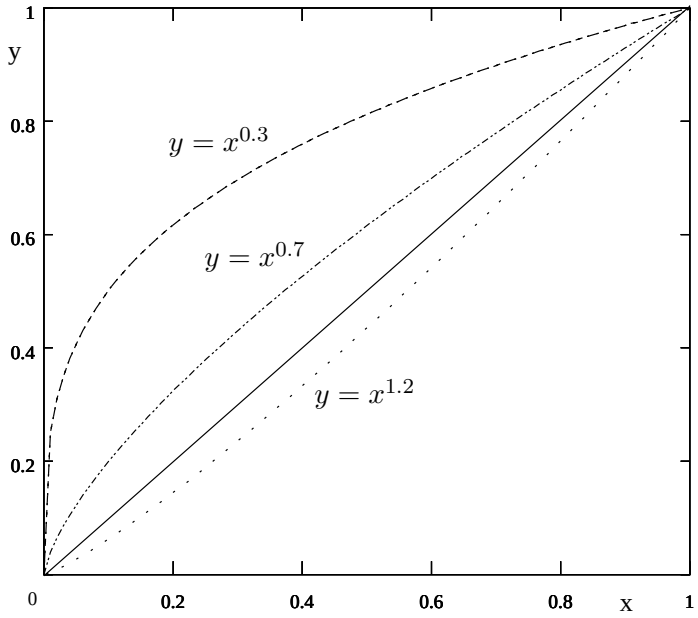


Bild 5.1: Beispiele für die Funktion $y = x^{\alpha}$ für verschiedene Kodebuchexponenten α. Die durchgezogene Linie ist die Identität $y = x$

lich näher zusammen. Die Gesamtwahrscheinlichkeit hängt dann stärker von den anderen Kodebüchern ab. Abbildung 5.2 zeigt die Funktion $x^{0.3} \cdot y^{0.7}$. Sie hängt im Bereich $x > 0.2$ vor allem von y ab. Im Extremfall ist ein Kodebuchexponent gleich Eins. Dann wird dieses Kodebuch für gänzlich unwichtig gehalten; es bleibt unberücksichtigt.

Liegt nur ein einzelnes Kodebuch vor, so resultiert aus einem kleinen Kodebuchexponenten lediglich eine Anhebung aller Wahrscheinlichkeitswerte. Da in der Dekodierungsphase der Spracherkennung die Suche nach der optimalen Wortfolge durch eine Strahlsuche beschleunigt wird, bedeutet eine Anhebung der Wahrscheinlichkeitswerte eine Vergrößerung der Strahlbreite und eine Verlangsamung des Dekodierungsprozesses. Da die Wahrscheinlichkeiten nichtlinear verzerrt werden, bedeutet eine auf diesem Wege erzwungene höhere Strahlbreite jedoch nicht zwingend eine Verbesserung der Erkennungsraten.

Wir betrachten nun wieder HMM mit mehreren Kodebüchern. Werden alle Kodebuchexponenten so verändert, dass das Verhältnis untereinander gleich bleibt, entspricht auch dies wohl nur einer Veränderung der Strahlbreite. Da zudem sehr große oder sehr kleine Exponenten α_n die Wahrscheinlichkeiten sehr verzerren und folglich weitaus mehr als eine reine Gewichtung der Kodebücher verursachen, bleibt der Suchraum nach optimalen α_n beschränkt. Die Experimente in den folgenden Kapiteln zeigen, dass Werte $0 < \alpha_n < 1$ sinnvoll sind. In Untersuchungen mit vielen Kodebüchern potenzieren sich allerdings die kombinatorischen Möglichkeiten, so

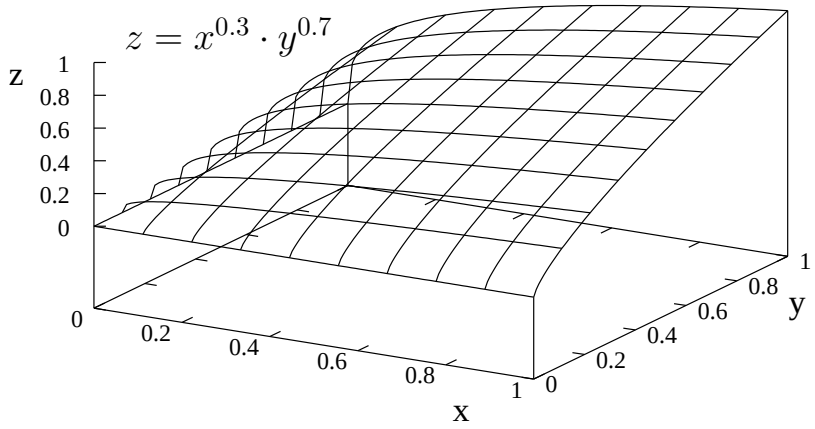


Bild 5.2: Die Funktion $x^{0.3} \cdot y^{0.7}$

dass eine exhaustive Gittersuche bald nicht mehr möglich ist. In dieser Arbeit wird also nach suboptimalen Gewichten α_n gesucht, um zu zeigen, dass verschiedene Ansätze mit mehreren gewichteten Kodebüchern Verbesserungen in der Erkennungsrate mit sich bringen. Die einzelnen Untersuchungen sind in den nachfolgenden Abschnitten beschrieben.

5.2 Experimente mit zwei Kodebüchern

In diesem Abschnitt werden Experimente mit zwei Kodebüchern für statische und dynamische Merkmale durchgeführt. In ersten Untersuchungen werden die Kodebuchexponenten für die Testphase optimiert. Anschließend wird für das beste Ergebnis ein optimaler Schwellwert zur Vektorquantisierung gesucht. Im Abschnitt danach werden Erkenner mit verschiedener Anzahl an Kodebuchklassen getestet. In den letzten Unterabschnitten werden die Kodebücher bereits in der Trainingsphase gewichtet. Optimale Ergebnisse auf der Validierungsstichprobe sind in den Tabellen der Übersichtlichkeit halber *kursiv*, optimale Ergebnisse auf der Teststichprobe **fett** gedruckt.

5.2.1 Training ohne Gewichtung der Kodebücher

In ersten Untersuchungen werden Experimente mit zwei Kodebüchern für 12 statische Merkmale (Energie und 11 Mel-Cepstrum Koeffizienten) und 12 dynamische Merkmale (Ableitungen der statischen) durchgeführt. Jedes Kodebuch hat 128 Klassen. Insgesamt werden also 256 12-dimensionale Normalverteilungen gelernt. Die Kodebuchexponenten α_{stat} und α_{dyn} werden beide in der Trainingsphase auf Eins gesetzt. Der Schwellwert zur Vektorquantisierung aus Ab-

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.70
α_{dyn}	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30
WA in %	75.6	76.6	78.0	77.7	***78.5***	77.8	78.1	***78.6***	77.3	78.0	77.6	75.2

Tabelle 5.1: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale. Summe der Kodebuchexponenten ist Eins.

schnitt 2.3.4 wird beim Training wie beim Referenzsystem konstant auf $\theta_{train} = 0.01$ gesetzt; beim Testen wird die Standardeinstellung des Schwellwertes $\theta_{test} = 0.0005$ beibehalten. Nach einem Training mit 10 Kodebuchneuschätzungen wie in Abschnitt 4.3.1 beschrieben, wird der Erkenner getestet. Die Erkennungsrate beträgt 71.3 % WA. Das beste Ergebnis des optimierten Referenzsystem mit 256 Klassen war 74.4 % WA. Mit gleichem $\theta_{test} = 0.0005$ konnten im Referenzsystem immerhin 73.5 % WA erzielt werden. Durch den Ansatz mit mehreren Kodebüchern hat sich die Spracherkennung also zunächst verschlechtert.

Im folgenden werden nun verschiedene Kodebuchexponenten beim Testen ausprobiert. Dabei werden zuerst auf der Validierungsstichprobe optimale Gewichte α_{stat} und α_{dyn} gesucht. Mit diesen optimalen Gewichten soll der Erkenner dann mit der disjunkten Teststichprobe getestet werden. Die Optimierung erfolgt also *nicht* auf der Teststichprobe; diese dient dazu, einen fertig trainierten Erkenner mit festen Kodebuchexponenten zu testen. Sie soll repräsentativ sein, so dass die Erkennungsraten für vergleichbare Äußerungen gleich gut sind,

Tabelle 5.1 zeigt Wortakkuratheiten für verschiedene Gewichte mit der Einschränkung $\alpha_{stat} + \alpha_{dyn} = 1$ auf der Validierungsstichprobe. Optimale Erkennungsraten ergeben sich für das Paar $\alpha_{stat} = 0.3$ und $\alpha_{dyn} = 0.7$ sowie für $\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$. Veranschaulicht sind die erzielten Wortakkuratheiten auch in Abbildung 5.3. Die Ergebnisse auf der Teststichprobe werden weiter unten diskutiert (Tabelle 5.4).

In einem optimalen Fall werden beide Kodebücher fast gleich gewichtet ($\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$). Auch hier gilt allerdings wie bei allen Kombinationen mit guten Ergebnissen $\alpha_{stat} < \alpha_{dyn}$. Das Kodebuch für dynamische Merkmale wird also als wichtiger erachtet. Dies widerspricht den Erwartungen, dass die statischen Merkmale wichtiger sind (experimentell in [ST95, S.324f] gezeigt), spiegelt aber die Ergebnisse aus Abschnitt 4.4 wieder. Dort erreicht ein Erkenner, der nur die statischen Merkmale berücksichtigt 63.7 % WA, ein Erkenner, der nur die Ableitungen berücksichtigt gar 65.6 % WA. Beide Erkenner arbeiten mit 256 Kodebuchklassen.

Nun schalten wir in unserem Erkenner mit zwei Kodebüchern jeweils eines ab, indem wir den Kodebuchexponenten auf Null setzen. Man beachte allerdings, dass die Kodebücher jetzt wieder 128 Klassen besitzen (im Gegensatz zu den Experimenten in Abschnitt 4.4). Werden nur die stati-

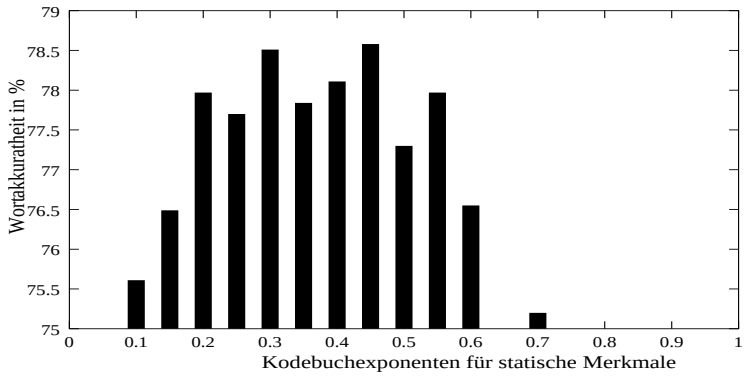


Bild 5.3: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.1

	Experimente mit der					
	Validierungsstichprobe			**Teststichprobe**		
α_{stat}	1.0	0.0	1.0	1.0	0.0	1.0
α_{dyn}	0.0	1.0	1.0	0.0	1.0	1.0
WA in %	69.9	69.7	75.6	66.7	68.7	71.3

Tabelle 5.2: Testen des Erkenners ohne Kodebuchexponenten einmal mit der Validierungsstichprobe und einmal mit der Teststichprobe: Erst wird das zweite Kodebuch ausgeschaltet, dann das erste und zuletzt werden beide Kodebücher ungewichtet betrachtet.

schen Merkmale betrachtet, so wird eine Erkennungsrate von 66.7 % WA auf der Teststichprobe erzielt. Im anderen Fall beträgt die Wortakkuratheit 68.7 % WA. Offenbar sind also für diese spezielle Anwendung mit Daten in Telefonqualität tatsächlich die dynamischen Merkmale aussagekräftiger. Werden die HMM-Parameter unter Berücksichtigung beider Kodebücher trainiert, werden insgesamt höhere Erkennungsraten erzielt, als in den Beispielen in Abschnitt 4.4; dort wurden jeweils ausschließlich 12 Merkmale betrachtet. Für einen guten schnellen Erkenner, der nur mit 12 Merkmale arbeitet lohnt sich also ein aufwändigeres Training mit zwei Kodebüchern. Die Ergebnisse auf der Validierungsstichprobe und auf der Teststichprobe mit “ausgeschalteten” Kodebuch sind in Tabelle 5.2 gegenübergestellt. Dort findet man auch den Versuch mit zwei ungewichteten Kodebüchern. Schon auf der Validierungsstichprobe erkennt man Vorteile der gewichteten Kodebücher, wenn man Tabelle 5.2 mit Tabelle 5.1 vergleicht.

Wir lassen nun ein Kodebuch abgeschaltet und gewichten das andere mit verschiedenen Kodebuchexponenten. Dabei hoffen wir separat optimale Gewichte zu finden, die dann vielleicht auch beim Verwenden beider Kodebücher gute Wortakkuratheiten liefern. Die Ergebnisse der Untersuchungen sind für die statischen Merkmale in Abbildung 5.4 und für die dynamischen

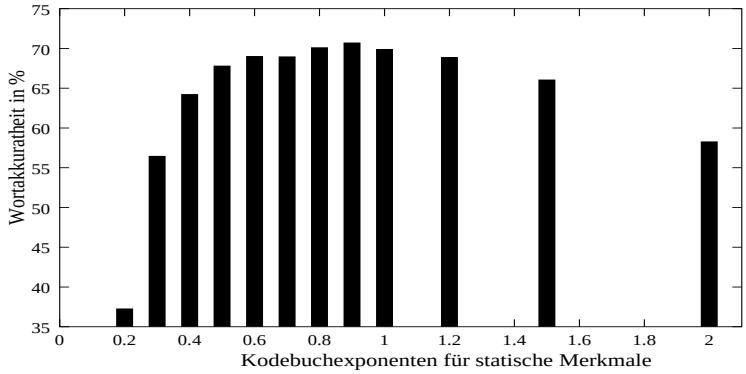


Bild 5.4: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale α_{stat}. Das andere Kodebuch ist abgeschaltet ($\alpha_{dyn} = 0$).

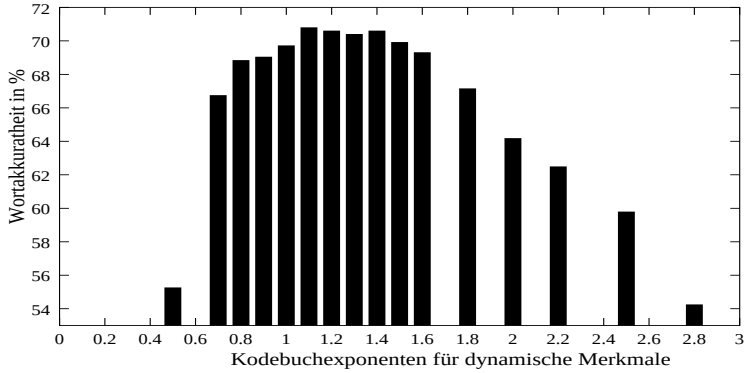


Bild 5.5: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für dynamische Merkmale α_{dyn}. Das andere Kodebuch ist abgeschaltet ($\alpha_{stat} = 0$).

in Abbildung 5.5 veranschaulicht. Es zeigt sich jedoch, wie bereits in Abschnitt 5.1 beschrieben, dass die Exponenten eines einzelnen Kodebuchs nicht als Gewichte zu interpretieren sind, sondern lediglich die Strahlbreite der Dekodierungsphase verändern. Zudem verzerren sie die Wahrscheinlichkeitswerte. Mit dem nichtverzerrenden Exponenten Eins werden also recht gute Erkennungsraten erzielt. Da die Gewichten $\alpha_{stat} = \alpha_{dyn} = 1$ auf der Validierungsstichprobe allerdings schlecht sind und auf der Teststichprobe nur 71.3 % WA erzeugen (Tabelle 5.2), ist die separate Suche nach optimalen Gewichten gescheitert.

Es wird nun die Suche nach einem optimalen Paar von Kodebuchexponenten auf der Validierungsstichprobe fortgesetzt, jedoch nicht exhaustiv. Nachdem suboptimale Paare α_{stat} und α_{dyn} mit der Nebenbedingung $\alpha_{stat} + \alpha_{dyn} = 1$ in Tabelle 5.1 gefunden wurden, testen wir jetzt einige Paare, die jene Nebenbedingung verletzen. Die Ergebnisse der Untersuchungen sind in Tabelle 5.3 zusammengefasst. Gehen man beispielsweise vom Paar $\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$ aus

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.35	0.40	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.50	0.55
α_{dyn}	0.90	0.90	0.65	0.70	0.80	0.85	0.90	0.95	1.00	1.10	0.90	0.90
WA in %	79.1	80.1	79.5	79.8	79.8	80.1	***80.3***	79.9	79.5	80.0	***80.5***	80.2

Tabelle 5.3: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale. Die Summe der Kodebuchexponenten ist beliebig.

	Experimente mit der **Teststichprobe**		
α_{stat}	0.30	0.45	0.50
α_{dyn}	0.70	0.55	0.90
WA in %	**76.5**	75.7	75.8

Tabelle 5.4: Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal (vgl Tabellen 5.3 und 5.1). Es wurden zwei Kodebücher mit je 128 Klassen trainiert.

und erhöht α_{dyn} so findet man ein besseres Optimum bei $\alpha_{dyn} = 0.9$. Nun variieren wir α_{stat} und finden ein Optimum $\alpha_{stat} = 0.5$ und $\alpha_{dyn} = 0.9$.

Nachdem wir nun drei verschiedene optimale Paare von Kodebuchexponenten gefunden haben, testen wir diese auf der Teststichprobe. Die Ergebnisse stehen in Tabelle 5.4. Es wird eine optimale Wortakkuratheit von 76.5 % erzielt. Die beste Erkennungsrate des optimierten Referenzsystems mit 256 Klassen liegt bei nur 74.4 % WA. Obwohl weniger Information zur Verfügung steht (die Korrelation zwischen statischen und dynamischen Merkmalen wird ignoriert), kann die Wortakkuratheit also um über zwei Prozentpunkte verbessert werden, was einer relativen Reduzierung der Fehlerrate um 8 % entspricht.

Würden die optimalen Kodebuchexponenten auf einer größeren Stichprobe gesucht, erhielten wir unter Umständen geringfügig andere Paare von Gewichten, die dem tatsächlichen Optimum näher kommen. Vielleicht ließen sich die Erkennungsraten weiter steigern. Einige weitere Experimente mit schlechteren Ergebnissen auf der Teststichprobe deuten jedoch darauf hin, dass das gefundene Optimum wohl im Bereich des tatsächlichen Optimum liegt. Ohne Einschränkung können wir feststellen, dass der Multi-Kodebuch-Ansatz deutliche Verbesserungen der Erkennungsraten mit sich bringt.

Es fehlt nun noch, den neuen besseren Erkenner so zu optimieren, wie es für das Referenzsystem in Kapitel 4.4 getan wurde. Dort wurde der Schwellwert θ_{test} für die Vektorquantisierung auf der Validierungsstichprobe optimiert. Hier betrachten wir nur den einfachen Fall, dass θ_{test} für beide Kodebücher gleich ist, obwohl die Anzahl der Klassen, die unter die Schwelle fallen,

für statische und dynamische Merkmale sehr unterschiedlich ist. Beim Quantisieren mit dem initialen Kodebuch werden 42 statische und 25 dynamische Klassen ausgewählt, nach zehn Kodebuchneuschätzungen sind es noch 10 statische und 16 dynamische Klassen. Auch zuletzt ist also die Gesamtzahl der Klassen (26) noch sehr hoch im Vergleich zum Referenzsystem (8) (vgl. Abbildung 4.3). Dadurch wird wohl ein Großteil der Rechenzeit, die im Multi-Kodebuch-Ansatz eingespart wird, wieder aufgebraucht.

Da weitere tiefere Untersuchungen im Rahmen der vorliegenden Diplomarbeit nicht mehr möglich waren, optimieren wir nur den gemeinsamen Schwellwert $\theta_{test,\ stat} = \theta_{test,\ dyn} = \theta_{test}$. Die Experimente dazu sind in Tabelle 5.5 zusammengestellt. Für den neuen Erkenner mit Kodebuchexponenten $\alpha_{stat} = 0.3$ und $\alpha_{dyn} = 0.7$ ist auch hier ein Schwellwert $\theta_{test} = 0.2$ geringfügig besser als der Standardwert $\theta_{test} = 0.0005$. Mit dieser veränderten Einstellung wird auf der Teststichprobe eine **Wortakkuratheit von 76.9 %** erreicht. Der Echtzeitfaktor kann von $EZF = 1.6$ auf $EZF = 1.0$ verbessert werden. Durch die Gewichtungen (Multiplikationen im Exponenten der Normalverteilungen) werden wir uns im Vergleich mit dem Referenzsystem langsamer. Optimierungen sind hier noch denkbar. Gegenüber dem Referenzsystem verkleinert sich aber die Wortfehlerrate von 25.6 % auf 23.1 %. Dies entspricht einer relativen Reduzierung um fast 10 %. Obwohl die Korrelation der statischen und dynamischen Merkmale verloren geht und die Anzahl der zu schätzenden Kodebuchparameter von etwa 83000 im Referenzsystem auf nun etwa 23000 verringert wurde (berücksichtigt werden Mittelwertvektoren und obere Dreiecksmatrix der symmetrischen Kovarianzmatrix für alle Klassen und alle Kodebücher), erzielt man deutliche Verbesserungen. Im nächsten Abschnitt erfolgen weitere Optimierungen durch Variation der Klassenzahl der Kodebücher.

5.2.2 Variation der Klassenzahl für die Kodebücher

Wie schon im letzten Abschnitt werden auch hier Untersuchungen mit zwei Kodebüchern für 12 statische Merkmale (Energie und 11 Mel-Cepstrum Koeffizienten) und 12 dynamische Merkmale (Ableitungen der statischen) durchgeführt. Die Kodebuchexponenten α_{stat} und α_{dyn} werden in der Trainingsphase wieder beide auf Eins gesetzt. Als Schwellwerte zur Vektorquantisierung aus Abschnitt 2.3.4 werden die Standardwerte $\theta_{train} = 0.01$ und $\theta_{test} = 0.0005$ gewählt.

Variiert wird in diesem Abschnitt die Anzahl der Kodebuchklassen. Der Einfachheit halber haben die Kodebücher für statische und dynamische Merkmale aber stets genauso viele Klassen. Nachdem im letzten Abschnitt zwei Kodebücher mit je 128 Klassen trainiert wurden, untersuchen wir nun zum einen ob eine Erhöhung der Klassenzahl eine Verbesserung der Wortakkuratheit mit sich bringt. Im Vergleich zum Referenzsystem, das mit Kodebüchern aus 256 24-

	θ_{test}	WA in %
Validierungsstichprobe:	0.00001	77.6
	0.0005	78.5
	0.01	78.0
	0.1	78.4
	0.2	78.7
	0.3	78.3
	0.5	78.2
	0.7	77.1
	0.9	76.6
Teststichprobe	0.0005	76.5
	0.2	**76.9**

Tabelle 5.5: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{stat} = 0.3$ und $\alpha_{dyn} = 0.7$.

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.70
α_{dyn}	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30
WA in %	75.0	74.9	75.7	77.0	77.3	***78.0***	77.8	77.7	77.1	76.1	74.3	74.1

Tabelle 5.6: Testen des Erkenners mit 2×100 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$.

dimensionalen Dichten arbeitet, ist der Informationsgehalt von insgesamt 128 12-dimensionalen Dichten sehr niedrig. Zum anderen zeigt aber Abbildung 3.1, dass durch zwei Kodebücher mit je 128 Dichten im Grunde ein 128×128-dimensionales Kodebuch modelliert werden kann. Diese Modellierungsfreiheit wird jedoch dadurch stark eingeschränkt, dass jeder HMM-Zustand nur insgesamt 256 Faktoren zur Gewichtung der Kodebuchkomponenten bereitstellt. Dennoch wird untersucht, ob vielleicht weniger als 128×128 Kodebuchklassen genügen.

Zuerst wird ein Erkenner mit 2×100 Klassen trainiert. Auf der Trainingsstichprobe werden die optimalen Kodebuchexponenten α_{stat} und α_{dyn} gesucht. In diesem Abschnitt betrachten wir nur den Spezialfall $\alpha_{stat} + \alpha_{dyn} = 1$. Die Ergebnisse sind in Tabelle 5.6 zusammengefasst und in Abbildung 5.6 verdeutlicht. Das optimale Paar ist $\alpha_{stat} = 0.35$ und $\alpha_{dyn} = 0.65$.

Anschließend werden Untersuchungen mit 2×200 Klassen unternommen. Tabelle 5.7 und Abbildung 5.7 zeigen das optimale Paar $\alpha_{stat} = 0.2$ und $\alpha_{dyn} = 0.8$. Ein letzter Erkenner wird mit 2×256 Klassen trainiert. Nun sind die optimalen Kodebuchexponenten $\alpha_{stat} = 0.15$ und $\alpha_{dyn} = 0.85$ (Tabelle 5.7 und Abbildung 5.7).

Mit den optimalen Paaren von Kodebuchexponenten, die auf der Validierungsstichprobe er-

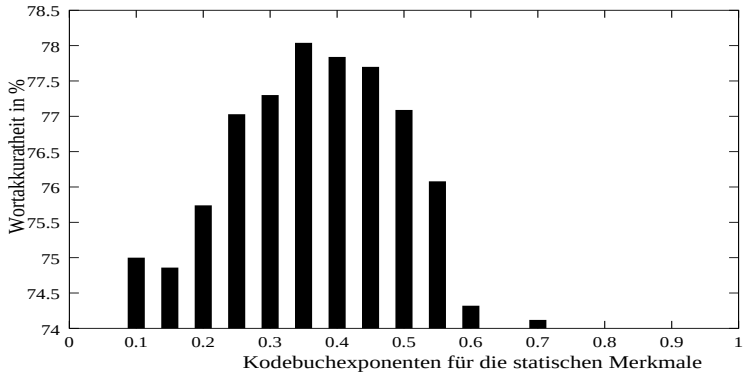


Bild 5.6: Testen des Erkenners mit 2×100 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.6

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.70
α_{dyn}	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30
WA in %	77.4	78.9	***79.5***	78.7	79.1	78.9	78.4	77.9	77.8	77.5	78.0	76.0

Tabelle 5.7: Testen des Erkenners mit 2×200 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$.

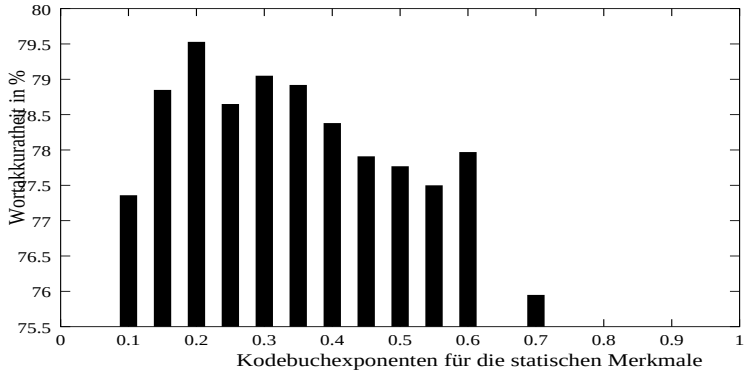


Bild 5.7: Testen des Erkenners mit 2×200 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.7

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.70
α_{dyn}	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30
WA in %	78.1	***79.3***	78.3	79.1	78.9	79.0	78.1	78.6	79.0	77.8	77.6	76.4

Tabelle 5.8: Testen des Erkenners mit 2×256 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$.

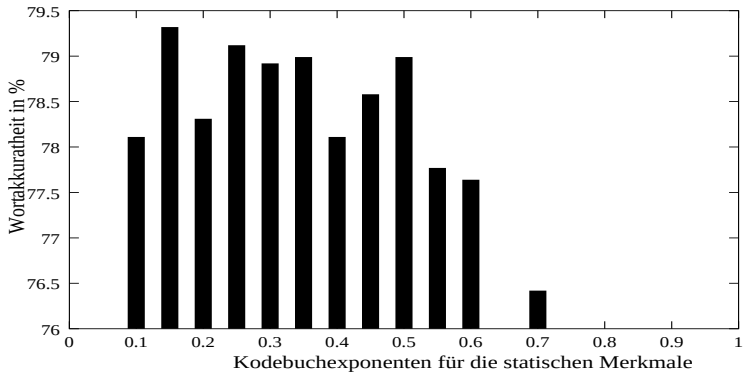


Bild 5.8: Testen des Erkenners mit 2×256 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.8

	Experimente mit der **Teststichprobe**							
# Klassen	2×100	2×128			2×200		2×256	
α_{stat}	0.35	0.30	0.45	0.50	0.20	0.30	0.15	0.25
α_{dyn}	0.65	0.70	0.55	0.90	0.80	0.70	0.85	0.75
WA in %	**75.8**	**76.5**	75.7	75.8	77.1	**77.3**	76.6	**77.4**

Tabelle 5.9: Testen der Erkenner mit mit verschieden vielen Klassen auf der Teststichprobe. Die Kodebuchexponenten sind jeweils auf der Validierungsstichprobe optimal (vgl. Tabellen 5.6, 5.7 und 5.8).

mittelt wurden, werden nun die Erkenner getestet. Die Ergebnisse findet man in Tabelle 5.9. Mit 2×100 Klassen werden 75.8 % WA erzielt, mit den 2×128 Klassen aus dem letzten Kapitel (nicht optimierter Fall mit Quantisierungsschwellwert $\theta_{test} = 0.0005$) 76.5 % WA. Im Versuch mit 2×200 Klassen kann die Erkennungsrate weiter auf 77.3 % WA erhöht werden. Im Experiment mit 2×256 Klassen erreichen wir **77.4 % WA**. Im Gegensatz zum Referenzsystem kann die Wortakkuratheit durch Erhöhen der Klassenzahl hier also erhöht werden. Mit nur 12-dimensionalen Dichtefunktionen ist die Zahl der freien Parameter (berücksichtigt werden Mittelwertvektoren und obere Dreiecksmatrix der symmetrischen Kovarianzmatrix für alle Klassen und alle Kodebücher) hier nämlich deutlich geringer. Im Referenzsystem mit 256 Klassen hat das Kodebuch etwa 83000 freie Parameter, im Ansatz mit 2 Kodebüchern mit je 256 Klassen nur etwa 46000.

In den weiteren Abschnitten betrachten wir wieder Erkenner mit 2×128 Klassen.

	Experimente mit der *Validierungsstichprobe*											**Test**
α_{stat}	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.60	0.70	0.25
α_{dyn}	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.40	0.30	0.75
WA in %	74.7	76.7	78.2	***79.3***	78.7	78.9	77.8	78.9	78.2	77.0	76.9	**76.5**

Tabelle 5.10: Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\ train} = \alpha_{dyn,\ train} = 0.5$ trainiert. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten.

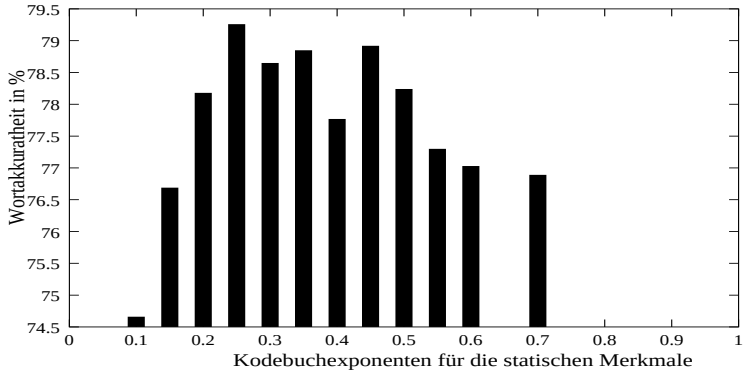


Bild 5.9: Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\ train} = \alpha_{dyn,\ train} = 0.5$ trainiert. Vgl. Tabelle 5.10

5.2.3 Training mit Gewichtung der Kodebücher

In diesem Abschnitt werden wieder Spracherkenner mit 2 Kodebüchern mit je 128 Klassen für die 12 statischen Merkmale und 12 dynamischen Merkmale trainiert. Die Schwellwerte zur Vektorquantisierung aus Abschnitt 2.3.4 werden auf die Standardwerte $\theta_{train} = 0.01$ und $\theta_{test} = 0.0005$ gesetzt. Nun werden in der Trainingsphase Kodebuchexponenten $\alpha_{stat,\ train} = 0.5$ und $\alpha_{dyn,\ train} = 0.5$ verwendet. Beide Kodebücher sind im Training wieder gleich gewichtet, die Kodebuchexponenten summieren sich jedoch zu Eins.

Anschließend werden wieder optimale Gewichte α_{stat} und α_{dyn} für die Testphase gesucht. Dazu wird der Erkenner mit verschiedenen Gewichten auf der Validierungsstichprobe getestet (Tabelle 5.10 und Abbildung 5.9). Die optimalen Gewichte sind $\alpha_{stat} = 0.25$ und $\alpha_{dyn} = 0.75$. Damit wird ein Erkenner getestet und eine Erkennungsrate von **76.5 % WA** auf der **Teststichprobe** erzielt.

Durch die Halbierung beider Gewichte in der Trainingsphase kann also keine Veränderung der Wortakkuratheit verursacht werden. Die aufwändige Gewichtung lohnt sich also nicht. Inter-

essant sind aber *verschiedene* Gewichte für die unterschiedlichen Kodebücher im Training. Da ein experimentelles Suchen nach optimalen Trainingsgewichten zu weit führen würde, wird eine spezielle Vorgehensweise im nächsten Abschnitt erläutert.

5.2.4 Zweites Training mit optimalen Gewichten

In diesem Abschnitt betrachten wir ein letztes Mal den Erkenner mir zwei Kodebüchern für 12 statische und 12 dynamische Merkmale. Jedes der Kodebücher hat 128 Klassen; die Schwellwerte zur Vektorquantisierung werden wieder auf die Standardwerte $\theta_{train} = 0.01$ und $\theta_{test} = 0.0005$ gesetzt. Schon im letzten Abschnitt sind wir der Frage nachgegangen, ob durch geeignete Kodebuchexponenten $\alpha_{stat\ train}$ und $\alpha_{dyn\ train}$ *in der Trainingsphase* die Erkennungsraten positiv verändert werden können. Um solche optimale Exponenten zu finden, ist es wohl zu zeitaufwändig, zahlreiche Erkenner zu trainieren.

In folgenden wird untersucht, ob die optimalen Kodebuchexponenten beim Testen des Erkenners mit zwei Kodebüchern aus Abschnitt 5.2.1 sich schon in der Trainingsphase positiv auswirken. In Abschnitt 5.2.1 wurden die optimalen Gewichte $\alpha_{stat} = 0.3$ und $\alpha_{dyn} = 0.7$ beim Testen mit der Validierungsstichprobe gefunden. Auf der Teststichprobe wurden so 76.5 % WA erzielt. Nun wird der Erkenner erneut trainiert, diesmal mit gewichteten Kodebüchern unter Verwendung eben dieser Kodebuchexponenten $\alpha_{stat\ train} = 0.3$ und $\alpha_{dyn\ train} = 0.7$.

Der Erkenner wird anschließend getestet. Dazu werden wieder optimale Kodebuchexponenten für die Testphase mit Hilfe der Validierungsstichprobe gesucht (siehe Tabelle 5.11 und Abbildung 5.10). Als optimales Paare wurde $\alpha_{stat} = 0.5$ und $\alpha_{dyn} = 0.5$ gefunden, sowie $\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$. Nach einem Training mit gewichteten Kodebüchern sind also wohl keine Gewichtungen in der Testphase mehr notwendig. Auf der **Teststichprobe** werden im ersten Fall 77.2 % WA erzielt und im zweiten Fall **77.8 % WA**. Die Wortakkuratheit kann also gegenüber dem Experiment aus Abschnitt 5.2.1 (76.5 % WA) nochmal um etwa einen Prozentpunkt gesteigert werden. Der Vergleich mit dem Referenzsystem (74.4 % WA) zeigt eine deutliche Verbesserung, was einer relativen Verkleinerung der Fehlerrate um 12 % entspricht.

Eine Optimierung des Schwellwertes für die Vektorquantisierung θ_{test} zeigt Tabelle 5.12. Die Erkennungsrate kann nicht verbessert werden jedoch der Echtzeitfaktor von $EZF = 1.1$ auf $EZF = 0.8$ für ein $\theta_{test} = 0.2$ (statt $\theta_{test} = 0.0005$).

In den folgenden Abschnitten werden Erkenner mit mehr als zwei Kodebüchern untersucht.

	Experimente mit der *Validierungsstichprobe*										**Test**	
α_{stat}	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.70	0.45	0.50
α_{dyn}	0.80	0.75	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30	0.55	0.50
WA in %	78.2	78.2	79.5	79.1	79.5	80.0	***80.2***	78.9	79.2	76.5	**77.8**	77.2

Tabelle 5.11: Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\,train} = 0.3$ und $\alpha_{dyn,\,train} = 0.7$ trainiert. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten.

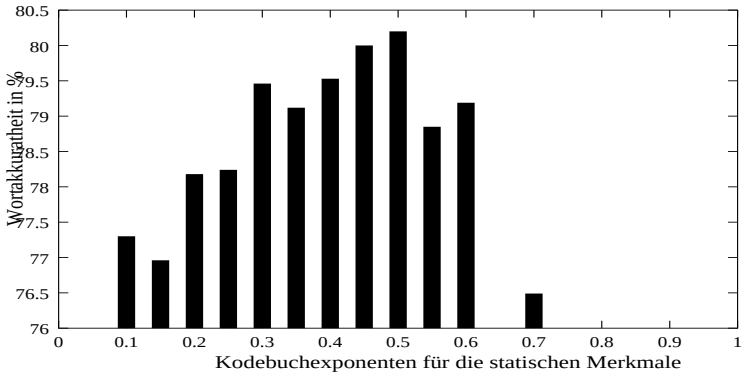


Bild 5.10: Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\,train} = 0.3$ und $\alpha_{dyn,\,train} = 0.7$ trainiert. Vgl. Tabelle 5.11

	θ_{test}	WA in %
Validierungsstichprobe:	0.0005	80.0
	0.005	80.3
	0.01	80.8
	0.05	80.5
	0.1	80.3
	0.2	79.9
	0.3	80.5
	0.4	79.2
	0.7	78.6
Teststichprobe	0.0005	**77.8**
	0.01	**77.8**
	0.3	77.5

Tabelle 5.12: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\,train} = 0.3$ und $\alpha_{dyn,\,train} = 0.7$ trainiert.

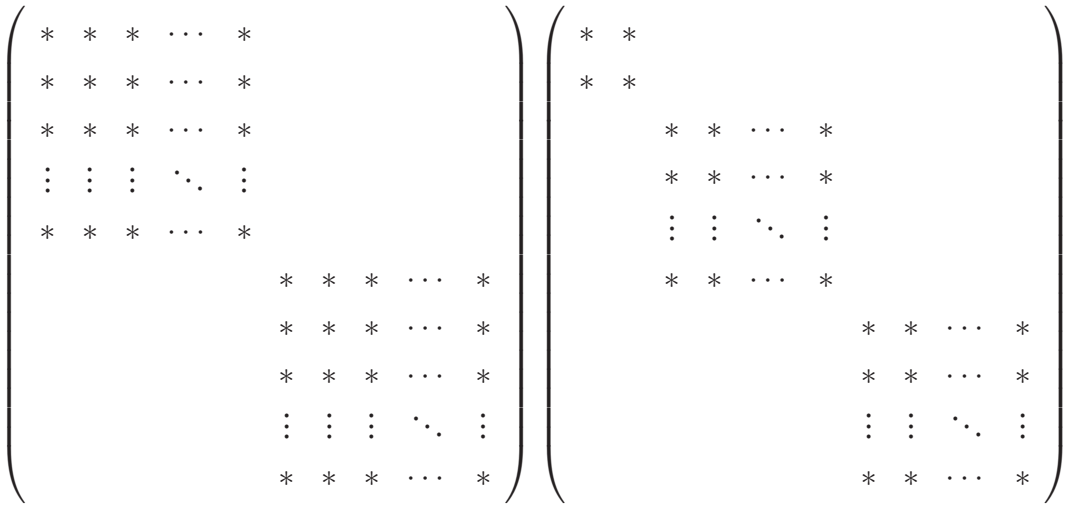
5.3 Experimente mit mehreren Kodebüchern

In den folgenden Abschnitten werden Untersuchungen mit drei bzw, vier Kodebüchern durchgeführt. Zuerst werden drei Kodebücher für Energie, Cepstrum und Ableitungen trainiert. In einem weiteren Experiment wird zu den beiden Kodebüchern für statische und dynamische Merkmale eines für die zweiten Ableitungen hinzugezogen. In den Untersuchungen im letzten Abschnitt werden Ableitungen für verschiedene Zeitauflösungen betrachtet. Optimale Ergebnisse auf der Validierungsstichprobe sind in den Tabellen der Übersichtlichkeit halber *kursiv*, optimale Ergebnisse auf der Teststichprobe **fett** gedruckt.

5.3.1 Separates Kodebuch für Energiemerkmale

In diesem Abschnitt wird ein Erkenner mit drei Kodebüchern untersucht. Das erste Kodebuch dient nun zum Quantisieren der Energie und ihrer Ableitung. Das zweite Kodebuch ist für die 11 Mel-Cepstrum Koeffizienten (MFCC) und das dritte für deren 11 Ableitungen. Erstmals in dieser Arbeit haben die Kodebücher also verschiedene Dimensionen. Im Vergleich mit den beiden Kodebüchern für statische und dynamische Merkmale aus dem letzten Abschnitt geht nun weitere Information verloren: Die Korrelation zwischen der Energie und den anderen 11 statischen Merkmalen sowie die Korrelation zwischen abgeleiteter Energie und den restlichen 11 dynamischen Merkmalen. Die Korrelation zwischen dem Energiemerkmal und seiner Ableitung dagegen bleibt im Modell erhalten. Zwischen den verschiedenen Merkmaltypen wird erneut statistische Unabhängigkeit angenommen (vgl. Gleichung 5.3). Energie und statische Merkmale gelten jetzt auch als unabhängig, Energie und deren Ableitung nicht.

Nachfolgend sind zwei Kovarianzmatrizen von Gaussdichten skizziert. Links werden die 24-dimensionale Merkmalvektoren von zwei Kodebüchern mit 12-dimensionalen Dichten erzeugt (vgl. Abschnitt 5.2). In der insgesamt 24×24-dimensionalen Kovarianzmatrix sind zwei Blöcke der Größe 12×12 besetzt. Rechts ist der Fall mit drei Kodebüchern aus diesem Abschnitt angedeutet. Dazu werden die Komponenten der Merkmalvektoren so umsortiert, dass Komponente Nummer zwei die Ableitung der Energie ist. Es ist ein Block der Größe 2×2 für die beiden Energiemerkmale in der 24-dimensionalen Kovarianzmatrix besetzt sowie zwei 11×11 Blöcke. Da die Kovarianzmatrizen symmetrisch sind, betrachten wir nur die obere Dreiecksmatrix. Es sind 21 Plätze weniger besetzt als im rechten Fall.



Summa summarum wird im 3-Kodebuch-Ansatz die Anzahl der freien Parameter weiter reduziert. Es soll nun untersucht werden, ob sich die Erkennungsraten trotzdem wenigstens nicht verschlechtern. Dazu wird ein Erkenner mit 100 Klassen pro Kodebuch trainiert. Insgesamt hat das Kodebuch hier 300 Klassen unterschiedlicher Dimension. Die Klassenzahl wird also gegenüber dem Versuch mit zwei Kodebüchern a 128 Klassen (siehe Abschnitt 5.2.1) erhöht. Insgesamt wird jedoch die Zahl der freien Parameter im Kodebuch (berücksichtigt werden Mittelwertvektoren und obere Dreiecksmatrix der symmetrischen Kovarianzmatrix für alle Klassen und alle Kodebücher) auf etwa 16000 reduziert. Im Referenzsystem beträgt diese Anzahl 83000 im Ansatz mit zwei Kodebüchern 23000. Der Schwellwert zur Vektorquantisierung behält im Training den Standardwert $\theta_{train} = 0.01$ und beim Test zunächst den Wert $\theta_{test} = 0.0005$.

Es müssen nun drei Kodebuchexponenten α_{en}, α_{ceps} und α_{abl} optimiert werden. Beim Training werden alle drei auf Eins gesetzt. Natürlich können die Gewichte in einem anschließenden weiteren Training — wie in Abschnitt 5.2.4 für zwei Kodebücher durchgeführt — mit den optimalen Werten besetzt werden. So können vielleicht noch bessere Erkennungsraten erzielt werden, aus Zeitgründen wird jedoch ein zweites Training mit diesem Experiment im Rahmen dieser Arbeit nicht durchgeführt.

Nach dem Training suchen wir nun wieder auf der Validierungsstichprobe optimale Kodebuchexponenten α_{en}, α_{ceps} und α_{abl}. Dazu betrachten wir zunächst die Kodebücher einzeln. In den drei Experimenten aus Tabelle 5.13 werden je zwei Kodebücher ausgeschaltet, indem die jeweiligen Kodebuchexponenten auf Null gesetzt werden. Die Experimente werden zusätzlich auf der Teststichprobe durchgeführt. An den Wortakkuratheiten erkennt man, dass das Kodebuch für die Ableitungen das beste ist, gefolgt von dem für die Mel-Cepstrum-Koeffizienten. Mit dem Kode-

	Experimente mit der					
	Validierungsstichprobe			**Teststichprobe**		
α_{en}	1.0	0.0	0.0	1.0	0.0	0.0
α_{ceps}	0.0	1.0	0.0	0.0	1.0	0.0
α_{abl}	0.0	0.0	1.0	0.0	0.0	1.0
WA in %	37.2	67.9	68.0	30.5	61.6	64.3

Tabelle 5.13: Testen der Kodebücher für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe.

	Experimente mit der *Validierungsstichprobe*											
α_{en}	0.30	0.30	0.30	0.30	0.35	0.35	0.40	0.45	0.45	0.45	0.50	0.50
α_{ceps}	0.30	0.30	0.30	0.35	0.35	0.35	0.35	0.45	0.45	0.45	0.40	0.40
α_{abl}	0.70	0.75	0.80	0.75	0.75	0.80	0.80	0.60	0.70	0.75	0.70	0.75
WA in %	79.8	80.0	79.3	79.9	80.0	80.1	***80.3***	78.7	80.0	78.7	***80.4***	79.2

Tabelle 5.14: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen.

buch für die Energiemerkmale werden die geringsten Erkennungsraten erzielt. Dementsprechend wird wohl der Kodebuchexponent für die Ableitungen am größten sein und der für das Energiekodebuch am geringsten. Durch diese Hypothese wird die aufwändige Suche nach einem Tripel optimaler Kodebuchexponenten im $\mathbb{R}^3$ etwas eingeschränkt. Durch systematisches Ausprobieren finden wir einige gute α_{en}, α_{ceps} und α_{abl}. Eine Auswahl zeigt Tabelle 5.14.

Die gefundenen optimalen Tripel von Gewichten (u.a. $\alpha_{en} = 0.5$, $\alpha_{ceps} = 0.4$, $\alpha_{abl} = 0.7$ bzw. $\alpha_{en} = 0.4$, $\alpha_{ceps} = 0.35$, $\alpha_{abl} = 0.8$) evaluieren wir auf der Teststichprobe. Die Ergebnisse findet man in Tabelle 5.15. Es werden Erkennungsraten bis zu 76.7 % WA für $\alpha_{en} = 0.3$, $\alpha_{ceps} = 0.3$ und $\alpha_{abl} = 0.75$ erzielt. Verglichen mit dem Referenzsystem aus Abschnitt 4.4 (dort werden 74.4 % WA erreicht) verbessert sich die Erkennung um über zwei Prozentpunkte. Gegenüber dem Ansatz mit zwei Kodebüchern aus Abschnitt 5.2.1 (dort werden 76.5 % WA erreicht) erkennt man nur geringfügige nicht signifikante Verbesserungen. Es ist allerdings zu beachten, dass hier die guten Ergebnisse mit einer geringeren Anzahl von freien Kodebuchparametern erzielt wurden. Durch Vergrößern der Kodebücher können die Erkennungsraten bestimmt noch gesteigert werden.

Der Erkenner mit zwei Kodebüchern (Abschnitt 5.2.1) wurde zudem noch bezüglich θ_{test} optimiert. Bei $\theta_{test} = 0.2$ werden dort 76.9 % WA erreicht. Diese Optimierung führen wir nun auch für den Erkenner mit $\alpha_{en} = 0.3$, $\alpha_{ceps} = 0.3$ und $\alpha_{abl} = 0.75$ durch. Bei drei Kodebüchern

	Experimente mit der **Teststichprobe**						
α_{en}	0.30	0.30	0.35	0.35	0.40	0.45	0.50
α_{ceps}	0.30	0.30	0.35	0.35	0.35	0.45	0.40
α_{abl}	0.70	0.75	0.75	0.8	0.80	0.79	0.70
WA in %	76.5	**76.7**	76.5	76.3	76.4	75.3	75.0

Tabelle 5.15: Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden drei Kodebücher für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen mit je 100 Klassen trainiert.

	θ_{test}	WA in %
Validierungsstichprobe:	**0.0005**	80.0
	0.01	80.3
	0.1	80.3
	0.2	79.7
	0.3	77.9
	0.5	78.2
	0.7	78.2
Teststichprobe	0.0005	76.7
	0.1	**77.5**

Tabelle 5.16: Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{en} = 0.3$ und $\alpha_{ceps} = 0.3$ und $\alpha_{abl} = 0.75$.

fallen insgesamt ohnehin viel mehr Klassen pro Merkmalvektor unter den Schwellwert. Es wird ein optimales $\theta_{test} = 0.1$ gefunden (siehe Tabelle 5.16). Auf der Teststichprobe werden mit diesem Schwellwert 77.5 % WA erreicht. Der Echtzeitfaktor verbessert sich von $EZF = 1.7$ auf $EZF = 1.0$. Die Verbesserung der Erkennungsrate fällt hier deutlicher aus als im Experiment mit zwei Kodebüchern.

Auch im nächsten Abschnitt wird ein Erkenner mit drei Kodebüchern trainiert, jedoch für statische Merkmale, Ableitungen und zweite Ableitungen.

5.3.2 Separates Kodebuch für zweite Ableitungen

Erneut wird in diesem Abschnitt ein Erkenner mit drei Kodebüchern trainiert. Hier wird zusätzlich die zweite Ableitung herangezogen. Diese entspricht der Regression der ersten Ableitung in 50 ms großen Zeitfenstern. Gegenüber dem Referenzsystem und allen bisherigen Experimenten in diesem Kapitel wird nun die Dimension der Merkmalvektoren erhöht. Die jetzt 36-dimensionalen Merkmalvektoren bestehen aus 12 statischen Merkmalen (Energie und 11 Mel-Cepstrum Koeffizienten), 12 ersten Ableitungen der statischen Merkmale und 12 zweiten Ab-

	Experimente mit der					
	Validierungsstichprobe			**Teststichprobe**		
α_{stat}	1.0	0.0	0.0	1.0	0.0	0.0
α_{abl1}	0.0	1.0	0.0	0.0	1.0	0.0
α_{abl2}	0.0	0.0	1.0	0.0	0.0	1.0
WA in %	68.4	68.5	57.1	66.9	68.0	53.0

Tabelle 5.17: Testen der Kodebücher für statische Merkmale, Ableitungen und zweite Ableitungen einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe.

leitungen. Es werden drei Kodebücher mit je 100 Klassen für statische Merkmale, deren Ableitungen sowie für die zweiten Ableitungen berechnet. Im Vergleich mit den Untersuchungen aus Abschnitt 5.2 mit zwei Kodebüchern wird nun zusätzliche Information herangezogen. Man hofft, dass der gestiegene Aufwand mit besseren Erkennungsraten belohnt wird. Verglichen werden die Ergebnisse aus diesem Abschnitt mit dem 2-Kodebuch-Experiment aus Abschnitt 5.2.2. Dort werden von einem Erkenner mit 2×100 Klassen 75.8 % WA erzielt.

Die drei Kodebuchexponenten α_{stat}, α_{abl1} und α_{abl2} werden auf der Validierungsstichprobe optimiert. Beim Training werden alle drei auf Eins gesetzt. Ein zweites "schiefes" Training mit den optimalen Werten, wie in Abschnitt 5.2.4 für zwei Kodebücher durchgeführt, wird aus Aufwandsgründen im Rahmen dieser Arbeit nicht durchgeführt. Der Schwellwert bei der Vektorquantisierung behält im Training und Test die Standardwerte $\theta_{train} = 0.01$ und $\theta_{test} = 0.0005$.

Zunächst betrachten wir wieder die Kodebücher einzeln. In den drei Experimenten aus Tabelle 5.17 werden je zwei Kodebücher ausgeschaltet, indem man die jeweiligen Kodebuchexponenten auf Null setzt. Die Wortakkuratheiten zeigen, dass das Kodebuch für die ersten Ableitungen am besten ist. Dasjenige für die zweiten Ableitungen ist dagegen am schlechtesten. Die Kodebuchexponenten für die ersten Ableitungen werden deshalb wohl am größten sein und diejenigen für die zweiten Ableitungen am kleinsten.

Nun suchen wir gute Tripel von Gewichten ($\alpha_{stat}, \alpha_{abl1}, \alpha_{abl2}$). Eine Auswahl der Ergebnisse zeigt Tabelle 5.18. Gute Gewichte werden beispielsweise durch die Tripel (0.5, 0.7, 0.3), (0.5, 0.7, 0.5), (0.55, 0.65, 0.45) oder (0.6, 0.6, 0.3) beschrieben. Da auf der kleinen Validierungsstichprobe geringfügige Unterschiede zwischen den Erkennungsraten rein zufällig und nicht signifikant sind, ist also die aus Tabelle 5.17 gewonnene Erkenntnis über die Wichtigkeit der Kodebücher ($\alpha_{abl2} < \alpha_{stat} < \alpha_{abl1}$) nicht immer zwingend. Deshalb wurde in diesem Abschnitt nach den optimalen Kodebuchexponenten systematisch gesucht. Erst wurden im $\mathbb{R}^3$ alle 64 Gitterpunkte ($\alpha_{stat}, \alpha_{abl1}, \alpha_{abl2}$) mit $\alpha_{stat}, \alpha_{abl1}, \alpha_{abl2} \in \{0.2, 0.4, 0.6, 0.8\}$ getestet.

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.20	0.30	0.35	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
α_{abl1}	0.80	0.70	0.70	0.60	0.60	0.70	0.70	0.80	0.80	0.80	0.80	0.80
α_{abl2}	0.40	0.40	0.40	0.50	0.60	0.40	0.50	0.20	0.30	0.40	0.50	0.60
WA in %	77.6	79.0	79.3	79.1	79.1	78.9	79.1	79.1	78.8	79.2	***79.4***	78.9

α_{stat}	0.45	0.45	0.50	0.50	0.50	0.50	0.50	0.50	0.55	0.55	0.60	0.60
α_{abl1}	0.70	0.70	0.60	0.70	0.70	0.70	0.70	0.80	0.65	0.70	0.60	0.70
α_{abl2}	0.35	0.50	0.50	0.30	0.35	0.40	0.50	0.30	0.45	0.45	0.30	0.40
WA in %	79.1	***79.5***	79.2	***79.5***	79.3	79.0	***79.5***	79.3	***79.5***	79.3	***79.5***	79.1

Tabelle 5.18: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale, Ableitungen und zweite Ableitungen.

Die besten Ergebnisse unterlagen den Einschränkungen $\alpha_{stat} \in [0.4, 0.6]$, $\alpha_{abl1} \in [0.4, 0.8]$ und $\alpha_{abl2} \in [0.2, 0.6]$. Danach wurden ausgewählte Zwischenpunkte untersucht. Gute Tripel zeigt Tabelle 5.18. Mit etlichen auf der Validierungsstichprobe sehr guten Kombinationen von Gewichten wird der Erkenner auf der Teststichprobe getestet. Die Ergebnisse zeigt Tabelle 5.19. Die beste Erkennungsrate beträgt 75.9 % WA, die anderen Ergebnisse liegen alle etwa bei 75.3 % WA. Die Wortakkuratheit von 75.8 % WA, die der entsprechende Erkenner mit zwei Kodebüchern (2×100 Klassen) erzielt, wird also durch Hinzunahme der zweiten Ableitungen nicht bzw. nicht signifikant übertroffen.

Es bleibt, die Anzahl der Kodebuchklassen zu optimieren. Insbesondere muss sie nicht für jedes der Kodebücher gleich sein. Ferner kann noch der Schwellwert θ_{test} für die Vektorquantisierung beim Testen optimiert werden. Bessere Erkennungsraten verspricht man sich auch, wenn man die zweiten Ableitungen über kleinere Zeitfenster berechnet. Da die zweite Ableitung aus der Regression der ersten Ableitung berechnet wird, gehen bei 50 ms Zeitfenstern zu viele statische Merkmale indirekt in die Berechnungen ein. Wir beenden jedoch den Ansatz mit der zweiten Ableitung und betrachten im nächsten Abschnitt erste Ableitungen für verschiedene Zeitauflösungen.

5.3.3 Ableitungen in verschiedenen Auflösungen

Gegenstand der Diskussion in diesem Abschnitt ist es, dynamische Merkmale über verschiedene Zeitauflösungen zu betrachten. Neben den 12 statischen cepstralen Merkmale und deren 12 Ableitungen über 50 ms, wie sie in Abschnitt 5.2 verwendet wurden, gehen nun noch zusätzlich die Ableitungen über 30 ms und 70 ms in die Berechnungen ein. In der Studienarbeit [Hac01]

	Experimente mit der **Teststichprobe**							
α_{stat}	0.35	0.40	0.40	0.45	0.50	0.50	0.55	0.60
α_{abl1}	0.70	0.60	0.80	0.70	0.70	0.70	0.65	0.60
α_{abl2}	0.40	0.60	0.50	0.50	0.30	0.50	0.45	0.30
WA in %	**75.9**	75.3	75.3	75.3	75.4	75.1	74.9	75.3

Tabelle 5.19: Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden drei Kodebücher für statische Merkmale, Ableitungen und zweite Ableitungen mit je 100 Klassen trainiert.

	Experimente mit der			
	Validierungsstichprobe		**Teststichprobe**	
α_{stat}	1.0	0.0	1.0	0.0
α_{mabl}	0.0	1.0	0.0	1.0
WA in %	73.8	67.9	67.6	67.4

Tabelle 5.20: Separates Testen der Kodebücher für statische Merkmale und Ableitungen in verschiedenen Zeitauflösungen. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe.

wurden diese Zeitfenster als optimale Kombination gefunden. Dort wurden die jeweils 12 Ableitungen aus verschiedenen Zeitauflösungen mit der Karhunen-Loève-Transformation auf 12 dynamische Merkmale reduziert.

In diesem Abschnitt verwenden wir die nicht transformierten Merkmale und trainieren verschiedene Kodebücher. Allerdings sind die Ableitungen aus verschiedenen Zeitfenstern stark linear abhängig. Die Unabhängigkeitsannahme, die Gleichung 5.3 zugrunde liegt wird also verletzt. Deshalb untersuchen wir in diesem Abschnitt zwei unterschiedliche Verfahren: Einmal trainieren wir HMM mit zwei Kodebüchern, eines ist für die 12 statischen, das andere für alle 36 dynamischen Merkmale. Erst im zweiten Ansatz wird die Unabhängigkeitsannahme verletzt; dann werden vier Kodebücher für die statischen und die dynamischen Merkmale aus je einer Auflösung trainiert. In beiden Ansätzen hat jedes der Kodebücher 128 Klassen. Der Schwellwert bei der Vektorquantisierung ist im Training und Test der Standardwert $\theta_{train} = 0.01$ bzw. $\theta_{test} = 0.0005$.

Betrachten wir zunächst den Fall mit zwei Kodebüchern für zwölf statische und 36 dynamische Merkmale und insgesamt 256 Klassen. Die Kodebücher sind im Training wieder ungewichtet. Die Kodebuchexponenten α_{stat} und α_{mabl} werden auf der Validierungsstichprobe optimiert. Zuerst betrachten wir in Tabelle 5.20 die Kodebücher einzeln und schalten jeweils das andere ab. Mit beiden Kodebüchern werden etwa gleich gute Erkennungsraten erzielt.

	Experimente mit der *Validierungsstichprobe*											**Test**
α_{stat}	0.20	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.80	0.40
α_{mabl}	0.80	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.35	0.30	0.20	0.60
WA in %	75.9	76.6	***76.8***	***76.9***	76.0	76.2	76.0	76.0	75.7	75.9	74.7	**75.3**

Tabelle 5.21: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale in verschiedenen Zeitauflösungen. Summe der Kodebuchexponenten ist Eins. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten.

	Experimente mit der							
	Validierungsstichprobe				**Teststichprobe**			
α_{stat}	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
α_{abl30}	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
α_{abl50}	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
α_{abl70}	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
WA in %	68.9	62.0	70.9	68.0	66.3	63.3	71.4	68.1

Tabelle 5.22: Testen der Kodebücher für statische Merkmale und Ableitungen über 30. 50 und 70 ms einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe.

In Tabelle 5.21 variieren wir die Kodebuchexponenten unter der Nebenbedingung, dass sie sich zu Eins summieren. Obwohl die Kodebücher im Einzelexperiment etwa gleich gut sind, wird in den optimalen Kombinationen $\alpha_{stat} = 0.35$ und $\alpha_{mabl} = 0.65$ sowie $\alpha_{stat} = 0.4$ und $\alpha_{mabl} = 0.6$ das Kodebuch mit dynamischen Merkmalen mit größeren Exponenten gewichtet. Auf der **Teststichprobe** erreichen wir in beiden Fällen allerdings nur **75.3 % WA**. Im Vergleich mit den Untersuchungen aus Abschnitt 5.2.1 erzielen wir keine Verbesserungen. Auch wenn wir die Kodebuchexponenten so variieren, dass ihre Summe größer Eins ist (mit den Gewichten $\alpha_{stat} = 0.4$ und $\alpha_{mabl} = 0.8$ erreicht man 77.4 % WA auf der Validierungsstichprobe), verbessert sich die Wortakkuratheit auf der Teststichprobe nicht. Bei konstant 128 Kodebuchklassen bringt es offensichtlich keinen Vorteil, 36 Merkmale, von denen je drei stark korreliert sind, statt der üblichen 12 Merkmale zu quantisieren. Auch sind im Kodebuch mit 36-dimensionalen Dichten sehr viele freie Parameter zu schätzen. Es empfiehlt sich, eine größere Stichprobe heranzuziehen.

Im zweiten Teil der Untersuchungen spalten wir also die 36 dynamischen Merkmale auf. Es werden vier Kodebücher mit je 128 Klassen trainiert, eines für 12 statische Merkmale, die anderen für je 12 dynamische Merkmale derselben Zeitauflösung. Das Training erfolgt wieder ungewichtet. Die optimalen Kodebuchexponenten α_{stat}, α_{abl30}, α_{abl50} und α_{abl70} suchen wir er-

	Experimente mit der *Validierungsstichprobe*											
α_{stat}	0.30	0.30	0.35	0.35	0.40	0.40	0.40	0.45	0.45	0.45	0.45	0.50
α_{abl30}	0.25	0.30	0.30	0.35	0.25	0.30	0.30	0.25	0.30	0.35	0.40	0.35
α_{abl50}	0.70	0.70	0.70	0.50	0.50	0.50	0.55	0.50	0.50	0.50	0.50	0.50
α_{abl70}	0.30	0.30	0.30	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
WA in %	79.6	***80.4***	79.7	90.4	80.3	80.3	80.1	80.1	80.1	80.1	89.9	80.1

Tabelle 5.23: Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale, sowie Ableitungen aus 30 ms, 50 ms und 70 ms Zeitauflösungen.

	Experimente mit der **Teststichprobe**				
α_{stat}	0.30	0.35	0.40	0.45	0.45
α_{abl30}	0.30	0.35	0.25	0.30	0.35
α_{abl50}	0.70	0.50	0.50	0.50	0.50
α_{abl70}	0.30	0.40	0.40	0.40	0.40
WA in %	**77.0**	76.9	**77.0**	76.8	76.6

Tabelle 5.24: Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden vier Kodebücher für statische Merkmale und Ableitungen aus 30 ms, 50 ms und 70 ms Zeitfenstern mit je 128 Klassen trainiert.

neut mit der Validierungsstichprobe. Wieder testen wir zunächst die Kodebücher einzeln. Tabelle 5.22 zeigt, dass das Kodebuch für Ableitungen aus 50 ms Zeitfenstern die besten Erkennungsraten erzielt. Danach folgen die Kodebücher für statische Merkmale und Ableitungen über 70 ms Zeitauflösungen. Die Merkmale mit 30 ms Zeitauflösungen liefern die geringste Wortakkuratheit.

Mit diesem Vorwissen beginnen wir die aufwändige Suche im $\mathbb{R}^4$ nach optimalen Kodebuchexponenten. Dabei beschränken wir uns aus Aufwandsgründen darauf, von den optimalen Gewichten $\alpha_{stat} = 0.3$ und $\alpha_{abl50} = 0.7$ bzw. $\alpha_{stat} = 0.45$ und $\alpha_{abl50} = 0.55$ sowie $\alpha_{stat} = 0.5$ und $\alpha_{abl50} = 0.9$, die in Abschnitt 5.2.1 für zwei Kodebücher gefunden wurden, auszugehen. Tabelle 5.23 zeigt einige Quadrupel von Gewichten, die auf der Validierungsstichprobe optimal sind, z.B. $\alpha_{stat} = 0.3$, $\alpha_{abl30} = 0.3$, $\alpha_{abl50} = 0.7$ und $\alpha_{abl70} = 0.3$ oder $\alpha_{stat} = 0.4$, $\alpha_{abl30} = 0.25$, $\alpha_{abl50} = 0.5$ und $\alpha_{abl70} = 0.4$. Auf der Teststichprobe werden, wie man aus Tabelle 5.24 entnehmen kann, bis zu **77.0 % WA** erzielt. Im Vergleich mit den Untersuchungen aus Abschnitt 5.2.1 mit Merkmalen in nur einer Zeitauflösung und zwei Kodebüchern (dort werden bis zu 76.5 % WA erzielt), wird keine signifikante Verbesserung erreicht. Dennoch kann der Erkenner noch optimiert werden, indem man die Klassenzahlen für die verschiedenen Kodebücher bzw. die Schwellwerte bei der Vektorquantisierung variiert. Jedenfalls liefert dieses

aufwändige Experiment mit vier Kodebüchern und insgesamt 512 Klassen wesentlich bessere Ergebnisse als jenes, das zu Beginn dieses Abschnittes vorgestellt wurde. Dort wurde mit den 36 dynamischen Merkmale in verschiedenen Zeitauflösungen nur ein einziges Kodebuch mit 128 Klassen trainiert.

Wir beenden nun die Untersuchungen verschiedener Ansätze, Erkenner mit mehreren Kodebüchern zu trainieren. Im folgenden Abschnitt prüfen wir, ob durch den Multi-Kodebuch-Ansatz die Robustheit des Erkenners gegenüber Hall erhöht wird.

5.4 Untersuchungen zur Robustheit gegen Hall

Werden neben der Sprache Störgeräusche aufgenommen, verschlechtert sich üblicherweise die Erkennungsrate stark. Auch in stiller Umgebung tritt ein Störgeräusch auf: Hall. Erfolgen die Aufnahmen über ein Nahbesprechungsmikrophon, so ist das Sprachsignal weitgehend frei von Hall. Dies gilt auch für Aufnahmen via Telefon, etwa bei den Anfragen an ein Zugauskunftsystem.

In vielen praktischen Anwendungen kann man aber Störgeräuschen nicht verhindern. Zur Bedienung eines Videorekorders per Sprachbefehle beispielsweise will der Benutzer bestimmt kein Nahbesprechungsmikrophon tragen müssen. Die Aufnahmen erfolgen mit einem Raummikrophon. In [Had02] wurde bereits eine Suche nach hallrobusten Merkmalen gestartet. Vom Ziel, dass sie Erkennungsraten des Raummikrophon mit denen des Nahbesprechungsmikrophon vergleichbar sind, ist man aber noch weit entfernt.

Wird der Erkenner mit unverhallten Sprachdaten trainiert, und mit einem Raummikrophon getestet, treten in der Anwendung nicht trainierte akustische Bedingungen auf. Wir untersuchen nun, ob der Erkenner mit mehreren Kodebüchern robuster gegenüber solchen Ereignissen ist. Dazu trainieren wir ein neues Referenzsystem mit einem Kodebuch mit 256 Klassen auf der VERBMOBIL_TINY Stichprobe. Die Schwellwerte zur Vektorquantisierung betragen hier $\theta_{train} = 0.01$ und $\theta_{test} = 0.005$. Getestet wird dieser Erkenner mit Daten, die im Rahmen des Müdigkeitsexperimentes aufgenommen wurden (siehe Kapitel 4.1). Die Äußerungen der fünf Sprecher wurden sowohl mit einem Nahbesprechungsmikrophon aufgezeichnet, als auch mit einem Raummikrophon. Auf diesen beiden Datensätzen wird ein Erkenner getestet, der auf HMM mit nur einem Kodebuch basiert. Im ersten Fall werden 73.8% WA erzielt, im zweiten Fall 33.5% WA. Die Erkennung auf den verrauschten Daten ist erwartungsgemäß sehr viel schlechter.

Nun wird ein zweiter Erkenner trainiert, der mit HMM mit mehreren Kodebüchern arbeitet.

Erkenner	Testdaten	α_{stat}	α_{dyn}	WA in %
Referenzsystem mit einem Kodebuch	Nahbesprechungsmikro			73.8
Referenzsystem mit einem Kodebuch	Raummikro			33.5
2 Kodebücher für statische und dynamische Merkmale	Nahbesprechungsmikro	0.30	0.70	75.1
2 Kodebücher für statische und dynamische Merkmale	Raummikro	0.30	0.70	29.3
2 Kodebücher für statische und dynamische Merkmale	Nahbesprechungsmikro	0.45	0.55	76.5
2 Kodebücher für statische und dynamische Merkmale	Raummikro	0.45	0.55	31.6

Tabelle 5.25: Erkennungsraten mit Raummikrophon und unverhallten Nahbesprechungsmikrophon mit verschiedenen Erkennern. α_{stat} und α_{dyn} sind die Kodebuchexponenten.

Es werden zwei Kodebücher für die 12 statischen bzw. die 12 dynamischen Merkmale wie im Abschnitt 5.2.1 berechnet. Jedes der Kodebücher hat 128 Klassen. Mit diesem Erkenner wird wieder die Wortakkuratheit auf der nichtverhallten und der verhallten Stichprobe getestet. Wir erwarten eine Verbesserung der Wortakkuratheit in beiden Fällen, jedoch hoffen wir, dass die Erkennungsraten auf verhallten Daten stärker zunehmen. Dann ist der neue Erkenner robuster gegen Hall.

Wie die Ergebnisse aus Tabelle 5.25 zeigen, ist die Robustheit des Erkenners gegenüber Hall nicht gestiegen. Die Erkennungsraten auf den Nahbesprechungsmikrophon-Daten wachsen im 2-Kodebuch-Ansatz auf bis zu 76.5 % WA an. Dies entspricht einer Verbesserung um über 2.5 Prozentpunkte. Die Kodebuchexponenten wurden allerdings nicht systematisch variiert, sondern nur einige wenige Paare getestet. Das Optimum in der Erkennungsrate ist also u.U. noch größer. Die Erkennungsraten mit den verrauschten Daten verschlechtern sich aber. Auch durch weiteres Variieren der Kodebuchexponenten kann wohl kein Paar mehr gefunden werden, mit dem die Wortakkuratheit auf den verrauschten Daten stärker steigt, als die nachgewiesene Steigerung auf den unverrauschten Daten. Der Erkenner mit zwei getrennten Kodebüchern für statische und dynamische Merkmale ist also nicht robuster gegen Hall. Im nächsten Abschnitt werden wichtige Experimente aus diesem Kapitel zusammengefasst.

5.5 Zusammenfassung

In diesem Kapitel wurde zuerst der Ansatz mit mehreren Kodebüchern für unterschiedliche Merkmaltypen motiviert. Insbesondere wurden die Auswirkungen unterschiedlicher Kodebuchexponenten erklärt. Sinnvolle Ergebnisse liefern Kodebuchexponenten $0 \leq \alpha \leq 1$. Das Kodebuch, das für das wichtiger gehalten wird, gewichtet man mit einem größeren Exponenten. Mit $\alpha = 0$ wird ein Kodebuch "abgeschaltet".

Es werden verschiedene Erkenner auf der Stichprobe ERDIAL_4999 trainiert. Das Training erfolgt mit ungewichteten Kodebüchern $\alpha = 1$. Variiert wird die Anzahl der Klassen, sowie die Anzahl der Kodebücher. Mit den üblichen 24-dimensionalen Merkmalvektoren werden einmal zwei Kodebücher für statische und dynamische Merkmale trainiert und später drei Kodebücher für die Energie und deren Ableitung, 11 MFCCs sowie 11 Ableitungen der MFCCs. In weiteren Untersuchungen werden die zweite Ableitung bzw. zusätzliche erste Ableitungen, die über verschieden breite Zeitfenster berechnet wurden, hinzugenommen. Es werden HMM mit bis zu vier Kodebüchern trainiert.

Mit dem fertig trainierten Erkenner werden auf der Validierungsstichprobe optimale Kodebuchexponenten gesucht. Mit diesen werden die Kodebücher gewichtet. Der Erkenner wird dann auf der Teststichprobe evaluiert. Ein erneutes "schiefes" Training mit diesen optimalen Gewichten ist denkbar und wird in einem Fall näher untersucht. Die Erkennungsraten können so weiter verbessert werden. Auch werden die Schwellwerte für die Vektorquantisierung θ_{test} ein einigen Fällen auf der Validierungsstichprobe optimiert. Für verschieden große Kodebücher und verschiedene Merkmaltypen erscheinen unterschiedliche θ_{test} als sinnvoll. Es werden aber nur gemeinsame θ_{test} für alle Kodebücher verwendet. Eine Zusammenfassung der wichtigsten Ergebnisse aus diesem Abschnitt zeigt Tabelle 5.26.

Ein optimales Ergebnis von 77.8 % WA wird mit zwei Kodebüchern mit je 128 Klassen für statische und dynamische Merkmale erzielt, wenn die Kodebücher schon während des Trainings gewichtet werden. Vielversprechend sind auch Untersuchungen, in denen die Klassenzahl weiter erhöht wird oder die Energiemerkmale in einem dritten Kodebuch ausgelagert werden. Man erwartet sich auch dort durch ein "schiefes" Training weitere Verbesserungen.

Mit einer weiteren Stichprobe (VERBMOBIL_TINY) werden zusätzlich Erkenner mit einem und zwei Kodebüchern trainiert und mit verhallten und unverhallten Stichproben getestet. Es zeigt sich allerdings keine verbesserte Hallrobustheit mit dem 2-Kodebuch Ansatz (Tabelle 5.25). Im nächsten Kapitel werden verschiedene Kodebücher für unterschiedliche Lautoberklassen trainiert.

Experiment	Para. /10^3	Klassen pro KB	# KB	KB-exponenten K1 / K2 / K3 / K4	θ_{test}	WA in %
Referenzsystem	83	256	1	1.00 / - / - / -	0.0005	73.5
Referenzsystem	83	256	1	1.00 / - / - / -	0.2	74.4
Referenzsystem	166	512	1	1.00 / - / - / -	0.2	74.5
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	23	128	2	0.30 / 0.70 / - / -	0.0005	76.5
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	23	128	2	0.45 / 0.55 / - / -	0.0005	75.7
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	23	128	2	0.50 / 0.90 / - / -	0.0005	75.8
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	23	128	2	0.30 / 0.70 / - / -	0.2	76.9
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	18	100	2	0.35 / 0.65 / - / -	0.0005	75.8
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	36	200	2	0.30 / 0.70 / - / -	0.0005	77.3
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2)	46	256	2	0.25 / 0.75 / - / -	0.0005	77.4
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2) Trainingsgewichte 0.5 u. 0.5	23	128	2	0.25 / 0.75 / - /-	0.0005	76.5
12 stat. Merkmale (K1) 12 dyn. Merkmale (K2) Trainingsgewichte 0.3 u. 0.7	23	128	2	0.45 / 0.55 / - / -	0.0005 bzw. 0.2	**77.8**
2 Energiemerkmale (K1) 11 MFCC (K2), 11 Abl. (K3)	16	100	3	0.30 / 0.30 / 0.75 / -	0.0005	76.7
2 Energiemerkmale (K1) 11 MFCC (K2), 11 Abl. (K3)	16	100	3	0.30 / 0.30 / 0.75/ -	0.1	77.5
12 stat. Merkmale (K1) 12 Abl. (K2) 12 zweite Abl. (K3)	27	100	3	0.35 / 0.70 / 0.40 / -	0.0005	75.9
12 stat. Merkmale (K1) 36 Abl. für verschiedene Zeitfenster (K2)	101	128	2	0.40 / 0.60 / - / -	0.0005	75.3
12 stat. Merkmale (K1) 12 Abl. (30ms) (K2) 12 Abl. (50ms) (K3) 12 Abl. (70ms) (K4)	46	128	4	0.30 / 0.30 / 0.70 / 0.30	0.0005	77.0

Tabelle 5.26: Wichtiger Experimente aus diesem Kapitel. Angegeben sind die Kodebuchparameter in Tausend, Anzahl der Klassen pro Kodebuch, Anzahl der Kodebücher, Exponenten für Kodebücher K1 - K4, Schwellwert bei der Vektorquantisierung und Wortakkuratheit in %

Kapitel 6

Kodebücher für verschiedene Lautoberklassen

In diesem Kapitel wird der zweite experimentelle Teil der Diplomarbeit diskutiert. Der Spracherkenner wird so erweitert, dass verschiedene Kodebücher für verschiedene Lautoberklassen bereitgestellt werden. Dieser Ansatz wurde bereits ausführlich in Abschnitt 3.2 motiviert. Auch ein Literaturüberblick wurde dort gegeben.

Im folgenden wird in einem ersten Abschnitt die genaue Vorgehensweise beschrieben. Experimente und Ergebnisse werden danach vorgestellt.

6.1 Vorgehensweise

Auch in diesem Kapitel werden semikontinuierliche HMM mit mehreren Kodebüchern untersucht. Im Gegensatz zum letzten Abschnitt wird in jedem HMM-Zustand wieder nur genau ein Kodebuch betrachtet. Zustände verschiedener Lautoberklassen verwenden dagegen unterschiedliche Kodebücher. Dies ist ein deutlicher Schritt vom semikontinuierlichen Hidden-Markov Modell hin zum kontinuierlichen. Man verspricht sich bessere Erkennungsraten.

Wir betrachten in diesem Kapitel $L = 47$ Lautoberklassen. Die L Teilkodebücher lassen sich wie schon im Abschnitt 3.2 beschrieben als ein einziges Kodebuch betrachten. In diesem Kapitel kommt nur der einfache Fall zur Anwendung bei dem die Klassenzahl K für alle Teilkodebücher gleich ist. Das gesamte Kodebuch hat dann $K_{ges} = L \cdot K$ Klassen.

In Tabelle 6.1 sind 81 im ISADORA-System verwendete Laute in SAMPA-Notation aufgelistet (vgl. [ST95, S.396 f.]). Beispiele sind der gedehnte “o”-Vokal /o:/, der Schwa /@/ oder der Affrikat /ps/, aber auch Stille /-/, Nonverbalien /NV/, Atmung /ATM/ oder der Signalton am Te-

lefon /TUT/. Jeder dieser elementaren Laute wird im HMM durch mehrere Zustände modelliert, die in Spalte zwei angegeben sind. Die Symbole in eckigen Klammern geben die phonemischen Basiseinheiten an, der der jeweilige Zustand zugeordnet wird. Es gibt 47 Symbole, die den Lautoberklassen entsprechen.

Für jeden Zustand wird ein Kodebuchindex l gespeichert, der die Nummer des Kodebuches angibt ($l \in \{0, 1, 2, ..., L-1\}$). Daraus lässt sich berechnen, welche Komponenten des gesamten Kodebuches zu dem jeweiligen zustandsspezifischen Kodebuch gehören. Es sind die K Klassen $l \cdot K, l \cdot K + 1, ..., (l+1) \cdot K - 1$

Zu Beginn der Experimente wird ein initiales Kodebuch berechnet. Dies geschieht mit einer beschrifteten oder etikettierten Stichprobe; für jeden Merkmalvektor ist bekannt, welcher der $L = 47$ Lautoberklassen er angehört. Diese Beschriftung erfolgt datengetrieben mit Hilfe der *hlb-Dateien* (siehe Abschnitt 4.3.1) des Referenzsystems. Für jede Äußerung ist dort mit dem Viterbi-Algorithmus die wahrscheinlichste Zustandsfolge berechnet worden. Jeder Zustand ist dabei mit dem Namen seiner Lautoberklasse bezeichnet. Die Länge der Zustandsfolge ist identisch mit der Anzahl der Merkmalvektoren in der Äußerung. So erhält man für jeden Merkmalvektor seine geschätzte Klassenzugehörigkeit.

Darauf folgt die Kodebuchberechnung. Man betrachtet immer nur die Merkmalvektoren einer bestimmten Lautoberklasse. Mit diesen Merkmalvektoren wird ein zustandsspezifisches Kodebuch mit K Normalverteilungsdichten berechnet. Die Dichten der Kodebücher aller Lautoberklassen werden nun zu einem einzigen initialen Kodebuch mit $K_{ges} = L \cdot K$ Klassen zusammengefasst.

Zur Neuschätzung des Kodebuchs und der HMM-Parameter ergeben sich die üblichen Formeln für semikontinuierliche HMM (Gleichungen 2.40). Allerdings muss man im Zustand j, dessen Kodebuchindex l ist, alle zustandsspezifischen Gewichte der Kodebuchkomponenten c_{jk} von "fremden" Kodebuchklassen Null setzen ($c_{jk} = 0$ für $k < l \cdot L$ und $k \geq (l+1) \cdot L$). In der Implementierung entfernen wir alle $(L-1) \cdot K$ auf Null gesetzte Komponenten c_{jk} des Vektors mit Ausgabegewichten. Übrig bleiben K Gewichte c_{jk}, wobei die k-te Komponente die Kodebuchklasse $l \cdot L + k$ gewichtet. Die Wahrscheinlichkeit, dass der Merkmalvektor $\boldsymbol{c}$ ausgegeben wird ist

$$b_j(\boldsymbol{c}) = \sum_{k=0}^{K} c_{jk} g_{l \cdot L + k}(\boldsymbol{c}). \tag{6.1}$$

Dabei ist g_k die k-te Normalverteilungskomponente im gesamten Kodebuch.

Im folgenden Abschnitt werden die Experimente zu diesem Ansatz erläutert.

Laut	**Zustände**	**Laut**	**Zustände**
/@/	[%]	/E:/	[e] [e] [e]
/6/	[@] [@] [@]	/f/	[f] [f] [f]
/e/	[E] [E] [E]	/g/	[G] [g] [g]
/e:/	[E] [E] [E]	/gr/	[G] [g] [r] [r]
/i/	[I] [I] [I]	/h/	[h] [h] [h]
/i:/	[I] [I] [I]	/I/	[i] [i] [i]
/N/	[N] [N] [N]	/j/	[i] [i] [i]
/o/	[O] [O] [O]	/k/	[K] [k] [9]
/o:/	[O] [O] [O]	/kr/	[K] [k] [x] [x]
/o6/	[O] [O] [@] [@]	/ks/	[K] [k] [s] [s]
/a6/	[a] [a] [@] [@]	/kv/	[K] [k] [v] [v]
/e6/	[E] [E] [@] [@]	/l/	[l] [l] [l]
/E6/	[e] [e] [@] [@]	/m/	[m] [m] [m]
/i6/	[I] [I] [@] [@]	/n/	[n] [n] [n]
/I6/	[i] [i] [@] [@]	/O/	[o] [o] [o]
/u6/	[U] [U] [@] [@]	/OY/	[o] [o] [q] [i]
/U6/	[u] [u] [@] [@]	/p/	[P] [p] [7]
/y6/	[Y] [Y] [@] [@]	/pf/	[P] [p] [f] [f]
/Y6/	[y] [y] [@] [@]	/pr/	[P] [p] [x] [x]
/26/	[Q] [Q] [@] [@]	/ps/	[P] [p] [s] [s]
/96/	[q] [q] [@] [@]	/9/	[q] [q] [q]
/O6/	[o] [o] [@] [@]	/9:/	[q] [q] [q]
/2/	[Q] [Q] [Q]	/r/	[r] [r] [r]
/2:/	[Q] [Q] [Q]	/s/	[s] [s] [s]
/S/	[S] [S] [S]	/t/	[T] [t] [8]
/Z/	[S] [S] [S]	/tS/	[T] [t] [S] [S]
/u/	[U] [U] [U]	/tr/	[T] [t] [x] [x]
/u:/	[U] [U] [U]	/ts/	[T] [t] [s] [s]
/y/	[Y] [Y] [Y]	/tz/	[T] [t] [z] [z]
/y:/	[Y] [Y] [Y]	/U/	[u] [u] [u]
/a/	[a] [a] [a]	/v/	[v] [v] [v]
/a:/	[a] [a] [a]	/w/	[u] [u] [u]
/aI/	[a] [a] [e] [i]	/x/	[x] [x] [x]
/aU/	[a] [a] [o] [u]	/Y/	[y] [y] [y]
/b/	[B] [b] [b]	/Y:/	[y] [y] [y]
/br/	[B] [b] [r] [r]	/z/	[z] [z] [z]
/C/	[c] [c] [c]	/-/	[-]
/d/	[D] [d] [d]	/NV/	[X]
/dj/	[D] [d] [i] [i]	/ATM/	[H]
/dr/	[D] [d] [r] [r]	/TUT/	[W]
/E/	[e] [e] [e]		

Tabelle 6.1: 81 elementare Laute unterteilt mit 47 phonemischen Basiseinheiten

6.2 Experimente

In den folgenden Experimenten trainieren wir initiale Kodebücher mit unterschiedlicher Klassenzahl. Bereits nach bis zu 20 Baum-Welch Iterationen wird noch vor der ersten Kodebuchneuschätzung des in Abschnitt 4.3.1 beschriebenen Trainingsverfahrens der Erkenner getestet. Training und Test erfolgen mit der Stichprobe ERDIAL_4999 (siehe Abschnitt 4.1).

Nachfolgend wird zuerst das phonetischen Mischungsverteilungskodebuch im Single-Kodebuch-Ansatz untersucht. Danach werden Ergebnisse und Experimente zum Multi-Kodebuch-Ansatz beschrieben.

6.2.1 Experimente mit einem phonetischen Mischungsverteilungskodebuch

In diesem Kapitel werden zwei verschiedene Erkenner untersucht, die ein phonetischen Mischungsverteilungskodebuch verwenden. Für beide Erkenner werden initiale Kodebücher wie im letzten Abschnitt beschrieben trainiert: Die Merkmalvektoren werden mit ihrer geschätzten Lautzugehörigkeit beschriftet. Für jede Lautoberklasse wird ein Kodebuch mit $K = 25$ Klassen berechnet. Nach Zusammenfassen der $L = 47$ Kodebücher erhält man ein großes phonetischen Mischungsverteilungskodebuch mit 1175 Klassen.

Damit trainieren wir nun einen Erkenner, der auf gewöhnlichen semikontinuierlichen HMM mit einem Kodebuch basiert. Schon nach 20 Baum-Welch-Iterationen erhalten wir auf der Teststichprobe eine Wortakkuratheit von 77.6 %. Nach einer weiteren Trainingsrunde mit Kodebuchneuschätzung und Baum-Welch-Training wie in Abschnitt 4.3.1 beschrieben (aufgelistete Schritte 1 - 5) verschlechtert sich die Wortakkuratheit auf 76.9 %. Mit dem Referenzsystem aus Abschnitt 4.4 erreicht man nach 10 Trainingsrunden 74.4 % WA. Durch das Einbringen von wissen über die Lautzugehörigkeit der Merkmalvektoren erhalten wir also zunächst ein Kodebuch, mit dem deutlich bessere Ergebnisse erzielt werden können. Wohl wegen der zu großen Anzahl an zu schätzenden Parametern versagt jedoch die Kodebuchneuschätzung.

Im zweiten Experiment wird der Ansatz aus [ST95, S.322f] wiederholt, der bereits im Literaturüberblick in Abschnitt 3.2.2 vorgestellt wurde. Wieder verwenden wir das phonetische Mischungsverteilungskodebuch mit 1175 Klassen, also weitaus mehr Klassen als in [ST95]. Ähnliche Normalverteilungskomponenten werden anschließend verschmolzen; man erhält 512 Klassen. Als Abstandsmaß wird der Informationsverlust verwendet, alternativ ist auch der mit der Varianz gewichtete euklidische Abstand der Mittelwertvektoren denkbar. Nach 20 Baum-Welch-Iterationen erzielen wir auf der Teststichprobe eine Wortakkuratheit von 76.3 %; nach

Beschreibung	WA ohne KB-Neuschätzung	WA mit KB-Neuschätzung
phonetischen Mischungsverteilungskodebuch 47 * 25 = 1175 Klassen	77.6 %	76.9 %
phonetischen Mischungsverteilungskodebuch 47 * 25 = 1175 Klassen zu 512 Klassen verschmolzen	76.3 %	76.8 %

Tabelle 6.2: Erkennungsraten mit einem phonetischen Mischungsverteilungskodebuch vor der ersten Kodebuchneuschätzung und nach einer weiteren Trainingsrunde

einer weiteren Trainingsrunde mit Kodebuchneuschätzung (Abschnitt 4.3.1, aufgelistete Schritte 1 - 5) verbessert sich die Erkennungsrate auf 76.8 % WA. Durch das Verschmelzen von Klassen geht also etwa ein Prozentpunkt verloren. Mit dem Referenzsystem mit 512 Klassen aus Abschnitt 4.4 erreicht man nach 10 Trainingsrunden nur 74.5 % WA.

Alle Ergebnisse aus diesem Abschnitt findet man in Tabelle 6.2. Im nächsten Abschnitt werden Experimente mit mehreren Kodebüchern durchgeführt.

6.2.2 Der Multi-Kodebuch Ansatz

In diesem Abschnitt wird der Ansatz untersucht, unterschiedliche Kodebücher für verschiedene Lautoberklassen zu verwenden. Dazu wird wie im letzten Abschnitt ein phonetisches Mischungsverteilungskodebuch trainiert: Die Merkmalvektoren werden mit ihrer geschätzten Lautzugehörigkeit beschriftet. Für jede Lautoberklasse wird ein Kodebuch mit K Klassen berechnet. Nach Zusammenfassen der $L = 47$ Kodebücher erhält man ein großes phonetischen Mischungsverteilungskodebuch mit $K \cdot 47$ Klassen.

Jeder HMM-Zustand gehört einer bestimmten Lautoberklasse an. Jeder Lautoberklasse ordnen wir ferner eine Kodebuchnummer zu. Daraus lässt sich berechnen, welche Kodebuchklassen die Ausgabeverteilung des jeweiligen Zustands modellieren. Die anderen Dichten werden von Zuständen derselben Lautoberklasse nicht betrachtet.

Nun wird die Anzahl der Kodebuchklassen K variiert. Nach dem Erzeugen eines initialen phonetischen Mischungsverteilungskodebuches werden 20 Baum-Welch-Iterationen durchgeführt. Der Erkenner wird vor der Kodebuchneuschätzung getestet. Wie Tabelle 6.3 zeigt, sind die Erkennungsraten im Vergleich zu den Ansätzen mit einem Kodebuch aus dem letzten Abschnitt (Tabelle 6.2) durchwegs schlecht. Mit 25 Kodebuchklassen für jedes Teilkodebuch erreichen wir ein Optimum von 51.9 % WA.

Vermutlich liegt dieses sehr schlechte Ergebnis daran, dass eine feste Anzahl von beispiels-

# Klassen pro Lautoberklasse	# Klassen insgesamt	WA in %
8	376	48.8
12	564	49.5
25 (vgl. Tabelle 6.2)	1175	**51.9**
50	2350	42.0

Tabelle 6.3: Erkennungsraten mit mehreren Kodebüchern für verschiedene Lautoberklassen vor der ersten Kodebuchneuschätzung

weise $K = 25$ Kodebuchklassen zu unflexibel ist. Zustände mancher Lautoberklassen benötigen wohl mehr als K Dichten, um die tatsächliche Ausgabeverteilung der Merkmalvektoren dieser Klasse gut zu approximieren. Für andere Lautoberklassen hingegen genügen weniger Dichten; gegebenenfalls reichen die Beobachtungen in der Teststichprobe auch gar nicht aus, um K Kodebuchklassen zu schätzen. Wir prüfen dies anhand des Erkenners mit $L \cdot K = 1175$ Klassen aus dem letzten Abschnitt nach. Dazu betrachten wir die zustandsspezifischen Gewichte c_{jk} für die einzelnen Dichten. Rein datengetrieben suchen wir nach einem Schwellwert w und betrachten nur solche Kodebuchdichten, deren Gewichte $c_{jk} > w$ sind. Bei Gleichverteilung würde für alle Gewichte $c_{jk} = 1/1175 \approx 0.0009$ gelten. Wir betrachten den Schwellwert $w = 0.001$ und ermitteln für alle Zustände einer Lautoberklasse die durchschnittliche Anzahl an verwendeten Dichten. Je nach Lautoberklasse liegt diese Zahl im Bereich zwischen 19 und 156. Die Lautoberklasse [W], mit der das Tuten des Telefons modelliert wird (vgl. Tabelle 6.1) gewichtet nur wenige Dichten stark. Dies entspricht den Erwartungen, da das Tuten kaum variiert. Zustände wie Stille [-], Nonverbalien [X] oder die Verschlußphase der “t”-Laute [T] benötigen sehr viele Klassen. Veranschaulicht wird die Anzahl der verwendeten Dichten für die 47 Lautoberklassen in Abbildung 6.1

Nach der Kodebuchneuschätzung verschlechtert sich die Erkennung. Dies liegt wie schon im letzten Abschnitt erwähnt daran, dass die Anzahl der zu schätzenden Parameter für die verwendete Trainingsmenge zu klein ist.

An dieser Stelle sind zahlreiche weitere Optimierungen denkbar: Andere Lautoberklassen, unterschiedliche Anzahl von Klassen pro Lautoberklasse und Verschmelzen von Klassen im Multi-Kodebuch-Ansatz. Die Ergebnisse der Ansätze mit einem Kodebuch aus Tabelle 6.2 haben gezeigt, dass durch Betrachtung der Lautoberklassen deutlich bessere Erkennungsraten erzielt werden können. Vergleicht man das Experiment, in dem das phonetische Mischungsverteilungskodebuch zu 512 Klassen verschmolzen wird mit dem Referenzsystem mit 512 Klassen, so erkennt man eine Reduktion der Fehlerrate um 9 % relativ schon nach nur einer Kodebuchneuschätzung. Dies soll dazu ermutigen weitere Untersuchungen mit getrennten Kodebüchern

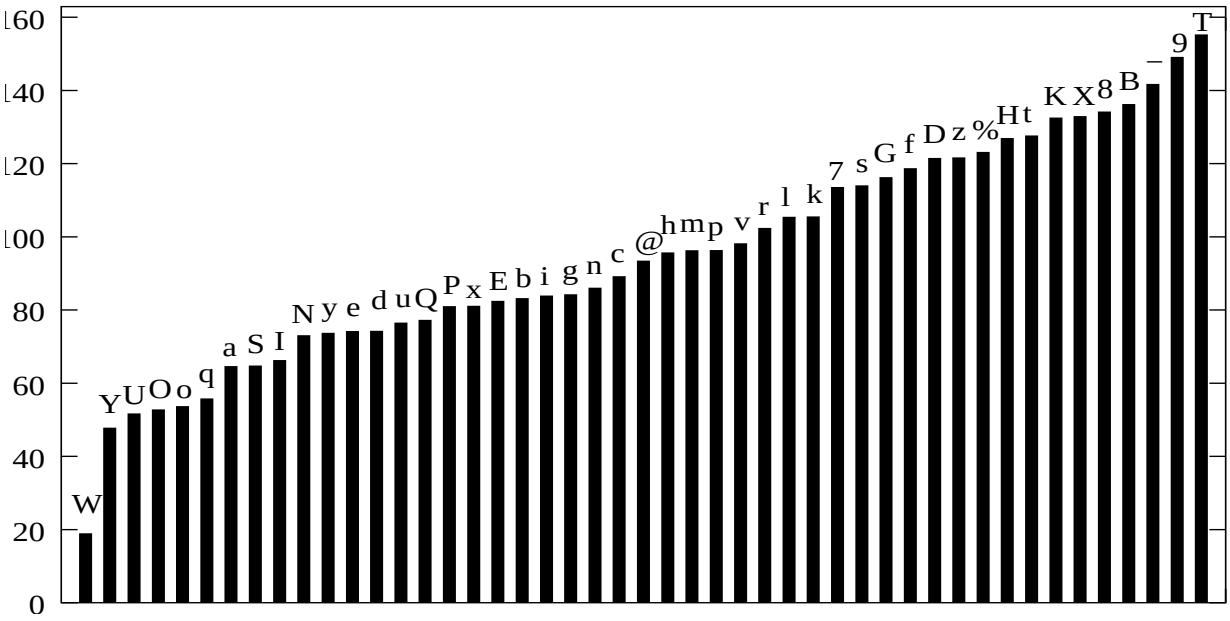


Bild 6.1: Anzahl der wichtigen Dichten für verschiedene Lautoberklassen. Es gilt $c_{jk} > 0.001$

durchzuführen, die aus Zeitgründen im Rahmen dieser Diplomarbeit nicht mehr möglich sind.

Im nächsten Kapitel erfolgt ein Ausblick, bevor die Arbeit mit der Zusammenfassung schließt.

Kapitel 7

Ausblick

In diesem Kapitel wird ein kurzer Überblick darüber gegeben, welche weiteren Untersuchungen mit semikontinuierlichen Hidden-Markov Modellen, die mehrere Kodebücher verwenden, sinnvoll erscheinen. In der vorliegenden Arbeit wurden zwei Teilgebiete unterschieden: Einmal werden Experimente mit getrennten Kodebüchern für verschiedene Merkmaltypen unternommen; im zweiten Teil werden Kodebücher für unterschiedliche Lautoberklassen betrachtet.

Im ersten Teil der Arbeit erfolgte eine aufwändige Optimierung zahlreicher Parameter: Die Anzahl der verwendeten Kodebücher, die Anzahl an Klassen pro Kodebuch, die Kodebuchexponenten zum Gewichten der einzelnen Kodebücher sowohl beim Training als auch beim Testen und schließlich die durchschnittliche Anzahl von Klassen, die beim Quantisieren eines Teilvektors betrachtet wird und durch einen Schwellwert festgesetzt ist. Ausführlich wurden die Optimierungen für das System mit zwei Kodebüchern für statische und dynamische Merkmale durchgeführt. Lediglich die Klassenzahl wurde stets für alle Kodebücher als gleich angenommen. Es erscheint jedoch als sinnvoll, dem durch höhere Exponenten begünstigten Kodebuch auch mehr Klassen zu gewähren. Eine optimale Klassenzahl könnte in der Trainingsphase gelernt werden, indem man Normalverteilungen, die von vielen Zuständen durch hohe Zustandsgewichte begünstigt werden, aufsplittet. Wenn man Kodebuchexponenten bereits in der Trainingsphase verwendet, kann unter Umständen auf das Gewichten in der Testphase verzichtet werden. So wird das Erkennen beschleunigt werden.

Als sinnvoll erschien es, ein separates drittes Kodebuch für Energiemerkmale und deren Ableitung zu trainieren, weil die Verteilung dieser Komponenten im Merkmalraum sich stark von der Verteilung der MFCCs unterscheidet. Hier wurden allerdings die zahlreichen Parameter nicht vollständig optimiert, da der zeitliche Aufwand bei drei Kodebüchern sehr hoch ist und im Rahmen dieser Diplomarbeit nicht mehr möglich war. Ein zweites Training mit den optimalen Ko-

debuchgewichten erscheint als sinnvoll, sowie eine Variation der Klassenzahl. Es kann auch untersucht werden, ob das Auftrennen des Energiemerkmals und seiner Ableitung weitere Vorteile mit sich bringt.

Zu den zwei Kodebüchern für statische und dynamische Merkmale werden in einem dritten Kodebuch die zweiten Ableitungen hinzugenommen. Es konnten keine Verbesserungen erzielt werden. Vermutlich liegt dies an einer falschen Zeitauflösung bei der Berechnung der zweiten Ableitungen. Es wurde dieselbe Zeitauflösung wie zur Berechnung der ersten Ableitung verwendet. Dieses Vorgehen wird in der Dissertation [Rie94] empfohlen. Die zweite Ableitung ist die Regression der ersten. Eine kleinere Zeitauflösung erscheint vielversprechender, da sonst statische Merkmale aus einem zu großen Zeitbereich in die Berechnungen mit eingehen. In einem weiteren Versuch können die Energie und deren beiden Ableitungen in ein separates Kodebuch ausgelagert werden.

Mit Ableitungen aus verschiedenen Zeitauflösungen sind im Multi-Kodebuch-Ansatz keine Verbesserungen zu erzielen, da die Merkmale nicht mehr als linear unabhängig angenommen werden können. Es wurde untersucht, ob die Erkennungsraten gesteigert werden können, wenn man mit Ableitungen verschiedener Zeitauflösungen ein einziges 36-dimensionales Kodebuch berechnet. Es wurden keine Erfolge erzielt. Sinnvoll erscheint es nun, die Klassenzahl für dieses Kodebuch mit dynamischen Merkmalen zu erhöhen. Gleichzeitig sollte eine größere Stichprobe herangezogen werden. Reduziert man die dynamischen Merkmale dagegen mit der Karhunen-Loeve-Transformation zu 12 dekorrelierten Komponenten, so werden bestimmt auch im Multi-Kodebuch-Ansatz, wie schon für den Single-Kodebuch-Ansatz in der Studienarbeit [Hac01] gezeigt, Verbesserungen erzielt.

Die Optimierung der Kodebuchexponenten erfolgte in dieser Arbeit ausschließlich auf der Validierungsstichprobe. Da diese sehr klein ist, sind die Ergebnisse wohl sehr ungenau. Deshalb wurden die Untersuchungen auf der Teststichprobe meist mit verschiedenen auf der Validierungsstichprobe optimalen Kodebuchexponenten durchgeführt. Ein konsequentes Optimieren mit einer größeren Validierungsstichprobe bringt unter Umständen noch geringfügig bessere Erkennungsraten mit sich.

In weiteren zukünftigen Untersuchungen können die unterschiedlichen Merkmalkomponenten unabhängig von ihrem Typ datengetrieben in Gruppen aufgeteilt werden, die dann jeweils von einem eigenen Kodebuch erzeugt werden. Dabei sind solche Merkmale zusammenzufassen, die stärker korreliert sind. Da die Korrelation zwischen Merkmalen aus unterschiedlichen Kodebüchern beim Multi-Kodebuch-Ansatz verloren geht, kann so der Verlust minimiert werden.

Im Artikel [Rog94] wurde bereits gezeigt, dass zustandsspezifische Kodebuchexponenten

gelernt werden können. So werden von verschiedenen Lautoberklassen unterschiedliche Kodebücher als wichtiger erachtet. Auch kann man im Verlauf eines Lautes verschiedene Kodebücher begünstigen.

Ferner könnte man verschiedene Kodebücher betrachten, die alle mit dem gesamten Merkmalvektor trainiert werden. Die Kodebücher unterscheiden sich dann in der Klassenzahl. So würde eine Art Auflösungshierarchie bezüglich der Dichten implementiert. Ein weiterer interessanter Versuch ist, getrennte Kodebücher für weibliche und männliche Sprecher zu implementieren. Anhand der Grundfrequenz wird in der Testphase entschieden, welches der Kodebücher stärker gewichtet werden muss.

Im zweiten Teil dieser Diplomarbeit wurden Kodebücher für verschiedene Lautoberklassen untersucht. Jedes Kodebuch hat 25 Klassen. Es wurde in zwei Verfahren, in denen nur ein Kodebuch verwendet wird, gezeigt, dass durch Betrachten von Lautoberklassen die Erkennungsraten stark gesteigert werden können. Während durch Zusammenfassen der Teilkodebücher zu einem großen Kodebuch mit 1175 Klassen die Wortakkuratheit des Erkenners deutlich verbessert wird, sind die Erkennungsraten jenes Erkenners, der in jedem Zustand nur genau ein Teilkodebuch betrachtet, sehr schlecht. Der Ansatz mit gleicher Klassenzahl für alle Kodebücher erweist sich als sehr unflexibel. Die Experimente sollten mit unterschiedlichen Kodebuchgrößen wiederholt werden. Dazu werden die zustandsspezifischen Gewichte c_{jk} der Zustände j im HMM für Klassen k ausgewertet. Die mittlere Anzahl der Gewichte, die über einem Schwellwert liegen ist für verschiedene Lautoberklassen stark unterschiedlich. Dementsprechend sollten auch im Multi-Kodebuch-Ansatz unterschiedliche Kodebuchgrößen verwendet werden. Um die große Anzahl von Kodebuchklassen stabil zu schätzen, sollten in weiteren Experimenten größere Stichproben herangezogen werden.

Im Rahmen der vorliegenden Diplomarbeit war es leider nicht mehr möglich andere Lautoberklassen zu untersuchen. Interessant wäre beispielsweise, verschiedene Kodebücher für Vokale, Frikative, Nasale und Plosive zu trainieren. Auch könnte datengetrieben gesucht werden, welche Lautoberklassen eine Vielzahl von Dichten gemeinsam verwenden. Ein völlig flexibles "*tied mixture model*" erscheint gegenüber dem letzten Ansatz allerdings als geeigneter. Jeder Zustand verwendet für seine Ausgabeverteilung eine beliebige Teilmenge von Dichten, die von einem gemeinsamen Kodebuch bereitgestellt werden. Dazu werden Klassen mit kleinen zustandsspezifischen Gewichten c_{jk} von diesem Zustand nicht mehr betrachtet. Im Gegenzug können Dichten, die an sehr viele Zustände gebunden sind, in mehrere Klassen aufgeteilt werden. Dieser Ansatz erwies sich bereits im Artikel [Wil97] als erfolgreich. Im Laufe des Trainings können ähnliche Dichten auch wieder verschmolzen werden.

In einem zusammenfassenden Experiment können beide Ansätze kombiniert werden. In einem HMM-Zustand werden dann zum Erzeugen einer Ausgabe einerseits mehrere Kodebücher für verschiedene Merkmalkomponenten berücksichtigt. Andererseits aber sind nur bestimmte Klassen aus jedem Kodebuch an den jeweiligen Zustand gebunden.

Auch Untersuchungen mit einem phonetischen Mischungsverteilungskodebuch pro Merkmaltyp, in dem ähnliche Klassen verschmolzen werden, erscheinen vielversprechend.

Beim semikontinuierlichen HMM sind im Vergleich zum kontinuierlichen deutlich weniger freie Parameter in der Trainingsphase zu lernen. Mit kleinen Stichproben erfolgt so ein stabileres Training; die Erkennungsraten sind besser. Dank leistungsfähigerer Rechner ist es heutzutage möglich einen deutlichen Schritt in Richtung kontinuierliche HMM zu gehen. Nach weiterer Forschung und experimentellen Untersuchungen können für unterschiedliche Aufgabenbereiche, die durch Stichproben hinreichend repräsentiert werden, optimale Hidden-Markov Modelle gefunden werden.

Kapitel 8

Zusammenfassung

Am Lehrstuhl für Mustererkennung wird ein Spracherkenner verwendet, der auf semikontinuierlichen Hidden-Markov Modelle (SCHMM) basiert. SCHMM arbeiten mit einem Kodebuch, das von allen Zuständen geteilt wird. In der vorliegende Diplomarbeit werden SCHMM mit mehreren Kodebüchern untersucht.

Aus dem Sprachsignal einer Äußerung wird eine Folge von Merkmalvektoren berechnet. Dazu wird es in kleine, überlappende Fenster zerlegt. Aus jedem dieser Kurzzeitanalysefenster wird eine Vielzahl von Merkmalen berechnet. Üblich sind als statische Merkmale u.a. die Kurzzeitenergie und Mel-Cepstrum Koeffizienten (MFCC). Zusätzlich verwendet man dynamische Merkmale, die auch den zeitlichen Kontext berücksichtigen. z.B. Ableitungen der statischen Merkmale.

Ein Hidden-Markov Modell (HMM) ist Modell eines Wortes oder einer Wortuntereinheit. Es besteht aus einer Menge von Zuständen, Übergangswahrscheinlichkeiten und Ausgabeverteilungen. In jedem Zustand erfolgt eine beobachtbare Ausgabe, der genaue Weg durch das HMM bleibt jedoch verborgen. So werden zeitliche (nichtlineare) Verzerrungen verschiedener Realisierungen eines Wortes modelliert. Ist eine Beobachtungsfolge gegeben, so wird für jedes HMM die Wahrscheinlichkeit berechnet, dass eben diese erzeugt wird. Eine effiziente Berechnungsmethode ist der Vorwärtsalgorithmus. Das HMM, das mit größter Wahrscheinlichkeit die unbekannte Lautfolge produziert, ist schließlich das Modell des Wortes, nach dem klassifiziert wird. Die unbekannte HMM-Parameter werden mit dem Baum-Welch-Algorithmus gelernt.

Bei diskreten HMM werden die Merkmalvektoren zunächst quantisiert. Ein Kodebuch unterteilt den Merkmalraum in Klassengebiete. Die Klassen können auch überlappen, sie werden dann durch Normalverteilungen modelliert. Der Vektorquantisierer bildet jeden Merkmalvektor auf die Klassennummer oder ein Lautsymbol ab. Aus dem Sprachsignal erhalten wir so eine

Folge von diskreten Werten. Diese entspricht der Ausgabe des diskreten HMM.

Den Informationsverlust der Vektorquantisierung vermeidet man beim kontinuierlichen HMM (CHMM). Von den HMM-Zuständen werden Merkmalvektoren ausgegeben. Die Ausgabeverteilungen sind zustandsspezifisch und werden durch Gaußsche Mischungsverteilungen modelliert. Diese K überlagerten Normalverteilungen kann man als Kodebuch mit K Klassen interpretieren. Jeder HMM-Zustand besitzt sein eigenes Kodebuch. Mit dem CHMM erreicht man bessere Erkennungsraten als mit dem diskreten HMM, allerdings nur, wenn ausreichend Trainingsmaterial zur Verfügung steht, um die Vielzahl von Kodebuchparametern stabil zu schätzen.

Einen Kompromiss stellt das am Anfang erwähnte semikontinuierliche HMM (SCHMM) dar. Wieder werden vom HMM Merkmalvektoren ausgegeben, es steht aber diesmal nur ein einziges Kodebuch zur Verfügung. Dieses wird von allen Zuständen geteilt, jedoch werden die einzelnen Normalverteilungen mit zustandsspezifischen Gewichten multipliziert. Bei wenig Trainingsmaterial werden beim SCHMM bessere Erkennungsraten erzielt; mit hinreichend vielen Daten ist das kontinuierliche HMM aber besser.

Im ersten Teil dieser Diplomarbeit wird der Spracherkenner, der auf SCHMM basiert, so erweitert, dass verschiedene Kodebücher für unterschiedliche Typen von Merkmalen (z.B. statische und dynamische Merkmale) verwendet werden. Wenn ein Zustand eine Ausgabe erzeugt, müssen alle Kodebücher gleichzeitig berücksichtigt werden. Es wird angenommen, dass die Merkmaltypen voneinander stochastisch unabhängig sind. Da von den HMM-Zuständen die Klassen aller Kodebücher individuell gewichtet werden, und so größere Modellierungsfreiheit besteht, lässt sich dieser Ansatz als Schritt in Richtung kontinuierliches HMM interpretieren. Er ist beispielsweise im SPHINX-System implementiert.

Auch im zweiten Teil der Diplomarbeit geht man einen Schritt Richtung CHMM. So erhofft man sich, die Erkennungsraten zu erhöhen. Wieder werden SCHMM mit mehreren Kodebüchern implementiert, jedoch teilen sich nun Zustände derselben Lautoberklasse je ein Kodebuch. In der Literatur findet man zahlreiche ähnliche Ansätze. Im "*tied mixture model*" steht ein Kodebuch mit sehr vielen Dichten zur Verfügung. Die Ausgabeverteilungen der einzelnen Zustände werden aber nur durch eine bestimmte Teilmenge der Dichten modelliert. In ähnlicher Weise betrachten wir die Kodebücher der verschiedenen Lautoberklassen als ein einziges. Jeder Zustand verwendet eine feste Anzahl bestimmter Dichten; jede Kodebuchklasse wird aber auch nur von Zuständen derselben Oberklasse verwendet.

Der Großteil der Untersuchungen erfolgt mit einer Teilmenge der EVAR-Stichprobe. Die Trainingsstichprobe umfasst 4999 Äußerungen, die Validierungsstichprobe 441 und die Teststichprobe 1998. In den Untersuchungen zur Hallrobustheit werden die VERBMOBIL-

Stichprobe und Daten aus dem Müdigkeitsexperiment verwendet (s.u.). Jeder Erkenner wird mit dem ISADORA-System mit zehn Kodebuchneuschätzungen trainiert und anschließend getestet. Dazu wird der LRBEAM-Erkenner verwendet. Im Referenzsystem mit 256 Kodebuchklassen werden mit 24-dimensionalen Merkmalvektoren (12 statische und 12 dynamische Merkmale) 74.4 % WA erzielt. Mit 512 Klassen erreichen wir keine signifikante Verbesserung.

In den Untersuchungen mit mehreren Kodebüchern für unterschiedliche Merkmaltypen hat sich das heuristische Vorgehen bewährt, die Kodebücher mit Kodebuchexponenten α zu gewichten ($0 \leq \alpha \leq 1$). Das wichtigere Kodebuch erhält einen höheren Exponenten. Im Experiment mit zwei Kodebüchern mit jeweils 128 Klassen für 12 statische bzw. 12 dynamische Merkmale zeigt sich, dass das Kodebuch für die Ableitungen wichtiger ist. Zunächst trainieren wir ohne Gewichte ($\alpha = 1$) und optimieren die Kodebuchexponenten zum Testen auf der Validierungsstichprobe. Auf der Teststichprobe erreichen wir so eine Wortakkuratheit von 76.9 %.

Mit dem Erhöhen der Klassenzahl steigen auch die Erkennungsraten. Mit 2×256 Klassen werden 77.4 % WA erreicht. Die Anzahl der Kodebuchparameter ist nur ein Drittel derer im Referenzsystem mit 512 Klassen; die Anzahl der Zustandsgewichte ist in beiden Fällen identisch.

Nun wird erneut das System mit 2×128 Klassen betrachtet. Die Kodebücher werden bereits im Training mit den optimalen Exponenten aus obigem Versuch gewichtet. Auf der Teststichprobe werden 77.9 % WA erreicht. Dies entspricht im Vergleich mit dem Referenzsystem mit 256 Klassen einer relativen Reduzierung der Fehlerrate um 12 %.

In den Versuchen mit drei und mehr Kodebüchern erfolgt das Training aus Zeitgründen nur ungewichtet. Mit drei Kodebüchern für die Energie und deren Ableitung, 11 MFCCs sowie deren 11 Ableitungen und 3×100 Klassen wird die Zahl der Kodebuchparameter weiter drastisch gesenkt. Die Erkennungsraten entsprechen trotzdem denen aus dem 2-Kodebuch-Experiment.

Wird zu den beiden Kodebüchern für statische und dynamische Merkmale ein drittes mit zweiten Ableitungen hinzugenommen können keine Verbesserungen erzielt werden. Evtl. müssen die zweiten Ableitungen über kleinere Zeitfenster berechnet werden. Auch erste Ableitungen für verschiedene Zeitauflösungen bringen keinen signifikanten zusätzlichen Erfolg. Hier ist die Unabhängigkeitsannahme für die verschiedenen Merkmaltypen verletzt.

Es wird nun die Robustheit des Ansatzes mit zwei Kodebüchern gegenüber Hall getestet. Der Erkenner wird nun mit einer Teilmenge der VERBMOBIL-Stichprobe von 940 Dateien trainiert. Im Rahmen des Müdigkeitsexperiments wurden Verbmobiltexte gesprochen und sowohl mit einem Nahbesprechungsmikrophon als auch mit einem Raummikrophon aufgezeichnet. Mit zwei Kodebüchern für statische und dynamische Merkmale werden mit der nichtverrauschten Stichprobe wieder deutliche Verbesserungen erzielt, nicht jedoch mit den Daten des Raummi-

krophons. Die Robustheit gegenüber Hall konnte nicht gesteigert werden.

Im zweiten Teil der Diplomarbeit werden Kodebücher für 47 Lautoberklassen trainiert. Es wird wieder die EVAR-Stichprobe herangezogen. Jeder HMM-Zustand verwendet genau ein Kodebuch, das mit einer etikettierten Stichprobe erstellt wird. In ersten Untersuchungen besitzt jedes Kodebuch 25 Klassen. Die Etikettierung erhalten wir aus dem Referenzsystem, mit dem für jede Sprachdatei die wahrscheinlichste Zustandsfolge berechnet wird.

Werden die Kodebücher zu einem einzigen zusammengefasst und ein Erkenner trainiert, so erzielen wir ohne Kodebuchneuschätzung 77.6 % WA. Nach der Neuschätzung verschlechtert sich die Erkennung, was wohl daran liegt, dass zu wenig Trainingsdaten für diese große Anzahl von Kodebuchklassen verwendet werden. Verschmilzt man ähnliche Klassen, so erhalten wir mit 512 Kodebuchklassen 76.3 % WA und verbessern uns nach einer Kodebuchneuschätzung auf 76.8 % WA. Im Vergleich mit dem Referenzsystem nimmt die Fehlerrate um 9 % relativ ab.

Nun werden 47 Kodebücher à 25 Klassen für verschiedene Lautoberklassen betrachtet und jedem Zustand genau eines zugeordnet. Die Erkennungsraten liegen bei nur 51.9 % WA und verschlechtern sich sowohl nach der Kodebuchneuschätzung als auch für andere Klassenzahlen. Die Ursache dafür liegt wohl an der festen Klassenzahl pro Kodebuch; das System ist so sehr unflexibel. In einem rein datengetriebenen Optimierungsverfahren konnte nachgewiesen werden, dass für Zustände verschiedener Oberklassen eine sehr unterschiedliche Anzahl von Kodebuchklassen favorisiert wird. In künftigen Untersuchungen sollten flexible Klassenzahlen implementiert werden. Zudem muss eine größere Stichprobe verwendet werden, um die große Anzahl von Parametern stabil zu schätzen. Auch sollten Untersuchungen mit weniger Lautoberklassen unternommen werden. Aus Zeitgründen konnten keine weiteren Versuche dazu durchgeführt werden. Die Erfolge, die durch das Betrachten von Lautoberklassen in Experimenten mit einem Kodebuch erzielt wurden, sollten aber zu weiterer Forschung motivieren.

Die Untersuchungen mit Kodebüchern für verschiedene Merkmaltypen müssen noch fortgesetzt werden. Der Ansatz mit drei Kodebüchern für Energie, MFCC und Ableitungen erscheint sehr vielversprechend. Es ist hier noch wie im 2-Kodebuch-Ansatz die Klassenzahl zu variieren. Auch gewichtete Kodebücher beim Training erscheinen sinnvoll. Zustandsspezifische Kodebuchgewichte brachten in der Literatur bereits Erfolge.

Insgesamt konnte in dieser Arbeit durch die Optimierung des Schwellwertes bei der Vektorquantisierung, mehrere Kodebücher für verschiedene Merkmaltypen und Optimierung der Kodebuchexponenten in der Trainings und Testphase die Fehlerrate um 16 % reduziert werden. Durch eine Kombination mit den Erfolgen des phonetischen Mischungsverteilungsansatzes können nach weiter Forschung zusätzliche Verbesserungen erzielt werden.

Anhang A

Berechnung von $\zeta_t(j,k)$

Zur Berechnung von $\zeta_t(j,k)$ gibt es für semikontinuierliche HMM die beiden Berechnungsmöglichkeiten aus Gleichung 2.36

$$\begin{aligned}\zeta_t(j,k) &= P(q_t = s_j, k_t = k|\boldsymbol{C}, HMM) \\ &= \frac{P(q_t = s_j, k_t = k, \boldsymbol{C}|HMM)}{P(\boldsymbol{C}|HMM)}\end{aligned}$$

sowie aus Gleichung 2.41

$$\begin{aligned}\zeta_t(j,k) &= P(q_t = s_j, k_t = k|\boldsymbol{C}, HMM) \\ &= P(q_t = s_j|\boldsymbol{C}, HMM)P(k_t = k|q_t = s_j, \boldsymbol{C}, HMM) \\ &= \gamma_t(j) \cdot \frac{g_k(\boldsymbol{c}_t)b_{jk}}{\sum_{\kappa=1}^{K} g_k(\boldsymbol{c}_t)b_{jk}}\end{aligned}$$

mit $\gamma_t(i) = P(q_t = s_i|\boldsymbol{o}, HMM)$

Der Beweis für die zweite Formel ist nachfolgend ausgeführt. Um die Übersichtlichkeit zu wahren wird die Bedingung HMM bei den Wahrscheinlichkeiten $p(\cdot|HMM)$ weggelassen.

$$
\begin{aligned}
\zeta_t(j,k) &= P(q_t = s_j, k_t = k|\boldsymbol{C}) \\
&= P(q_t = s_j|\boldsymbol{C})P(k_t = k|q_t = s_j, \boldsymbol{C}) \\
&= \gamma_t(j) \cdot P(k_t = k|q_t = s_j, \boldsymbol{C}) \\
&= \gamma_t(j) \cdot \frac{P(k_t = k, q_t = s_j, \boldsymbol{C})}{P(q_t = s_j, \boldsymbol{C})} \\
&= \gamma_t(j) \cdot \frac{P(k_t = k, q_t = s_j, \boldsymbol{c}_t)P(\boldsymbol{c}_1\boldsymbol{c}_2...\boldsymbol{c}_{t-1}\boldsymbol{c}_{t+1}...\boldsymbol{c}_T)}{\sum_{\kappa=1}^{K} P(k_t = k, q_t = s_j, \boldsymbol{c}_t)P(\boldsymbol{c}_1\boldsymbol{c}_2...\boldsymbol{c}_{t-1}\boldsymbol{c}_{t+1}...\boldsymbol{c}_T)} \quad \text{(*)} \\
&= \gamma_t(j) \cdot \frac{P(k_t = k, q_t = s_j, \boldsymbol{c}_t)}{\sum_{\kappa=1}^{K} P(k_t = k, q_t = s_j, \boldsymbol{c}_t)} \\
&= \gamma_t(j) \cdot \frac{P(\boldsymbol{c}_t|k_t = k, q_t = s_j)P(k_t = k|q_t = s_j)P(q_t = s_j)}{\sum_{\kappa=1}^{K} P(\boldsymbol{c}_t|k_t = k, q_t = s_j)P(k_t = k|q_t = s_j)P(q_t = s_j)} \\
&= \gamma_t(j) \cdot \frac{P(\boldsymbol{c}_t|k_t = k, q_t = s_j)P(k_t = k|q_t = s_j)}{\sum_{\kappa=1}^{K} P(\boldsymbol{c}_t|k_t = k, q_t = s_j)P(k_t = k|q_t = s_j)} \\
&= \gamma_t(j) \cdot \frac{P(\boldsymbol{c}_t|k_t = k)P(k_t = k|q_t = s_j)}{\sum_{\kappa=1}^{K} P(\boldsymbol{c}_t|k_t = k)P(k_t = k|q_t = s_j)} \quad \text{(**)} \\
&= \gamma_t(j) \cdot \frac{g_k(\boldsymbol{c}_t)b_{jk}}{\sum_{\kappa=1}^{K} g_k(\boldsymbol{c}_t)b_{jk}}
\end{aligned}
$$

In Zeile (*) wird die Unabhängigkeit zeitlich benachbarter Merkmalvektoren angenommen; Voraussetzung für Zeile (**) ist, dass das Erzeugen eines Vektors $\boldsymbol{c}_t$ durch die Mischungskomponente k des Kodebuches unabhängig vom Zustand q_t ist. Beides gilt für semikontinuierliche HMM.

Verzeichnis der Bilder

1.1 Statistisches Modell der Sprachproduktion für das Wort *'haben'*. Aus [ST95, S.126] 16
1.2 Wortfehlerrate verschiedener HMM-Typen in Abhängigkeit von der Größe der Stichprobe. Aus [Ace01, S.440] 18

2.1 Struktur eines Klassifikationssystems. Nach: [Nie83, S.14] 21
2.2 Spektrogramm des Wortes "Bahnhof" 27
2.3 Diskretes Hidden-Markov Modell 32
2.4 Kontinuierliches Hidden-Markov Modell 34
2.5 Semikontinuierliches Hidden-Markov Modell 36

3.1 Semikontinuierliches Hidden-Markov Modell mit zwei orthogonalen Kodebüchern 41
3.2 Beliebige Bindungen zwischen Zuständen und Kodebuchklassen bei HMM mit mehreren Kodebüchern 46
3.3 Drei Kodebücher für verschiedene Lautoberklassen mit je drei Normalverteilungskomponenten 47

4.1 Architektur des ISADORA-Systems, aus [ST95, S.279] 54
4.2 Konvergenz der Bewertung b beim Training 56
4.3 Durchschnittliche Anzahl von Klassen mit Bewertung unter einer Schwelle mit initialem Kodebuch (0) und den zehn Neuschätzungen im Referenzsystem 58

5.1 Beispiele für die Funktion $y = x^{\alpha}$ für verschiedene Kodebuchexponenten α. Die durchgezogene Linie ist die Identität $y = x$ 63
5.2 Die Funktion $x^{0.3} \cdot y^{0.7}$ 64
5.3 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.1 66

5.4 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale α_{stat}. Das andere Kodebuch ist abgeschaltet ($\alpha_{dyn} = 0$). 67
5.5 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für dynamische Merkmale α_{dyn}. Das andere Kodebuch ist abgeschaltet ($\alpha_{stat} = 0$). 67
5.6 Testen des Erkenners mit 2×100 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.6 71
5.7 Testen des Erkenners mit 2×200 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.7 71
5.8 Testen des Erkenners mit 2×256 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Vgl. Tabelle 5.8 72
5.9 Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\, train} = \alpha_{dyn,\, train} = 0.5$ trainiert. Vgl. Tabelle 5.10 . 73
5.10 Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\, train} = 0.3$ und $\alpha_{dyn,\, train} = 0.7$ trainiert. Vgl. Tabelle 5.11 75

6.1 Anzahl der wichtigen Dichten für verschiedene Lautoberklassen. Es gilt $c_{jk} > 0.001$ 95

Verzeichnis der Tabellen

4.1 Variation von θ_{test} und resultierende Wortakkuratheit beim Referenzsystem mit 256 Klassen 57

4.2 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 512 Klassen 59

4.3 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 256 Klassen, der nur die 12 *statischen* Merkmale verwendet 59

4.4 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit 256 Klassen, der nur die 12 *dynamischen* Merkmale verwendet 60

5.1 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale. Summe der Kodebuchexponenten ist Eins. 65

5.2 Testen des Erkenners ohne Kodebuchexponenten einmal mit der Validierungsstichprobe und einmal mit der Teststichprobe: Erst wird das zweite Kodebuch ausgeschaltet, dann das erste und zuletzt werden beide Kodebücher ungewichtet betrachtet. 66

5.3 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale. Die Summe der Kodebuchexponenten ist beliebig. 68

5.4 Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal (vgl Tabellen 5.3 und 5.1). Es wurden zwei Kodebücher mit je 128 Klassen trainiert. 68

5.5 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{stat} = 0.3$ und $\alpha_{dyn} = 0.7$. 70

5.6 Testen des Erkenners mit 2×100 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$. . . . 70

5.7 Testen des Erkenners mit 2×200 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$. . . . 71

5.8 Testen des Erkenners mit 2×256 Klassen auf der Validierungsstichprobe für verschiedene Kodebuchexponenten α_{stat} und α_{dyn}. Es gilt $\alpha_{stat} + \alpha_{dyn} = 1$. . . . 71

5.9 Testen der Erkenner mit mit verschieden vielen Klassen auf der Teststichprobe. Die Kodebuchexponenten sind jeweils auf der Validierungsstichprobe optimal (vgl. Tabellen 5.6, 5.7 und 5.8). 72

5.10 Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\, train} = \alpha_{dyn,\, train} = 0.5$ trainiert. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten. 73

5.11 Testen des Erkenners auf der Validierungsstichprobe für verschiedene Kodebuchexponenten für statische Merkmale α_{stat}. Es gilt $\alpha_{dyn} = 1 - \alpha_{stat}$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\, train} = 0.3$ und $\alpha_{dyn,\, train} = 0.7$ trainiert. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten. 75

5.12 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{stat} = 0.45$ und $\alpha_{dyn} = 0.55$. Der Erkenner wurde mit Gewichten $\alpha_{stat,\, train} = 0.3$ und $\alpha_{dyn,\, train} = 0.7$ trainiert. 75

5.13 Testen der Kodebücher für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe. 78

5.14 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen. 78

5.15 Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden drei Kodebücher für Energiemerkmale, Cepstrum-Koeffizienten sowie deren Ableitungen mit je 100 Klassen trainiert. 79

5.16 Variation von θ_{test} und resultierende Wortakkuratheit beim Erkenner mit Kodebuchexponenten $\alpha_{en} = 0.3$ und $\alpha_{ceps} = 0.3$ und $\alpha_{abl} = 0.75$. 79

5.17 Testen der Kodebücher für statische Merkmale, Ableitungen und zweite Ableitungen einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe. 80

5.18 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale, Ableitungen und zweite Ableitungen. . 81
5.19 Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden drei Kodebücher für statische Merkmale, Ableitungen und zweite Ableitungen mit je 100 Klassen trainiert. . . 82
5.20 Separates Testen der Kodebücher für statische Merkmale und Ableitungen in verschiedenen Zeitauflösungen. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe. 82
5.21 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische und dynamische Merkmale in verschiedenen Zeitauflösungen. Summe der Kodebuchexponenten ist Eins. Rechts das Ergebnis auf der Teststichprobe mit optimalen Gewichten. 83
5.22 Testen der Kodebücher für statische Merkmale und Ableitungen über 30. 50 und 70 ms einzeln. Links sind die Versuche auf der Validierungsstichprobe durchgeführt, rechts auf der Teststichprobe. 83
5.23 Testen des Erkenners mit der Validierungsstichprobe und verschiedenen Kodebuchexponenten für statische Merkmale, sowie Ableitungen aus 30 ms, 50 ms und 70 ms Zeitauflösungen. 84
5.24 Testen des Erkenners mit der Teststichprobe. Die Kodebuchexponenten sind auf der Validierungsstichprobe optimal. Es wurden vier Kodebücher für statische Merkmale und Ableitungen aus 30 ms, 50 ms und 70 ms Zeitfenstern mit je 128 Klassen trainiert. 84
5.25 Erkennungsraten mit Raummikrophon und unverhallten Nahbesprechungsmikrophon mit verschiedenen Erkennern. α_{stat} und α_{dyn} sind die Kodebuchexponenten. 86
5.26 Wichtiger Experimente aus diesem Kapitel. Angegeben sind die Kodebuchparameter in Tausend, Anzahl der Klassen pro Kodebuch, Anzahl der Kodebücher, Exponenten für Kodebücher K1 - K4, Schwellwert bei der Vektorquantisierung und Wortakkuratheit in % 88

6.1 81 elementare Laute unterteilt mit 47 phonemischen Basiseinheiten 91
6.2 Erkennungsraten mit einem phonetischen Mischungsverteilungskodebuch vor der ersten Kodebuchneuschätzung und nach einer weiteren Trainingsrunde 93
6.3 Erkennungsraten mit mehreren Kodebüchern für verschiedene Lautoberklassen vor der ersten Kodebuchneuschätzung 94

Literaturverzeichnis

[Ace01] Acero, A.; Huang, X.; Hon, H.: *Spoken Language Proceeding*, Prentice Hall, 2001.

[Bak75] Baker, J.: *The DRAGON System - an Overview*, *IEEE Trans. on Acoustics, Speech and Signal Processing*, Bd. 23, 1975, S. 24–29.

[Dem77] Dempster, A.; Laird, N.; Rubin, D.: *Maximum Likelihood from Incomplete Data via the EM Algorithm*, *J. Royal Statistical Society*, Bd. 39, Nr. 1, 1977, S. 1–38.

[Dig96] Digalakis, V.; Monaco, P.; Murveit, H.: *Genones: Generalized Mixture Tying in Continuous Hidden Markov Model Based Speech Recognizers*, 1996.

[Gal98] Gallwitz, F.; Aretoulaki, M.; Boros, M.; Haas, J.; Harbeck, S.; Huber, R.; Niemann, H.; Nöth, E.: *The Erlangen Spoken Dialogue System EVAR: A State–of–the–Art Information Retrieval System*, in *Proceedings of 1998 International Symposium on Spoken Dialogue (ISSD 98)*, Sydney, Australia, 1998, S. 19–26.

[Hac01] Hacker, C.: *Optimierung der Merkmalberechnung für die automatische Spracherkennung*, Studienarbeit, Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen–Nürnberg, 2001.

[Had02] Haderlein, T.: *Using the ISADORA System for Analysing Fatigue Symptoms and Robustness of Features against Reverberation*, Internal Report, Lehrstuhl für Multimediakommunikation und Signalverarbeitung (Institut für Elektrotechnik, Elektronik und Informationstechnik) und Lehrstuhl für Mustererkennung (Institut für Informatik), Universität Erlangen–Nürnberg, 2002.

[Hua89] Huang, X.; Jack, M.: *Semi–Continuous Hidden Markov Models for Speech Signals*, *Computer Speech & Language*, Bd. 3, Nr. 3, 1989, S. 239–251.

[Hua90] Huang, X.; Lee, K.; Hon, H.: *On Semi-Continous Hidden Markov Modeling*, in *Proc. ICASSP '90*, 1990, S. 689–692.

[Hua91] Huang, X.; Lee, K.; Hon, H.; Hwang, M.: *Improved Acoustic Modeling with the SPHINX Speech Recognition System*, in *Proc. ICASSP '91*, 1991, S. 345–348.

[Hua93] Huang, X.; Alleva, F.; Hon, H.-W.; Hwang, M.-Y.; Lee, K.-F.; Rosenfeld, R.: *The SPHINX-II Speech Recognition System: An Overview*, *Computer Speech and Language*, Bd. 7, Nr. 2, 1993, S. 137–148.

[Jel76] Jelinek, F.: *Continuous Speech Recognition by Statistical Methods*, *Proceedings of the IEEE*, Bd. 64, Nr. 4, 1976, S. 532–556.

[Lev66] Levensthein, V.: *Binary Codes Capable of Correcting Deletions Insertions and Reversals*, *Cybernetics and Control Theory*, Bd. 10, 1966, S. 707–710.

[Lin80] Linde, Y.; Buzo, A.; Gray, R.: *An Algorithm for Vector Quantizer Design*, *IEEE Trans. on Communications*, Bd. 28, Nr. 1, 1980, S. 84–95.

[Nie83] Niemann, H.: *Klassifikation von Mustern*, Springer–Verlag, Berlin, 1983.

[Nie90] Niemann, H.: *Pattern Analysis and Understanding*, Bd. 4 von *Springer Series in Information Sciences*, Springer, Heidelberg, 1990.

[Nöt01] Nöth, E.; Boros, M.; Fischer, J.; Gallwitz, F.; Haas, J.; Huber, R.; Niemann, H.; Stemmer, G.; Warnke, V.: *Research Issues for the Next Generation Spoken Dialogue Systems Revisited*, in Matoušek, V.; Mautner, P.; Mouček, R.; Taušer, K. (Hrsg.): *Proc. 4th International Conference on Text, Speech and Dialogue (TSD 2001)*, Bd. 2166 von *Lecture Notes for Artificial Intelligence*, Springer–Verlag, Berlin, September 2001, S. 341 – 348.

[Rie94] Rieck, S.: *Parametrisierung und Klassifikation gesprochener Sprache*, Dissertation, Technische Fakultät der Universität Erlangen–Nürnberg, 1994.

[Rog94] Rogina, I.; Waibel, A.: *Learning State-Dependent Stream Weights for Multi-Codebook HMM Speech Recognition Systems*, in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Bd. 1, Adelaide, Australia, 1994, S. 217–220.

[Sch95] Schüssler, M.: *Realisierung von Sprecheradaptionsverfahren für ein sprecherunabhängiges Spracherkennungssystem*, Diplomarbeit, Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen–Nürnberg, 1995.

[ST93a] Schukat-Talamazzini, E.; Bielecki, M.; Niemann, H.; Kuhn, T.; Rieck, S.: *A Non–Metrical Space Search Algorithm for Fast Gaussian Vector Quantization*, in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, 1993.

[ST93b] Schukat-Talamazzini, E.; Niemann, H.: *ISADORA — A Speech Modelling Network Based on Hidden Markov Models*, University Erlangen–Nuremberg, 1993, http://www.minet.uni-jena.de/www.fakultaet/schukat/MYPUB/SchukatTalamazzini93:IAS.ps.

[ST95] Schukat-Talamazzini, E. G.: *Automatische Spracherkennung – Grundlagen, statistische Modelle und effiziente Algorithmen*, Vieweg, Braunschweig, 1995.

[Ste02] Stemmer, G.; Nöth, E.; Niemann, H.; Zeissler, V.; Hacker, C.: *A Phone Recognizer helps to Recognize words better*, to appear, 2002.

[Wil97] Willett, D.; Rigoll, G.: *A New Approach to Generalized Mixture Tying for Continous HMM-Based Speech Recognition*, in *5th European Conference on Speech Communication and Technology*, Rhodes, Greece, 1997, S. 1175–1178.

[You92] Young, S.: *The General Use of Tying in Phoneme-Based HMM Speech Recognisers*, in *Proc. ICASSP '92*, 1992, S. 569–572.