Studienarbeit:

SICHERHEITSMAßE FÜR DIE ERKENNUNG SPONTANER SPRACHE

Stefan Kopp
(skopp@ira.uka.de)

20. April 1995

Zusammenfassung

Verbesserung der Spracherkennerleistung durch ein Sicherheitsmaß, das die Wahrscheinlichkeit für die richtige Erkennung eines Wortes in den Erkennungsprozeß mit einbezieht und bei Unterschreitung des zuvor gewählten Grenzwertes das erkannte Wort als falsch ablehnt.

am
Institut für Logik, Komplexität und Deduktionssysteme

Betreuer : Monika Woszczyna
Professor : Dr. Alex Waibel

Inhaltsverzeichnis

1 Einführung

1.1 Das JANUS-Projekt

Bei dem JANUS-Sprachübersetzer-System handelt es sich um eine Kooperation der Universität Karlsruhe mit der Carnegie-Mellon University Pittsburgh.

JANUS läßt sich grob in drei Hauptgruppen einteilen: Spracherkennung, Übersetzung und Sprachsynthese.

Die Verbesserung der Spracherkennung, die einen Schwerpunkt der Forschungen am Institut für Logik, Komplexität und Deduktionssysteme darstellt, ist Thema dieser Studienarbeit.

Der im Spracherkennungssystem von JANUS verwendete Spracherkenner basiert auf einem Hidden Markov Modell (HMM), welches eine auf der Bayes'schen Gleichung basierende Funktion auswertet, um den Score der Hypothesen zu berechnen. Erkenner diesen Typs liefern die für die akustische Eingabe wahrscheinlichste Wortfolge bezüglich der Formel:

$$\mathbf{P(W|A)} = \frac{\mathbf{P(A|W)P(W)}}{\mathbf{P(A)}} \tag{1}$$

$P(A|W)$ steht für die Wahrscheinlichkeit der akustischen Sequenz A für eine gegebene Wortfolge W und wird vom HMM für jedes Wort im Lexikon errechnet. P(W) repräsentiert die A-Priori-Wahrscheinlichkeit der Wortfolge W und wird durch ein statistisches Sprachmodell in Form von Bigrammen bereitgestellt. P(A) ist die A-Priori-Wahrscheinlichkeit der Folge der akustischen Signale.

Wie in den meisten Spracherkennern wird auch im JANUS-Spracherkenner P(A) nicht abgeschätzt, mit der einfachen Begründung, daß P(A) für alle Hypothesen eines gesprochenen Satzes gleich ist. So ändert also die Division durch P(A) nicht die relative Güte der Hypothesen des Erkenners für die Wortfolge, wodurch das Vernachlässigen dieser Wahrscheinlichkeit an der Wahl der besten Hypothese nichts ändert.

1.2 Neue Ansätze

Auswirkung dieser vereinfachten Score-Berechnung ist, daß die vom Erkenner errechneten Scores keine absoluten, sondern nur relative Warscheinlichkeitsmaße sind, was zur Folge hat, daß wir zwar wissen, welche Hypothese am ehesten zutrifft, jedoch nicht, wie gut sie die Realität wiedergibt.
Wenn der Erkenner also mehr machen soll, als nur die wahscheinlichste Folge von Worten eines vorgegebenen Lexikons auszugeben, so muß ein Weg gefunden werden, ein Gütemaß für die erkannte Sequenz zu bestimmen.
Damit wäre zum Beispiel auch die Detektion und Weiterverarbeitung von neuen, nicht im Lexikon befindlichen Worten möglich.
Diese Studienarbeit befaßt sich mit dem Versuch der Abschätzung dieses Gütemaßes und der daraus möglichen Verbesserung der Erkennerleistung.

2 Spracherkennung

Abbildung 1 zeigt ein Blockdiagramm des JANUS-Worterkennersystems, das in den folgenden Kapiteln genauer betrachtet wird.

2.1 Signalverarbeitung

Die erste Frage, die sich beim Entwurf eines Spracherkennungssystems erhebt ist, wie das Sprachsignal maschinenverwertbar zu repräsentieren bzw. zu codieren ist, bevor die eigentliche Erkennung erfolgt. Grundsätzlich könnte man natürlich einfach die digitalisierte Wellenform des Signals als Eingabe benutzen. Jedoch ist die riesige benötigte Rechnerleistung bei einer Abtastrate von 16000 Werten in der Sekunde sehr hinderlich. Abgesehen davon beinhaltet die Wellenform auch Informationen, die für die Spracherkennung redundant sind (z.B. die Phaseninformation) und die Erkennerleistung eher mindern als verbessern. Darum wurde eine Anzahl von Codierungsverfahren entwickelt, die versuchen, eine kompaktere Repräsentation der Sprache zu erzeugen, während die wichtigen Spracheigenschaften erhalten bleiben bzw. sogar besonders herausgearbeitet werden.
Der wichtigste Punkt vor dem Erkennungsvorgang selbst ist im Besonderen die Wahl der parametrischen Darstellung, die die sprachliche Eingabe möglichst genau wiedergeben, Redundanzen aufheben und sprachliche Eigenschaften hervorheben soll, so daß die Daten möglichst kompakt und informativ vorliegen. Danach wird eine Normalisierung des Sprachsignals durchgeführt, um ungewünschtes Rauschen zu reduzieren.

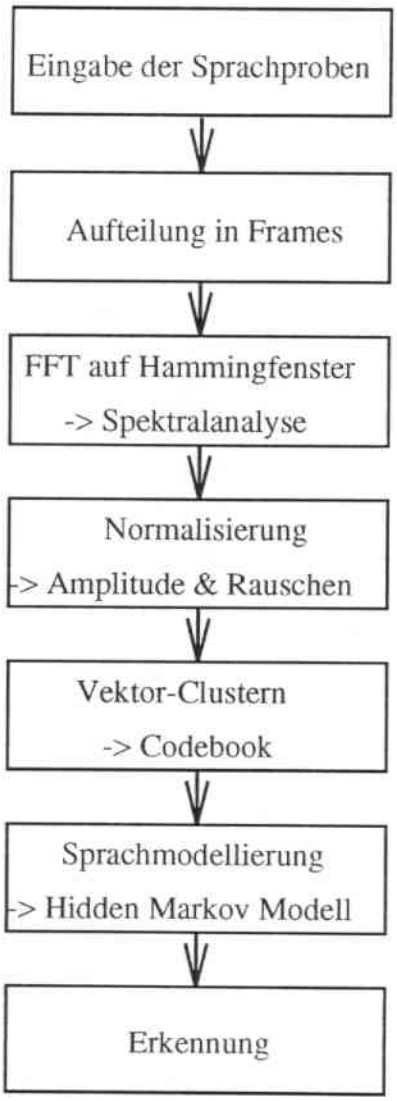

Abbildung 1: JANUS-Worterkennersystem

2.1.1 Parametrische Darstellung des Sprachsignals

Wegen der großen Menge anfallender Daten bedarf es einer kritischen Auswahl der parametrischen Darstellung. Es müssen also verschiedenste Techniken der Datenreduzierung gegeneinander abgewogen werden, wobei die höchste Priorität auf der wahrheitsgetreuen Wiedergabe der für die Spracherkennung wichtigsten Merkmale liegt.

Drei der wichtigsten Kriterien für die Auswahl der parametrischen Darstellung, neben der Güte der Spracherkennung im Ganzen, sind die benötigte Rechenzeit, der Speicherbedarf und die Komplexität der Implementierung. Da der Mensch über die Fähigkeit der Spracherkennung im höchsten Grade verfügt und es diese Fähigkeit in der maschinellen Spracherkennung nachzubilden gilt, liegt es nah, die akustische Verarbeitung der Sprache im menschlichen Gehör als Vorbild zu nehmen.

Die vom Ohr/Mikrofon aufgenommenen Schalldruckwellen werden dabei zunächst einer Frequenzanalyse unterzogen und der Energiegehalt der einzelnen Frequenzbereiche wird dann an das Hirn/den Rechner weitergeleitet. Das Spektrogramm sollte also einen ausreichenden Informationsgehalt für eine gute Spracherkennung haben.

2.1.2 Frames und Fenster

Weil die Sprache ein kontinuierlicher Vorgang ist, gilt es quasi-stationäre Sprachsignale zu erzeugen. Dies gelingt durch die Betrachtung des fortschreitenden Sprachsignals in kurzen Zeitintervallen. Die Länge der Intervalle wird dabei so gewählt, daß innerhalb eines Intervalls keine wesentlichen Änderungen des Spektrums auftreten. In der Praxis hat sich bewährt, alle 10 ms ein 256 Werte, also 16 ms breites Fenster, herauszugreifen. Durch die entstehende Überlappung der Fenster wird die Kontinuität des Signalverlaufs gewährleistet. Das Verwenden noch schmalerer Frames wäre mit erhöhtem Rechenaufwand zu bezahlen.

Das Ausblenden eines Frames aus dem gesamten Sprachsignal wird durch eine sogenannte Fensterfunktion erreicht. Diese Funktion muß folgende zwei Kriterien erfüllen: erstens soll sie den für die Spracherkennung relevanten Frequenzbereich möglichst nicht verändern und zweitens, um Signalverzerrungen an den Rändern der Frames zu vermeiden, im Zeitbereich schnell abklingen. Bezüglich dieser Bedingungen zeigt sich die als Hamming-Fenster bekannte Funktion als gut geeignet. Sie hat die Form:

$$w_H(k) = \begin{cases} 0.54 + 0.46\cos(\frac{2\pi k}{N}) & f\ddot{u}r \ -\frac{N}{2} \leq k \leq \frac{N}{2} - 1 \\ 0 & sonst \end{cases} \tag{2}$$

Hierbei steht N für die Anzahl der Abtastwerte pro Frame, k stellt den Index für den Abtastwert dar. Um nun ein Teilstück des Sprachsignals herauszugreifen, wird das mittels eines A/D-Wandlers bei einer Abtastrate von 16kHz (bei dieser Abtastrate ist nach dem Theorem von Nyquist sichergestellt, daß die obere Grenzfrequenz des gewünschten Spektrums (7250 Hz) noch korrekt und ohne aliasing abgetastet wird) zuvor abgetastete diskrete Signal s(n) mit der am Analysezeitpunkt n zentrierten Fensterfunktion multipliziert. Dadurch erhält man bei N Abtastungen innerhalb eines Frames zum Zeitpunkt n (n ist der Index der Abtastungen) folgende Ausgabe:

$$\bar{s}(n) = \sum_{k=n-\frac{N}{2}}^{n+\frac{N}{2}-1} s(k) w_H(n-k) \tag{3}$$

2.1.3 Fourier - und Hartleytransformationen

Um nun das gewünschte Frequenzspektrum der einzelnen Frames zu erhalten, wendet man auf die erhaltenen Werte die sogenannte *diskrete Fouriertransformation* an. Sie hat die Form:

$$D_\mu = \sum_{n=0}^{N-1} d_n \ exp\ (-i2\pi \frac{\mu n}{N}) \ , \quad mit \quad 0 \leq \mu, n \leq N-1 \tag{4}$$

Hierbei bezeichnet n die diskrete Zeitvariable, μ stellt eine Frequenzvariable dar. Wird μ inkrementiert, so steigt die Frequenz um $\frac{16khz}{N}$ bis maximal $\frac{N}{2}16kHz$. Danach fällt sie wieder um den gleichen Betrag bis auf Null für n=N. Bei einer Abtastrate von 16kHz und einer Fensterbreite von 16 ms, was zu N=256 Abtastwerten/Spektralkoeffizienten je Fenster führt, entspricht dieser Betrag 62.5Hz.

Wenn das Eingabesignal nur reelle Werte annimmt, werden durch die DFT doppelt soviele Multiplikationen wie nötig ausgeführt. Abhilfe schafft in diesem Fall die diskrete Hartley Transformation, die sich aus der DFT in folgender Weise ableitet:

$$H_\mu = Re(D_\mu) - Im(D_\mu) \tag{5}$$

Die DFT bzw. DHT haben in den vergangenen Jahren für die Signalverarbeitung besonders dadurch an Bedeutung gewonnen, daß es gelungen ist, mit der Schnellen Fouriertransformation bzw. Hartleytransformation (FFT bzw. FHT) leistungsfähige Algorithmen zu entwickeln, die es gestatten, die Berechnung auch einer größeren Anzahl von Abtastwerten genügend schnell durchzuführen. Für weitere Informationen sei auf entsprechende Literatur verwiesen [2.1],[2.2].

2.1.4 Melscalekoeffizienten

Das durch die Verwendung der Fast Fourier Transformation aus dem Sprachsignal erhaltene Frequenzspektrum wird zunächst gemäß

$$P(\mu) = Re(D(\mu))^2 + Im(D(\mu))^2 \tag{6}$$

in ein Energiespektrum gewandelt. Dabei ist zu beachten, daß wegen der vorliegenden Symmetrie durch das Quadrieren die vormals 256 Spektralkoeffizienten auf 128 Leistungskoeffizienten reduziert werden. Um nun eine weitere Datenreduzierung zu erreichen, werden im nächsten Schritt aus den erhaltenen Leistungskoeffizienten 16 Melscalekoeffizienten berechnet, was eine Aufteilung der Frequenzen in Frequenzbänder bedeutet.
Die hierfür benutzte Mel-Skala teilt dabei die Frequenzbänder, in Anlehnung an das menschliche Hörvermögen, in den unteren Frequenzbereichen enger ein als in den oberen. Tabelle 1 zeigt die berechneten Melscale-Koeffizienten und die zugehörigen Frequenzbereiche bei einer Abtastrate von 16kHz. Hierbei wurden die Melscale-Koeffizienten mittels der über die Hartleytransformation erhaltenen Spektralkoeffizienten berechnet.

Band	Melscale-Koeffizient	Frequenzbereich [Hz]
0	m(0)=H(0)+H(1)+H(2)/2	0 - 125
1	m(1)=H(2)/2+H(3)+$\cdots$+H(6)/2	125 - 375
2	$\cdots$	375 - 625
3	$\cdots$	625 - 875
4	$\cdots$	875 - 1125
5	$\cdots$	1125 - 1375
6	$\cdots$	1375 - 1625
7	$\cdots$	1625 - 1875
8	$\cdots$	1875 - 2187.5
9	$\cdots$	2187.5 - 2562.5
10	$\cdots$	2562.5 - 3000
11	$\cdots$	3000 - 3562.5
12	$\cdots$	3562.5 - 4250
13	$\cdots$	4250 - 5062.5
14	m(14)=H(81)/2+$\cdots$+H(97)/2	5062.5 - 6062.5
15	m(15)=H(97)/2+$\cdots$+H(116)/2	6062.5 - 7250

Tabelle 1: Frequenzbereiche der Mel-Skala im Spektrum 0 - 8000 Hz

2.1.5 Normalisierung

Die Normalisierung der zuvor berechneten Melscalekoeffizienten erfüllt zwei unterschiedliche Zwecke: erstens wird ein gewisses Maß an Rauschen herausgefiltert und zweitens werden die Schwankungen in der Aussteuerung des Sprachsignals auf das Intervall [0;1] normiert. Die Vorgehensweise des JANUS-Systems sieht dabei wie folgt aus.

Zunächst wird der Mittelwert der 16 Melscalekoeffizienten über den gesamten zu betrachtenden Zeitbereich berechnet. Dieser Mittelwert des untersuchten Satzes wird als Maß für das konstante Rauschen von allen Koeffizienten subtrahiert, die Differenz des ebenfalls bestimmten Minimum und Maximum dient als Skalierungsfaktor, durch den die Koeffizienten zum Zweck der Normierung dividiert werden.

Für jeden Frame wird letztendlich ein Vektor extrahiert, der neben den 16 Melscalekoeffizienten noch weitere Parameter umfaßt wie z.B. Delta Melscale Koeffizienten oder Power. Auf die Berechnung von Cepstralen Parametern, wie in der Mehrzahl der verwendeten Spracherkenner, wird im JANUS-Spracherkennersystem verzichtet.

2.2 Hidden Markov Modelle

In dem vorangegangenen Kapitel Signalverarbeitung ging es darum, gesprochene Sprache in ein für Maschinen verwertbares Format zu bringen. Damit ist die zur Spracherkennung nötige Vorverarbeitung beendet und es ist nun von Interesse, aus den gewonnenen Daten mit Hilfe mathematischer Modelle eine Aussage zu treffen, um welche Folge von Worten es sich bei dem gesprochenen Satz handelt.

Das im JANUS-Spracherkennungssystem verwendete mathematische Modell ist ein statistisches[1], das unter der Annahme benutzt wird, daß sich Sprache mit Hilfe eines parametrischen Wahrscheinlichkeitsprozesses beschreiben läßt und daß die Parameter des statistischen Prozesses auf eine präzise, vorher genau definierte Art und Weise bestimmt werden können. Die probabilistischen Modelle werden verwendet, um trotz unsicherer und unvollständiger Informationen eine sinnvolle Erkennung zu ermöglichen. Unsicherheiten bei der Erkennung können viele Gründe haben, zum Beispiel Unterschiede in der Aussprache, gleichklingende Worte (Homophone), Hintergrundgeräusche

[1] demgegenüber steht die Gruppe der deterministischen Modelle

usw. Aus diesem Grund bieten statistische Modelle offensichtlich auch geeignete Methoden für die Spracherkennung.

Der in den meisten Erkennern und auch im vorliegenden JANUS-System verwendete Ansatz ist der des Hidden Markov Modells (HMM). Diese Modelle bestehen aus einer sogenannten Markov-Kette, die eine Folge von Zuständen wiedergibt und beschreiben einen doppelt eingebetteten statistischen Prozeß, dem ein nicht direkt beobachtbarer statistischer Prozeß zugrundeliegt, der nur durch eine weitere Menge eben solcher Prozesse, die eine Folge von Beobachtungen erzeugen, beobachtet werden kann.

2.2.1 Elemente von HMMs

Hier sollen kurz die Bestandteile eines HMM definiert werden und eine Erklärung für die vom Modell geleistete Generierung einer Beobachtungsfolge gegeben werden. Ein HMM wird durch folgende Elemente definiert:

1) Die Anzahl **N** der Zustände innerhalb des Modells
Auch wenn die Zustände versteckt sind, gibt es doch für viele praktische Anwendungen einen greifbaren Zusammenhang der einzelnen Zustände oder einer Menge von Zuständen und dem beobachteten Ereignis. So beschreiben bei der späteren Anwendung im Spracherkenner mehrere Zustände einzelne Phoneme bzw. Worte.

Es existiert eine Vielzahl möglicher Modelle. Für die Spracherkennung wird im Regelfall und so auch im JANUS-Spracherkenner das später beschriebene links-rechts Modell verwendet.

Die einzelnen Zustände werden mit S=$\{S_1, S_2, \ldots, S_N\}$ und der Zustand zum Zeitpunkt t mit q_t bezeichnet.

2) Die Anzahl **M** der maximal in einem Zustand zu beobachtenden verschiedenen Ausgabewerte oder in anderen Worten, die Größe des Alphabets der zu beobachtenden physikalischen Ausgaben des nachzubildenden Prozesses
Die einzelnen verschiedenen Ausgabesymbole werden mit V=$\{v_1, v_2, \ldots, v_M\}$ bezeichnet.

3) Die Wahrscheinlichkeitsverteilung der Zustandsübergänge $\mathbf{A} = \{\mathbf{a_{ij}}\}$, wobei

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i] \quad , \quad 1 \leq i, j \leq N \tag{7}$$

für die Wahrscheinlichkeit steht, daß dem Zustand S_i der Zustand S_j folgt. Außerdem gilt:

$$\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i \quad und \quad a_{ij} \geq 0 \quad \forall i, j \tag{8}$$

4) Die Wahrscheinlichkeitsverteilung der beobachtbaren Werte $\mathbf{B} = \{\mathbf{b_j(k)}\}$ im Zustand j, wobei

$$b_j(k) = P[v_k zum Zeitpunkt t | q_t = S_j] \quad , \quad 1 \leq j \leq N \; und \; 1 \leq k \leq M \tag{9}$$

für die Wahrscheinlichkeit steht, daß zum Zeitpunkt t der Zustand S_j erreicht ist, und der Vektor v_k ausgegeben wird.

5) Die Startzustandsverteilung $\pi = \{\pi_\mathbf{i}\}$, wobei

$$\pi_i = P[q_1 = S_i] \quad , \quad 1 \leq i \leq N \tag{10}$$

für die Wahrscheinlichkeit steht, daß sich das System zu Beginn im Zustand S_i befindet.

Zur Spezifikation eines HMM bedarf es also der Bestimmung der zwei Modellparameter **N** und **M** und der Festlegung der drei Wahrscheinlichkeismaße **A**, **B** und π. Um das HMM zu beschreiben wird abkürzend $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ geschrieben.

2.2.2 Probleme der HMMs und deren Lösung

Aus der Literatur [2.7] bekannt, gibt es drei Grundprobleme für Hidden Markov Modelle, die im folgenden mit den zugehörigen Lösungsvorschlägen aufgeführt werden.

1) Das Evaluierungsproblem

Wie können wir effizient die Wahrscheinlichkeit berechnen, daß bei gegebenem Modell und Beobachtungssequenz die Folge von Ausgabewerten von dem Modell erzeugt wurde? Man kann das Problem auch so beschreiben, daß man mittels eines Bewertungsschemas testet, wie gut ein gegebenes Modell eine Folge von Ausgabewerten wiedergibt.

Kurz: bei gegebenem Modell $\lambda = (A, B, \pi)$ und gegebener Ausgabefolge $O = O_1 O_2 \ldots O_T$ ist ein effizienter Algorithmus für die Berechnung von $P(O|\lambda)$ gesucht.

2) Das Kodierungsproblem

Wie kann man das Geheimnis, also den versteckten Teil des HMM, lüften und die bestmögliche Abfolge der Zustände finden? Es ist natürlich klar, daß es hierbei nicht darum gehen kann, die korrekte Zustandsfolge zu finden, sondern es gilt durch ein Gütekriterium das Problem bestmöglich zu lösen.

Kurz: bei gegebenem Modell $\lambda = (A, B, \pi)$ und gegebener Ausgabefolge $O = O_1 O_2 \ldots O_T$ ist ein effizienter Algorithmus für die Suche nach einer Zustandssequenz $Q = q_1 q_2 \ldots q_T$ gesucht, für die $P(O, Q|\lambda)$ maximiert wird.

3) Das Lernproblem

Wie sind die Modellparameter zu wählen, um die Ausgabefolge bestmöglich zu beschreiben? Die zum Anpassen der Modellparameter verwendeten Sprachsequenzen werden auch *Trainingsset* genannt. Das Lernproblem ist das für die Spracherkennung kritischste Problem, da es direkt die Güte der Modellparameter und damit des Modells selbst betrifft.

Kurz: bei einer gegebenen Menge von Beobachtungssequenzen ist ein effizienter Algorithmus zur Einstellung der Modellparameter $\lambda = (A, B, \pi)$ gesucht, so daß $P(O|\lambda)$ maximiert wird.

Um diese Probleme mit akzeptablem Rechenaufwand zu bewältigen, bedarf es einiger effizienter Algorithmen auf die im Folgenden nur kurz eingegangen wird, um dann im nächsten Abschnitt die eigentliche Anwendung in der Spracherkennung zu zeigen.

Beim Evaluierungsproblem gilt es, die Wahrscheinlichkeit einer Folge von Ausgabewerten bei gegebenem Modell zu berechnen:

$$P(O|\lambda) = \sum_{\forall Q} P(O, Q|\lambda) = \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{11}$$

Bei dieser Art der Berechnung, wird also zunächst im Zustand q_1 mit der Wahrscheinlichkeit π_{q_1} begonnen, in dem die Ausgabe O_1 mit der Wahrscheinlichkeit $b_{q_1}(O_1)$ erzeugt wird. Dann zählt die Uhr weiter von Zeitpunkt t=1 zu t=2, der Zustand wechselt von q_1 zu q_2 mit der Wahrscheinlichkeit $a_{q_1 q_2}$ und im Zustand q_2 wird die Ausgabe O_2 mit der Wahrscheinlichkeit $b_{q_2}(O_2)$ erzeugt ...

Wie leicht zu erkennen ist, werden $2T \cdot N^T$ Berechnungen benötigt, um auf diese Weise $P(O|\lambda)$ zu berechnen. Zu jedem Zeitpunkt t=1,2,...,T gibt es N mögliche Zustände, die zu erreichen sind, und für jede solche Zustandsfolge werden 2T Berechnungen für jeden Term der Summe benötigt. Bereits für kleine Werte von N (Anzahl der Zustände) und T (Anzahl der Ausgabewerte) nimmt die Berechnung immense Ausmaße an.

Eine Lösung bietet der sogenannte *Forward-Algorithmus*, der auf der Idee aufbaut, daß nicht der gesamte Pfad vom Startzustand bis zum Endzustand betrachtet wird, sondern jeweils nur Schritt für Schritt Teilfolgen der Ausgabewerte in die Berechnung eingehen. So definiert sich die Forward-Variable also folgendermaßen:

$$\alpha_t(i) = P(O_1 O_2 \ldots O_t, q_t = S_i|\lambda) \tag{12}$$

Dies entspricht der Wahrscheinlichkeit der partiellen Ausgabesequenz $O_1 \ldots O_t$ bis zum Zeitpunkt t und dem zum Zeitpunkt t erreichten Zustand S_i, bei gegebenem Modell λ.

α berechnet sich rekursiv durch:

1) Initialisierung:

$$\alpha_1(i) = \pi_i b_i(O_1) \quad , \quad 1 \leq i \leq N \tag{13}$$

2) Induktion:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \quad , \quad 1 \leq t \leq T-1 \quad \textit{und} \quad 1 \leq j \leq N \tag{14}$$

3) Terminierung:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{15}$$

Der Rechenaufwand beschränkt sich beim Forward-Algorithmus auf N^2T Berechnungen, da zu jedem Zeitpunkt jeder der N möglichen Zustände N Vorgänger haben kann. Zur Verdeutlichung zeigt Abbildung 2 die Menge von Operationen, die für die Berechnung von der Forward-Variable $\alpha_{t+1}(j)$ benötigt wird.

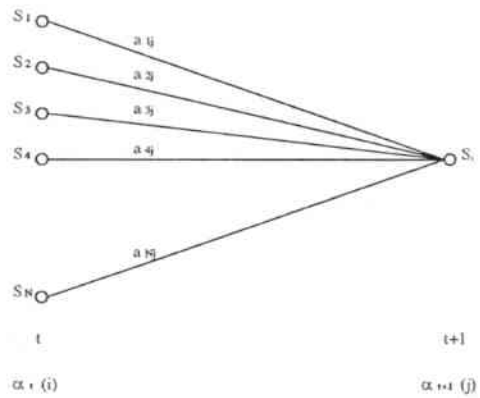

Abbildung 2: Berechnung der Forward-Variable $\alpha_{t+1}(j)$

Anders als beim Evaluierungsproblem, für das eine exakte Lösung gefunden werden kann, gibt es einige mögliche Wege, auf denen man das Kodierungsproblem lösen kann oder genauer gesagt, die optimale Zustandsfolge für die zugehörige gegebene Beobachtungssequenz finden kann. Die Schwierigkeit liegt hier im Bestimmen eines für die Suche nach einer passenden Zustandsfolge geeigneten Gütekriteriums. Zum Beispiel könnte ein mögliches Gütekriterium darauf basieren, jeweils den Zustand q_t zu wählen, der, unabhängig von vorhergehendem und nachfolgendem Zustand, am wahrscheinlichsten ist. Obwohl dieses Kriterium den Erwartungswert der korrekten Zustände maximiert, kann es einige Probleme mit der insgesamt zu betrachtenden Folge von Zuständen geben, zum Beispiel wenn es im HMM unzulässige Zustandsübergänge gäbe[2], so daß die bestmögliche Zustandsfolge nicht einmal zulässig wäre. Abhilfe würde ein Kriterium schaffen, das zum Beispiel den Erwartungswert der korrekten Zustandspaare q_t, q_{t+1} maximiert. Das jedoch

[2]in diesem Fall ist die Übergangswahrscheinlichkeit $a_{ij} = 0$ für einige i und j

am häufigsten verwendete Kriterium sucht den besten Zustandspfad, maximiert also $P(Q|O,\lambda)$ was gleichbedeutend ist mit der Maximierung von $P(Q,O|\lambda)$. Der zu diesem Zweck verwendete *Viterbi-Algorithmus* ist folgendermaßen definiert[3]:
1) Initialisierung:

$$\delta_1(i) = \pi_i b_i(O_1) \quad , \quad 1 \leq i \leq N \tag{16}$$

$$\psi_1(i) = 0 \quad , \quad 1 \leq i \leq N \tag{17}$$

2) Induktion:

$$\delta_{t+1}(j) = \max_{1 \leq i \leq N} [\delta_t(i) a_{ij}] b_j(O_{t+1}) \quad , \quad 1 \leq t \leq T-1 \quad \textit{und} \quad 1 \leq j \leq N \tag{18}$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} [\delta_t(i) a_{ij}] \quad , \quad 1 \leq t \leq T-1 \quad \textit{und} \quad 1 \leq j \leq N \tag{19}$$

3) Terminierung:

$$p^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{20}$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{21}$$

4) Pfadrückverfolgung (backtracking):

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad , \quad t = T-1, T-2, \ldots, 1 \tag{22}$$

Es ist anzumerken, daß sich die beiden Algorithmen sehr ähneln und hauptsächlich (abgesehen vom backtracking) in der Maximierung in (18) unterscheiden, die anstelle der Aufsummierung in (14) steht.
Das dritte und bei weitem schwierigste Problem stellt sich bei der Suche nach einer Methode zur Einstellung der Modellparameter (A, B, π), so daß die Wahrscheinlichkeit der Ausgabesequenz für das gegebene Modell maximal wird. Da es keinen Weg gibt, die Parameter optimal einzustellen um eine

[3] $\delta_t(i)$ steht für die größte Wahrscheinlichkeit, die sich längs eines Pfades, der sich bis zum Zeitpunkt i aus den ersten t Beobachtungen zusammensetzt und im Zustand S_i endet, errechnet.
$\psi_t(j)$ ist ein Array, in dem für jedem Zeitpunkt t und Zustand S_j das Argument festgehalten wird, das $\delta_{t-1}(i)$ maximiert

globale Lösung zu finden, muß man sich damit begnügen, $\lambda = (A, B, \pi)$ so zu wählen, daß $P(O|\lambda)$ ein lokales Maximum erreicht. Ein effizienter, iterativer Algorithmus ist der hier verwendete *Forward-Backward-* bzw. *Baum-Welch-Algorithmus.*
Die Vorgehensweise ist dabei, aus der jeweils gegebenen Abschätzung für λ die zu erwartenden Wahrscheinlichkeiten zu berechnen, um dann λ neuabzuschätzen, was die modifizierten Parameter $\bar{\lambda}$ liefert, die wiederum neuabgeschätzt werden. Dies geschieht solange, bis entweder ein kritischer Punkt erreicht ist, an dem $\bar{\lambda} = \lambda$, oder $P(O|\bar{\lambda}) - P(O|\lambda) \leq \epsilon$, also ein Grenzwert erreicht ist.
Um den Neuabschätzungsvorgang für die Modellparameter zu beschreiben, bedarf es zuerst einiger Definitionen. Zuerst definieren wir $\xi_t(i,j)$ als die Wahrscheinlichkeit, zum Zeitpunkt t im Zustand S_i und zum Zeitpunkt t+1 im Zustand S_j zu sein, bei gegebenem Modell und Beobachtungssequenz:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{23}$$

In Abbildung 3 ist diese Betrachtungsweise verdeutlicht.

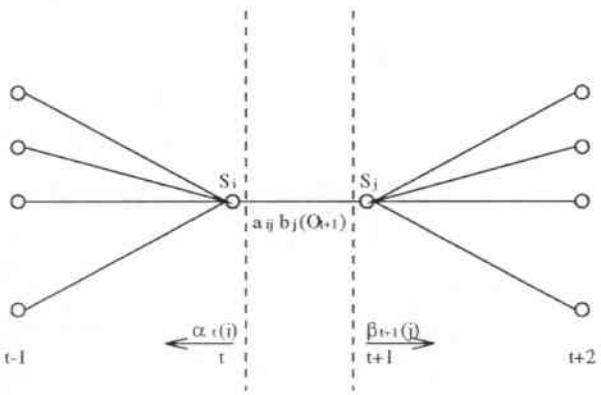

Abbildung 3: benötigte Folge der Operationen zur Berechnung des Falles, daß $q_t = S_i$ und $q_{t+1} = S_j$ gilt

So wird klar, daß man $\xi_t(i,j)$ auch in folgender Form schreiben kann:

$$\xi_t(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{24}$$

Die verwendete Backward-Variable $\beta_t(i)$ ist ähnlich wie die Forward-Variable

$\alpha_t(i)$ als die Wahrscheinlichkeit der partiellen Ausgabesequenz $O_{t+1} \ldots O_T$ ab dem Zeitpunkt t+1 bis zum Ende definiert, bei gegebenem Modell λ und zum Zeitpunkt t erreichtem Zustand S_i:

$$\beta_t(i) = P(O_{t+1} O_{t+2} \ldots O_T | q_t = S_i, \lambda) \tag{25}$$

Die *Neuabschätzungsformeln* errechnen sich folgendermaßen:
1) $\bar{\pi}_i$ = *erwartete Häufigkeit, zum Zeitpunkt* $t = 1$ *im Zustand* S_i *zu sein*

$$\bar{\pi}_i = \sum_{j=1}^{N} \xi_1(i,j) \tag{26}$$

2) $\bar{a}_{ij} = \frac{\textit{erwartete Zahl der Übergänge von } S_i \textit{ zu } S_j}{\textit{erwartete Zahl der Übergänge von } S_i}$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)} \tag{27}$$

3) $\bar{b}_j(k) = \frac{\textit{erwartetes Vorkommen in Zustand } j \textit{ bei Beobachtung von Symbol } v_k}{\textit{erwartetes Vorkommen in Zustand } j}$

$$\bar{b}_j(k) = \frac{\sum_{t=1:O_t=v_k}^{T} \sum_{i=1}^{N} \xi_t(i,j)}{\sum_{t=1}^{T} \sum_{i=1}^{N} \xi_t(i,j)} \tag{28}$$

Wie bereits vorher erwähnt, konvergiert der Algorithmus, das heißt, es kann bewiesen werden, daß nach jeder Iteration (Neuabschätzung) das neue Modell $\bar{\lambda}$ entweder eine höhere Wahrscheinlichkeit besitzt, die Beobachtung zu erzeugen ($P(O|\lambda) > P(O|\bar{\lambda})$) oder das ursprüngliche Modell einen kritischen Punkt der Wahrscheinlichkeitsfunktion definiert ($\bar{\lambda} = \lambda$). Das Endresultat des Neuabschätzungsvorgangs ist der *Maximum-Likelihood-Schätzer* des Hidden-Markov-Modells.

2.3 Verwendung des HMMs

Eine Frage, die sich jetzt aufdrängt lautet, wie man nun das mittels der genannten Algorithmen entworfene Hidden-Markov-Modell bei der Spracher-

kennung eigentlich anwendet.
Wenn man $W = w_1, w_2, \ldots, w_n$ als eine Folge von n gesprochenen Worten und $A = a_1, a_2, \ldots, a_k$ als die beobachteten akustischen Daten[4]auffaßt, so bezeichnet $P(W|A)$ die Wahrscheinlichkeit, daß die Wortfolge W in Abhängigkeit von den beobachteten akustischen Daten A gesprochen wurde. Der Spracherkenner entscheidet sich zugunsten der Wortfolge $\hat{W}$, die das folgende Entscheidungskriterium erfüllt:

$$P(\hat{W}|A) = \max_{W} P(W|A) \tag{29}$$

Nach der Bayes'schen Gleichung gilt:

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)} \tag{30}$$

Aus Gleichung (30) folgt, daß man in der Spracherkennung vor vier grundlegende Probleme gestellt ist.

2.3.1 Akustische Verarbeitung - von Melscalekoeffizienten zum Codebuch

Zunächst muß die Form der akustischen Aussage A bestimmt werden, auf der die Entscheidung des Spracherkenners aufbaut. Diese Umwandlung des Sprachsignals in Akustikdaten A, also die akustische Verarbeitung, wurde bereits in Kapitel 2.1 eingehend beschrieben.

Die häufig angewandte Vektorquantisierung, bei der die kontinuierlichen Beobachtungsvektoren auf diskrete Codebuch-Indizes abgebildet werden, findet im JANUS-Erkenner nicht statt, sondern es werden für jedes Phonem Verteilungen bestimmt.

Wenn ein Codebuch erst einmal durch die Tainingsdaten für ein Phonem erstellt ist, handelt es sich bei der Zuordnung der Codebuch-Indizes zu den kontinuierlichen Beobachtungsvektoren nur noch um eine 'nearest neighbour Berechnung', das heißt, daß der Beobachtungsvektor O_l dem nach einem Distanzmaß (z.B. euklidische Distanz) nächstgelegenen Codebuch-Vektor zugeordnet wird. Um für jedes Phonem ein gutes Codebuch zu erstellen, des-

[4]auf der Basis, auf der der Erkenner seine Entscheidung für ein Wort aufbaut

sen Vektoren eine ähnliche Wahrscheinlichkeitsverteilung haben wie die Beobachtungsvektoren, bedarf es einer großen Trainingsmenge und geeigneter Algorithmen. Es gibt mehrere gebräuchliche iterative Verfahren zur Codebuchgenerierung, z.B. den LBG-Algorithmus, der zunächst den Zentroid aller Trainingsdaten berechnet, darauf die Trainingsdaten in zwei Teile aufteilt, wiederum die Zentroide berechnet, wieder splittet ... Die Iteration bricht ab, sobald die gewünschte Zahl an Zentroiden=Codebookeinträgen erreicht ist.

Eine andere Vorgehensweise ist, die Trainingsvektoren in M (M ist die gewünschte Zahl an Codebucheinträgen) disjunkte Mengen zu teilen, die Zentroide dieser Mengen zu berechnen, um dann in weiteren Schritten iterativ die Partitionierung und damit das Codebuch zu optimieren. Ein Clusterverfahren dieser Art, das k-Means-Verfahren, wird im JANUS-Spracherkenner benutzt, um für jedes Phonem ein Codebuch mit M=50 Vektoren zu erzeugen.

Durch die 'nearest neighbour Berechnung' und damit die Wahl eines Vektors als nächsten Vektor zum Beobachtungsvektor, erhält man eine Verzerrung. Die Erkennerleistung ist natürlich stark von der Größe dieser Verzerrung abhängig. Dabei gerät man in den Zielkonflikt, zum einen die Distorsion möglichst gering halten zu wollen und zum anderen das Codebuch nicht zu groß werden zu lassen, um nicht mit zu vielen Parametern auf Probleme bei der Implementierung des HMMs zu stoßen, die Rechenzeit zu erhöhen und desweiteren in den Konflikt zu geraten, daß bei geringer Menge an Trainingsdaten diese gelernt werden, statt eine geeignete Generalisierung zu erreichen. Untersuchungen zur Codebuchgröße [2.7] haben gezeigt, daß nach einem Wert von M=32 Codebookeinträgen die Verzerrung nur noch in geringem Maße verkleinert werden kann, wohingegen die Anzahl der Parameter und damit die zur Erkennung benötigte Rechenzeit stark ansteigt. Aus diesem Grund werden im Allgemeinen Codebücher mit einer Größe von 32 bis 256 Vektoren verwendet. Beim JANUS-Erkenner werden im Speziellen für jedes Basisphonem 50 Codebookeinträge geclustert.

Besonders hervorzuheben ist, daß die minimale Distanz eines jeden Phonems zum zuvor parametrisierten Signal abgespeichert und kein Stellvertretervektor für jedes einzelne Phonem ausgewählt wird. Es wird also exakt betrachtet, wie nahe das beobachtete Signal an einen der Einträge im jeweiligen Phonem-Codebuch heranreicht.

2.3.2 Akustische Modellierung

Als nächstes ist es wichtig, ein akustisches Modell zu entwickeln, das statistisch die Beziehung zwischen der vom Sprecher ausgegebenen Wortsequenz W und der vom Akustikprozessor ausgegebenen Symbolfolge beschreibt, also die Berechnung der Wahrscheinlichkeit $P(A|W)$ erlaubt. Im JANUS-System besteht das akustische Modell aus Hidden-Markov-Ketten, die die Aussprache einzelner Phoneme nachbilden. Um genau zu sein, besteht das Modell für Wortfolgen aus einer Aneinanderreihung von einzelnen Wortmodellen und diese wiederum aus der Verkettung von Modellen der in diesen Worten auftretenden Phoneme, die die grundsätzliche Aussprache einzelner Worte definieren.

Zu jedem Wort w des Vokabulars existiert im sogenannten Wörterbuch[5] eine Grundform $B(w) = b_1, \ldots, b_k$, die aus einer Folge von Phonemen b_i aus einem Phonemalphabet besteht und zu jedem Phonem b aus dem Phonemalphabet gibt es wiederum eine Hidden-Markov-Kette. Die Übergänge zwischen den Zuständen der HM-Ketten sind mit den Wahrscheinlichkeiten für die Phoneme oder genauer gesagt den Distanzen jedes einzelnen Phonems zum Beobachtungsvektor besetzt. Die Aneinanderkettung der Phonem-Modelle zu Wortmodellen soll in Abbildung 4 verdeutlicht werden.

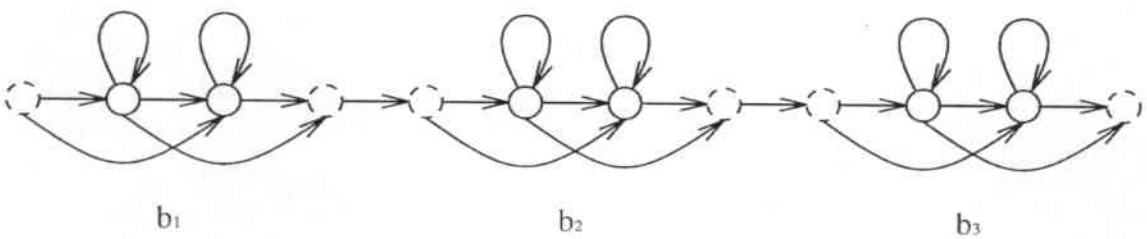

Abbildung 4: Zusammengesetzte Hidden-Markov-Kette für eine Wortfolge

$P(A|W)$ ist demnach die Wahrscheinlichkeit, daß die zusammengesetzte Markov-Kette die Folge $A = a_1, a_2, \ldots$ erzeugt, wenn man sie vom Anfangs- bis zum Endzustand durchläuft, wobei nochmal darauf hingewiesen sei, daß die Zustandsübergänge nicht für ein Phonem und dessen Wahrscheinlichkeit stehen, sondern alle möglichen Phonemwahrscheinlichkeiten bzw. Distanzen in Betracht ziehen.

Der tatsächliche Vorgang der Erkennung wird in der Hypothesensuche unter

[5] das in der vorliegenden Studienarbeit verwendete Dictionary umfaßte ca 2000 Worte und 43 Phoneme

Punkt 4 mit dem *One-Stage Dynamic Time Warping Algorithmus* genauer beschrieben.

2.3.3 Sprachmodellierung - das Trigramm-Modell

Eine weitere Aufgabe die sich stellt, liegt in der Konstruktion eines Sprachmodells, das P(W), also die Wahrscheinlichkeit, daß der Sprecher die Wortfolge W aussprechen will, berechnet.

Wenn man annähme, daß die Wahrscheinlichkeit ein Wort zu sprechen von allen vohergehenden Worten abhängt, dann ließe sich P(W) wie folgt berechnen:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_{i-1}, \ldots, w_1) \tag{31}$$

Jedoch stößt man bei dieser Berechnung bereits an die Grenzen des Machbaren, denn selbst bei einem kleinen Wortschatz der Größe N wäre die Anzahl N^i der benötigten Werte zu groß, um gespeichert zu werden. Deshalb reduziert man die Betrachtung der Historie eines Wortes auf die letzten beiden Vorgänger w_{i-1} und w_{i-2}. So entsprechen die Faktoren $P(w_i|w_{i-1}, \ldots, w_1)$ aus Gleichung 31 also $P(w_i|w_{i-1}, w_{i-2})$ was sich in der folgenden 'Trigramm-Gleichung' ausdrückt:

$$P(W) = P(w_1) \cdot P(w_2|w_1) \cdot \prod_{i=3}^{n} P(w_i|w_{i-1}, w_{i-2}) \tag{32}$$

Ein nicht unerhebliches Problem ergibt sich jedoch daraus, daß es keine genügend großen Datenbasen gibt, um diese Wahrscheinlichkeiten direkt durch die relative Häufigkeit im Text zu berechnen; die Zahl der verschiedenen Trigramme liegt bereits bei dem verwendeten eingeschränkten Wortschatz von 2000 Worten bei 8 Milliarden. Deshalb gilt es, eine Methode[6] zu finden, die die benötigten Wahrscheinlichkeiten - also die Unigramme, Bigramme und Trigramme - abschätzt, statt sie zu berechnen.

[6] zum Beispiel die von S. Katz ausgearbeitete Methode

2.3.4 Hypothesensuche - der One-Stage Dynamic Time Warping Algorithmus

Die Punkte zuvor haben nur dem Zähler der Bayes'schen Gleichung gegolten und da der Nenner nicht eine Funktion ist, die von W abhängt, muß P(A) auch nicht ausgewertet werden, um die Wortfolge $\hat{W}$ zu finden, die die Gleichung 29 erfüllt. Somit bleibt also nur noch das Problem, schnell eine Hypothese für $\hat{W}$ zu finden, für die $P(W|A)$ maximal ist.

Diese Hypothesensuche beginnt mit dem ersten Beobachtungsvektor und dem Versuch, diesen, wie auch alle weiteren, auf eine Wortfolge zu matchen, die aus Wörtern besteht, die wiederum aus Phonemen bestehen. Also beginnt die Suche damit, jedes im Wörterbuch vorkommende Wort zunächst auf den Beginn der Beobachtungsfolge zu matchen und jeden einzelnen Versuch solange auszubauen, bis für einige Ansätze offenbar wird, daß sie soviel schlechter als andere sind, daß sie einen bestimmten Grenzwert überschreiten und fallen gelassen werden können (oder auch nicht). Durch dieses Fallenlassen wird der Rechenaufwand erheblich verringert, jedoch kann natürlich ein eigentlich richtiger Weg irrtümlich als falsch abgelehnt werden.

Die übrigen Ansätze werden bis zum Wortende weiterverfolgt, um dann bei dem nächsten Beobachtungsvektor wiederum mit dem Versuch zu beginnen, jedes im Wörterbuch vorkommende Wort auf die weitere Beobachtungsfolge zu matchen. Dieser Vorgang setzt sich fort, bis das Ende der Akustikfolge erreicht ist. Danach beginnt die Auswertung der möglichen Wege von Anfang bis Ende und das beste Gesamtergebnis wird als Lösungshypothese vorgeschlagen.

Diese Vorgehensweise ermöglicht es, die gesamte Wortfolge zu betrachten, ohne zunächst eine Worterkennung durchzuführen, deren Ergebnis dann erst in eine Wortfolge zusammengesetzt werden muß. Es werden also keine Wortgrenzen gesucht. Diese entstehen sozusagen von alleine dort, wo das eine Wort im Erkennungsprozeß aufhört und das nächste beginnt.

Logischerweise gibt es zwei Übergangsregeln von Zuständen. Zum einen innerhalb der Wort-templates von Phonem zu Phonem, wobei auch die Phoneme in die 3 Zustände Anfang, Mitte, Ende aufgeteilt sind. Zum anderen zwischen den Wort-templates, also an den Wortgrenzen, wobei dieser Part durch die Bigramme des Sprachmodells geleistet wird.

2.4 Phonemerkennung oder Worterkennung

Der Unterschied zwischen Phonem- und Worterkennung liegt im Grunde lediglich darin, daß der Wortschatz bei der Phonemerkennung auf 43 unterschiedliche Worte oder besser Phoneme reduziert ist. Das andere Dictionary bringt natürlich auch mit sich, daß hier die Bigramme und Trigramme nicht auf den Worten, sondern auf den Phonemen aufbauen. Es entsteht also eine neue Grammatik, die während der Studienarbeit erst angefertigt werden mußte. Außerdem baut natürlich neben der Spracherkennung auch das nachträglich durchgeführte Forced Alignment auf diesen unterschiedlichen Sprachmodellen und Dictionaries auf.

3 Idee des Confidence Measures

3.1 Einführung

Wie wir in Kapitel 2.2.2 gesehen haben, verwendet der auf einem Hidden-Markov-Modell basierende Spracherkenner eine Bewertungsfunktion, der die Bayes'sche Gleichung zugrundeliegt. Dabei wird also, wie bereits beschrieben, die wahrscheinlichste Wortsequenz für eine gegebene akustische Eingabe nach folgender Formel ausgewählt:

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)} \tag{30}$$

Jedoch haben wir auch gesehen, daß die a-priori-Wahrscheinlichkeit der von der Signalverarbeitung ausgegebenen Folge von Akustik-Vektoren P(A) nicht in die Berechnung eingeht. Die Begründung für das Ignorieren der a-priori-Wahrscheinlichkeit P(A) liegt in der Tatsache, daß P(A) für alle Möglichkeiten, eine Wortfolge zusammenzustellen, gleich ist. Deshalb hat ein Weglassen in der Berechnung von $P(W|A)$ keine Auswirkungen auf die relative Reihenfolge der Güte der Satzhypothesen.
Weil bisher nur nach der besten Hypothese gesucht wurde, egal wie gut oder schlecht sie eigentlich wirklich war, genügte dieser Ansatz. Das bedeutet aber auch, daß die Bewertungen (Scores) des Spracherkenners für die Wort- und Satzhypothesen kein absolutes Wahrscheinlichkeitsmaß sondern nur eine relative Größe sind. Wir wissen, welche Äußerung am wahrscheinlichsten ist,

nicht aber, ob sie ein von uns erwartetes Mindestmaß an Übereinstimmung mit der akustischen Eingabe hat. Dieses in der Studienarbeit behandelte Vertrauensmaß für die Spracherkennung ist eine wichtige Information, um zum Beispiel neue Worte, die nicht im Wörterbuch aufgeführt sind, in der Spracheingabe zu bemerken oder um den Benutzer des Systems davon zu unterrichten, daß Mißverständnisse durch eine undeutliche Aussprache auftreten könnten, so daß er die Gelegenheit zu einer Wiederholung des Satzes bekommt. Natürlich gibt es viele Wege, ein Vertrauensmaß für eine Hypothese zu erlangen. Man kann die Syntax, Semantik, Pragmatik oder die Struktur spontaner Sprache als Wissensbasis verwenden, jedoch haben alle Kriterien ihre Schwächen. In dieser Arbeit wird der Versuch unternommen, dieses Konfidenz-Maß über die akustische 'Paßform' zu schätzen und das durchgeführte Experiment soll den Erfolg dieser Maßnahme dokumentieren.

3.2 Normalisierung des Wort-Scores

In dieser Arbeit wird untersucht, wie gut das System falsch erkannte Worte zurückweisen kann, wenn es dazu nur die akustische Information als Anhaltspunkt nimmt. Der Gedanke dabei ist, akustische Merkmale herauszuarbeiten, die es erlauben, korrekt erkannte Worte von falschen Worthypothesen möglichst gut zu unterscheiden.
Wie bereits zuvor eingehend betrachtet, benutzt der Spracherkenner des JANUS-Systems ein akustisches Modell auf der Basis eines Hidden-Markov-Modells und ein Sprachmodell auf der Basis von Unigrammen, Bigrammen und Trigrammen um eine Maximum-Likelihood-Schätzung für die gesuchten Wortfolgen zu erzeugen. Dem Benutzer bleibt die Wahl, ob eine Suche nur nach dem besten Pfad oder den n-besten Pfaden durchgeführt wird. Die vom Erkenner erzeugten Scores, die eine gewichtete Summe der logarithmierten Wahrscheinlichkeiten des Akustik- und des Sprachmodells darstellen, geben das Maß, wie wahrscheinlich ein Suchpfad nach der Wortfolge ist. Die Pfadsuche wird wie bereits zuvor beschrieben, jeweils um ein weiteres Wort vorangetrieben, indem zunächst der Score für die akustische Übereinstimmung jedes möglichen Wortes berechnet wird, um dann diesen Score mit der Übergangswahrscheinlichkeit des Sprachmodells für das jeweilige Wort und dem Score des bisher betrachteten Pfades zu verknüpfen. Wie bereits kritisiert, sind diese Scores jedoch nicht normalisiert, geben also auch kein absolutes Maß für die Güte eines Wortpfades, sondern sind nur im Vergleich mit den Scores für andere Satzhypothesen desselben Sprachbeispiels von Bedeutung.

Die Scores sind daher nicht geeignet, falsche Hypothesen zu erkennen und diese als solche zurückzuweisen.

In der Absicht, falsch erkannte Worte zurückweisen zu können, also die Güte der Erkennerleistung zu bestimmen, bedarf es einer geeigneten Normalisierung der vom Erkenner erzeugten Scores. In der vorliegenden Arbeit liegt der Ansatz in einer zur Worterkennung parallel laufenden Phonemerkennung (siehe Abbildung 5), wobei die Scores der Phonemerkennung zur Normalisierung der Worterkennnerscores dienen.

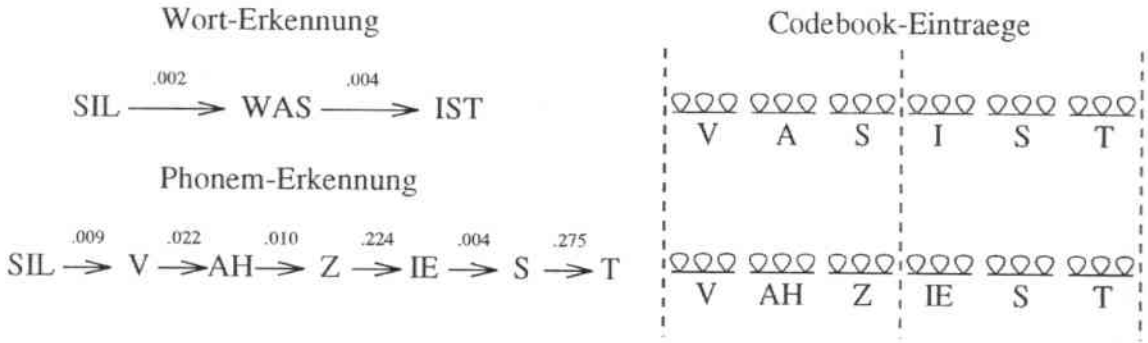

Abbildung 5: parallele Durchführung von Wort- und Phonemerkennung zur Normalisierung

Dieser Vorgang ist in folgender Formel beschrieben:

$$normscore = \frac{wortscore \; - \; phonemscore}{Anzahl \; der \; Frames} \tag{33}$$

Der Phonemerkenner benutzt als 'Sprachmodell' Bigramme von Phonemübergängen, genauso wie der Worterkenner Bigramme für die Wortübergänge verwendet. Das Sprachmodell des Worterkenners liegt demnach in reduzierter Form vor, da sich die Vergangenheit jedes Wortes lediglich auf ein Wort stützt.

Bei der durchgeführten Normalisierung, weicht die vorliegende Ausarbeitung etwas von dem zugrundeliegenden Paper von Sheryl R. Young und Wayne Ward [1.1] ab. Zwar wird auch hier der Score der Phonemerkennung von dem Score der Worterkennung für dieselbe Menge von Frames subtrahiert, jedoch wird die Phonemerkennung nicht innerhalb der Wortgrenzen durchgeführt. Stattdessen findet eine durchgehende Phonemerkennung auf dem gesamten Sprachbeispiel statt.

Wenn man Gleichung 33 betrachtet und sich verdeutlicht, daß es sich bei den Scores um logarithmierte Wahrscheinlichkeiten handelt, dann erkennt man, daß die Subtraktion der Scores einer Division der Wahrscheinlichkeiten entspricht. Das Ergebnis dieser Subtraktion wird anschließend durch die Framelänge des Wortes dividiert. Durch das Verwenden der Phonemerkennung als 'Normierungsfaktor', wird der Bezug zum akustischen Signal weitgehend unabhängig von dem verwendeten Sprachmodell und dem zur Verfügung stehenden Vokabular hergestellt.

3.3 Erstellung von Histogrammen

Mit Hilfe der normalisierten Wortscores wird anhand von Trainingsbeispielen für jedes Wort ein Histogramm über diesem normalisierten Wortscore erzeugt. Dabei sollen die Trainingsbeispiele Aufschluß darüber geben, wie wahrscheinlich ein Wort richtig oder falsch erkannt wurde, wenn es einen bestimmten normalisierten Score erhalten hat. Zu diesem Zweck werden die normalisierten Scores der einzelnen Vorkommen mit dem Vermerk 'richtig/falsch erkannt' abgespeichert und später in den Scorebereichen des Histogramms ausgewertet, so daß für jeden Scorebereich eine Wahrscheinlichkeit für die Richtigkeit des erkannten Wortes vorliegt. Diese Wahrscheinlichkeitsverteilung entspricht dem gesuchten Vertrauensmaß und stellt somit das Endprodukt dieser Studienarbeit dar.

4 Umsetzung der Aufgabe im JANUS-System

Die Bearbeitung der Aufgabe gliederte sich in eine Abfolge von Schritten, die hier stichwortartig den Verlauf der Studienarbeit verdeutlichen sollen.

(i) Vorbereitung

Zunächst galt es, für die Phonemerkennung sowohl ein geeignetes Sprachmodell als auch ein Dictionary zu erstellen. Das Dictionary läßt sich als einfache 1:1-Übersetzung der Phoneme auf sich selbst realisieren. Zur Erstellung einer Phonem-Grammatik bedurfte es zunächst eines Programms, das größere Mengen von Text mittels eines passenden Wörterbuchs in Phoneme übersetzt. Aus diesem Phonemtext konnten dann Übergangswahrscheinlichkeiten in Form von Bigrammen und a-priori-Wahrscheinlichkeiten in Form von Unigrammen erzeugt werden.

(ii) Erkennung

Nachdem geeignete Parameter für die Wort- und Phonemerkennung gefunden waren, konnte die Erkennung von 1138 gesprochenen Sätzen beginnen, um die für das Confidence Measure benötigten Daten zu liefern.

(iii) Erstellen von Utterances

Aus den zuvor erhaltenen Wort- und Phonemhypothesen mußten nun mittels eines kleinen Programms Utterances erstellt werden, die für den nächsten Schritt, das Forced Alignment, benötigt wurden.

(iv) Forced Alignment

Das Forced Alignment wurde durchgeführt, um zum einen die genauen Wortgrenzen mit den dazugehörigen Wortscores zu erhalten. Zum anderen wurden für die Phonemerkennung die entsprechenden Phonempfade ausgegeben. Diese geben für jeden Frame den Zustand, als auch den aufsummierte Score an.

(v) Alignment

Das zur Verfügung gestellte Tool `align` ermöglicht es, die korrekt erkannten Worte bzw. Phoneme von den falsch erkannten zu unterscheiden, um eine Bewertung für die später aufzustellenden Histogramme zu ermöglichen.

(vi) Hauptprogramm

Hier wurde aus den bisher erhaltenen Daten die für die Erstellung der Histogramme benötigte Information herausgezogen. Das bedeutet, daß zunächst die Wortscores mittels der Phonemscores normalisiert wurden, um dann den normalisierten Wortscores die aus `align` erhaltene Bewertung (richtig/falsch) zuzuordnen. Die somit erhaltenen Werte wurden in einem Binären Suchbaum abgelegt, um später für die Auswertung in Form von Histogrammen schnell auffindbar zu sein.

(vii) Anwendung und Auswertung

In diesem Schritt wurden die fertigen Histogramme auf einer Menge von 124 Beispielsätzen getestet, um den Beweis zu erbringen, daß mit Hilfe des Sicherheitsmaßes in Form von Histogrammen ein relativ sicheres Verwerfen von falsch erkannten Worten möglich ist. Dies geschah in der Form, daß zunächst über das Confidence Measure bewertet wurde, ob ein Wort richtig oder falsch erkannt wurde, um danach die erhaltene Bewertung mit der Wirklichkeit, also dem tatsächlichen Ergebnis[7], zu vergleichen. Die daraus resultierenden Versuchsergebnisse sind im nächsten Abschnitt nachzulesen.

[7] wieder mit Hilfe des bereitgestellten Tools align erhalten

5 Ergebnisse

In dem am Ende der Studienarbeit durchgeführten Experiment wurde auf eine geringe Anzahl von Test-Sprachbeispielen zurückgegriffen. Um bessere Werte zu erhalten, ist es natürlich anzuraten, sowohl die Anzahl der Trainingsbeispiele als auch die Anzahl der Testbeispiele zu erhöhen. Außerdem ist durch die aktuelle Umstellung auf einen größeren Phonemsatz eine Verbesserung der Erkennerleistung im Gesamten zu erwarten.

5.1 Versuchsergebnisse

Bei dem durchgeführten Experiment wurden zunächst Histogramme für jedes trainierte Wort aufgestellt, die darüber Auskunft geben sollen, wie wahrscheinlich ein Wort richtig erkannt wurde, wenn es in einem bestimmten Score-Bereich liegt. Die Scorebereiche der Histogramme sind unterschiedlich groß, da die Aufteilung der Bereiche dynamisch geschieht. Die Ergebnisse erlangen dadurch eine größere Aussagekraft, da es bei einer statischen Aufteilung der Scorebereiche in gleichgroße Teile vorkommt, daß *Ausreißern* eine zu starke Aussagekraft zugewiesen wird. Wenn also ein solcher *Ausreißer* am Rand eines Bereiches liegt, wird im Extremfall von diesem einen Wert die spätere Bewertung des gesamten Scorebereiches abhängen.

Grenzwert	corr acc [%]	corr rej [%]
20 %	95.2	15.5
40 %	88.0	34.0
45 %	85.5	38.7
50 %	81.1	43.5
55 %	77.6	47.4
65 %	66.9	61.0
75 %	50.7	72.4
85 %	35.9	81.3

Tabelle 2: Grenzwerte und Resultate

Diese Histogramme wurden in dem durchgeführten Experiment in der Weise gebraucht, daß ein vorher festgesetzter Grenzwert jeweils mit dem Prozentrang verglichen wurde, der in dem für den Score gültigen Score-Bereich erreicht wurde. Für den Fall, daß der Prozentrang größer war, wurde das Wort

Wort	corr acc [%]	corr rej [%]	Wort	corr acc [%]	corr rej [%]
bißchen	100.0	50.0	da	90.0	18.2
dann	88.5	66.7	der	100.0	16.7
ein'	100.0	66.7	habe	100.0	100.0
ich	100.0	2.6	ist	66.7	75.0
könnten	100.0	100.0	möglich	100.0	100.0
mit	100.0	28.6	morgens	100.0	33.3
okay	100.0	100.0	sie	100.0	46.7
um	100.0	12.5	würde	100.0	23.1

Tabelle 3: korrektes Akzeptieren und Zurückweisen von Worten bei einem Grenzwert von 40 %

als richtig angenommen, für den anderen Fall als falsch abgelehnt.
Für verschiedene Grenzwerte durchgeführt, ergaben sich dabei die in Tabelle 2 aufgeführten Werte. In Tabelle 3 sind bei einem Grenzwert von 40 % die erzielten Resultate für einzelne Worte dargestellt. Hierbei haben die Werte folgende Bedeutung:

$$corr\ acc = \frac{korrekt\ akzeptierte\ Worte}{gesamt\ korrekt\ erkannte\ Worte}\ nicht\ \frac{korr\ akzeptierte\ Worte}{ges.\ akzeptierte\ Worte}$$

$$corr\ rej = \frac{korrekt\ zurückgewiesene\ Worte}{gesamt\ falsch\ erkannte\ Worte}\ nicht\ \frac{korr\ zurückgew.\ Worte}{ges.\ zurückgew.\ Worte}$$

5.2 Ausblick

Die in dieser Studienarbeit erzielten Ergebnisse sollen einen Einblick vermitteln, welche Möglichkeiten durch ein geeignetes *Sicherheitsmaß* bestehen, die Erkennerleistung des JANUS-Spracherkenners zu verbessern. Durch den zuletzt erfolgten Umstieg auf einen größeren Phonemsatz sollten sogar noch bessere Ergebnisse zu erwarten sein. Es ist nun daran, die erzielten Resultate in geeigneter Weise in den Spracherkenner einzubinden, so daß aus puren Zahlen wahrnehmbare Verbesserungen in der Spracherkennung entstehen.
Zum Beispiel wäre durch eine Bestimmung der Erkennungsqualität dem System die Möglichkeit gegeben, bei einem schlecht erkannten Wort nochmals nachzufragen, um Mißverständnisse zu vermeiden. Außerdem wäre es vorstellbar, neue, noch nicht im Wortschatz befindliche Worte zu erkennen, um diese dann zu trainieren und in das Vokabular aufzunehmen. In jedem Fall

besteht durch ein geeignetes Sicherheitsmaß die Möglichkeit, die Erkennerleistung abzuschätzen und bei kritischen Erkennungsvorgängen den Wahrheitsgehalt der Ergebnisse zu bewerten.

A Kurzbeschreibung und Benutzung der Software

In diesem Abschnitt soll der Aufruf der zur Bestimmung des Confidence Measures benötigten Programme genauso wie das Zusammenstellen der Ergebnisse beschrieben werden. Noch sind Verbesserungsmöglichkeiten denkbar, die eine benutzerfreundlichere Bedienung bieten würden. Hier ist jedoch lediglich der Umgang mit der bisher zur Verfügung stehenden Grobform dargestellt.

A.1 Umgang mit dem Programm text-in-phon.c

Kurzbeschreibung:
Dieses Programm transferiert eine Textdatei mit Hilfe des verwendeten Wörterbuchs in eine Phonemdatei, das heißt, jedes Wort der Textdatei wird durch die Übersetzung in Phonemschreibweise ersetzt. Das entstehende Phonemfile dient der späteren Erzeugung der Phonemgrammatiken, also der Berechnung der Mono - und Bigramme.

Aufruf des Programms text-in-phon.c

tip *textdatei-name*

Der Anwender hat dafür Sorge zu tragen, daß die von Ihm angegebene Textdatei existiert. Das verwendete Dictionary `dict.220` ist bereits festgelegt, da die Versuchsdaten einheitlich darauf basieren. Die Ergebnisdatei, in der nach Programmablauf der in Phoneme übersetzte Text liegt, trägt den Namen `phonemutterance`. Für den Fall, daß Schwierigkeiten beim Öffnen der Files entstehen, wird der Anwender durch die entsprechenden Fehlermeldungen darauf hingewiesen. Das Programm bricht dann ab, und der Aufruf muß mit korrekten Angaben wiederholt werden.

A.2 Zwischenschritte

Unter Zuhilfenahme des zuvor entstandenen Phonemtextfiles, kann im nächsten Schritt eine Phonemgrammatik, also ein Phonem-Sprachmodell, berechnet werden. Diese Grammatik wird im darauffolgenden Phonemerkennungsvorgang verwendet. Dazu müssen folgende Files bereitliegen:

1) die `fft-filenamenliste`

2) das `phonemdictionary`

3) die `phonemgrammatik`

4) das dementsprechend geänderte `dafaultsfile`

Aufruf des Phonemerkenners

janus *-X phonemerkennungs-defaultsfile*

Die Ausgabe der Phonemhypothesen erfolgt in die Datei `hypo_1.all`. Diese Datei wird zur weiteren Verwendung benötigt.

Parallel dazu verläuft der Vorgang der Worterkennung. Die Unterschiede liegen im wesentlichen in den verschiedenen Wörterbüchern und Grammatiken. Daraus folgt natürlich auch, daß das defaultsfile anders aussieht. Die fft-files *müssen* jedoch gleich sein, da ja auf denselben Sprachbeispielen sowohl die Wort- als auch die Phonem-Erkennung stattfindet.

Aufruf des Worterkenners

janus *-X worterkennungs-defaultsfile*

A.3 Umgang mit den Programmen hypo-in-utts-p/w.c

Kurzbeschreibung:

Dieses Programm wandelt die Phonem- bzw. Worthypothesen in sogenannte Utterances um, mit denen dann ein Forced-Alignment durchgeführt werden kann. Außerdem wird eine reine Worthypothesendatei erzeugt, bei der der Score und der Bezeichner des zugehörigen fft-Files entfernt werden. Diese dient nachher zum Vergleich zwischen Hypothese und Referenz.

Aufruf des Programms hypo-in-utts-p/w.c

hyutt-p/w *p/w-hypofile p/w-hypofile(bereinigt) p/w-utt-filenamenliste*

Die zu den fft-files gehörenden utterances werden mit den Bezeichnern der fft-files benannt, wobei zur Unterscheidung der Daten zusätzlich am Kopf phon_ bzw word_ eingefügt und am Ende .fft durch .utt ersetzt wird.

A.4 Zwischenschritte

Die zuvor erzeugten Utterancefiles werden zur Durchführung des *forced alignment* benötigt, bei dem die exakten Wortgrenzen und Wortscores bzw. der Phonempfad und die aufsummierten Framescores bestimmt werden. Die Durchführung gleicht dem Erkennervorgang, so daß prinzipiell ähnliche Daten vorliegen müssen. Hier werden lediglich die fft-filenamenliste gegen die utt-filenamenliste und die utt-files gegen die fft-files ausgetauscht. Als Ausgabe liegen nach dem forced alignment die zu den Phonem-Utterances gehörenden Phonem-Pfade und -Scores im fft-file-Bezeichner mit Endung .path vor. Für die Wort-Utterances liegen die Wortgrenzen und Wortscores in Files mit der Endung .wbs.

In einem weiteren Schritt werden mittels des Tools `align` die korrekt erkannten Worte von den falsch erkannten unterschieden. Dazu muß zunächst eine Referenzdatei erstellt werden, in der die tatsächlich gesprochenen Sätze liegen. Diese Datei wird dann mit dem zuvor erstellten bereinigten Hypothesenfile verglichen. In der enstehenden Ausgabedatei werden zum einen die Sätze der Hypothese und der Referenz so markiert, daß korrekt erkannte Worte klein, falsch erkannte groß geschrieben sind. Die zum anderen mitgelieferte Statistik über die Erkennerleistung ist für die weitere Arbeit nicht von Belang.

Aufruf des Tools align

align -def *referenz* **-hyp** *hypothese* > *align-file*

A.5 Umgang mit den Programmen histodist.c/histoquant.c

Kurzbeschreibung:

Die Programme `histodist.c` und `histoquant.c` berechnen Histogramme für jedes in der Erkennung vorkommende Wort. Das heißt bildlich gesprochen, die Anzahl der korrekten bzw. falschen Erkennungen eines Wortes wird über dem normalisierten Score aufgetragen. Die Normalisierung ist ein bereits zuvor geschilderter Vorgang, bei dem zunächst der Phonemscore vom Wortscore subtrahiert und das Ergebnis dann durch die Wortlänge in Frames dividiert wird.

Der Unterschied der beiden Programme liegt in der Einteilung des normalisierten Scores also wiederum bildlich gesprochen in der Einteilung der Säulen der zu bildenden Histogramme. Bei `histodist.c` wird die Spanne der erzielten Scores in gleiche Teile aufgeteilt, so daß manche Bereiche mit weniger Beispielen besetzt sind als andere. Dagegen ist die Einteilung der Scores bei `histoquant.c` dynamisch und das Augenmerk liegt darauf, daß in jedem Bereich gleich viele Worterkennungen liegen. Die Anzahl der Bereiche ist frei wählbar.

Aufruf der Programme histodist.c/histoquant.c

hisdis/hisqua *wbs-listenname path-listenname alignfile*

Die Ergebnisse werden in die Datei `histogram` zur optischen Kontrolle und beim später auch verwendeten Programm `histoquant.c` ebenfalls in die Datei `prueffile` zur weiteren Verarbeitung in `histpruef.c` geschrieben und sind im Prinzip das Endprodukt, also das Confidence Measure für jedes erkannte Wort.

Anmerkung zum Programm: Um das Einfügen der Einträge schnell zu gestalten, wurde die Struktur eines binären Suchbaums benutzt. Die Einträge dieses Baumes bestehen aus den Worten und einer zu den Worten gehörenden Kette von Vorkommen mit dem Hinweis, ob die Erkennung korrekt oder falsch war.

A.6 Die Überprüfung mittels histpruef.c

Kurzbeschreibung:

Im wesentlichen werden in `histpruef.c` die Confidence Measures oder ge-

nauer die zuvor berechneten Histogramme für jedes Wort dazu verwendet, die Erkennung auf bereits beschriebene Weise zu überprüfen. Dazu werden die Test-Beispiele zunächst durch die Schritte Wort/Phonemerkennung und Forced Alignment wie zuvor beschrieben aufbereitet. Danach werden in `histpruef.c` die Worte und die dazugehörigen normalisierten Scores mit Hilfe der Histogramme bewertet. Der zu Beginn angegebene Grenzwert wird zu diesem Zweck mit dem Prozentrang des Wortes in dem entsprechenden Scorebereich des Histogramms verglichen. Ist der Prozentrang größer, so wird das Wort als korrekt bewertet, sonst als falsch.

Die Ausgabedatei des Programms beinhaltet die Qualität der Bewertung für jedes Wort und die Güte der Confidence Measures im gesamten.

Aufruf des Programms histpruef.c

cmpruef *wbs-listenname path-listenname alignfile ergebnisfile*

Die Angabe der Namen muß natürlich auch hier korrekt sein. Beim Ergebnisfile ist darauf zu achten, daß nicht eine bereits existierende Datei überschrieben wird.

Anmerkung zum Programm: Der Grenzwert ist im Moment noch 'von Hand' einzustellen, sprich, es ist ein direkter Eingriff ins Programm nötig, um den Wert von **thresh** zu ändern.

B Literatur

[1.1] Young, S.R., Ward, W.: Recognition Confidence Measure For Spontaneous Spoken Dialog. IEEE ICASSP-93, S. 1177 ff

[2.1] Achilles, D.: Die Fourier-Transformation in der Signalverarbeitung. 2. A. Springer 1985

[2.2] Programs for digital signal processing. IEEE Acoustics, Speech and Signal Processing, New York, 1979

[2.3] Raj Reddy, D.: Speech Recognition by Machine: A Review. IEEE Proceedings 64(4), S. 502 ff, 1976

[2.4] Davis, S.B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continiously Spoken Sentences. IEEE 1980

[2.5] Gray, R.M.: Vector Quantization. IEEE 1984

[2.6] Rabiner, L.R., Levinson, S.E.: Isolated and Connected Word Recognition - Theory and Selected Applications. IEEE 1981

[2.7] Rabiner, L.R.: A Tutorial on Hiddden Markov Models and Selected Applications in Speech Recognition. IEEE 1989

[2.8] Rabiner, L.R., Wilpon, J.G., Soong, F.K.: High Performance Connected Digit Recognition Using Hidden Markov Models. IEEE 1989

[2.9] Jelinek, F.: The Development of an Experimental Discrete Dictation Recognizer. IEEE 1985

[2.10] Ney,H.: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. IEEE 1984