Graz University of Technology

Automatische Spracherkennung für Fluglotsen

Masterarbeit

an der

Technischen Universität Graz

eingereicht von

Mario Ackerl

am Insitut für Signalverarbeitung und Sprachkommunikation,
Inffeldgasse 12, A-8010 Graz, Austria

September 2010

Begutachter: o.Univ.-Prof. Dipl.-Ing. Dr. techn. Gernot Kubin
Betreuer: Dipl.-Ing. Dr. techn. Stefan Petrik

Kurzfassung

In dieser Arbeit wird untersucht, ob automatische Spracherkennung in der Flugverkehrssicherung als Unterstützung für Fluglotsen bei der Dokumentation ihrer Arbeit verlässlich eingesetzt werden kann. Trotz der einfachen Kommandosprache, die im Funkverkehr verwendet wird, haben frühere Untersuchungen den Einsatz dieser Technologie bisher nicht empfehlen können. Anhand einer Sprachdatenbank von realistischen Flugverkehrskommunikationen werden nun aktuelle Lösungsansätze für die bekannten Probleme wie Sprecheradaption, statistische Sprachmodellierung oder diskriminatives Training evaluiert.

Dazu wurde auf der Basis von verfügbaren Werkzeugen ein Framework zum Training und Testen eines triphon-basierten Spracherkenners implementiert, das die Kontrolle über alle Stufen der statistischen Modellierung erlaubt und mit beliebigen Sprachdatenbanken verwendet werden kann. Das Framework unterstützt verschiedene Trainingsstrategien wie Flat-start oder Bootstrapping, statistische Uni-, Bi-, und Trigramm-Sprachmodelle, Sprecheradaption und diskriminatives akustisches Training. Die besten Resultate wurden mit einer Kombination aus Bootstrapping, Sprecheradaption und einem statistischen Bigramm-Sprachmodell erzielt. Insgesamt wurde im Mittel über 10 Sprecher eine Worterkennungsrate von 95% und eine Satzerkennungsrate von 70% erzielt. Dieses Ergebnis ist eine deutliche Verbesserung im Vergleich zu früheren Untersuchungen auf derselben Sprachdatenbank und ein großer Schritt hin zur tatsächlichen Einsetzbarkeit in der Praxis.

Abstract

The goal of this thesis is to investigate whether automatic speech recognition can reliably be used in air traffic control as support for air traffic controllers in documenting their work. Despite the simple structure of the air traffic control language, earlier studies could not recommend this technology to the authorities so far. In this work, currrent solutions to the known problems such as speaker adaptation, statistical language models or discriminative training are evaluated on a database of realistic air traffic control recordings.

A framework for training and evaluation of a triphone-based speech recognizer was implemented based on available toolkits. This framework can be used with different speech databases and provides control over all stages of statistical modelling. Furthermore, it supports flat-start and bootstrapping training strategies, statistical uni-, bi-, and trigram language models, speaker adaptation, and discriminative acoustic training. The best results were achieved with a combination of bootstrapping, speaker adaptation, and a bi-gram language model. A word recognition rate of 95% and a sentence recognition rate of 70% were measured averaged over 10 speakers. This result is a significant improvement compared to earlier studies performed on the same database and a big step forward towards applicability in air traffic control.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Mario Ackerl

Digital unterschrieben von Mario Ackerl
DN: cn=Mario Ackerl, o, ou, email=mario.ackerl@gmail.com, c=AT
Datum: 2010.09.11 22:01:32 +02'00'

Graz, am 11. September 2010 (Unterschrift)

Danksagung

Die Diplomarbeit ist die interessanteste Zeit des Studiums. So kann man endlich sehen, ob sich die letzten Jahre des Studierens gelohnt haben. Ich möchte mich an dieser Stelle bei den Personen bedanken, die mir bei der Erarbeitung dieser Diplomarbeit geholfen haben.

Bei Herrn Prof. Gernot Kubin bedanke ich mich für die Vergabe und Begutachtung der Diplomarbeit.

Mein herzlicher Dank gilt Herrn Dr. Stefan Petrik, der mich durch seine engagierte Betreuung und stete Diskussionsbereitschaft mit vielseitigen Denkanstößen bereichert und bei der Erstellung dieser Arbeit unterstützt hat.

Des weiteren danke ich Wilfried Mittendrein für die orthographische Korrektur der Arbeit

Besonders möchte ich mich bei meiner Freundin Barbara bedanken, denn aus ihrer Liebe konnte ich immer wieder Kraft und Motivation schöpfen, wenn es mal nicht so rund gelaufen ist.

Nicht zuletzt möchte ich mich bei meinen Eltern bedanken, denn ohne ihre finanzielle und moralische Unterstützung wäre dieses Studium nie möglich gewesen.

Danke!

Mario Ackerl
Graz, September 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ASR	Automatic Speech Recognition
ATC	Air Traffic Control
ATCOSIM	Air Traffic Control Simulation Speech Corpus
CART	Classification And Regression Tree
CFG	Context Free Grammar
CYK	Cocke-Younger-Kasami-Algorithmus
DFT	Diskrete Fourier Transformation
DTW	Dynamic Time Warping
EM	Expectation-Maximisation (algorithm)
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
ICAO	International Civil Aviation Organization
IPA	International Phonetic Alphabet
LVCSR	Large Vocabulary Continuous Speech Recognition
MAP	Maximum A-Posteriori
MFCC	Mel Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MLLR	Maximum Likelihood Linear Regression
NLP	Natural Language Processing
OOV	Out-Of-Vocabulary
PCFG	Probabilistic Context Free Grammar
PM	Pronunciation Model
PV	Pronunciation Variant
SAMPA	Speech Assessment Methods Phonetic Alphabet
WER	Word Error Rate

1. Einleitung

> Verständnis ist nichts anderes als die durch die Sprache verursachte Wahrnehmung.
>
> *(Thomas Hobbes)*

Computer und Roboter, die sich in natürlicher Sprache mit den Menschen unterhalten und alle Sprachen beherrschen, sind schon lange fixer Bestandteil der allermeisten Science Fiction Filme. Parallel zur Science Fiction wird an diesem Thema auch schon seit den 60er Jahren weltweit intensiv geforscht. Obwohl wir vom Ziel des perfekten Spracherkenners für alle Sprecher und Sprachen in jeder beliebigen Umgebung noch weit entfernt sind, hat es doch stetig große Fortschritte gegeben, sodass heutzutage für bestimmte Anwendungen konzipierte und trainierte Spracherkennungssysteme sehr gute Erkennungsraten von über 90% erreichen können. Ein Beispiel dafür wäre der COST 249 SpeechDat Multilingual Reference Recogniser, der dies für alle SpeechDat Datenbanken erreicht. [1] Die vorliegende Arbeit konzentriert sich auch auf einen Teilbereich der menschlichen Kommunikation: nämlich darauf, die von Fluglotsen verwendete Sprache gut erkennen zu können. Im 1. Kapitel wird die Aufgabenstellung näher erläutert. Kapitel 2 umreißt danach kurz die theoretischen Grundlagen, die nötig sind, um ein automatisches Spracherkennungssystem (ASR) basierend auf Hidden Markov Modellen (HMMs) zu konstruieren. Daraus ergibt sich dann das Design des für diese Arbeit konstruierten ASR Systems, welches in Kapitel 3 vorgestellt wird. Kapitel 4 ist der experimentelle Teil der Arbeit, indem der Weg der Optimierung eines ASR Systems für diese Problemstellung näher erläutert wird. Dabei werden alle Parameter so angepasst, dass eine sehr gute Erkennung erzielt werden kann.

1.1. Automatische Spracherkennung in der Flugsicherung

Das Thema der vorliegenden Arbeit ist der Entwurf eines automatischen Spracherkennungssystems (ASR), das von Fluglotsen in der Flugverkehrssicherheit eingesetzt werden soll.

Konkret sieht das Szenario folgendermaßen aus: Die ICAO (zuständig für die weltweite Vereinheitlichung der Regelungen für die Luftfahrt) sieht vor, dass die Fluglotsen (Air Traffic Controller) in einem Kontrollraum (Tower) arbeiten. Ein Fluglotse ist dafür zuständig, dass alle Flugzeuge einen genügend großen Sicherheitsabstand zueinander aufweisen. Dazu ist ein Team von 2-3 Fluglotsen für einen bestimmten Sektor (Teil des Luftraums) zuständig. Sie sind im ständigen Kontakt mit den Piloten der Flugzeuge. Diese Kommunikation läuft aus Sicherheitsgründen über einen amplitudenmodulierten Funkkanal, wodurch Funksprüche auch bei schlechten Signal-Rausch-Verhältnis (zum Beispiel durch atmosphärische Störungen verursacht) noch verstanden werden können. Dabei ist die Qualität allerdings schlechter als bei Frequenzmodulation, weshalb sich der Flugfunk für ungeübte Hörer unverständlich anhört. [2]

Um eine effiziente und weltweit einheitliche Kommunikation möglich zu machen, werden die Fluglotsen speziell darauf geschult eine von der ICAO genau definierte Kontrollsprache zu verwenden. [3] Die Sprache der Flugverkehrssicherheit ist Englisch. Es ist nicht das Ziel

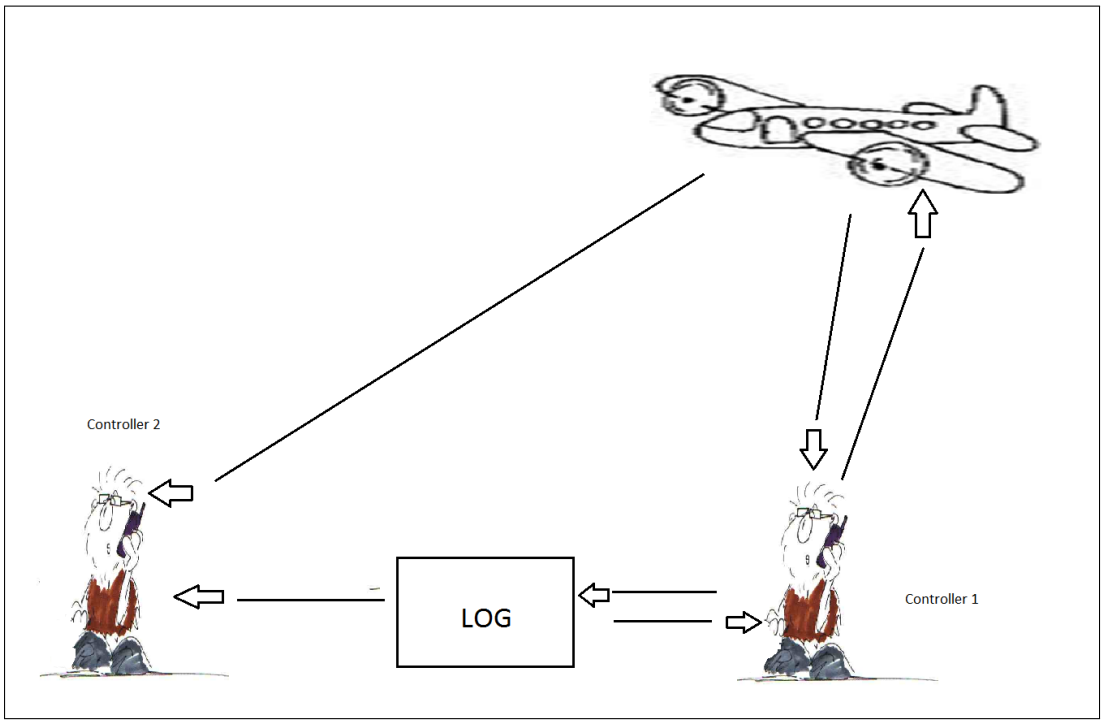

Abbildung 1.1.: Kommunikation zwischen Pilot und Fluglotsen in der Luftfahrt

dieser Arbeit, den gesamten Funkverkehr zwischen Piloten und Fluglotse mit automatischer Spracherkennung zu erfassen, sondern es geht nur um einen Teilbereich. Nämlich die vom Fluglotsen durchgegebenen Kommandos mit automatischer Spracherkennung zu erfassen und zu protokollieren (siehe Abbildung 1.1). Das vereinfacht die Aufgabenstellung, da man bei der Spracherkennung nicht mit dem Funkkanal arbeiten muss, sondern der Fluglotse spricht ja in ein Headset, sodass man hier eine hervorragende Aufnahmequalität erwarten kann. Diesen Protokollierungsschritt müssen die Fluglotsen zur Zeit manuell durchführen. Mit einer automatischen Spracherkennung wird den Fluglotsen ein Arbeitsschritt abgenommen. Zusätzlich soll die Arbeit zeigen, wie gut automatische Spracherkennung mit so einem Task umgehen kann. Schließlich wird hier nicht der Dialog zwischen Fluglotsen und Pilot so angepasst, dass eine Erkennung möglichst einfach möglich ist, sondern der Erkenner muss sich an die gegebenen Bedingungen im Kontrollraum inklusive möglicher Hintergrundgeräusche anpassen. Ein sekundärer Task dieser Arbeit ist es, im Zuge des Erkennerentwurfs ein gut dokumentiertes und auf andere Anwendungen adaptierbares Erkennungssystem zu bauen.

1.1.1. Kontrollersprache

Die verwendete Sprache ist Englisch und sollte in einer ATC Umgebung nur eine Menge einfacher Kommandos enthalten. Diese werden von der ICAO genau spezifiziert. Die Struktur einer ATC Instruktion soll so aussehen, wie in Abbildung 1.2 dargestellt.

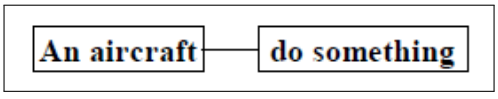

Abbildung 1.2.: Aufbau einer ATC Instruktion

Obwohl die bei der Ausbildung zum Fluglotsen gelernte ICAO Phraseologie genau spezifiziert ist, prägen Jahre der praktischen Arbeit als Fluglotse die Menschen, sodass sich ihr Verhalten mit der Zeit leicht anpasst. Deswegen ist nur ein kleiner Prozentsatz der Anweisungen eines erfahrenen Fluglotsen zu 100% mit den ICAO Empfehlungen konform.

1.2. Die ATCOSIM Sprachdatenbank simulierter Flugverkehrskommunikation

Um einen Spracherkenner für eine bestimmte Anwendung zu bauen, ist es zuerst einmal notwendig, diesen zu trainieren. Für das Training sind repräsentative Daten aus einer Sprachdatenbank notwendig. Zur Erfüllung der Aufgabenstellung einen automatischen Spracherkenner für die Flugverkehrssicherheit zu konstruieren, wurde die sogenannte ATCOSIM (Air Traffic Control Simulation Speech Corpus) Sprachdatenbank verwendet. Diese Datenbank besteht aus Sprachaufnahmen im Gesamtumfang von 10 Stunden. Diese Aufnahmen wurden im Zuge einer realitätsnahen Flugverkehrssimulation erstellt. Zu jedem Audiosample steht eine dazu passende Transkription auf Wortebene zur Verfügung. Die ATCOSIM Datenbank ist aus mehreren Gründen für diese Aufgabe besser geeignet als andere Datenbanken (NIST, HIWIRE, nnMATC, VOCALISE): Der Corpus ist öffentlich und frei erhätlich. Er stellt direct-microphone wideband Aufnahmen der operationellen Flugsicherung zur Verfügung, welche in einer realitätsnahen Simulation gemacht wurden. Des Weiteren ist der Corpus auf Äußerungsebene segmentiert: Jedes Kommando aus der Simulation entspricht einer eigenen Wave Datei mit zugehöriger Textdatei gleichen Namens.

1.2.1. Aufnahmesituation

Die Aufnahmen wurden im Air Traffic Control Raum des Eurocontrol Experimental Centre (siehe Abbildung 1.3) in Frankreich aufgenommen. Sowohl die Arbeitsbedingungen der Fluglotsen als auch die Flugverkehrs-Szenarios werden realitätsnah simuliert. Die Fluglotsen kommunizieren per Headset mit Piloten, die von einem anderen Raum aus die simulierten Flugzeuge steuern. Es arbeiten immer mehrere Fluglotsen gleichzeitig, um auch die Zusammenhänge zwischen den Kontrollsektoren zu simulieren. Die Stimme des Piloten wurde während der Simulationen nicht aufgenommen, nur die Anweisungen der Fluglotsen.

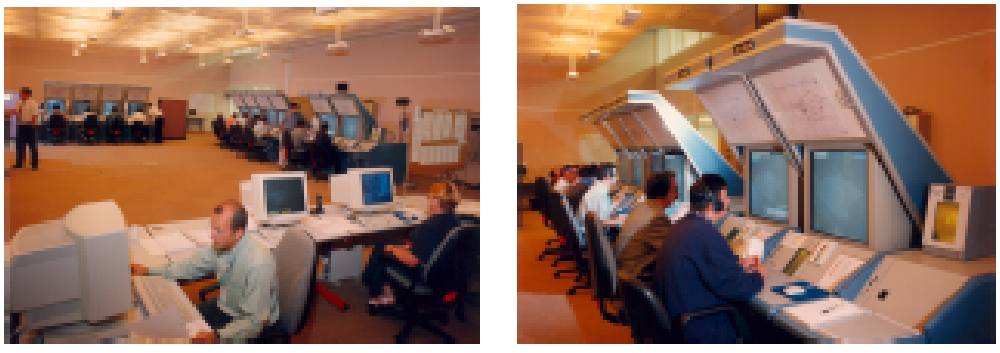

Abbildung 1.3.: Kontrollraum des EUROCONTROL Experimental Centre aus [4] mit Genehmigung des Autors

1.2.2. Aufnahme der Sprecher

Die Aufnahmen enthalten Simulationen von Luftraumsektoren in Deutschland (Söllingen) und der Schweiz (Zürich und Genf). Für maximale Realitätsnähe wurden für die Simulation Fluglotsen ausgesucht, die genau in diesen Sektoren mehrere Jahre gearbeitet hatten. Die Muttersprache der Sprecher, ist deutsch und französisch, was bei der Spracherkennung berücksichtigt werden muss. Insgesamt enthält der Corpus Sprachmaterial von 10 unterschiedlichen Sprechern, (4 Frauen und 6 Männer), siehe Tabelle 1.1 Pro Sprecher wurden zwischen 238 und 1848 Äußerungen aufgenommen.

ID	Muttersprache	Geschlecht	Sektor	#Äußerungen
sm1	Deutsch	Mann	Söllingen	1167
sm2	Deutsch	Mann	Söllingen	1848
sm3	Deutsch	Mann	Söllingen	808
sm4	Deutsch	Mann	Söllingen	1162
gf1	Schweizer Franz.	Frau	Genf	238
gm1	Schweizer Franz.	Mann	Genf	384
gm2	Schweizer Franz.	Mann	Genf	378
zf1	Schweizer Deutsch	Frau	Zürich	1716
zf2	Schweizer Deutsch	Frau	Zürich	1739
zf3	Schweizer Deutsch	Frau	Zürich	638

Tabelle 1.1.: Liste der Sprecher des ATCOSIM Corpus

Die Fluglotsen verwendeten ein Sennheiser HME 45-KA Headset. Das Sprachsignal wurde zusammen mit einem push-to-talk Signal von einem Sony DTC-60ES DAT im LongPlay Modus aufgenommen. Die Audiodaten des DAT Recorders wurden dann über eine RME Hammerfall HDSP 9632 Soundkarte mit S/PDIF auf den Computer übertragen.

1.2.3. Transkription

Anhand des Push-to-talk Signals wurden dann beim Transfer auf den Computer die Äußerungen segmentiert. Die orthographische Transkription der einzelnen Äußerungen erfolgte dabei auf Wortebene in Standard British-English, kleingeschrieben und ohne Satzzeichen.

1.2.3.1. Besonderheiten des ATCOSIM Corpus

ICAO Buchstabiertafel: alfa, bravo, charlie, delta, echo, foxtrott, golf, hotel, india, juliett, kilo, lima, mike, november, oscar, papa, quebec, romeo, sierra, tango, uniform, victor, whiskey, xray, yankee, zulu

Akronyme: k, l, m, nato, opec, icao

nicht-verbale Laute: ah hm ahm yeah aha nah ohh

Zahlen: zero, oh, one, two, three, four, five, six, seven, eight, nine, ten, hundred, thousand, decimal

nicht verifizierte Radio Kennungen (radio call sign): @aerovic, @alpha, @aviva, @bama, @cheena, @cheeseburger, @color, @devec, @foxy, @hanseli, @indialook, @ingishire, @jose, @metavec, @nafamens, @period, @roystar, @sunwing, @taitian, @tele

1.2.4. Aufbau der Sprachdatenbank (Corpus)

Der Corpus ist in 4 Unterordnern strukturiert: WAVdata enthält die Sprachaufnahmen im WAV Format, TXTdata die dazugehörigen Transkriptionen. Im Ordner HTMLdata sind die

Transkriptionen zusätzlich im HTML Format gespeichert, um auf alle Samples mit dazugehöriger Transkription über einen Browser zugreifen zu können. Der Ordner DOC enthält schließlich die Dokumentation des Corpus. [4]

- WAVdata enthält die aufgenommenen Sprachdaten. Jedes File gehört zu einer Fluglotsen Äußerung im Microsoft WAVE Format. Die Aufnahmen sind 1-Kanal Aufnahmen mit einer Samplerate von 32 kHz und einer Auflösung von 16 bit pro Sample. Insgesamt befinden sich 10,078 Audiodateien im WAVdata Verzeichnis, welche in Unterverzeichnissen verteilt sind. Es existieren insgesamt 10 Unterverzeichnisse für die 10 Sprecher im WAVdata Verzeichnis, welche jeweils noch einmal in Unterverzeichnisse für jede einzelne Session unterteilt sind. Der Dateiname setzt sich aus „speakerId_sessionId_number.wav zusammen.“ (z.B. zf2_04_010.wav)
- TXTdata enthält die Transkriptionen im plain-text 7-bit ASCII Format. Die Files entsprechen der ISO/IEC 8859-1 (ISO Latin-1) und der Unicode (UTF-8) Enkodierung. Die Verzeichnisstruktur entspricht der des WAVdata Verzeichnisses. Im Rootverzeichnis des TXTdata Ordners wird zusätzlich eine Datei namens **fulldata.csv** zur Verfügung gestellt. Diese Datei liegt im comma-separated value (CSV) Format vor und enthält die gesamten Transkriptionen, um zum Beispiel mit einem Datenbankprogramm oder Spreadsheet Programm darauf übersichtlich zugreifen zu können. Weiters liegt im Rootverzeichnis noch die Datei **wordlist.txt**, welche eine alphabetisch sortierte Liste aller auftretenden Wörter enthält.
- HTMLdata enthält die HTML Files, welche in diesem Verzeichnis vorliegen, wird nicht gearbeitet. Diese dienen nur der bequemen Darstellung im Browser Table Format. Aber der Zugriff über HTML ist praktisch, um in einzelne Audio Files reinzuhören und dazu die Transkription zu sehen.

1.3. Themenbezogene Arbeiten

Bereits im Jahr 1998 wurden im Auftrag des Eurocontrol Experimental Centre drei kommerzielle Spracherkenner anhand der ATCOSIM Datenbank auf ihre Eignung für den praktischen Einsatz in der Flugverkehrssicherheit geprüft. Durch die Automatisierungsmöglichkeiten, die ein gut funktionierendes Spracherkennungssystem ermöglicht, sollten in erster Linie Kosten bei den Simulationen gespart werden, wobei auch viele weitere Einsatzmöglichkeiten im praktischen Flugverkehr nicht ausgeschlossen wurden. Allerdings wurden, nach der deutlich unter den Erwartungen liegenden Ergebnissen der Spracherkennung, alle Pläne die Spracherkennung einzusetzen verworfen. [5] Um festzustellen, warum die Erkennungsraten so schlecht waren, werden wir uns die Experimente, die damals gemacht worden sind, genauer anschauen.

1.3.1. Spracherkennungssysteme

Die Auswahl der drei zu testenden Spracherkennungssysteme wurde durch folgende Minimalanforderungen bestimmt.

- Sprecherunabhängigkeit
- mittlere Vokabularsgröße (bis zu 500 Wörter)
- Vokabular kann durch den Benutzer angepasst werden
- Ergebnis unabhängig vom verwendeten Mikrofon

Die drei ausgewählten auf (bewährten) Hidden Markov Modellen basierenden Spracherkennungssysteme waren folgendermaßen konfiguriert:

- **Spracherkenner „A“** : ist ein Hardware Spracherkennungssystem basierend auf einer DSP Add-on Karte mit der Möglichkeit, eine Grammatik in Baumstruktur zu verwenden.
- **Spracherkenner „B“** : ist eine Software Spracherkennungssystem, wo ebenfalls eine Grammatik mit Baumstruktur verwendet werden kann.
- **Spracherkenner „C“** : ist schließlich auch eine reine Software Lösung, wo jedoch eine stochastische Grammatik verwendet werden kann.

1.3.2. Ergebnisse Eurocontrol 1998

Zur Beurteilung der Ergebnisse eines Spracherkennungssystems wird die Worterkennungsrate herangezogen

$$WR = (1 - \frac{Substitutions + Insertions + Deletions}{totalnumberofWords}) \cdot 100\% \tag{1.1}$$

Die erreichten Ergebnisse der drei verwendeten ASR Systeme sahen folgendermaßen aus:

System	# gesprochene Worte	# erkannte Worte	Worterkennungsrate
System A	33150	8766	26,4%
System B	48568	11280	23,2%
System C	48046	18582	38,7%

Tabelle 1.2.: Erkennungsraten der drei getesteten ASR Systeme [5]

Zusätzlich zu den normalen Ergebnissen wurden spezielle Aufnahmen unter Laborbedingungen durchgeführt, die folgendermaßen aussahen:

- 50 ICAO konforme Standardphrasen
- reduziertes Vokabular mit 80 Wörtern
- komplettes Vokabular bekannt und trainiert
- nur eine Instruktion pro Äußerung
- 7 EEC Fluglotsen sprechen diese Phrasen für diesen speziellen Test

System	# gesprochene Worte	# erkannte Worte	Worterkennungsrate
System A	521	277	52%
System B	521	439	84%
System C	521	420	81%

Tabelle 1.3.: Erkennungsraten der drei getesteten ASR Systeme unter Laborbedingungen [5]

1.4. Interpretation der Eurocontrol Ergebnisse

Oberflächlich betrachtet könnte man jetzt schlußfolgern, dass die Spracherkennung für eine Anwendung in der Flugverkehrssicherheit nicht geeignet ist. So wurde auch am Eurocontrol Simulation Center festgestellt, dass die Spracherkennung den Anforderungen nicht genügt. [5] Die Hauptmotivation dieser Arbeit ist es diesen Schluss zu widerlegen und zu zeigen, dass Spracherkennung auch mit diesem schwierigen Szenario umgehen kann. Der erste Ansatzpunkt ist festzustellen, warum kommerzielle Spracherkennungssysteme, die auch schon 1998 recht gute Erkennungsraten erzielten, diese Performance bei den Simulationen der Eurocontrol ganz und gar nicht lieferten.

1.5. Zusammenfassung: Komplexität der Aufgabenstellung

Um die Schwierigkeit der Aufgabe einzuschätzen, hier noch einmal eine Zusammenfassung der Lernbedingungen:

- Szenario: sehr realitätsnahe simulierte Flugkontrollsituationen
- 10 Sprecher (4 Frauen / 6 Männer)
- Sprachtypus: controlled language (ICAO-Standard)
- 10.086 Aufnahmen (Zw. 238 und 1848 Äußerungen pro Sprecher)
- Sprecher stammen aus Genf/Söllingen/Zürich
  Englisch (non-native)
  Muttersprache der Sprecher: Französisch (Schweiz), Deutsch (Schweiz), Deutsch (Deutschland)
- Samplingfrequenz: 32.000 Hz, bei 16 Bit Auflösung
- Close-talking microphone
- Auf Wortebene transkribiert
- mid-range Vokabular (859 Wörter)
- Sprechermodus: kontinuierliche Sprache
- Sprecherstil: spontane Sprache
- Enrollment: sprecherunabhängig (da 10 Sprecher)
- Umgebungsgeräusch: nur Controlroom Umgebungsgeräusche

Bevor wir uns an das Design eines Spracherkenners für diese Aufgabenstellung wagen, wird zuerst im nächsten Kapitel eine theoretische Einführung in das Thema geliefert.

2. Automatische Spracherkennung (ASR)

> Reports and theses in speech recognition often begin with a cliche, namely that speech is the most natural way for human beings to communicate with each other
>
> *(Melvyn J. Hunt)*

2.1. Aufbau eines stochastischen Spracherkenners

Seit den 80er Jahren folgt die Spracherkennung ausschließlich dem statistischen Verarbeitungsparadigma. (siehe Abbildung 2.1)

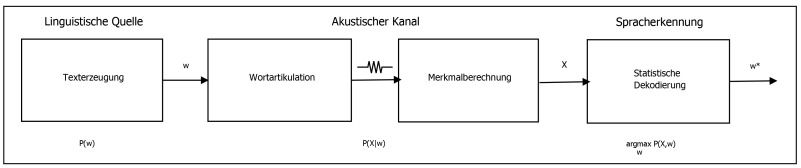

Abbildung 2.1.: statistisches Modell der Spracherkennung

Das statistische Modell sieht die Spracherkennung als einen Dekodierungsprozess, der die sprecherseitige Verschlüsselung einer Sequenz **w** gesprochener Wörter $w_1, ..., w_m$ in eine Sequenz **X** reelwertiger Merkmalvektoren $x_1, ...x_T$ umkehrt. Ausgehend von **X** (Merkmalsvektoren des Sprachsignals) ist eine Näherung w^* der ursprünglichen Wortfolge w zu berechnen.

Sind die a priori Verteilung $P(w)$ sowie die likelihood Verteilung $P(X|w)$ bekannt, so ist auch die gemeinsame Verteilungsfunktion $P(X, w)$ von Wort und Vektorfolge bekannt und die gesuchte Lösung ergibt sich aufgrund der **Bayesregel** zu:

$$w^* = argmax_w P(w|X) = argmax_w \frac{P(w) \cdot P(X|w)}{P(X)} = argmax_w P(w) \cdot P(X|w) \qquad (2.1)$$

Die Bayesregel minimiert auf lange Sicht die Fehlerwahrscheinlichkeit bei der Worterkennung. Allerdings sind die autretenden Verteilungsfunktionen nicht bekannt. Zum Entwurf eines Spracherkennungssystems sind möglichst exakte Näherungen für $P(X|w)$ (akustisches Modell) und $P(w)$ (Sprachmodell) zu bestimmen, sowie eine geeignete Dekodierungsfunktion $argmax_w$ zu finden. Die Marginalverteilung $P(X)$ im Nenner ist konstant bezüglich w und folglich irrelevant für die Maximierung. [6]

Im Folgenden werden diese Module im Detail erklärt.

Zuerst muss eine geeignete Parametrisierung des Eingabesignals vorliegen. In dieser Arbeit werden dafür nur Mel Frequency Cepstral Coefficients (MFCCs) als Merkmalsvektoren verwendet. Das Mel im Namen beschreibt die wahrgenommene Tonhöhe. Danach folgt das Hauptaugenmerk dieser Arbeit - die akustische Modellierung mit Hidden Markov Modellen. Die Theorie wird schließlich mit einem kurzen Überblick über den Einfluss des Sprachmodells abgerundet.

2.2. Merkmalsextraktion

Die Merkmalsextraktion dient neben der Repräsentation des Sprachschalls und der Reduktion der Datenmenge dazu die Variabilitäten der verschiedenen Phoneme oder Wörter hervorzuheben. Dabei sollen die Unterschiede, die durch verschiedene Sprecher, Sprechweise oder Umgebungseinflüsse verursacht werden, ausgeblendet werden.

Abbildung 2.2 zeigt den Prozess dieser Merkmalsgewinnung für die in dieser Arbeit verwendeten MFCCs. Bei Berechnung des Cepstrums wird die Faltungsoperation auf Grund des Logarithmus in eine Addition transformiert, womit man das Sprachsignal in Anregung und Quelle trennen kann. Die Mel Bandpassfilterung dient dazu Frequenzbänder zusammenzufassen und dabei die psychoakustischen Phänomene des menschlichen Gehörgangs nachzubilden.

2.2.1. DFT

Das Sprachsignal in der Zeitdarstellung ist zu komplex, um daraus aussagekräftige Informationen zu gewinnen. Deshalb wird zuerst eine Transformation in den Frequenzbereich mittels DFT vorgenommen.

$$X(n) = \sum_{k=0}^{N-1} x[k] \cdot e^{-j \cdot \frac{2\pi}{N} \cdot n} \tag{2.2}$$

2.2.2. Fensterung

Da Sprachsignalfolgen nichtperiodisch sind, muss für diese Fouriertransformation ein kurzer Ausschnitt mittels Fensterung entnommen und periodisch fortgesetzt werden. Eine geeignete und in der Sprachverarbeitung häufig eingesetzte ist das Hamming-Fenster , welches Sprungstellen an den Kanten vermeidet und durch eine stärkere Dämpfung der Nebenmaxima einen größeren Kontrast im Spektrum bewirkt. [7]

$$w_{hamm}[n] = 0,54 - 0,46 cos(\frac{2\pi n}{N-1}), 0 \leq n < N \tag{2.3}$$

2.2.3. Mel-Filterbank

Das menschliche Hörempfinden ist je nach Frequenzbereich unterschiedlich gut ausgeprägt. Töne höherer Frequenz werden leiser wahrgenommen als Töne tieferer Frequenz. Die Mel-Skala (siehe Abbildung 2.3) wurde eingeführt, um dieses psychoakustische Phänomen nachzuempfinden. [7]

$$mel(f) = 1127 \cdot ln(1 + \frac{f}{700} Hz) \tag{2.4}$$

Die Umsetzung kann als eine Bank von Dreiecksfiltern mit größerer Bandbreite für höhere Frequenzen erreicht werden.

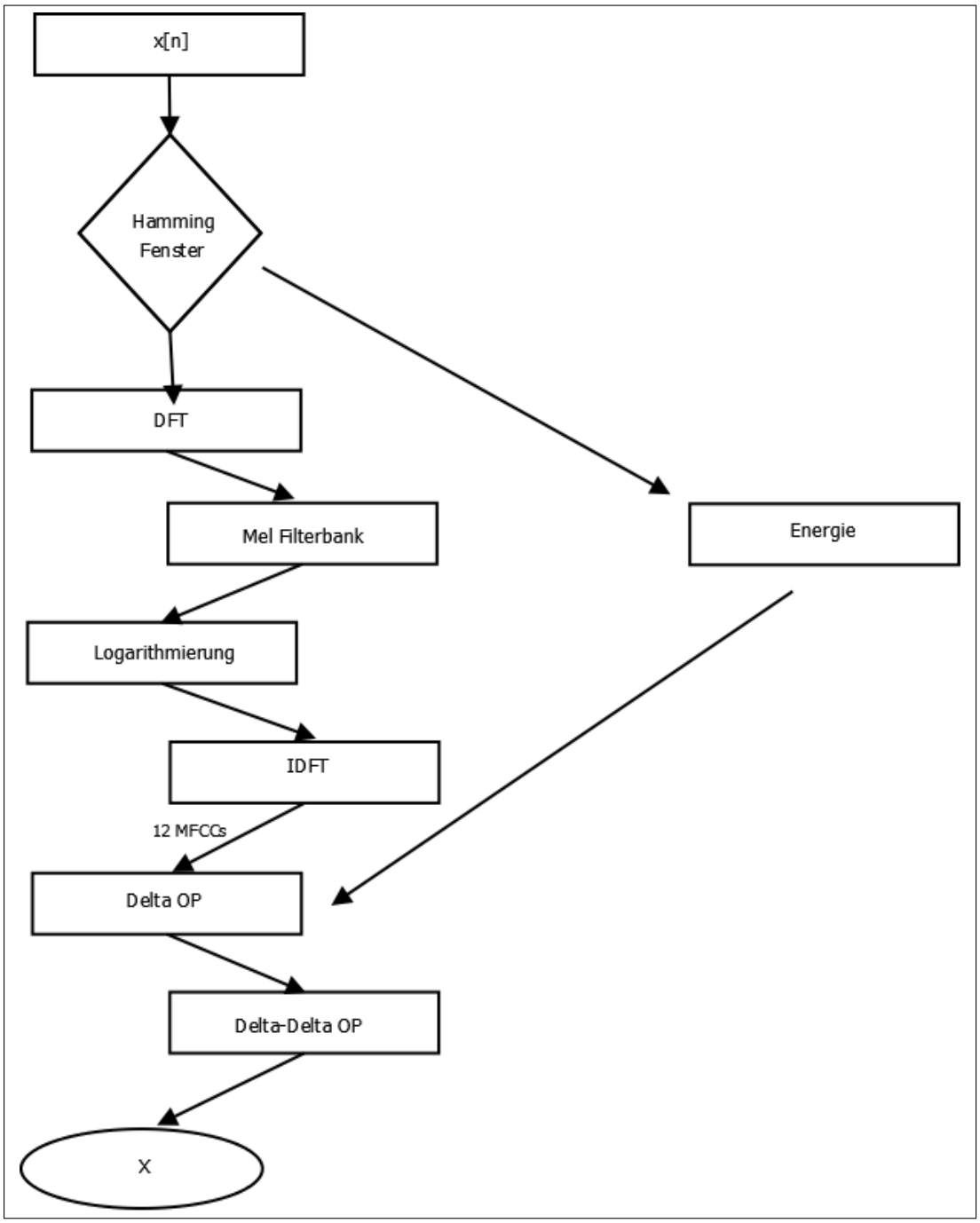

Abbildung 2.2.: Prozess der Merkmalsgewinnung

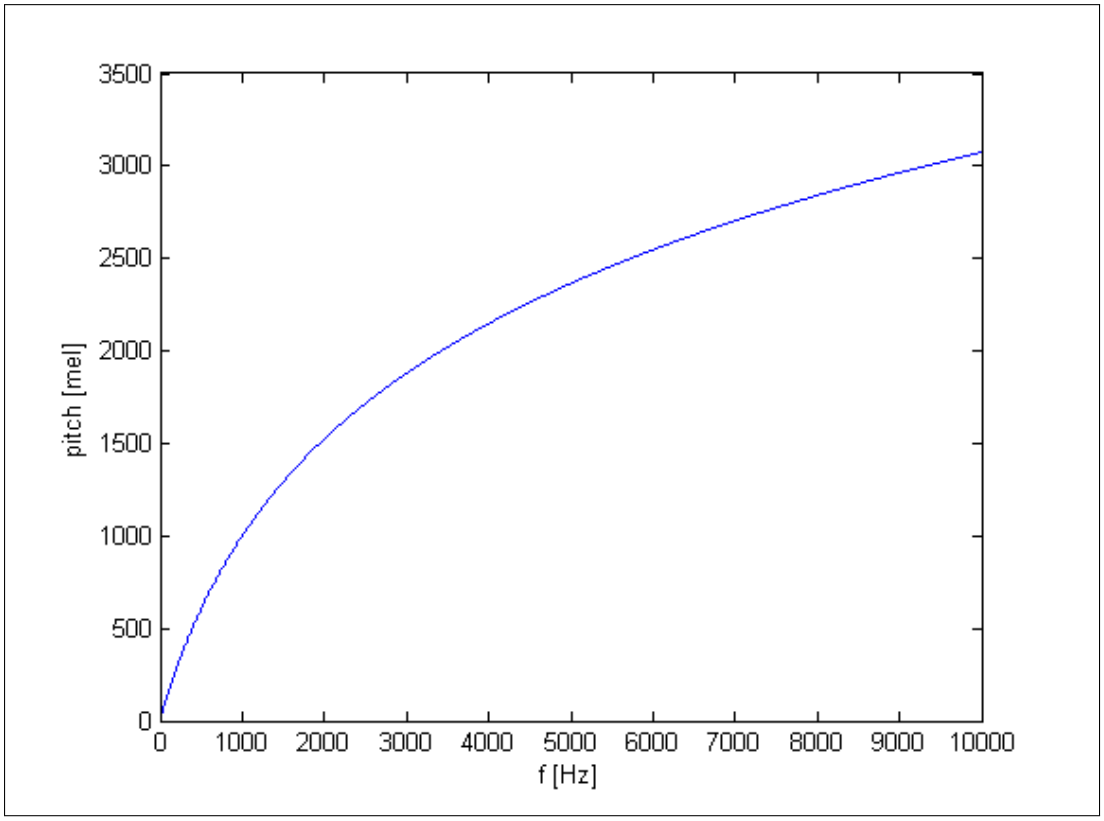

Abbildung 2.3.: Mel Skala

2.2.4. Cepstrum Koeffizienten

Nach der Filterung wird das Signal noch logarithmiert und daraufhin wieder mit einer Inversen Fourier oder Kosinustransformation in den Cepstralbereich transformiert (dadurch sind die Features unkorreliert). Die Cepstrumkoeffizienten niedrigerer Ordnung stellen die Grobstruktur des Sprachsignals dar. Nur diese enthalten für die Phonemunterscheidung relevante Informationen. [7]

2.2.4.1. Zeitliche Änderung

Systematische Untersuchungen aus der Mustererkennung zeigen, dass die zeitliche Änderung eines Features mehr Informationen für den Erkennungsprozess enthalten als der absolute Wert eines Merkmals. Deshalb wird die 1. zeitliche Ableitung und die 2. zeitliche Ableitung berechnet und dem Merkmalsvektor hinzugefügt.

$$\Delta x(n) = \frac{x(n+\tau) - x(n-\tau)}{2} \tag{2.5}$$

Die resultierenden 39 Dimensionen eines MFCC Vektors $\mathbf{X}$ setzen sich nun folgendermaßen zusammen:
x_1 ... Energie
$x_2 - x_{13}$ ... Cepstrum Koeffizienten
x_{14} ... Δx_1
$x_{15} - x_{26}$ ... $\Delta x_2 - \Delta x_{13}$ ... Delta Features (velocity)

$x_{27} \ldots \Delta\Delta x_1$
$x_{28} - x_{39} \ldots \Delta\Delta x_2 - \Delta\Delta x_{13} \ldots$ Delta-Delta Features (acceleration)

2.3. Markov Modelle

Nachdem uns nun anstatt des Sprachsignals ein Merkmalsvektor X zur Verfügung steht, der dieses Sprachsignal möglichst gut repräsentiert, können wir ausgehend von unserer Gleichung

$$w^* = argmax_w P(w) \cdot P(X|w) \tag{2.6}$$

eine akustische Modellierung finden, um die likelihood $P(X|w)$ möglichst genau zu ermitteln. In dieser Arbeit werden für diese akustische Modellierung die bewährten Hidden Markov Modelle verwendet.

2.3.1. Markov Modelle

Formal definiert man ein Markov Modell durch eine Menge von Zuständen

$$S = s_1, s_2, ..., s_N s \tag{2.7}$$

durch die a-priori Wahrscheinlichkeiten (Wahrscheinlichkeit dafür, dass s_i der erste Zustand einer Zustandssequenz ist)

$$\pi_i = P(q_1 = s_i)\pi = pi_1, pi_2, ..., pi_N s \tag{2.8}$$

und durch die Übergangswahrscheinlichkeiten a_{ij}.

$$\begin{gathered} a_{ij} = P(q_{n+1} = s_j | q_n = s_i) \\ A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1Ns} \\ ... & ... & ... & ... \\ a_{N_s1} & a_{N_s2} & ... & a_{NsNs} \end{pmatrix} \end{gathered} \tag{2.9}$$

Man sieht, dass beim Übergang zu Zustand j nur der Zustand i einen Einfluss hat. Frühere Zustände haben keinen Einfluss. Dies ist eine Vereinfachung, die häufig angenommen wird. Man spricht hier von einem Markov Modell 1. Ordnung.

Ein Markov Modell produziert im Laufe der Zeit $1 \leq n \leq N$ eine Zustandssequenz:

$$Q = q1, ..., q_N, q_n \epsilon S \tag{2.10}$$

2.3.2. Hidden Markov Modell (HMM)

Bei einem Hidden Markov Modell kann man nicht beobachten, in welchem Zustand s_i sich das Modell gerade befindet. Stattdessen erzeugt ein zweiter Zufallsprozess zu jedem Zeitpunkt beobachtbare Ausgangssymbole (oder Beobachtungen) gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung. (siehe Abbildung 2.4) Diese Emissionswahrscheinlichkeiten

$$b_{i,k} = P(x_n = v_k | q_n = s_i) b_{i,k} \tag{2.11}$$

geben also die Wahrscheinlichkeit an, dass zum Observationszeitpunkt v_k x_n beobachtet werden kann, wenn das System im Zustand s_i ist. $X = x_1, x_2, ..., x_N$ wird dabei als Observationssequenz bezeichnet. Insgesamt ergeben sich folgende HMM Parameter.

$$\Theta = (A, B, \pi) \tag{2.12}$$

[8]

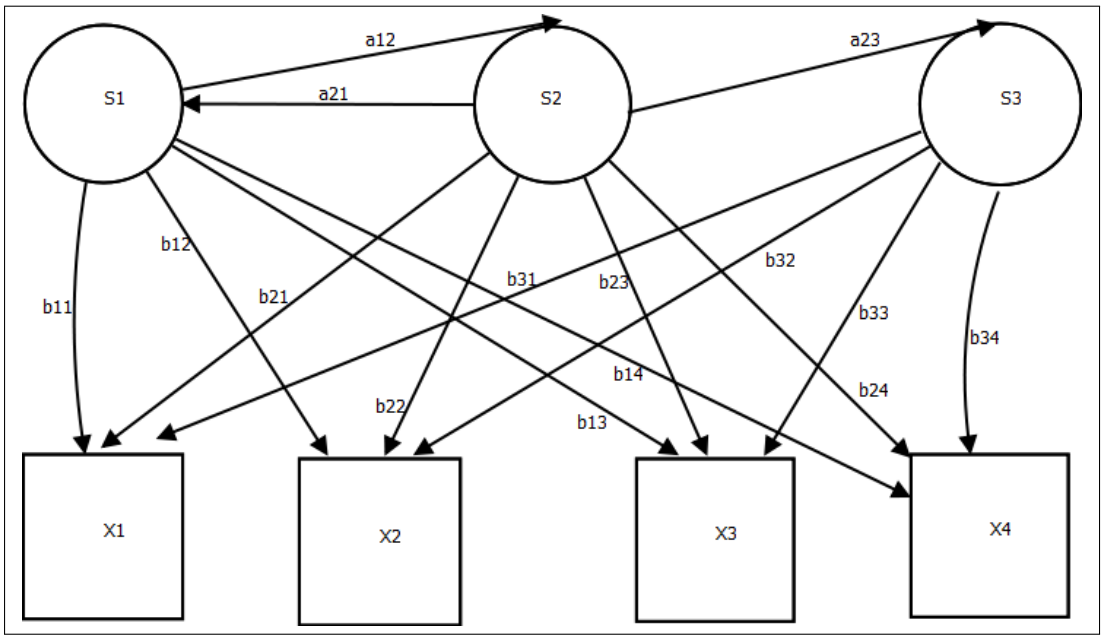

Abbildung 2.4.: HMM

2.3.3. Die drei grundlegenden HMM Probleme

Um dieses HMM Modell (als akustisches Modell) verwenden zu können, müssen drei fundamentale Probleme gelöst werden.

Klassifikationsproblem
Gegeben ist ein HMM Modell mit den Parametern $\Theta = (A, B, \pi)$. Finde die Produktionswahrscheinlichkeit $P(X|\theta)$ einer beobachteten Sequenz X.

$$P(X|\Theta) =? \tag{2.13}$$

Eine effiziente Implementierung ist der Forward/Backward Algorithmus.

Dekodierungsproblem
Gegeben ist wiederum ein HMM Modell Θ. Gesucht ist die hidden state Sequenz Q, die eine beobachtete Sequenz X am besten erklärt.

$$Q^* = argmax_Q P(X, Q|\Theta) =? \tag{2.14}$$

Eine effiziente Implementierung der Dekodierung ist der Viterbi Algorithmus.

Lernproblem
Wie muss man die Modelparameter Θ adjustieren um die Produktionswahrscheinlichkeit $P(X|\Theta)$ zu maximieren.

$$\hat{\Theta} = (\hat{A}, \hat{B}, \hat{\pi}) =? \Rightarrow P(X|\hat{\Theta}) = \max_{\Theta} P(X|\Theta) \tag{2.15}$$

Eine effiziente Lösung liefert der Baum-Welch Algorithmus. [8]

2.3.4. Forward Algorithmus

Der Forward Algorithmus berechnet mit Hilfe der Forward-Variablen $\alpha_n(j)$ für ein gegebenes HMM Modell Θ die Wahrscheinlichkeit einer Beobachtung $P(X|\Theta)$. Wobei die Forward Variable $\alpha_n(j)$ die Wahrscheinlichkeit, zum Zeitpunkt t die Beobachtung X gemacht zu haben und im Zustand s_j zu sein, beschreibt. [8]

$$\alpha_n(j) = P(x_1, ...x_n, q_n = s_j|\Theta) \tag{2.16}$$

$\alpha_n(j)$ (und somit auch $P(X|\Theta)$) lässt sich rekursiv berechnen:
1. Initialisierung:

$$\forall j : \alpha_1(j) = \pi_i \cdot b_{j,x1}, \tag{2.17}$$

2. Rekursion:

$$\begin{gathered} \forall n > 1 und \forall j = 1...N_s : \\ \alpha_n(j) = (\sum_{i=1}^{N_s} \alpha_{n-1}(i) \cdot \alpha_{i,j}) \cdot b_{j,xn} \end{gathered} \tag{2.18}$$

3. Termination:

$$P(X|\Theta) = \sum_{j=1}^{N_s} \alpha_N(j) \tag{2.19}$$

2.3.5. Backward Algorithmus

Der Backward-Algorithmus berechnet mit Hilfe der Backward Wahrscheinlichkeiten β die Wahrscheinlichkeit, dass bei einem gegebenen Hidden-Markov-Modell Θ und einer beobachteten Symbolsequenz X das Modell zur Zeit i im Zustand s_i war. [8]

$$\beta_n(i) = P(x_{n+1}, ...x_N | q_n = s_i, \Theta) \tag{2.20}$$

1. Initialisierung:

$$\begin{aligned} \forall i = 1...N_s \\ \beta_N(i) = 1 \end{aligned} \tag{2.21}$$

2. Rekursion:

$$\begin{aligned} \forall n < N \cap \forall i = 1...N_s \\ \beta_n(i) = \sum_{j=1}^{N_s} a_{i,j} \cdot b_{j,x_n+1} \cdot \beta_{n+1}(j) \end{aligned} \tag{2.22}$$

3. Termination:

$$P(X|\Theta) = \sum_{j=1}^{N_s} \pi_j \cdot b_{j,x1} \cdot \beta_1(j) \tag{2.23}$$

2.3.6. Viterbi Algorithmus

Der Viterbi Algorithmus bestimmt die wahrscheinlichste Sequenz von verborgenen Zuständen Q^* bei einem gegebenen Hidden Markov Model Θ und einer beobachteten Sequenz X von Symbolen. Diese Zustandssequenz wird auch als Viterbi-Pfad bezeichnet. [8]
1. Initialisierung

$$\begin{aligned} \forall j = 1...N_s \\ \delta_1(j) = \pi_j \cdot b_{j,x1} \\ \psi_1(j) = 0 \end{aligned} \tag{2.24}$$

2. Rekursion

$$\begin{aligned} \forall n > 1 \cap \forall j = 1...N_s \\ \delta_n(j) = max_i(\delta_{n-1} \cdot a_{i,j}) \cdot b_{j,xn} \\ \psi_n(j) = argmax_i(\delta_{n-1}(i) \cdot a_{i,j}) \end{aligned} \tag{2.25}$$

3. Termination

$$\begin{aligned} P^*(X|\Theta) = max_j(\delta_N(j)) \\ q_N^* = argmax_j(\delta_N(j)) \end{aligned} \tag{2.26}$$

4. Backtracking

$$\begin{aligned} q_n^* = \psi_{n+1}(q_{n+1}^*) \\ n = N-1, N-2, ..., 1 \end{aligned} \tag{2.27}$$

2.3.7. Baum-Welch Algorithmus

Der Baum-Welch Algorithmus (eine Instanz des Expectation-Maximization kurz EM Algorithmus) ist die Lösung des dritten und schwierigsten HMM Problem. Das Lernproblem wird iterativ in zwei Teilschritten gelöst. Im ersten verwendet er den Forward-Backward Algorithmus zur Findung der bereits bekannten Forward-Backward Variablen. Auf der Basis dieses ersten Schrittes berechnet er im zweiten Schritt die neuen Modellparameter.

2.3.7.1. Schritt 1: Estimation-Step

Mit Forward/Backward Algorithmus alle α und β Variablen berechnen. Daraus alle Übergangswahrscheinlichkeiten $\xi_n(i,j)$ für s_i nach s_j zum Zeitpunkt n bei gegebenem Modell Θ berechnen.

$$\xi_n(i,j) := P(q_n = s_i, q_{n+1} = s_j | X, \Theta) = \frac{\alpha_n(i) \cdot a_{i,j} \cdot b_{j,x_{n+1}} \cdot \beta_{n+1}(j)}{P(X|\Theta)} \tag{2.28}$$

mit

$$P(X|\Theta) = \sum_{i=1}^{N_s} \alpha_n(i) \cdot \beta_n(i) \tag{2.29}$$

Zusätzlich wird noch die Wahrscheinlichkeit im Zustand s_i zum Zeitpunkt n zu sein benötigt:

$$\delta_n(i) := P(q_n = s_i | X, \Theta) = \sum_{j=1}^{N_s} \xi_n(i,j) \tag{2.30}$$

Wenn man diese Wahrscheinlichkeiten ξ und δ über alle Zeitabschnitte n aufsummiert, so ergeben sich Erwartungswerte für den Aufenthalt in state s_i bzw. für den Übergang von s_i nach s_j.

2.3.7.2. 2. Schritt: Maximization-Step

Ausgehend von den Gleichungen des E-Steps werden die neuen Modellparameter $\hat{\Theta}$ folgendermaßen ermittelt.

$$\begin{aligned} \pi_i &= \gamma_1(i) \\ a_{i,j} &= \frac{\sum_{n=1}^{N-1} \xi_n(i,j)}{\sum_{n=1}^{N-1} \gamma_n(i,j)} \\ b_{j,k} &= \frac{\sum_{n=1}^{N} \gamma_n(i,j) \cdot [x_n = v_k]}{\sum_{n=1}^{N} \gamma_n(i,j)} \end{aligned} \tag{2.31}$$

Nach dem M-Step ist eine Iteration abgeschlossen und die nächste E-Step/M-Step Iteration kann beginnen. Das Ganze wird solange wiederholt, bis die Maximum-Log Likelihood $L(X|\Theta)$ konvergiert. [8]

2.4. Akustisch-phonetische Modellierung

Als geeignete Modelltopologien werden in der Sprachverarbeitung Links-Rechts Modelle verwendet. (siehe Abbildung 2.5) Da Sprachsignale einem zeitlichen Ablauf folgen, ist auch dies eine sinnvolle Vereinfachung.

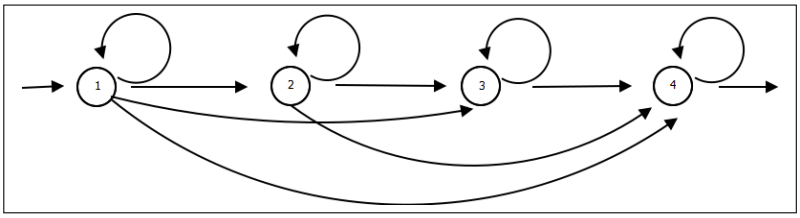

Abbildung 2.5.: Links-Rechts Modelle als geeignete Modelltopologien für die Sprachverarbeitung [6]

Bei der Anwendung von Hidden Markov Modellen in der Spracherkennung sind die im vorherigen Kapitel vorgestellten Algorithmen noch etwas komplizierter, da die Emissionswahrscheinlichkeiten $b_{i,j}$ nicht diskret sind, sondern einem kontinuierlichen Gaußschem Emissionsmodell entsprechen.

Diese Verteilung enthält meistens noch mehrere Mischkomponenten (Mixtures of Gaussians), d.h. eine gewichtete Summe mehrerer Gaußverteilungen.

$$bj(x) = \sum_{k=1}^{K} c_{jk} \mathrm{N}(\mu_{jk}, \Sigma_{jk}) \tag{2.32}$$

mit $\sum_{k=1}^{K} c_{jk} = 1$.

Die Kovarianzmatrix Σ_{jk} wird dabei als diagonal angenommen. Diese Vereinfachung impliziert, dass die 39 Dimensionen der Feature Vektoren unkorreliert sind. Durch die Kosinustransformation bei der Merkmalsextraktion ist das auch gewährleistet. [6]

	Wortmodell	Monophonmodell	Triphonmodell
Präzision	+	-	+
Robustheit	-	+	+
Modularität	-	+	+

Tabelle 2.1.: Vergleich Ganzwortmodell mit Wortuntereinheiten

2.4.1. Vom Ganzwortmodell zu Wortuntereinheiten

Bei der Erkennung von Sprache liegen die Wortrealisierungen in kompletten Sätzen eingebettet vor und sind unter Umständen durch Stillebereiche unterbrochen. Das Wort ist aus vielfältigen Gründen für die Spracherkennung nur mäßig geeignet:

- Das erforderliche Trainingsmaterial ist proportional zum Wortschatzumfang L.
- Die Zahl freier HMM-Parameter ist enorm, was zu labilen Schätzwerten und hohem Rechen- und Speicheraufwand führt.
- Die Modelle sind sehr unflexibel, weshalb das Erkennungsvokabular dem Trainingsvokabular entsprechen muss.
- Wortübergreifende Aussprachevarianten können nicht berücksichtigt werden.

Deswegen bietet sich die Verwendung von geeigneten Wortuntereinheiten an. Am naheliegendsten ist die Verwendung eines Monophonmodells.

2.4.2. Monophone

Jede Sprache enthält zwischen 20 und 60 Phoneme. Die Verwendung eines einzelnen Phonems als Wortuntereinheit für die Spracherkennung löst scheinbar alle Probleme auf einen Schlag. Die freien Parameter sind gering und selbst mit einer verhältnismäßig kleinen Trainingsstichprobe können diese robust geschätzt werden. Das Markovmodell eines beliebigen Wortes lässt sich allein aufgrund der Kenntnis seiner Lautschrift aufbauen. Unglücklicherweise bewirkt die Kontextabhängigkeit der Phonemartikulation, dass das Monophonmodell zu unpräzise wird und damit nur schwer eine gute Erkennungsrate erreicht werden kann.

2.4.3. Kontextabhängige Wortuntereinheiten - Triphonmodelle

Aus linguistischer Sicht wären Wortuntereinheiten, wie Silben, Halbsilben, Sylparts oder Diphone denkbar. Allerdings sind diese Untereinheiten schwer modellierbar oder unmodular. Als State-of-the-art der Spracherkennung hat sich die Verwendung von Triphonmodellen als Wortuntereinheit durchgesetzt. Diese kombinieren die Flexibilität des Monophonmodells mit der Kontextabhängigkeit durch die phonologische Umgebung.

Man betrachtet dabei jedes Monophon in seiner unmittelbaren lautlichen Umgebung, welche durch die angrenzenden Phoneme der Transkription repräsentiert werden. Beim Training werden zur Initialisierung in einem Vorlauf die Modelle kontextunabhängiger (Monophone) Laute erzeugt; jedes Triphonmodell wird anschließend mit den Parametern der Monophon HMMs vorbesetzt, das zu seinem Kernphonem gehört. Die zentrale Schwierigkeit liegt im Transfer; normalerweise sind nicht alle Wörter des Anwendungsbereiches in der Lernstichprobe vertreten, in der Regel werden nicht einmal alle zu ihrer Synthese benötigten Triphonmodelle in ausreichender Zahl vorliegen.

Allgemein sind mit den Triphonen sehr präzise, aber eher unsolide geschätzte Modelle verknüpft, während die der Monophone statistisch robust, doch in ihren Eigenschaften verwaschen sind. Um diesem Problem der Triphone zu begegnen, gibt es verschiedene Ansätze, wie Rückgriffstrategien (auf Monophonmodell), Interpolation oder Generalisierungsverfahren.

In dieser Arbeit wird das Generalisierungsverfahren CART (Classification and Regression Tree) verwendet. Diese Generalisierung geschieht mittels Konstruktion eines Entscheidungsbaumes durch linguistische Ja/Nein Fragen an den linken und rechten Phonemkontext, wobei in jedem Knoten aus einem festen Vorrat erlaubter Fragen diejenige ausgewählt wird, welche die Aufspaltung mit dem maximalen Informationsgewinn nach sich zieht. [6] Dadurch werden unter der Annahme, dass ein Laut in ähnlichen Kontexten auch ähnlich realisiert wird, Kontexte in gröbere Klassen zusammengefasst. Die resultierenden Grundelemente heißen generalisierte Triphone. Durch die Gruppierung müssen weniger Triphone unterschieden werden, sodass die Anzahl der Modelle ab und die Trainierbarkeit zunimmt.

2.4.4. Wortgrenzmodellierung

Aufgrund der Koartikulation wird in der Spracherkennung das Triphonmodell gegenüber dem Monophonmodell bevorzugt. Da bei kontinuierlicher Sprache nicht zwischen jedem Wort eine Pause eingelegt wird, ergeben sich aber auch wortübergreifende Koartikulationsverschleifungen.

Werden nur wortinterne Triphone verwendet, spricht man von einem wortinternen Triphonmodell. Kommen hingegen auch wortübergreifende Triphonmodelle zum Einsatz, spricht man von einem cross-word Triphonmodell.

Bei der Verwendung eines cross-word Triphonmodells explodiert die Anzahl der möglichen Triphone, weshalb der vorgestellte Generalisierungsmechanismus (state-tying+Cart) umso wichtiger ist. Jedoch müssen die verwendeten phonetischen Fragen eigentlich für jeden Korpus von einem Phonologieexperten definiert werden. Weshalb nicht immer gewährleistet ist, dass ein cross-word Triphonmodell bessere Ergebnisse liefert als ein wortinternes.

2.4.5. Wörterbuch

Die Voraussetzung, um ein Wortuntereinheitenmodell (siehe Abbildung 2.6) basierend auf Triphonen verwenden zu können, ist ein Wörterbuch mit Lautschrift. Am einfachsten konstruiert man sich so ein Wörterbuch, indem man sich eine Liste der vorkommenden Wörter aus der Lernstichprobe extrahiert und diese mittels einem Standardwörterbuch in Lautschrift transkribiert.

2.5. Sprachmodell

Mittlerweile liegt das Eingabesignal als Merkmalsvektor X vor und ein akustisches Modell $P(X|w)$ wurde gewählt. Ausgehend von unserer Gleichung

$$w^* = argmax_w P(w) \cdot P(X|w) \tag{2.33}$$

muss noch eine Näherung für das Sprachmodell $P(w)$ gefunden werden. Das Sprachmodell $P(w)$ dient dazu, falsch erkannte Wörter mit Hilfe des Kontextes zu korrigieren, weil das akustische Modell $P(X|w)$ auch bei deutlicher Aussprache einige Wörter nicht richtig ermitteln kann. [7]

Grundsätzlich unterscheidet man zwei Arten von Sprachmodellen:

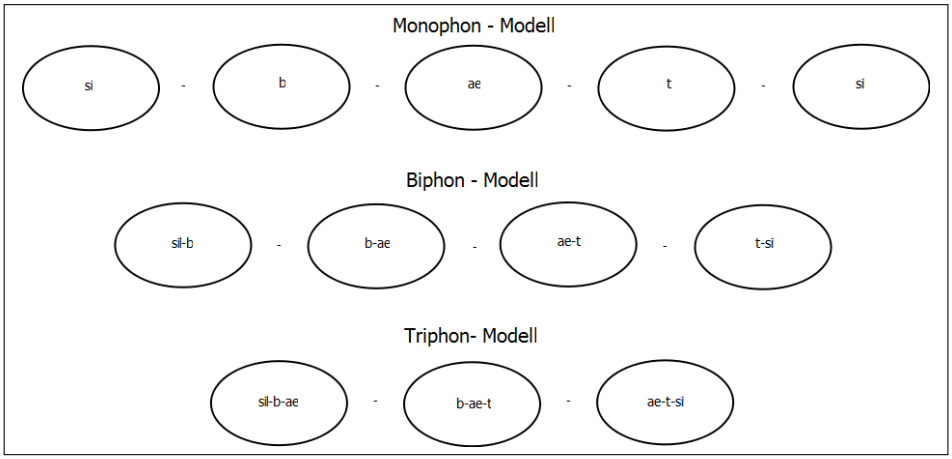

Abbildung 2.6.: Phonemmodelle[8]

- regelbasierte Sprachmodelle (Kontextfreie Grammatiken)
- stochastische Sprachmodelle (n-gramme)

2.5.1. Kontextfreie Grammatik

Eine kontextfreie Grammatik hat 4 Parameter $< \Sigma, V, S, P >$.

1. Σ ... Menge der Terminalsymbole (Wörter)
2. V ... Grammatikvariablen (Nicht-Terminale)
3. SϵV ... Startsymbol
4. P ... Menge der Produktionsregeln

Die formale Sprache, die durch eine kontextfreie Grammatik (kontextfrei, weil Wortkontext nicht modelliert wird) definiert wird, ist die Menge von Strings, welche von einem designierten Startsymbol S mittels der Produktionsregeln P abgeleitet werden kann.

Zur Erklärung ein Beispiel für eine kontextfreie Grammatik für ein Subset der englischen Sprache.

2.5.1.1. Beispiel für eine kontextfreie Grammatik

$\Sigma = I, prefer, a, morning, flight$
V=$S, NP, VP, Pro, Verb, Det, Nom, Noun, Noun$

Grammatikregeln:

- S -> NP+VP
- NP -> Pro
- NP -> Det + Nom
- Nom -> Noun+Noun

lexikalische Regeln:

- Pro -> I
- Verb -> prefer
- Det -> a
- Noun -> morning

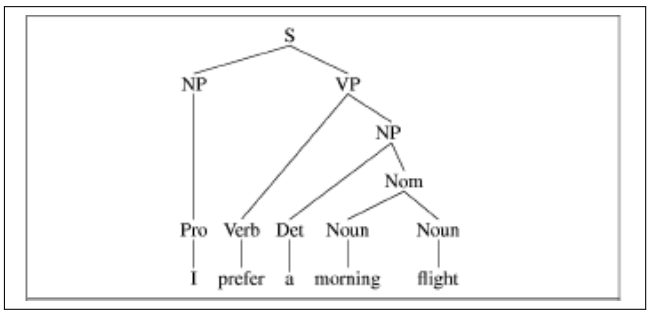

Abbildung 2.7.: Baum für den Satz „I prefer a morning flight“[9]

- Noun -> flight

Anhand dieser Regeln lässt sich dann für den Satz „I prefer a morning flight“ ein Baum bauen. (siehe Abbildung 2.7)

Der Satz „I prefer a morning flight“ist anhand der von uns definierten Grammatikregeln ein korrekter englischer Satz, da er mit unserem Grammatikmodell abbildbar ist. Um eine kontextfreie Grammatik zu verifizieren, ist ein sogenannter Part of Speech Tagger, der den Wörtern die entsprechenden Terminalsymbole zuordnet, nötig. (Kann mittels HMM und Viterbi gelöst werden) Danach wird mit einem Parser basierend auf dem Cocke-Younger-Kasami-Algorithmus (CYK) ALgorithmus rekursiv aus den Speech Tags der Baum konstruiert, falls eine Konstruktion möglich ist. Eine Erweiterung der kontextfreien Grammatik wäre die Verwendung einer PCFG (probabilitistic context free grammar), wobei alle Produktionsregeln zusätzlich mit Wahrscheinlichkeiten verknüpft sind. [9]

2.5.2. Stochastisches Sprachmodell

Stochastische Sprachmodelle basieren auf der Faktorisierung in bedingte Wortwahrscheinlichkeiten. Die Vorgeschichte fließt in der Berechnung der Auftrittswahrscheinlichkeit eines Wortes mit ein. Das ist aufgrund der Redundanz natürlicher Sprache sinnvoll. Somit können auf gewisse Wortfolgen nur bestimmte weitere Folgen sinnvoll sein. [7]

Im einfachsten probabilitischen Wortmodell hätte jedes Wort die gleiche Wahrscheinlichkeit auf ein anderes zu folgen. Ein etwas komplexeres Wortsequenzmodell würde jedem Wort seine relative Auftrittswahrscheinlichkeit in Bezug auf einen Trainingskorpus zuordnen. Wenn wir dieses Modell um bedingte Wahrscheinlichkeiten erweitern - also nicht die individuelle relative Auftrittswahrscheinlichkeit eines Wortes verwenden, sondern auf die bedingte Auftrittswahrscheinlichkeit, dass ein Wort y kommt, wenn der Vorgänger x war. $P(y|x)$ [9]

Allgemein kann diese Wahrscheinlichkeit so ausgedrückt werden:

$$P(w) = P(w_1) \cdot P(w_2|w_1) \cdot \prod_{i=3}^{L} P(w_i|w_1....w_i - 1) \tag{2.34}$$

Doch es gibt keinen einfachen Weg die Wahrscheinlichkeit eines Wortes bei gegebener Sequenz von Vorgängern zu berechnen. Das Problem wird durch eine einfache Approximation gelöst. Es wird nicht auf die gesamte Sequenz von Vorgängern geschaut, sondern nur auf einen Vorgänger (Bigramme) bzw. auf zwei Vorgänger (Trigramme). Ein Bigramm ist auch ein Markov Modell 1. Ordnung, ein Trigramm ein Markov Modell 2. Ordnung und ein n-Gramm Modell ein Markov Modell n. Ordnung.

$$P_{bigramm} = P(w_n|w_{n-1}) \tag{2.35}$$

Bei einem Wörterbuch mit L Einträgen existieren L^2 Bigramme und

$$P_{trigramm} = P(w_n|w_{n-2}w_{n-1}) \tag{2.36}$$

L^3 Trigramme. Abhilfe zur Reduktion der Anzahl zu speichender Wahrscheinlichkeiten ist die Einteilung in Wortkategorien, z.B. die Kategorie Numerale für alle Zählwörter.

2.5.2.1. Bigramm Beispiel

Gegeben ist folgender Korpus: $$ John sat on the old chair ##
$$ John read the old book ##
$$ John was interesting ##
$$ the book was interesting ##

$$P(sat|John) = \frac{\#(\text{John sat})}{\#(\text{John})} = \frac{1}{3} \tag{2.37}$$

aber anhand dieses Trainingskorpus wäre die Wahrscheinlichkeit

$$P(sat|Max) = \frac{\#(\text{Max sat})}{\#(\text{Max})} = 0 \tag{2.38}$$

.

Um dem Problem von 0 Wahrscheinlichkeiten zu begegnen, gibt es 3 Lösungsmöglichkeiten:

- Glättung der Schätzwerte (dicsounting oder smoothing)
- Rückgriffstrategie (Backoff)
- Interpolation

2.5.2.2. Discounting

State of the art beim Discounting ist die Verwendung eines **Kneser-Ney Discounting** Algorithmus. Wegen der Komplexität wird an dieser Stelle stattdessen das „Laplace Smoothing “mit n=1. erklärt, weil dadurch die Idee einfacher erkennbar ist.

Gegeben sind K=3 Pins w_i mit der relativen Häufigkeit N_i.

$$\begin{aligned} N_1 &= 4 \\ N_2 &= 3 \\ N_3 &= 0 \end{aligned} \tag{2.39}$$

Die dazugehörigen Wahrscheinlichkeiten wären:

$$\begin{aligned} P(w_1) &= \frac{N_1}{N_1 + N_2 + N_3} = \frac{4}{7} \\ P(w_2) &= \frac{N_2}{N_1 + N_2 + N_3} = \frac{3}{7} \\ P(w_3) &= \frac{N_3}{N_1 + N_2 + N_3} = \frac{0}{7} \end{aligned} \tag{2.40}$$

Laplace Smoothing mit n=1 macht daraus:

$$\begin{aligned}\hat{P}(w_1) &= \frac{N_1 + 1}{N_1 + N_2 + N_3 + K} = \frac{5}{10} \\ \hat{P}(w_2) &= \frac{N_2 + 1}{N_1 + N_2 + N_3 + K} = \frac{4}{10} \\ \hat{P}(w_3) &= \frac{N_3 + 1}{N_1 + N_2 + N_3 + K} = \frac{1}{10}\end{aligned} \tag{2.41}$$

2.5.2.3. Backoff

Wenn die Wahrscheinlichkeit eines n-Gramms 0 ergibt, wird auf das (n-1)-Gramm zurückgegriffen, um diese Wahrscheinlichkeit zu ermitteln.

$$\hat{P}(w_i|w_{i-2}w_{i-1}) = \begin{cases} P(w_i|w_{i-2}w_{i-1}), & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 P(w_i|w_{i-1}), & \text{if } C(w_{i-2}w_{i-1}w_i) = 0 \\ & \text{and } C(w_{i-1}w_i) > 0 \\ \alpha_2 P(w_i), & \text{otherwise.} \end{cases}$$

Abbildung 2.8.: Backoff für Trigramme[9]

2.5.2.4. Interpolation

Bei der Interpolation werden verschiedene n-Gramm Ordnungen kombiniert, indem ein Trigramm über eine Linearkombination der Unigramm, Bigramm und Trigramm-Wahrscheinlichkeit interpoliert wird.

$$\hat{P}(w_3|w_2w_1) = \lambda_1 \cdot P(w_1) + \lambda_2 \cdot P(w_2|w_1) + \lambda_3 \cdot P(w_3|w_2w_1) \tag{2.42}$$

mit $\sum_{i=1}^{3} \lambda_i = 1$. [9]

In dieser Arbeit wird ein Bigramm Sprachmodell mit einer Kombination aus Kneser-Ney Discounting und Backoff verwendet.

2.6. Sprecher Adaptation

Kommerzielle Spracherkenner sind in der Regel sprecherunabhängig trainiert. Unter Umständen wird die Möglichkeit geboten mittels einer kleinen Menge an Sprachproben (Development oder Enrollment Data) einen bestimmten Sprecher anzulernen und dessen Sprache dann besser zu erkennen.

Die in dieser Arbeit dafür verwendete Methode basiert auf MLLR (Maximum likelihood linear regression). Dabei wird eine Transformation berechnet, welche den Unterschied zwischen einem initialen Modell und dem Development Material reduziert. Um das Ganze technisch auszudrücken: MLLR ist eine Modelladaptionstechnik, welche eine lineare Transformationsvorschrift für die Mittelwerte und Varianzparameter eines Gaussian mixture HMM Systems

berechnet. Mit dieser Transformation wird also ein neues HMM System mit angepassten Parametern erzeugt, welches eher die Daten des Development Materials erzeugt.

Wenn nur die Mittelwerte verschoben werden, spricht man von einem MLLR-Mean Algorithmus. Analog dazu heißt eine Variante des MLLR, welcher nur die Kovarianzmatrizen transformiert MLLRCOV. Die 3. Variante des MLLR genannt CMLLR (Constrained MLLR) transformiert die Features anstatt Mittelwerte und Varianzen.

2.7. Wortdekodierung

Ausgehend von der Gleichung

$$w^* = argmax_w P(w) \cdot P(X|w) \tag{2.43}$$

fehlt nun nur mehr eine geeignete Modellierung für die Bildung von $argmax_w$. Da das gesamte System als Kombination von akustischem Modell und Sprachmodell ein riesiges Markovmodell ist, eignet sich hierfür der bereits beschriebene Viterbi Algorithmus. Es gibt jedoch zwei Hauptprobleme bei der Verwendung des Viterbi Algorithmus. Zum ersten berechnet der Viterbi Algorithmus nicht die Wortsequenz, welche am ehesten einer Akustik entspricht, sondern eine Approximation davon. Und zum zweiten kann der Viterbi Algorithmus von keinem Sprachmodell, welches komplexer als ein Bigramm Sprachmodell ist, profitieren, da er auf dynamischem Programmieren basiert. [9] Grundsätzlich gibt es zwei Möglichkeiten diesen Problemen zu begegnen. Entweder man verwendet einen komplett anderen Dekodierungsalgorithmus (z.B. A* Decoder) oder man verwendet den Viterbi Algorithmus als Teil eines Zweistufendekoders. In dieser Arbeit wird ein Zweistufendekoder verwendet, dessen Aufbau in Abbildung 2.9 dargestellt ist.

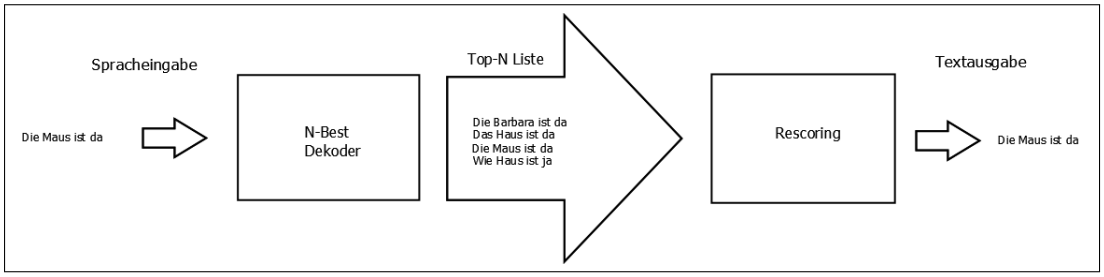

Abbildung 2.9.: N-best Decoding als Teil eines zweistufigen Dekoders[9]

Der Viterbi Algorithmus wird so verändert, dass er nicht nur eine Lösung, sondern eine ganze Liste von N-best Pfaden zurückliefert. Dabei verwendet der Viterbi ein einfaches Sprachmodell, wie zum Beispiel ein Bigramm-Modell. In einem zweiten Dekodierungsschritt wird dann ein Rescoring basierend auf einem besseren Sprachmodell vorgenommen mit dem auf N reduzierten Suchraum.

3. Design

Im vorherigen Kapitel wurden die theoretischen Grundlagen für den Entwurf eines Spracherkenners gelegt. Darauf aufbauend wird jetzt ein Spracherkenner für die ATCOSIM Sprachdatenbank entwickelt. Die Entwicklung basiert auf dem sogenannten HTK (Hidden Markov Toolkit), welches speziell zum Einsatz für wissenschaftliche Untersuchungen im Bereich der Spracherkennung an der Universität von Cambridge entwickelt wurde. HTK ist zum freien Download erhältlich und bietet alle nötigen Einstellungs- und Konfigurationsmöglichkeiten, um einen guten Spracherkenner zu bauen. Weiters fängt man mit dem Design eines Spracherkenners damit nicht bei Null an, sondern in der Dokumentation liegt ein Tutorial bei, welches den Entwurf eines Basissystems vorskizziert. [10]

3.1. Einstiegspunkte für den Entwurf des Spracherkenners

Der entwickelte Erkenner basiert auf drei verfügbaren Trainings- und Testskripten. Das Ziel war die Vor- und Nachteile der drei Varianten zu analysieren und das Beste aus allen dreien in ein neues Framework zu überführen.

3.1.1. HTK Tutorial

Das HTK Tutorial zeigt Schritt für Schritt, wie man einen Spracherkenner für ein Telefonwählsystem konstruiert. Dabei sollen Zahlen kontinuerlich gesprochen werden können und aus einer limitierten Menge von Namen auch direkt über den Namen gewählt werden können. Das Tutorial gliedert sich in folgende Abschnitte: Datenvorbereitung, Monophon HMM Modell erstellen, state-tied Triphonmodell erstellen, Erkennung und Sprecher Adaptierung. Obwohl im Prinzip alle Schritte vorgezeigt werden, liefert das Tutorial keinen vollautomatischen . Viele Schritte müssen von Hand per Editor gemacht werden und deswegen ist es zwar ein guter Startpunkt, aber für einen kompletten Spracherkenner, der mit großen Datenbanken umgehen soll, wäre das viel zu mühsam und größere Experimente wären damit kaum machbar. [10]

3.1.2. Wall Street Journal Corpus (WSJ0) Trainingsprozedur

HTK Recipe ist ein von Keith Vertanen zur Verfügung gestellter Erkenner basierend auf dem HTK Toolkit. Dabei wird ein Erkenner für die Daten des Wall Street Journal WSJ0 Corpus erzeugt. Als Aussprachewörterbuch fungiert das CMU pronunciation dictionary der Carnegie Mellon Universität für nordamerikanisches Englisch.

Prinzipiell folgt Keith Vertanen in seinem Entwurf dem Tutorial. Dabei hat er eine volle Ablaufsteuerung mittels Perl Skripten realisiert, die den ansonsten händisch zu erledigenden Anpassungsschritt übernehmen. Die Ablaufreihenfolge aller HTK Tools und Perl Skripte wird mittels Bash Skripts erledigt.

Sein System erreicht mit einem word-internal Triphonmodell eine WER von 7,85% und mit dem cross-word Triphonmodell eine WER von 6,91% .

[11] Dem System fehlt die Möglichkeit, es leicht auf andere Datenbanken zu portieren. Außerdem kann man das System nur sprecher-unabhängig trainieren. Eine Adaption auf andere Sprecher ist nicht vorgesehen. Außerdem kann man in den Ablauf nur schwer eingreifen. Eine Evaluation zwischen Modellgenerationen ist nicht vorgesehen. Diskriminatives Training wurde auch nicht implementiert.

3.1.3. SpeechDat-II Referenz-Spracherkenner - Refrec

Der Speechdat-II Referenz Spracherkenner ist ein Erkennerbausatz, der ebenfalls, basierend auf dem HTK Tutorial, für alle Speechdat Datenbanken konzipiert ist. Im Unterschied zum Recipe ist Refrec komplett in Perl geschrieben - Aufrufe der HTK Tools erfolgen von Perl aus. Interessant ist die zweistufige flat Bootstrapping Prozedur, die in Version 0.94 eingefügt wurde und signifikante Performanzverbesserungen mit sich gebracht haben soll. [1] Portierbarkeit auf andere Datenbanken, freie Konfigurierbarkeit, Evaluation zwischen den Modellgenerationen, Sprecher Adaption und diskriminatives Training fehlen aber auch in diesem Bausatz.

3.2. Software Framework

Ähnlich wie das HTK Recipe ist auch das für diese Arbeit entwickelte Framework mittels Bash Skripten aufgebaut. Jedes Modul entspricht dabei einem Bash Skript im Rootverzeichnis. Die Programmierung mittels Shell Skripten hat sich deswegen angeboten, weil alle HTK Tools mit einem traditionellen Command-line basierenden Interface ausgestattet sind. Der Aufruf eines HTK Tools soll an dieser Stelle anhand des fiktiven HTK Tools HFoo demonstriert werden:

HFoo C config f 34.3 a s myfile file1 file2

Dabei sind alle optionalen Parameter durch den Bindestrich gekennzeichnet, alle notwendigen Parameter für den Aufruf eines Tools werden hinten angereiht. Durch das Verwenden eines Configfiles kann man sich Parameter einsparen.

Obwohl das Programmieren per Shell antiquiert wirkt, liegen die Vorteile auf der Hand. Alle Zwischenschritte liegen als Textdateien vor. Diese können auch mit anderen Programmen analysiert oder sogar verändert werden, bevor sie mit den HTK Tools weiterverarbeitet werden. Abbildung 3.1 zeigt, wie das HTK Toolkit verwendet wird.

Alle Operationen, die das Editieren eines Textfiles erfordern, sind schließlich im Framework über Perl/Python Skripten automatisiert, die dann vom jeweils laufenden Shell Skript zwischen den HTK Tools aufgerufen werden.

3.2.1. Struktur

Im Rootverzeichnis des Frameworks liegen die einzelnen Module des Erkenners:

- loop.sh: Schleife über alle Sprecher für cross-speaker Evaluation
- go.sh: zeigt ein komplettes Trainingsrezept für einen einzelnen Sprecher
- prep_atc.sh: zeigt, wie man die ATCOSIM Datenbank für die Spracherkennung aufbereitet
- prep_data.sh: teilt die Daten in Trainings, Development und Testset (inkl. MLF und SCP Files)
- train_mono.sh: Monophon Training
- mono_mixup.sh: optionales Monophon Mixup (Erhöhung der Mixtures)

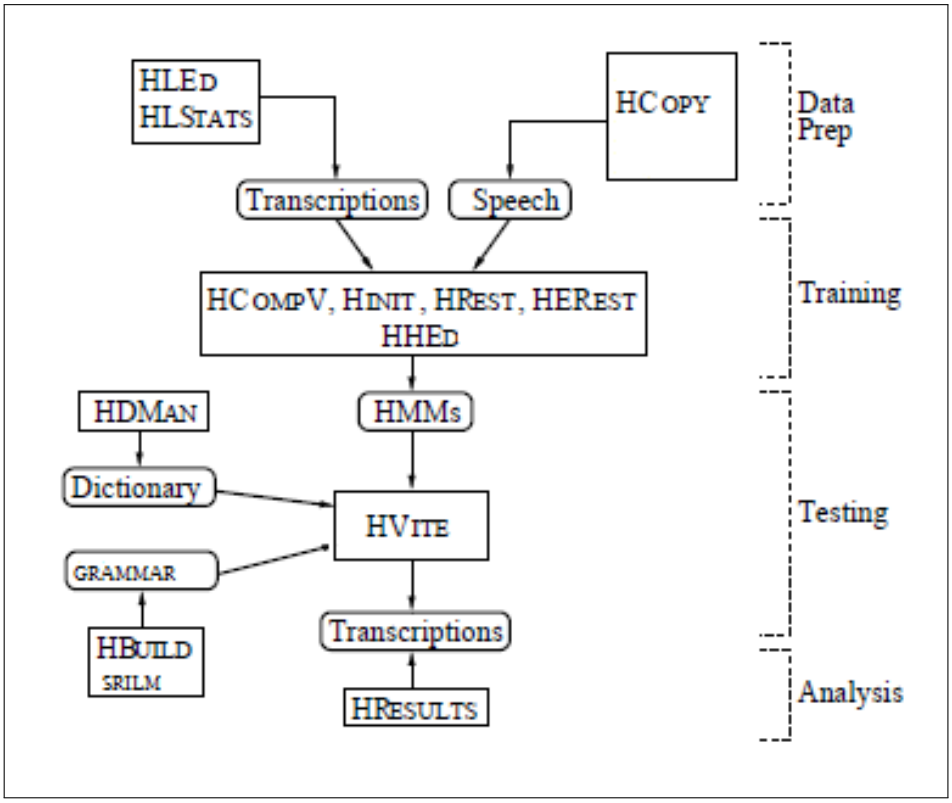

Abbildung 3.1.: Übersicht über die Verwendung von HTK[10]

- eval_mono.sh: Evaluierungsfunktion für Monophone
- train_tri.sh: Triphon Training (benötigt Monophonmodell)
- train_tied.sh: erzeugt Tied-State Triphones aus einem Triphonmodell
- eval_at.sh: finales Evaluierungsskript für Erkennungsrate
- train_mixup.sh: Skript, um die Anzahl der Gaußschen Mixtures bei Tied-state Triphonen zu erhöhen
- adapt.sh: Skript für Sprecher Adaption
- record.sh: zeigt, wie man per Konsole aufnimmt
- run_live.sh: Skript zur Demonstration von live input
- train_disc.sh: Skript, welches ein diskriminatives Nachschätzen der Parameter ermöglicht

Zusätzlich zu diesen Modulen liegen die Konfigurationseinstellungen für den Ablauf in der Datei env.sh.

Die gesamte Ordner Struktur des Frameworks sieht folgendermaßen aus:

- common: Konfigurationsfiles
- log: Log Files
- prep: Output von prep_atc und prep_data
- output: Output aller Trainingsskripte (= Modellgenerationen)

Die Konfigurationsfiles für die HTK Tools liegen im Ordner Common, die Log Files werden für jedes einzelne Tool im Ordner Log Files gespeichert. Die aufbereitete Datenbank (MFCC Files, MLF Files, Wörterbuch, Sprachmodell) liegt nach der Verarbeitung im Ordner prep.

Der eigentliche Output jedes Erkennungstools liegt schließlich im Ordner output, in dem dann Unterordner für jede Modellgeneration erstellt werden. Dabei ist hmm1 bis hmm9 das Monophonmodell, hmm10 bis hmm15 das Triphonmodell. Weitere Modellgenerationen, wie für die Adaption, Erhöhung der Anzahl der Mischkomponenten oder für das diskriminative Training liegen in eigenen Unterverzeichnissen hmmXX.

3.3. Datenvorbereitung

Bevor ein Spracherkenner für eine vorhandene Datenbank trainiert werden kann, muss diese entsprechend aufbereitet werden. Dies erledigt das Skript prep_atc.sh, welches dies für die ATCOSIM Datenbank demonstriert. Für andere Datenbanken muss dieses Skript von Hand modifiziert werden. Nach einem Durchlauf von prep_atc.sh liegen alle Wave Samples der ATCOSIM Datenbank als MFC Feature Vektoren vor und es existiert eine Datei, welche die Pfade zu allen Transkriptionen enthält. Außerdem werden das Sprachmodell und ein Wörterbuch erstellt. All das wird im Ordner prep abgelegt.

3.3.1. Wörterbuch

Mit der ATCOSIM Datenbank wird eine Liste der auftretenden Wörter mitgeliefert, was die Erstellung eines Wörterbuchs vereinfacht. Jedoch ist die gesamte Datenbank nur auf Wortebene und nicht auf Phonemebene transkripiert. Die für die Spracherkennung erforderliche Transkription auf Phonemebene muss manuell über die Transkription der Einzelwörter durchgeführt werden. Dazu wurde das CMU Dictionary der Carnegie Mellon Universität für nordamerikanisches Englisch zur Hilfe genommen. Dieses Wörterbuch besteht aus 39 Phonemen, welche in Anhang A inklusive Beispiel aufgelistet sind.

700 der vorhandenen 850 Wörter konnten damit in die Phonemebene überführt werden. Die verbleibenden 150 Wörter mussten manuell transkripiert werden. (Vom Autor ohne linguistischen Background) Beispiele:

- AIRALS EH R AE L S
- AIRFRANS EH R F R AE N S
- AIRTAS EH R T AE S
- LUXAIR L AH K S EH R
- SUNWING S AH N W IH NG

Mit dem Wörterbuch können dann MLF (Master Label) Transkriptionen der Wave Files auf Phonemebene durch Einsetzen aus der Wortebene erzeugt werden.

3.3.2. Sprachmodell

Im Framework ist grundsätzlich nur die Verwendung von stochastischen Grammatiken vorgesehen. (n-gramme) Diese Modelle werden direkt mit einem Tool (ngram-count [12]) aus den Trainingsdaten erstellt. Zu beachten ist, dass der unter HTK verwendete Viterbi Dekoder HVite mit einer textuellen n-gram Grammatik nicht arbeiten kann, sondern ein Wortnetzwerk benötigt, welches mit HBuild erstellt wird. Jedoch unterstützt HBuild nur n-gram Modelle bis zur Ordnung 2, sprich Bigrammmodelle. Wobei, wie bereits in der Theorie erwähnt, der Viterbi Algorithmus mit Modellen höherer Ordnung sowieso nicht umgehen kann. Will man komplexere Sprachmodelle verwenden, arbeitet man mit dem Large Vocabulary Decoder HDecode anstatt mit HVite. HDecode kann auch direkt mit dem von n-gram count erzeugten Sprachmodell arbeiten. Jedoch unterstützt HDecode keine Wort-Internen Triphonmodelle, weshalb es im Framework so gelöst ist, dass bei Verwendung eines Wort-Internen Modells HVite anstatt HDecode verwendet wird.

3.4. Trainingsprozedur

3.4.1. Monophon Modell

Der Ablauf des Trainings eines Spracherkenners mittels HTK entspricht einer sukzessiven Verbesserung bestehender Modellparameter über die Baum-Welch Gleichungen. (HERest) Zwischendurch werden mittels manueller oder toolbasierter Eingriffe Änderungen am Modell vorgenommen und danach wird wieder mit HERest verbessert. Bevor jedoch damit begonnen werden kann, ist die Erstellung eines initialen Modells notwendig. Zur Erstellung dieses initialen Modells gibt es zwei Strategien: Bootstrapping und Flat Start. Beim Bootstrap Training werden phonetisch segmentierte Dateien benötigt und aus denen wird mit dem Viterbi Algorithmus jedes Phonem HMM individuell initialisiert. Dazu werden die Tools HInit und HRest verwendet, um die Segmente der Sprache, welche zu dem gerade bearbeiteten Phonem gehören, isoliert zu trainieren.

Einfacher ist die Verwendung einer Flat Start Prozedur. Dabei wird für jede Äußerung ein Super-HMM als Summe der Einzel-HMMs erstellt. Dieses wird trainiert und dann wieder auf die Einzel-HMMs zurückgerechnet. [10] Da bei vielen Datenbanken, wie auch bei der ATCOSIM Datenbank, keine phonetisch segmentierten Daten zur Verfügung stehen und zusätzlich zur Flat Start Prozedur dennoch die Möglichkeit, über Bootstrapping zu initialisieren, gewünscht wurde, stehen dafür folgende Möglichkeiten zur Verfügung: 1. Initialisierung mit der TIMIT Datenbank 2. two-stage bootstrapping mit der ATCOSIM Datenbank.

Bei Methode 1 wird mit der TIMIT Datenbank vorinitialisiert. Die TIMIT Datenbank ist speziell für die Entwicklung und Evaluierung von Spracherkennungssystemen am Massachusets Institute of Technology zusammen mit SRI und Texas Instruments Inc. entwickelt worden. Diese Datenbank ist so ausgerichtet, dass alle Phoneme in ausreichender Menge abgedeckt werden. [13] Deswegen eignet sich die TIMIT Datenbank besonders dazu die Phoneme für ein Monophonmodell zu initialisieren. Dabei werden mit der TIMIT Datenbank die Phoneme mittels Bootstrapping initialisiert und diese Phoneme werden als Modellgeneration 5 ins Training der ATCOSIM Datenbank übernommen.

Variante 2 ist etwas komplexer. Dabei wird zuerst ein Monophonmodell unter Verwendung eines Teils der Datenbank mit 4 Mixtures trainiert. Anhand dieses Monophonmodells wird eine Segmentierung der Trainingsdaten ermittelt. Diese Segmentierung (relativ ungenau) wird dann zur Initialisierung mit der normalen HInit/HRest Bootstrapping Prozedur verwendet.

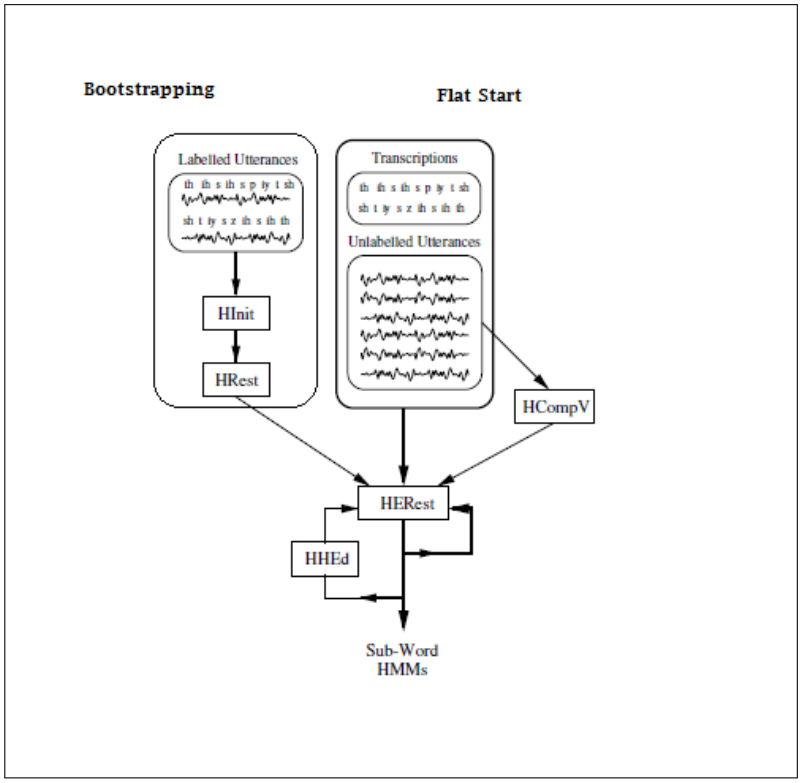

Abbildung 3.2.: Trainingsstrategien Bootstrapping vs. Flat Start in HTK[10]

In Abbildung 3.2 werden die Ergebnisse aller drei vorgestellten Initialisierungsvarianten gegenübergestellt.

3.4.1.1. Topologie

Die verwendete Modelltopologie (des initialen und aller folgenden Modelle) ist ein 3-state links-rechts Hidden Markov Modell. Für jedes Monophon existiert ein lineares Hidden Markov Modell. (Keine Zustände werden übersprungen.) Lediglich das Pausemodell (sp) und Ruhemodell (sil) für Stille entsprechen einem Links-Rechts Modell, in dem Zustände übersprungen werden können, um gut modellieren zu können, dass die Stille optional ist. (siehe Abbildung 2.5)

3.4.2. Vom Triphon Modell zum generalisiertes Triphon Modell

Das Triphon Modell wird über das Monophonmodell erzeugt, indem jedes Monophon in allen vorkommenden Umgebungen betrachtet wird. Diese Triphon-HMMs werden dann mit den Werten des eingeschlossenen Monophon-HMMs initialisiert. Dabei unterstützt das Framework sowohl wortinterne (word-internal) als auch wortübergreifende (cross-word) Triphonmodelle (siehe vorheriges Kapitel). Das Zusammenfassen von Zuständen (state-tying) erfolgt nach einigen Baum-Welch Iterationen mittels linguistischer Fragen bezüglich des Kontexts. (Clustering siehe vorheriges Kapitel) Nach insgesamt 15 Modellgenerationen liegt somit ein generalisiertes Triphone (eng. tied-state) Ausgangsmodell für weitere Verbesserungsmöglichkeiten im Ordner hmm15.

3.5. Evaluierung

Ein typischer Trainigsverlauf durch das Framework mit laufender Evaluierung auf den Trainingsdaten sieht in etwa so aus, wie in Abbildung 3.3 dargestellt. Dabei wird die Evaluierung ab Modellgeneration 5 (nach Abschluß der Initialisierung mit TIMIT) dargestellt. Man sieht eine sukzessive Verbesserung des Monophonmodells bis zur 9. Modellgeneration. In Modellgeneration 10 wurde das Monophonmodell in ein Triphonmodell transformiert. Modellgeneration 13 ist schließlich bereits ein generalisiertes Triphonmodell, welches bis zur Modellgeneration 17 weitertrainiert wird. Danach kommt zum Abschluss dieses Trainingsverlaufs ein Adaptionsverfahren zum Einsatz.

3.6. Optimierung der Erkennungsrate

Das generalisierte Triphonmodell dient als Basis für weitere Verbesserungen. Dabei stehen Skripte zur Erhöhung der Anzahl an Gaußschen Mischkomponenten (Mixtures), zur Sprecheradaption *(mittels einem kleinen Entwicklungsdatensatz (eng. development set) ein sprecherunabhängiges Modell auf die Charakteristiken eines bestimmten Sprechers anpassen)* und zum diskriminativen Nachschätzen der Modellparameter zur Verfügung.

3.6.1. Erhöhung der Anzahl an Mischkomponenten

Bei der Anwendung von Hidden Markov Modellen in der Spracherkennung entsprechen die Emissionswahrscheinlichkeiten $b_{i,j}$ einem kontinuierlichen Gaußschem Emissionsmodell. Wenn man diese Verteilung als gewichtete Summe mehrerer Gaußverteilungen (Mischkomponenten) annimmt, können die Trainingsdaten genauer gelernt werden. Allerdings erhöht sich damit auch die Anzahl freier Parameter, was einerseits zu einer längeren Trainings- und Testzeit führt und andererseits auch dazu führen kann, dass diese mit der Menge an zur Verfügung

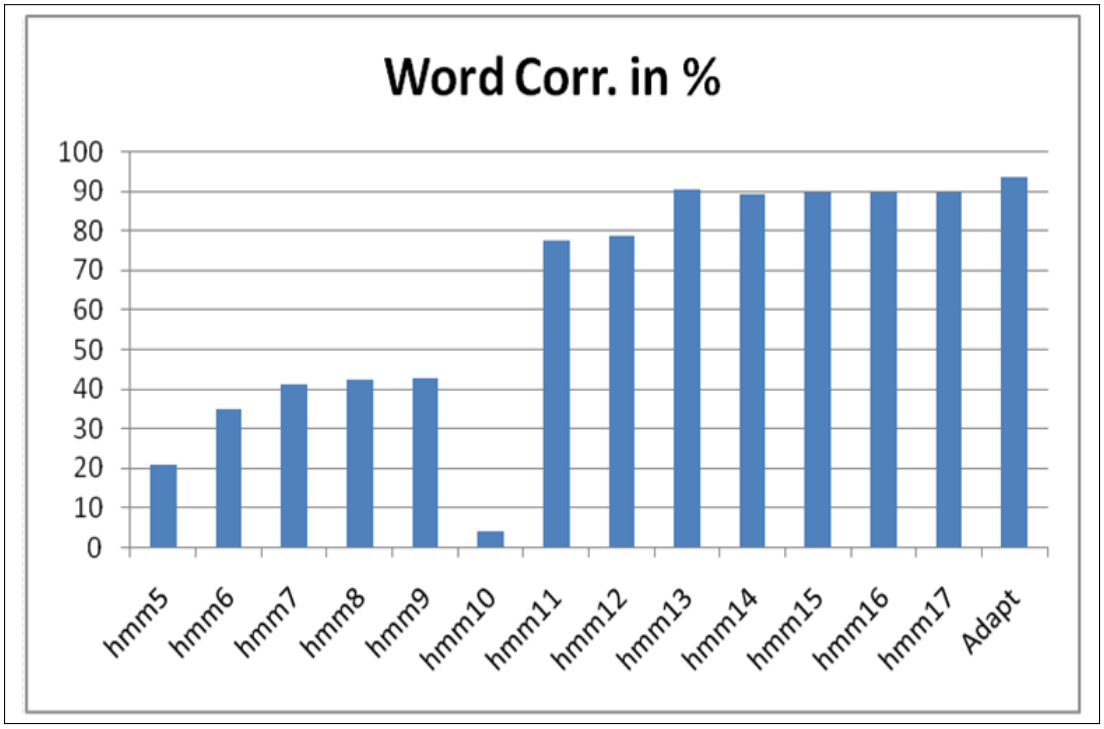

Abbildung 3.3.: Worterkennungsrate in Abhängigkeit von der Modellgeneration hmmXX

stehenden Trainingsdaten nicht robust geschätzt werden können. Um hier die richtige Balance zu finden, wird mit bis zu 32 Mischkomponenten trainiert.

Um die Anzahl an Mischkomponenten in HTK zu erhöhen, wird der Editor (bzw. HTK Editor HHEd) verwendet. Die Modellparameter werden danach durch 4 Baum-Welch Iterationen (HERest) ermittelt. Um ein genaueres Modell zu ermitteln, wird die Anzahl an Mischkomponenten stufenweise erhöht: Verdoppelung der Anzahl an Mischkomponenten gefolgt von 4 Baum-Welch Iterationen, bis die Anzahl an Mischkomponenten erreicht wird, dessen Performanz evaluiert werden soll.

3.6.2. Diskriminatives Training

Angelehnt an das HTK Tutorial werden mittels HDecode.mod und HLRescore die für das diskriminative Training notwendigen Numerator und Denumerator Gitter (eng. Lattices) erstellt. Danach kann erst das diskriminative Nachschätzen der Modellparameter in mehreren Iterationen mittels HMMIRest durchgeführt werden. (HMMIRest anstatt HERest)

3.6.3. Sprecher Adaptation

Die Sprecher Adaption wird auch mittels HERest im Adaptionsmodus durchgeführt. Zum Einsatz kommt dabei der Maximum Linear Regression Algorithmus angewendet auf die Development Daten des Sprechers, auf den adaptiert werden soll.

4. Experimente

In Kapitel 2 wurden die theoretischen Grundlagen zum Bau eines automatischen Spracherkenners erörtert und in Kapitel 3 dann das Design eines Spracherkenners für die Verwendung in der Flugverkehrssicherheit. In diesem Kapitel werden nun die Experimente, die zum Finden sehr guter Parameter für die ATCOSIM Datenbank nötig waren, erklärt. Die Endergebnisse werden danach mit den Ergebnissen der Eurocontrol aus dem Jahre 1998 (siehe Einleitung) verglichen.

4.1. Aufbau der Experimente

In den Experimenten wird mit der gesamten ATCOSIM Datenbank getestet. Zur Genauigkeitsabschätzung wird eine Cross-Validation verwendet. Dabei wird die ATCOSIM Datenbank in Lerndaten und Validationsdaten aufgeteilt. Im Lerndatensatz sind jeweils 9 der 10 Sprecher und im Validationsdatensatz dann der fehlende 10. Sprecher. Dies wird 10 mal wiederholt, sodass jeder Sprecher einmal getestet wird. So werden die Charakteristiken des zu testenden Sprechers nicht mitmodelliert und dennoch können alle Daten zum Trainieren und Testen verwendet werden. So wird der Einsatz als kommerzieller Spracherkenner simuliert, da der Kunde natürlich nicht in der Trainingsdatenbank sein kann.

Die folgenden Experimente sind so aufgebaut, dass die Basiserkennungsrate mit HTK Standardparametern sukzessive verbessert wird. Dabei wird der Einfluss einzelner Parameter auf die Worterkennungsrate untersucht. Ein optimaler Wert für einen Parameter wird dann für das nächste Experiment weiterverwendet, wenn der nächste Parameter untersucht wird.

Zum anschaulichen Vergleich werden die Einzelresultate in einem Matlab Boxplot zusammengefasst und so gegenübergestellt. Die Y-Achse entspricht dabei der Worterkennungsrate in % der korrekt klassifizierten Wörter.

4.2. Tuningmöglichkeiten

In Gleichung 2.1 wurde die Findung von $argmax_w P(w|X)$ als Dekodierung, die Berechnung von $P(X|w)$ als akustisches Modell und die Berechnung von $P(w)$ als Sprachmodell identifiziert. Dies sind auch die drei Ansatzpunkte zur Verbesserung der Erkennungsrate.

4.3. Untersuchung verschiedener akustischer Modellierungsstrategien

Im akustischen Modell wird, wie in Kapitel 2 und 3 erläutert, ein Hidden Markov Modell verwendet. Jedes Triphon erhält dabei ein eigenes Links-Rechts Hidden Markov Modell. Die Emissionsverteilung folgt einem Gaußschen Emissionsmodell mit variabler Anzahl von Mischkomponenten. Dabei wird nicht auf Wave Dateien, sondern auf aus den Wave-Dateien berechneten Merkmalsvektoren operiert. Die Zordnung von Triphonen zu Wörtern erfolgt über das Aussprachewörterbuch.

Die Verwendung von MFCC Merkmalsvektoren ist defacto 1. Wahl bei vielen Spracherkennungsanwendungen, [10] weshalb auf Experimente mit anderen Merkmalsvektoren verzichtet wurde.

Da das Triphonmodell über ein Monophonmodell generiert wird, hat auch das Tuning des Monophonmodells einen Einfluss auf die finale Erkennungsrate. Im Paper des htk Recipe [11] wurde festgestellt, dass beim Bau eines Spracherkenners auf der WSJ0 Datenbank Bootstrapping der Monophone mit der Timit Datenbank bessere Ergebnisse liefert als ein Flat Start mit der WSJ0 Datenbank.

4.3.1. Vergleich von Flat Start Initialisierung und Bootstrapping

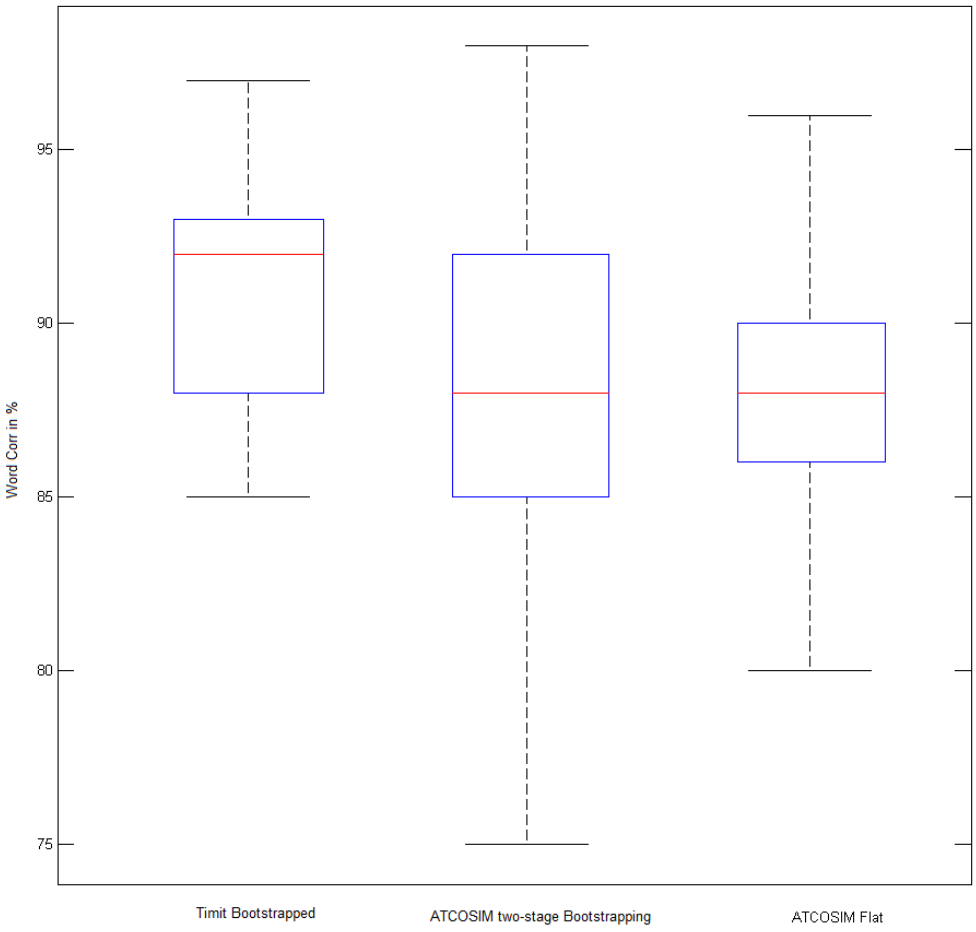

Abbildung 4.1.: Vergleich der finalen Erkennungsrate bei Verwendung von den zwei Bootstrapping Verfahren im Vergleich zu Flat Start

Im Unterschied dazu verwendet der Erkennerbausatz Refrec für Speechdat Datenbanken ein „zweistufiges (eng. two-stage) Bootstrapping“ Verfahren. Bootstrapping, was normalerweise nur für eine phonetisch transkribierte Datenbank verwendet werden kann, wird dabei auf Datenbanken, welche auf Wortebene transkribiert sind, adaptiert. [1] Dazu wird vorab

mit einem Teil der Datenbank ein Monophonmodell trainiert. Am Ende wird der Viterbi Algorithmus auf die Trainingsdaten zur Dekodierung verwendet und daraus eine phonetische Transkription erstellt. (first stage) Im zweiten Schritt (second stage) wird dann diese Segmentierung in den Viterbi Algorithmus von HInit zum Bootstrapping verwendet. [10] Um die Trainingszeit zu verkürzen, wurde getestet, welche Menge an Trainingsdaten ausreichend ist, um die maximale Erkennungsrate für dieses Verfahren zu erreichen. Dabei wurde festgestellt, dass eine Größe von 1000 Äußerungen ausreichend ist.

Aus Abbildung 4.1 sieht man, dass Bootstrapping mit der TIMIT Datenbank leichte Vorteile gegenüber dem zweistufigen Bootstrapping mit der ATCOSIM Datenbank oder einem Flat Start mit sich bringt. Das Ergebnis ist im Mittel besser und die Varianz ist kleiner.

Das zweistufige Bootstrapping Verfahren ist also für die ATCOSIM Datenbank weniger geeignet. Begründen lässt sich das damit, dass bei den Speechdat Datenbanken a-priori Informationen zur Verfügung stehen, welche Teile der Datenbank sich durch eine große Aussprachevielfalt und phonetische Ausbalanciertheit auszeichnen, sowie durch eine fehlerfreie klare Aussprache. Während die Speechdat Datenbank eine Sprachdatensammlung speziell für die Entwicklung von Spracherkennern ist, ist die ATCOSIM Datenbank nur eine Aufnahme aus der Flugverkehrssicherheit, wobei natürlich all diese vereinfachten Bedingungen wie Studiolaboraufnahme und klare Aussprache nicht gegeben sind. Deswegen kann auch unmöglich ein kleiner Ausschnitt aus der Datenbank gewählt werden, welcher repräsentativ für die englische Sprache von 10 Sprechern ist.

Fazit: Die TIMIT Datenbank mit 6300 Äußerungen von 630 Sprechern liefert wie erwartet die beste Monophoninitialisierung.

4.3.2. Wort-Interne im Vergleich zu wortübergreifenden Triphonmodellierungsverfahren

Wenn man wortübergreifende (engl. cross-word) Verschleifungseffekte berücksichtigt, erhöht sich die Anzahl der Triphone von 2000 auf 6000. Keith Vertanen bestätigt in seinem technischen Report die Aussage der Literatur, dass im Allgemeinen mit wortübergreifenden Triphonen bessere Erkennungsraten erreicht werden können. [11] In Abbildung 4.2 werden diese beiden Modellierungsverfahren gegenübergestellt.

Das Experiment zeigt, dass diese allgemeine Annahme für die ATCOSIM Datenbank nicht gilt. Das lässt sich mit der Datenbankgröße begründen. Bei 6000 Hidden Markov Modellen für 8000 Sampleaufnahmen steht einfach nicht genug Trainingsmaterial zur Verfügung, um 6000 Modellparameter robust schätzen zu können.

4.3.3. Variable Anzahl von Mischkomponenten sprecherunabhängig und sprecherabhängig nach Adaption

Im Paper zum htk recipe [11] sowie im Paper zum refrec Bausatz [1] wird festgestellt, dass sich für Modelle mit bis zu 32 Mischkomponenten Verbesserungen in der Wortfehlerrate einstellen. Im Experiment wird dies für die ATCOSIM Datenbank untersucht. (siehe Abbildung 4.3)

Getestet wurde mit 2, 4, 8, 16 und 32 Mischkomponenten - jedoch konnte bereits ab 2 Mischkomponenten keine Verbesserung durch eine Erhöhung erzielt werden, wie auch die Gegenüberstellung von 2 zu 32 Mischkomponenten zeigt. Begründet kann dies mit dem verhältnismäßig kleinen Datensatz werden. Eine Verdoppelung der Anzahl der Mischkomponenten verdoppelt auch die Anzahl freier Parameter im Emissionsmodell, welche dann mit dem kleinen Datensatz nicht mehr robust geschätzt werden können. Außerdem schlägt sich die Anzahl

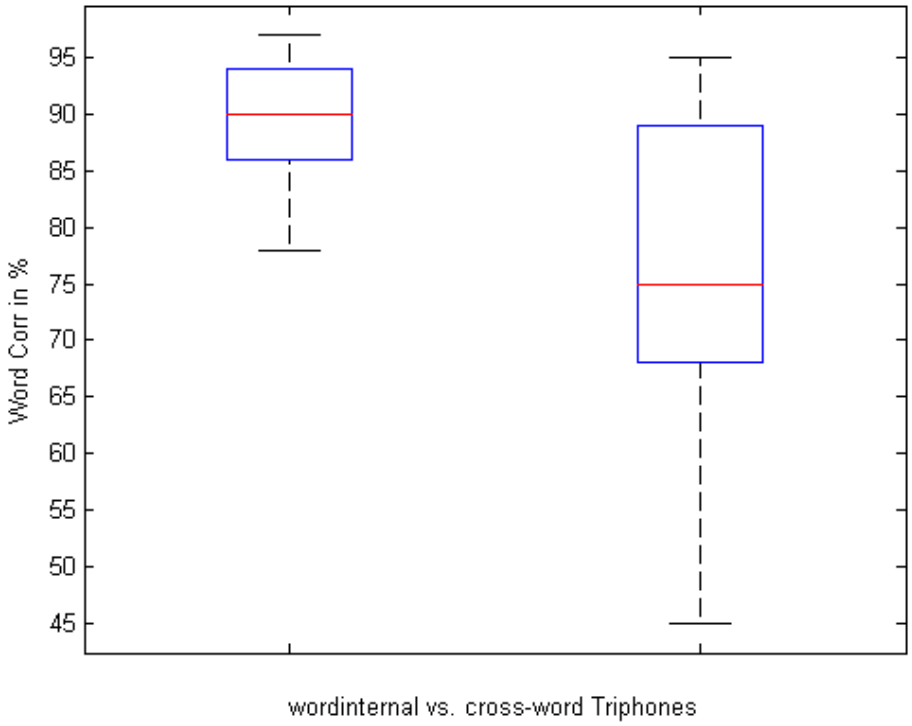

Abbildung 4.2.: Erkennungsraten von word internal Triphones im Vergleich zu cross-word Triphones

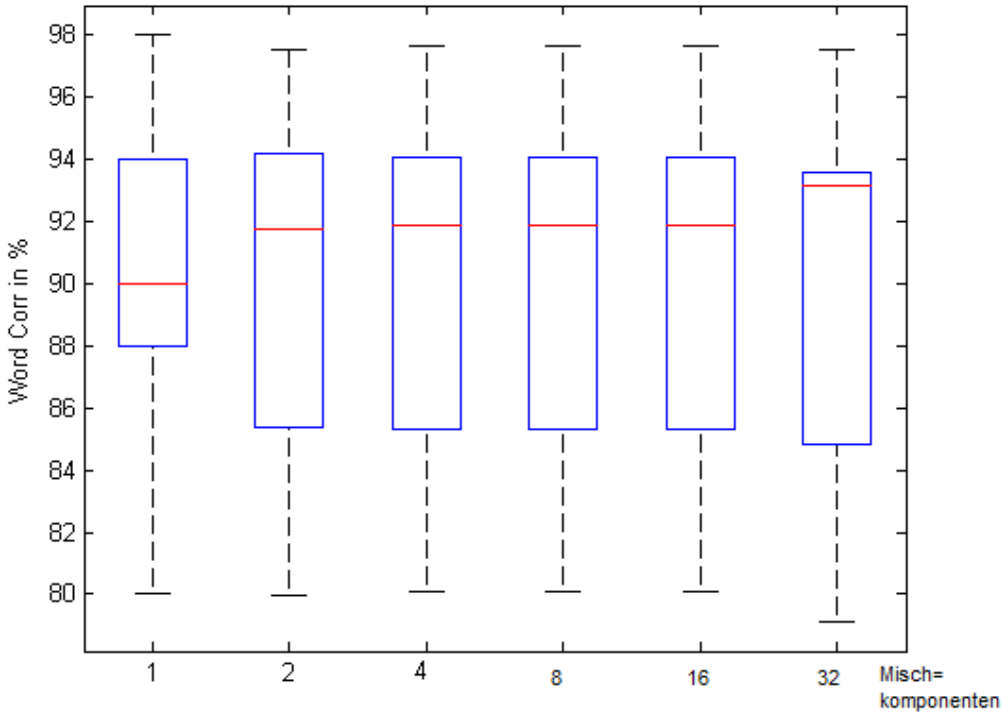

Abbildung 4.3.: Erhöhung der Mischkomponenten sprecherunabhängig trainiert

der Mischkomponenten massiv in der Reaktionszeit nieder, sodass mit einer größeren Anzahl an Mischkomponenten keine Echtzeitspracherkennung möglich ist.

4.3.4. Sprecheradaption

Die Verwendung verschiedener Adaptionsverfahren (mllrmean, mllrcov, cmllr) hatte keinen Einfluss auf die Erkennungsrate. Im folgenden Experiment soll der Gewinn durch Adaption bei 2 Mischkomponenten untersucht werden.

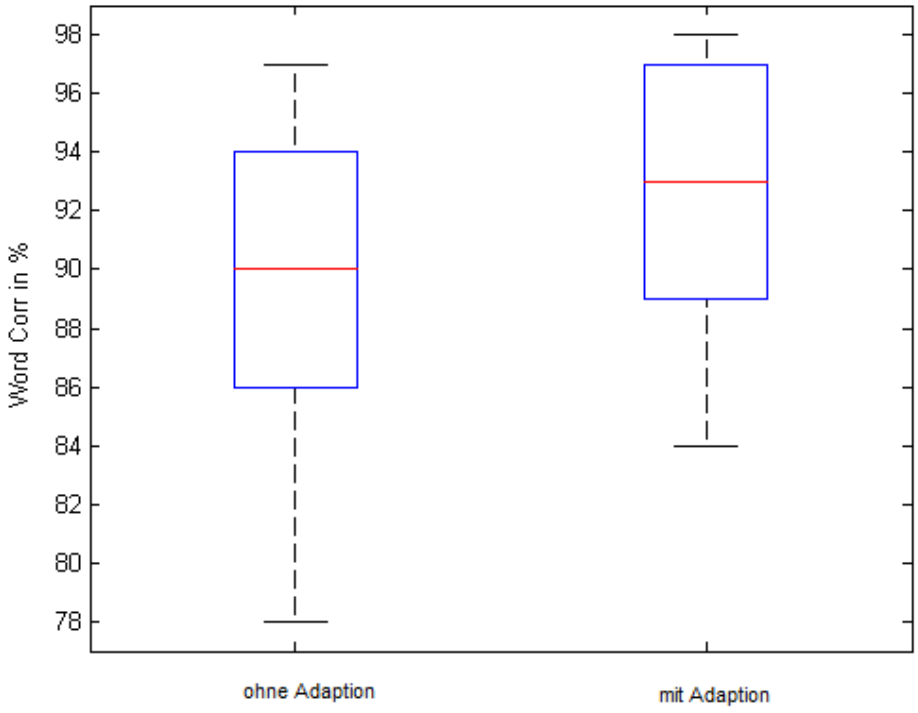

Abbildung 4.4.: Verbesserung durch Adaption

Durch die Sprecheradaption, die eine Anpassung auf die Charakteristik des zu testenden Sprechers bedeutet (der getestete Sprecher ist nicht im Testdatensatz), wird eine deutliche Steigerung der Erkennungsrate erreicht. Das Adaptionsverfahren, das hier zum Einsatz kommt, ist CMLLR. (siehe 3.6.3) Um Material für das Adaptionsverfahren zu bekommen, wird der Validationsdatensatz (welcher den gerade zu testenden Sprecher enthält) noch einmal im Verhältnis 3:1 in Validationsdatensatz und Entwicklungsdatensatz (eng. development set) geteilt. Die Entwicklungsdaten werden dazu verwendet, das Modell auf die Charakteristiken des zu testenden Sprechers zu adaptieren und mit dem reduzierten Validationsdatensatz wird danach dieser Sprecher getestet. Aus Abbildung 4.4 ist erkennbar, dass durch Adaption im Mittel ein Gewinn von 3% der Worterkennungsrate erzielbar ist.

Aus Tabelle 4.1 sind die Auswirkungen der Adaption auf einzelne Sprecher abzulesen. Interessant ist hier auch die Abhängigkeit von der Größe des Datensatzes, der von einem einzelnen Sprecher zur Verfügung steht. So führt die Adaption beim Sprecher mit der ID gf1 zu einer leichten Verschlechterung, während zum Beispiel der Sprecher sm2 einen massiven Gewinn bezüglich der Worterkennungsrate durch Adaption erfährt.

ID	#Train-	#Dev-	#Testäußerungen	WER in % vor	und nach Adaption
sm1	875	73	219	89%	98%
sm2	1386	116	346	81%	97%
sm3	606	51	151	90%	96%
sm4	799	72	291	85%	93 %
gf1	179	15	44	86%	84%
gm1	288	24	72	84%	89%
gm2	284	24	70	91%	88%
zf1	1287	107	322	85%	98%
zf2	1304	109	326	94%	97%
zf3	479	40	119	90%	91%

Tabelle 4.1.: Auswirkungen der Adaption auf einzelne Sprecher

4.3.5. Einfluss des Aussprachewörterbuches

Das verwendete Wörterbuch CMU Dict ist ein Aussprachewörterbuch für amerikanisches Englisch. Die 10 Sprecher sind von der französisch und deutschsprachigen Schweiz sowie aus Deutschland. Deshalb gibt es eine große Varianz innerhalb der Aussprache mancher Wörter. Deswegen wäre es möglich die Erkennungsrate zu verbessern, indem Aussprachealternativen eingebaut werden. Um dies für die ATCOSIM Datenbank durchzuführen, wäre ein Linguist nötig, der die nötigen Transkriptionen durchführt. Dies ist im Rahmen dieser Arbeit nicht vorgesehen. Der Einfluss des Vokabulars zeigt sich bei einem Experiment, in welchem das Vokabular im Umfang von 900 Wörter reduziert werden soll. Dazu wurde eine Liste von erlaubten Wörtern erstellt. Erlaubte Wörter sind Ortsangaben, Zählwörter, Funkkennungen, flugverkehrsspezifische Navigationsangaben, ICAO Buchstabiertafel (sprich ICAO Standard konforme Phrasen). Ein Perl Skript des Frameworks filtert dann diejenigen Aufnahmen heraus, die Wörter enthalten, die nicht in der jeweiligen Liste sind. Um genügend Trainingsmaterial zu haben, wurde die Liste dann noch etwas erweitert. Die vollständige Liste des mittleren Vokabulars befindet sich in Anhang B und die entfernten Worte (sprich die Differenz) in Anhang C

Aus der Gegenüberstellung (siehe Abbildung 4.5) lässt sich erkennen, dass durch die Reduktion des Vokabulars eine bessere Erkennungrate erzielt werden kann.

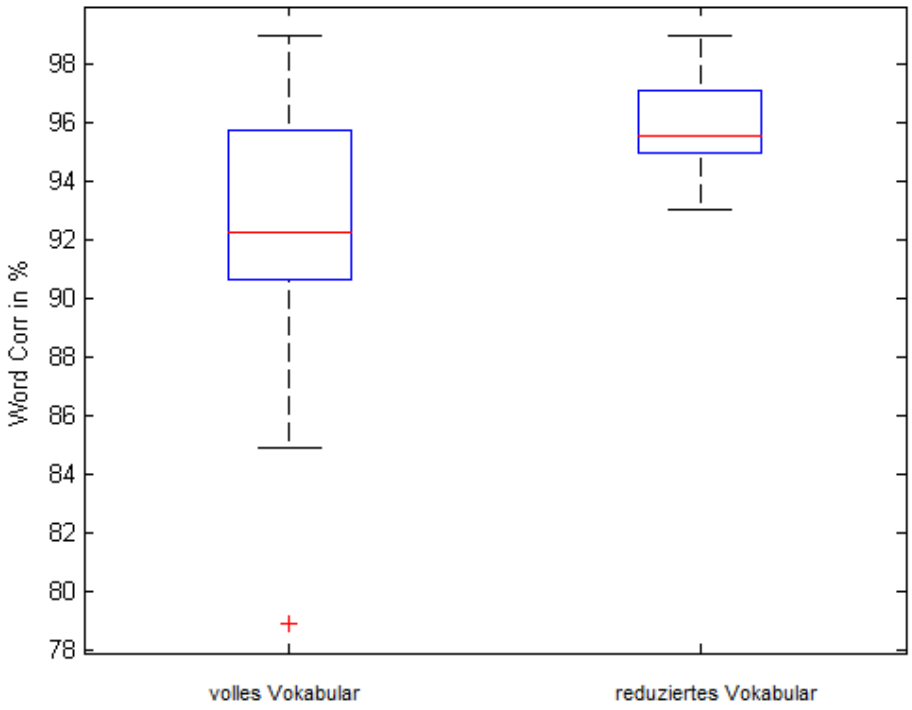

Abbildung 4.5.: Erkennungsrate ohne Vokabularreduktion im Vergleich zu mittlerem Vokabular (200 Wörter)

4.3.6. Trennung von Männern und Frauen

Da Männer und Frauen einen unterschiedlichen Vokaltrakt haben, wird in diesem Experiment untersucht, ob sich die Erkennungsrate verbessert, wenn man Männer und Frauen trennt.

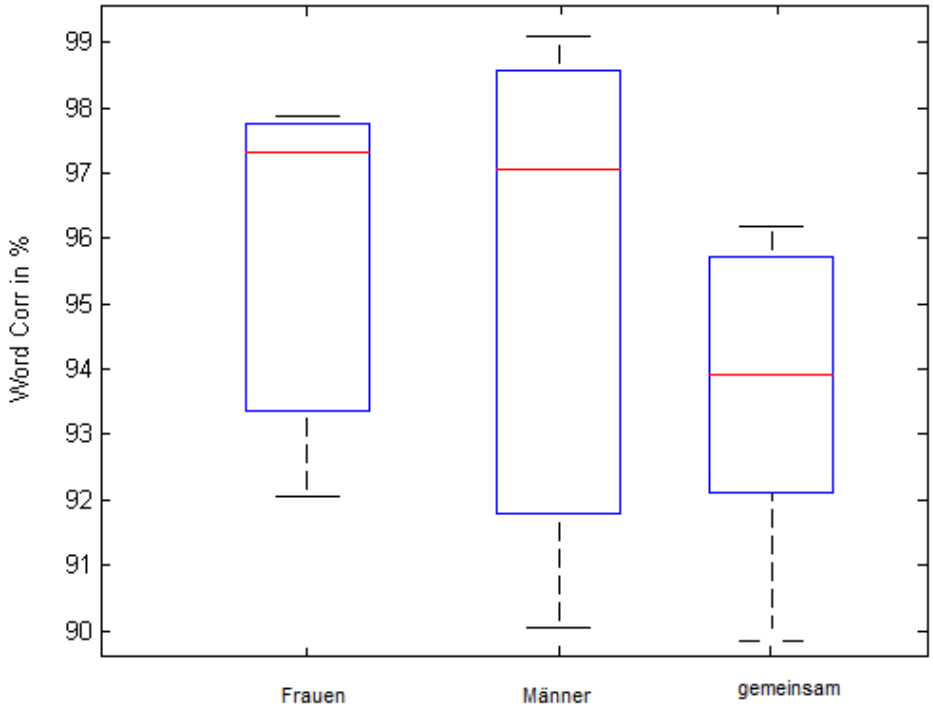

Abbildung 4.6.: Frauen und Männer bei getrenntem Training

In Abbildung 4.6 wird die ATCOSIM Datenbank in Männer und Frauen geteilt, beide Teile werden unabhängig voneinander trainiert und getestet. Wie man sieht, führt das tatsächlich zu einer deutlichen Verbesserung im Vergleich zu einem gemeinsamen Training, wobei die Frauen eine etwas bessere Worterkennungsrate erreichen.

4.4. Untersuchung verschiedener Sprachmodelle und Dekodierungsansätze

In Kapitel 1.1.1 wurde festgestellt, dass es nicht viel Sinn macht, eine kontextfreie Grammatik nach dem ICAO Standard zu konstruieren. Deswegen bleibt nur die Verwendung einer stochastischen Grammatik. Da zur Dekodierung in HTK nur der Viterbi Dekoder verwendet werden kann, bietet sich auch nur die Verwendung eines Bigramm-Modells an. Um zu sehen, welcher Gewinn durch das Sprachmodell erzielt wird, wird in Abbildung 4.7 die Performanz eines Unigramm Modells evaluiert.

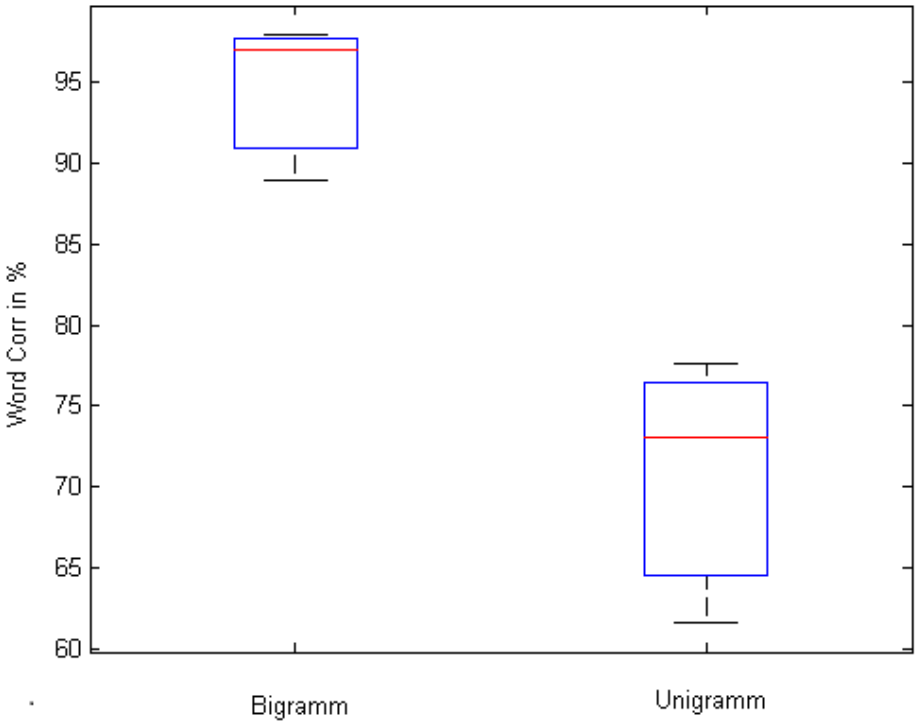

Abbildung 4.7.: Einfluss des Sprachmodells

Man sieht, dass das Bigramm Modell einen Gewinn von 25% im Vergleich zum Unigramm Modell erzielt. Denkbar wäre die Verbesserung mittels Rescoring, welches auch Trigramm Modelle unterstützt.

Mit dem seperat erhältichen Large Vocabulary Decoder HDecode sind im Gegensatz zum Standard Viterbi Algorithmus auch Trigramm Sprachmodelle möglich. Da das standardmäßig verwendete Bigramm Modell im Vergleich zum Unigramm 25% Performanz bringt, ist zu untersuchen, ob ein Trigrammmodell noch weitere Steigerungen erzielen kann. Jedoch kann HDecode nur mit Cross-Word Triphonmodellen arbeiten. Weshalb eine direkte Gegenüberstellung von Trigramm Modell mit cross-word Triphonen und einem Bigramm Modell mit wortinternen Triphonen nicht sinnvoll ist. In Abbildung 4.8 werden die Sprachmodelle miteinander verglichen.

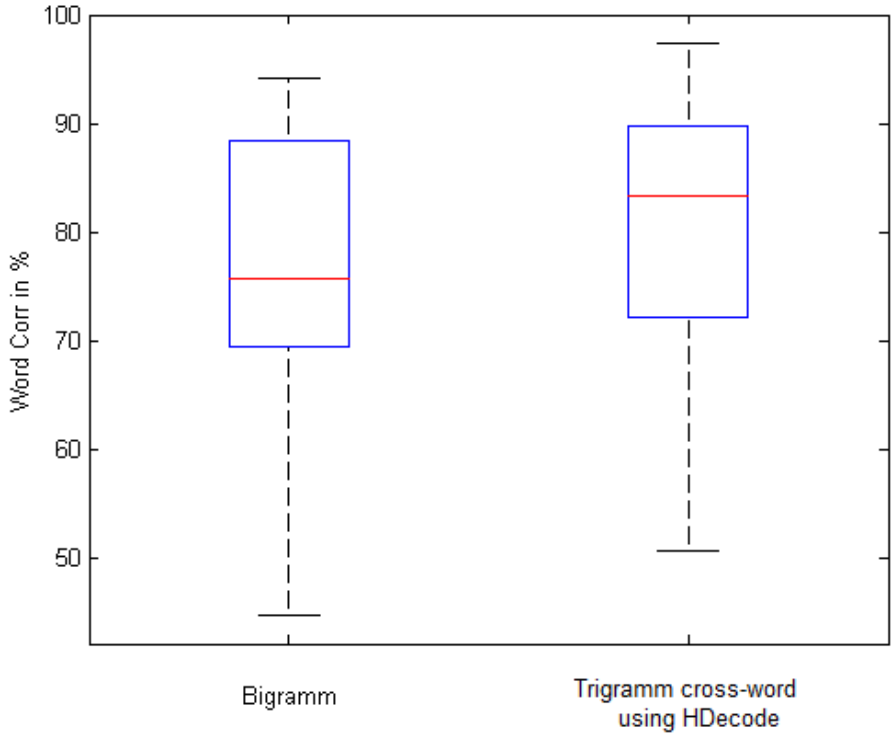

Abbildung 4.8.: Erkennungsrate Bigramm vs. Trigramm x-word Modell

Das Trigramm Modell bringt Gewinne, allerdings reicht es nicht, um an die Performanz des wortinternen Triphonmodells mit Bigrammgrammatik heranzukommen.

4.5. Optimierung durch diskriminative Nachschätzung der Parameter

Alle bisherigen Ergebnisse wurden mit dem Maximum Likelihood Verfahren erzielt, welches sich seit vielen Jahren als Standardverfahren in der Spracherkennung etabliert hat. Bei gegebenen Trainingsdatensätzen und ihren korrespondierenden Klassenzuordnungen werden dabei die Modellparameter so geschätzt, dass die Klassenzugehörigkeitswahrscheinlichkeit maximiert wird. Jedem Modell werden nur die Musterfolgen zugeordnet, die seiner Klasse entsprechen. Auch werden nur die Produktionswahrscheinlichkeiten für eine korrekte Transkription w_r genutzt. Wie die Produktionswahrscheinlichkeit für eine Wortfolge $w \neq w_r$ aussieht, bleibt unberücksichtigt. Das Training jedes Modells ist daher unabhängig vom Training der anderen.

Die Klassenüberschneidungen werden nicht berücksichtigt. Der Erkennungsprozess lässt sich jedoch verbessern, wenn die Produktionswahrscheinlichkeiten der Wortfolge w_r auch zwischen konkurierenden Wortfolgen maximal sind. Bei den diskriminativen Verfahren werden daher auch klassenfremde Trainingsdaten zur Parameterschätzung herangezogen, um somit eine bessere Klassentrennung zu erreichen. [7] Im Hidden Markov Toolkit werden drei unterschiedliche diskriminative Lernverfahren unterstützt: Maximum Mutual Information (MMI), Minimum Phone Error (MPE) und Minimum Word Error (MWE). Eine genaue Evaluierung und Optimierung mit diskriminativen Verfahren und aller dabei einstellbarer Parameter würde den Rahmen dieser Arbeit sprengen und wurde deswegen nicht gemacht. Der Vollständigkeit halber wurde ein Experiment mit diskriminativer Nachschätzung der HMM Parameter unter Verwendung von Standardparametern durchgeführt. (siehe auch Kapitel 3.6.3)

Nach der Studienarbeit von Ingo Siegert [7] kann die diskriminative Nachschätzung der Parameter auf der WSJ0 Datenbank (welche auch im htk-recipe verwendet wird) eine Steigerung der Erkennunsrate um bis zu 5% bei Singlemix Modellen erzielen. Jetzt wird untersucht, ob die gleichen Parameter auch bei der ATCOSIM Datenbank eine Verbesserung der finalen Erkennungsrate erzielen können.

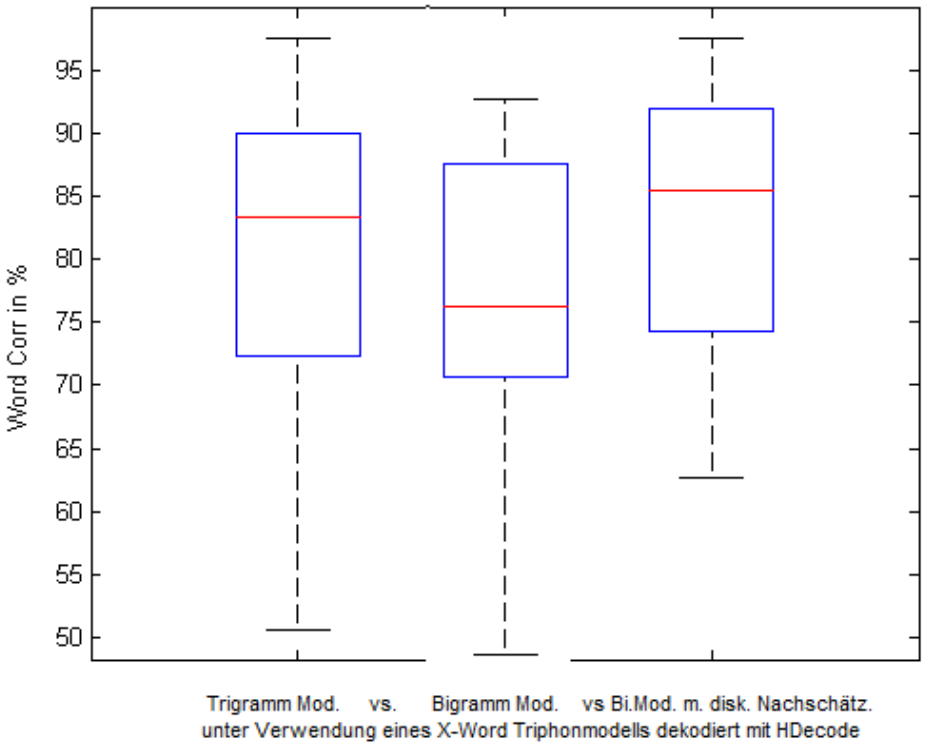

Abbildung 4.9.: Verbesserung durch diskriminatives Training

Aus Abbildung 4.9 lässt sich erkennen, dass ein Bigramm Modell nach Anwendung von diskriminative Nachschätzung bessere Ergebnisse liefert, allerdings können die bisherigen Bestwerte unter Verwendung von HVite und normalem Maximum Likelihood Verfahren nicht erreicht werden. Zu den Gründen ist folgendes zu sagen: Der seperat erhältliche Large Vocabulary Dekoder HDecode, der zum diskriminativen Training verwendet werden muss, ist für große Sprachdatenbanken optimiert. Eine solche steht mit der ATCOSIM Datenbank nicht

zur Verfügung, sodass dieser schlechtere Ergebnisse liefert als der Standarddekoder HVite. Weiters kann bei HDecode kein wortinternes Triphonmodell verwendet werden, was für die Sprachdatenbank ATCOSIM ebenfalls bedingt durch die größere Anzahl freier Parameter bei kleiner Datenbankgröße als Nachteil zu werten ist. (siehe auch Kapitel 4.3.2) Außerdem wurden die Parameter nicht so genau evaluiert und optimiert wie zuvor beim Maximum Likelihood Verfahren. Das Potential von diskriminativen Training lässt sich also durchaus erkennen und dies ist ein Ansatzpunkt für weitere Optimierungen späterer Arbeiten!

4.6. Finale Erkennungsrate

Zusammenfassend lässt sich sagen, dass in den Experimenten auf der ATCOSIM Datenbank die besten Ergebnisse unter Verwendung eines 2-Mixture Gaußschen Emissionsmodell mit Sprecheradaption, getrenntem Training von Männern und Frauen, und einem wordinternen Triphonmodell basierend auf einem Bigramm Sprachmodell erzielen lassen.

Die finalen Erkennungsraten nach Sprechern geordnet sind in Tabelle 4.10 ersichtlich.

	finale Erkennungsrate in %		
	Word	Sent	Acc
gf1	92	51	91
gm1	91	55	89
gm2	94	48	93
sm1	99	90	99
sm2	96	78	94
sm3	97	81	96
sm4	96	77	95
zf1	98	85	97
zf2	97	76	97
zf3	92	57	89
	95,2	69,8	94

Abbildung 4.10.: finale Ergebnisse

Wenn man die Worterkennungsrate über alle getesteten Äußerungen anstatt das Mittel über alle Sprecher verwendet, ergibt sich eine Worterkennungsrate von $96,5\%$ und eine Satzerkennungsrate von 75%.

4.7. Vorschlag für konkret zu implementierendes System

Die erzielten Erkennungsraten sind durchaus praxistauglich. Um das System einsetzen zu können, ist ein skriptgesteuertes, kommandozeilenbasiertes System nicht ausreichend. Eine Möglichkeit um zu einem einsatzfähigen System zu kommen ohne einen kompletten Spracherkenner neu zu entwerfen ist es, die HTK Bibliotheken für C/C++ zu verwenden, womit die Funktionalität des Frameworks (siehe Kapitel 3) in einigen Wochen realisierbar wäre. Dazu müsste dann nur noch eine praxistaugliche Oberfläche entwickelt werden. Die Ergebnisse aus

Kapitel 4.3.4 zeigen die Vorteile eines sprecherabhängigen Modells. Deswegen empfiehlt es sich einige vorgegebene Sätze vom jeweiligen Benutzer (Fluglotse X) vorsprechen zu lassen, damit das System auf seine Charakteristiken optimal trainiert ist. Um die Trefferquote weiter zu erhöhen empfiehlt sich auch die Ausgabe einer Liste der besten Ergebnisse (n-best Liste, mit n=Anzahl der auszugebenden Sätze) der Erkennung anstatt nur des besten Treffers. Aus dieser Liste kann der Benutzer dann zum Beispiel mit einem Drop-Down Menü den richtigsten Eintrag der Spracherkennung auswählen. Jedoch sollte diese Liste nicht zu lange sein um den Fluglotsen nicht unnötig abzulenken. Ein praktischer Einsatz des Systems würde die erreichten Ergebnisse der Wort- und Satzerkennung weiter verifizieren und zeigen ob die erzielten $96,5\%$ Worterkennungsrate auch in der Praxis möglich sind. Mit relativer Sicherheit sind 80% erreichbar, da diese Mindesterkennungsrate auch mit schlechten Parametern immer erreicht werden konnte.

5. Zusammenfassung und Ausblick

Die Aufgabenstellung war, einen Spracherkenner für die Fluglotsen bei der Flugverkehrssicherheit zu entwerfen. Ausgangsmaterial war eine Untersuchung bei der Eurocontrol, bei dessen Studium sich die Schlussfolgerung aufdrängt, dass die Spracherkennung für ein so schwieriges Szenario nicht geeignet ist.

Um das zu widerlegen, wurden neue aufwendigere Experimente durchgeführt. Dazu wurde ein Framework entwickelt, womit freies Konfigurieren und automatisches Testen ermöglicht wurde. Durch den allgemeinen Entwurf kann dieses Framework auch für Experimente auf anderen Datenbanken verwendet werden.

Nach vielen Optimierungsexperimenten konnte schließlich eine Worterkennungsrate von 97% erreicht werden, was ein hervorragendes Ergebnis bedeutet. Wobei nicht unerwähnt bleiben soll, dass es bei den Ergebnissen einen positiven Bias gibt: Das Sprachmodell wird auch aus den Daten einer einzigen Aufnahmesession erstellt. Auch wenn Trainings- und Testdaten getrennt sind, so sind die zum Beispiel die Flugzeugnummern (Lufthansa four three five six) in Test- und Trainingsdaten identisch.

Um die hier erzielten Erkennungsraten auch in der Praxis zuverlässig erreichen zu können, ist allerdings viel mehr Trainingsmaterial notwendig, um einerseits das komplette Vokabular abzudecken und andererseits auch die Charakteristiken von mehr Sprechern und möglichen Umgebungen zu modellieren. So sind in der ATCOSIM Datenbank nur die Daten von Fluglotsen aus 3 verschiedenen simulierten Regionen (siehe Kapitel 1.2.2) enthalten. Zusätzlich wäre es wünschenswert, dass zumindest etwas repräsentatives Trainingsmaterial auf Phonemebene von einem Linguisten transkribiert zur Verfügung steht. Das würde Bootstrapping Training mit korrekten Transkriptionen anstatt mit näherungsweise aus dem Monophonmodell ermittelten, ermöglichen. Außerdem gehört zu einem Sprachtraining ein Wörterbuch, welches regionale Einfärbungen der englischen Aussprache hinreichend berücksichtigt. Hier konnte nur ein Wörterbuch für amerikanisches Englisch verwendet werden wobei die Trainingssprecher aber deutscher bzw. Schweizer Herkunft waren. Zusätzlich mussten noch Transkriptionen vom Autor (Nicht-Linguist) erstellt werden, um alle Wörter der Datenbank komplett abzudecken.

Eine größere Datenbank würde auch die Verwendung von cross-word Triphones mit dem Large Vocabulary Dekoder HDecode sinnvoll machen und darauf aufbauend kann man dann die Erkennungsraten mit diskriminativen Training weiter verbessern.

Zusammenfassend lässt sich schlussfolgern, dass die technischen Voraussetzungen für einen hervorragenden Spracherkenner zur Verwendung in der Flugverkehrssicherheit durchaus gegeben sind.

A. Phoneme des CMU Wörterbuches

Phoneme	Beispiel	Transkription
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

Tabelle A.1.: CMU Pronunciation Dictionary

B. Wortliste mittleres Vokabular

ABLE
ABOUT
ABOVE
ACCEPT
ACOSTA
ADDRESS
ADRIA
ADVISED
AERO
AFTER
AGAIN
AGERI
AGREE
AHEAD
AIR
AIRFRANS
AIX
ALBIX
ALITALIA
ALL
ALLOWED
ALMOST
ALRIGHT
ALSO
AND
ANOTHER
ANY
ANYHOW
APPROVAL
APPROVED
APPROXIMATELY
ARBOS
AS
ASK
AT
AUTUN
AWHILE
BACK
BALE
BANKO
BE
BECAUSE
BECOME
BEEN
BEFORE
BEGAR
BEHIND
BEING
BELOW
BERLIN
BERN
BETTER
BIG
BILSA
BLOCKED
BLUE
BOTH
BRAVO
BRITANNIA
BRITISH
BULL
BUT
BY
BYE
CALL
CALLED
CALLING
CAN
CAN'T
CASE
CHANCE
CHANGE
CHECK
CHECKED
CIAO
CLEAR
CLEARED
CLIMB
CLIMBING
CODE
COME
COMING
CONDOR
CONFIRM
CONFIRMATION
CONFIRMED
CONTACT
CONTACTED
CONTINENTAL
CONTINUE
CONTROL
CONVENIENCE
COOPERATION
COORDINATION
COPIED
CORRECT
CORSAIR
COSTA
COULD
COURSE
CROSS
CROSSING
CRUISE
CRUISING
CURRENT
CUT
DAY
DECIMAL
DECREASE
DEFINITELY
DEGREES
DESCENDING
DESCENT
DESTINATION
DETAILS
DIJON
DIKMI
DINKELSBUHL
DIRECT
DISREGARD
DISTANCE
DITOR
DO
DOES
DON'T
DOWN
EAST
ECHO
EGYPTAIR
EIGHT
EIGHTTEEN
EIGHTY
EITHER
ELEVEN
ELEVEN
ENABLE
ENTER
EPINAL
EQUIPPED
ESTABLISHED
EXACT
EXERCISE
EXPECT
EXPEDITE
FAMILIAR
FASTER
FEET
FIFTEEN
FIFTEEN
FIFTY
FINAL
FINE
FINISH
FINISHED
FINNAIR
FIVE
FIVE
FLIGHT
FLY
FOLLOW
FOR
FORCE
FORTY
FOUR

FOUR
FOURTEEN
FOURTY
FOX
FOXTROT
FRANKFURT
FRENCH
FREQUENCY
FRIBORG
FROM
FRONT
FULL
FULLY
FURTHER
FUSSE
GENEVA
GERMAN
GERMANIA
GET
GETTING
GIVE
GO
GOING
GONE
GOOD
GOT
GREAT
GULF
HALLO
HAPAG
HAREM
HAVE
HAVING
HE
HEADING
HELLO
HELP
HERBI
HERE
HIGH
HIGHER
HIM
HOCHWALD
HOUNDRED
HOW
HOWEVER
HUNDRED
IBERIA
IDENTIFICATION
IDENTIFIED
IF
I'M
IMMEDIATELY
IN
INBOUND
INCREASE
INDEED
INDICATION
INFORMATION
INITIAL
INITIALLY
INSTEAD
INSTRUCTED
INTER
INTERESTED
INTERFERE
INTERSECTION
INTO
IS
IT
IT'S
JAMBA
JET
JUST
KARLSRUHE
KEEP
KEMPTEN
KILL
KILLED
KINES
KNOW
LARVI
LASON
LAST
LATER
LAUDA
LEAST
LEAVE
LEFT
LEVEL
LIBERTY
LIGHT
LIKE
LIKEWISE
LINE
LINK
LISMO
LITTLE
LLOYD
LOHRE
LONDON
LOOKING
LOOKS
LOT
LOUD
LOWER
LUFTHANSA
LUPEN
LUXAIR
LUXEUIL
LYON
MACH
MAINTAIN
MAINTAINING
MAKE
MALAYSIAN
MALTA
MANY
ME
MED
MERAIR
MERCI
MERIDIANA
MIDLAND
MIKE
MILAN
MILANO
MILES
MILITARY
MINIMUM
MINUTE
MINUTES
MISSED
MISSING
MISSION
MISTAKE
MIXED
MORNING
MUNCHEN
MUNICH
MY
NATTENHEIM
NAVIGATION
NEAR
NEED
NEGATIVE
NEGRA
NETHERLANDS
NEW
NICE
NINE
NINE
NINETEEN
NO
NON
NORMAL
NORMALLY
NORTH
NOT
NOW
NUMBER
OBSERVE
OBVIOUSLY
OCCUPIED
O'CLOCK
ODINA
OF
OFF
OH
OHH
OK
OKAY
OLYMPIC
OMEGA
ON
ONE
ONE
OPPOSITE
OR
ORDER
OTHER
OTHERS
OUT
OVER
OWN
PASSEIRY
PASSING
PERMIT
PLANNED
PLEASE
POINT
PORTUGAL
POSITION
POSSIBILITY
POSSIBLE
PRECEDING

PREFER
PRESENT
PRESSURE
PREVIOUS
PROBLEM
PROBLEMS
PROCEED
QUITE
RADAR
RADIO
RAMSTEIN
RATE
REACH
REACHING
READ
READING
READY
REALLY
REASON
RECEIVED
RECLEARANCE
RECLEARED
REDUCE
REIMS
REMAIN
REPORT
REPORTED
REQUEST
REQUESTED
REQUESTING
REROUTED
RESPECT
RESTRICTING
RESTRICTIONS
RESUME
RHEIN
RIGHT
ROGER
ROMEO
ROUTE
ROUTING
RUNWAY
SABENA
SAID
SAME
SARONNO
SATA
SAUDIA
SAY
SCANDINAVIAN
SECOND
SECTOR
SEE
SEEMS
SENT
SEPARATION
SERVUS
SET
SEVEN
SEVEN
SEVENTEEN
SHORT
SHORTLY
SHOULD
SIERRA
SIMPLY
SINCE
SIR
SIX
SIX
SIXTEEN
SKYFOX
SLIGHTLY
SLOWER
SO
SOBELAIR
SOMEONE
SOON
SORRY
SOUTH
SPACE
SPAIR
SPEED
SPEEDBIRD
SPEEDWAY
STABLE
STAND
STATE
STAY
STEEL
STILL
STOP
STRASSBURG
SUCCEEDING
SUFFICIENT
SUGGEST
SUNWING
SURE
SWISS
SWISSAIR
SWITCH
TAG
TAKE
TALK
TANGO
TAROM
TELL
TEN
TEN
THAN
THANK
THANKS
THAT
THAT'S
THE
THEN
THERE
THINK
THIRTEEN
THIRTEEN
THIRTY
THIRTY
THIS
THOUSAND
THOUSAND
THREE
THREE
THROUGH
TILL
TIME
TIMES
TO
TODAY
TOLD
TOO
TOPSWISS
TORINO
TOUR
TOWARDS
TRACK
TRADITION
TRAFFIC
TRANSAVIA
TRANSLIFT
TRANSMISSION
TRASA
TRASADINGEN
TROUBLES
TRY
TSCHU
TUNAIR
TUNIS
TURKAIR
TURKISH
TURN
TWELVE
TWENTY
TWO
TYPE
UKAY
UNDERSTAND
UNIFORM
UNITED
UNTIL
UP
US
VECTORS
VERY
VICTOR
VISCINITY
VIVA
WALKING
WANTED
WARBURG
WAS
WAY
WE
WEEKEND
WELCOME
WELL
WERE
WE'RE
WHAT
WHAT'S
WHEN
WHERE
WHEREVER
WHETHER
WHICH
WHO
WHO'S
WIND
WITH
WITHIN
WORK
WORTH

WOULD	YET	YOU'VE
WRONG	YOU	ZERO
YEAH	YOUR	ZULU
YES	YOU'RE	ZURICH

C. Entfernte Worte

A
ADIEU
ADIOS
ADJACENT
ADJO
ADVI=
@AEROVIC
AFFIRM
AFTERNOO=
AFTERNOON
AFTERWARDS
AH
AHA
AHM
AI=
AIR=
AIRALS
AIRBUS
AIRCRAFT
AIRFORCE
AIRLINK
AIRSPACE
AIRTAS
AIRTOURS
AIRWAY
AL=
ALFA
ALGERIE
ALI=
ALITA=
ALREADY
ALWAY=
AMENDMENT
AN
=AN
AN=
AOSTA
ARE
AREA
ARRIVEDERCI
ASCOT
AU
AUF
AUSTRIAN
B
B=
BAFAIR
BALKAN
@BAMA
BEE
BELGIAN
BELSTAR
BEST
BIT
BONJOUR
BRE=
BREAK
BRI=
BRITAN=
BUONGIORNO
C
C=
CALLSIGN
CANN=
CANNE
CENTENNIAL
CHARL=
CHARLIE
@CHEENA
@CHEESEBURGER
CLEARANCE
CLI=
CO=
@COLOR
COMMENCE
COMPUTER
CON=
CONSTELLATION
CONT=
CORNA
CORRECTION
CORSAIR
COT=
COTAM
D=
DAG
DANKE
DECIM=
DEL=
DELTA
DES=
DESCEND
@DEVEC
DID
DIR=
DU
DUE
DUSSELDORF
EARLIER
EAS=
EGYPT
EIGH=
EMIRATES
EUROPA
EVA
EVEN
EVENING
EXPEDITIOUS
EXPEDITIOUSLY
F
F=
FI=
</FL>
<FL>
FO=
FOKKER
FOR=
@FOXY
FRANCE
FREQ
FREQUENC=
FRIBOURG
FU=
FUTURA
GAVE
GE=
GENE=
GENEV=
GEORGIA
GIBAIR
GOLF
GOO=
GOTIL
GOTT
GRUSS
GUTEN
H=
HALF
HAMBURG
HANSA
@HANSELI
HAP=
HAS
HEJDA
HER=
HM
HO=
HOI
HOT=
HOTE=
HOTEL
HUNGRY
I
IDEA
IDEN=
IDENT
I'LL
IND=
INDIA
@INDIALOOK
@INGISHIRE
IRISH
ISRAELI
JAT

JETAVIATION
JETCOM
@JOSE
JULIETT
K
KAMAS
=KAY
KILO
KIR=
KNOTS
L
LA
LE=
LEISURE
LESS
LEV=
LIMA
LU=
LUFTHA=
LUFTY
LUHA
M
M=
MALAY=
MARSEILLE
MASP
MAY
@METAVEC
MIL=
MIS=
MONARCH
MOR=
MORE
MORGEN
MORN=
MORNI=
MOROK
MOUTH
MUCH
MUN=
@NAFAMENS
NAH
NATO
NAVI=
NAVIG=
NAVIGATIO=
NECKAR
NELLI
NI=
NIN=
NO=
NORVENICH
NOVEMBER
O
O=
OKAYDOKE
OLBEN
ON=
ONUR
OSCA=
OSCAR
</OT>
<OT>
P
PAPA
PARIS
PASSE=
PER
@PERIOD
PICK
PIN
PLEASURE
PORTUGALIA
PRESENTLY
PREX
PROC=
PROCEE=
PUB
R
R=
RA=
RAD=
RE=
REALIZED
REI=
RESIA
REVOIR
RHEI=
RHI=
ROGE=
ROLAMPONT
ROLLEN=
@ROYSTAR
RV=
S
S=
SCHONEN
SEC
SEV=
SHAMROCK
SHORTCUT
SI=
SINGA
SO=
SPAR
SQUAWK
SQUAWKING
ST
STANDARD
STANDING
STATION
STOPPING
STRESS
@SUNWING
SURE
T
T=
TA=
@TAITIAN
TAN=
@TELE
TH=
THAN=
THEREAFTER
THR=
TRA=
TRAN=
TRANSU=
TRANSWEDE
TRASAD=
TRIPLE
TSCH=
TSCHUSS
TUN=
TURNING
TW=
U
UPON
V
VALDA
VI=
VIA
W
WASN'T
WHISKEY
WHITESTAR
WIEDERHOREN
WILL
WILLISAU
XRAY
YANKEE
Z=
ZE=
ZER=
ZURI=

Literaturverzeichnis

[1] Konrad Hofbauer and Stefan Petrik. ATCOSIM Air Traffic Control Simulation Speech Corpus. Technical Report TR TUG-SPSC-2007-11, EUROCONTROL Experimental Centre, 2008.

[2] Ernst Guenter Schukat-Talamazzini. *Automatische Spracherkennung.* Vieweg Verlag, 1995.

[3] Biing-Hwang Juang Lawrence Rabiner. *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.

[4] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2000.

[5] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4.* Cambridge University Engineering Department, Cambridge, UK, 2006.

[6] Finn Tore Johansen, Narada Warakagoda, Børge Lindberg, Gunnar Lehtinen, Zdravko Kacic, Andrej Zgank, Zdravko Ka Ci C, Kjell Elenius, and Giampiero Salvi. The cost 249 speechdat multilingual reference recogniser. In *COST 249 MCM, Technical Annex*, 2000.

[7] S.Berger J. Hinkelbein. *Prüfungsvorbereitung für die Privatpilotenlizenz.* Hördt, 2007.

[8] International Civil Aviation Organization. Manual of Radio Telephony. Fourth Edition(2007).

[9] Hering H. Comparative Experiments: Speech Recognizers for ATC Simulations. Technical Report EEC Note No. 09/98, EUROCONTROL Experimental Centre, 1998.

[10] Ingo Siegert. Integration neuer Standardverfahren der Spracherkennung in ein vorhandenes Spracherkennungssystem. Technical report, Otto-von-Guericke-Universität Magdeburg, 2007.

[11] Keith Vertanen. Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments. Technical report, Cavendish Laboratory, University of Cambridge, http://www.keithv.com/pub/baselinewsj, 2006.

[12] Andreas Stolcke. SRILM - An Extensible Language Modeling Toolkit. Technical Report Doc 9432 AN/925, Speech Technology and Research (STAR) Laboratory, 2002.

[13] Linguistic Data Consortium. *TIMIT Acoustic-Phonetic Continuous Speech Corpus.* http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1, online zitiert am 24. August 2010.

[14] Klaus Beulen. Phonetische Entscheidungsbäume für die automatische Spracherkennung mit großem Vokabular. Technical report, RWTH Achen, 1999.

[15] H. Niemann. *Klassifikation von Mustern.* Springer, 1983.