Diplomarbeit

Erkennung von Sprache in Telephonqualität

Steven Bird
Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe

27. Juni 1995

Ich ekläre hiermit an Eides Statt, daß ich die vorliegende Diplomarbeit selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Literaturquellen sind im Literaturverzeichnis vollständig aufgeführt.

Karlsruhe, den 27.6.1995

Steven Bird
Klosterweg 28
76131 Karlsruhe

Mein Dank gilt Prof. Waibel und Prof. Wolf, die mir diese Diplomarbeit ermöglicht haben.

Ganz besonders herzlichen Dank möchte ich Herman Hild und Martin aussprechen.

Unterstützung fand ich auch bei den Mitarbeiten des Instituts für Logik, Komplexität und Deduktionssysteme.

I also owe a great debt to Dr. E.V. Jones, at the Univerity of Essex, without whose advice and motivation I may never have completed the Tripartite program.

Finally, thanks to Susanne Habermann, whose multilingual abilities have made my German infinately more understandable, than it ever otherwise would have been.

Inhaltsverzeichnis

Zusammenfassung

Das Ziel dieser Diplomarbeit ist, verschiedene Vorverarbeitungsmethoden zu erforschen, um die automatische Erkennung, von geräuschhaltigen Sprachsignalen mit Computern zu verbessern. Der Schwerpunkt liegt auf der Erkennung von Sprache in Telephonqualität. Mehrere Methoden wurden getestet und kombiniert, um eine optimale Lösung zu finden. Jede Methode wird mit zwei verschiedenen Erkennern getestet: dem Janus-Spracherkenner - er basiert auf einem HMM (englisch: *Hidden Markov Model*) und einem Erkenner basierend auf neuronalen Netzen.

Kapitel 1

Einführung

Nach einer kurzen Motivation zur Spracherkennung in Telephonqualität, werden die grundsätzliche Sprachverarbeitungstheorie und die wichtigsten Merkmale der Sprachsignale eingeführt.

1.1 Automatische Buchstabierung

Automatische Auskunftssysteme werden in der Zukunft eine immer größere Bedeutung erlangen. Dabei wird die Spracherkennung über das Telefon eine wichtige Rolle spielen. Eine typische Anwendung ist etwa eine automatische Telefonnummerauskunft, bei der buchstabierte Namen erkannt werden müssen (siehe Abbildung 1.1).

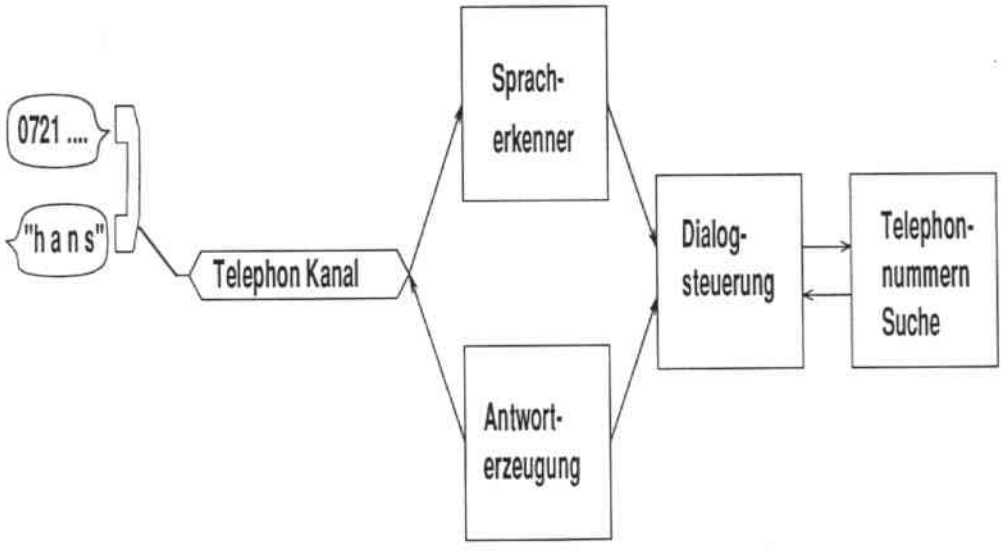

Abbildung 1.1: Ein Automatisches Telephonnummerauskunftsystem

Diese Diplomarbeit versucht die Erkennungsphase des Systems zu verbessern. Zwei Spracherkenner werden getestet:

Der MSTDNN - Spracherkenner (englisch:*Multi-State Time Delay Neural Network*) basiert auf neuronalen Netzen und erkennt beliebige Sequenzen von Buchstaben. Er arbeitet bisher mit Sprachdaten, die in einem Sprachlabor mit guten Mikrofonen in geräuscharmer Umgebung aufgenommen wurden. Die Erkennungsgenauigkeit mit dieser "high quality" Sprache ist gut (98.5/92.0 % *word accuracy* für sprecherabhängige/sprecherunabhängige Erkennung uf englishen Buchstaben), aber die Erkennungsfehler wachsen ziemlich schnell, wenn er mit einem geräuschhaltigen Sprachsignal getestet wird.

Der Janus-Spracherkenner, der für das Übersetzungsystem JANUS entwickelt worden ist, basiert auf einem HMM *Hidden Markov Model.* Das Janusprojekt ist sehr groß, und innerhalb des C-STAR[1] Projekts versucht es den Computer zu benutzen um die Kommunikation zweier Menschen zu ermöglichen, die verschiedene Sprachen benutzen. In dieser Diplomarbeit wird, nur der Janus-Spracherkenner benuzt.

Die theoretischen Grundlagen der zwei Erkenner und seine Implementierung werden im Kapiteln 8 und 9 dargestellt.

1.2 Das Sprachsignal

Das Spachsignal hat gewisse Eigenschaften oder Merkmale, die klar zu identifizieren sind. Deshalb hat man mehrere spezielle Verarbeitungsmethoden entwickelt um das Sprachsignal zu behandeln. Einige von diesen Methoden sind Varianten der normalen digitalen Signalverarbeitung, während andere speziell für die Aufgabe der Sprachverarbeitung entwickelt worden sind.

Zuerst nimmt man die folgenden Sprachsignalcharakteristika an:

- Obwohl das Signal nicht stationär ist, kann man es als stationär betrachten, wenn man einem kürzen Ausschnitt analisiert (ungefähr 10ms [7]).
- Sprache wird von drei tonerzeugenden Prozeßen bestimmt:
  - Stimmhafte Laute (englisch: *Voiced Sounds*): Geschaffen durch die Anregung des Vokaltrakts durch quasi-periodischen Luftdruckpulse.
  - Frikativlaute (englisch:*fricative sounds*): Der Vokaltrakt wird verenkt. Luft strömt hindurch und erzeugt Turbulenzen im Vokaltract.

[1] Eine Kooperation der Univerität Karlsruhe mit der Carnegie-Mellon University (Pittsburg), ATR Interpreting Telephone Laboratories (Osaka) und der SIEMENS AG.

  - Plosivlaute (englisch:*plosive sounds*): Das Vokaltrakt wird verschlossen, um eine Drucksteigerung zu erlauben. Dieser Druck wird dann durch schnelles Öffnen freigegeben.

- Die Vokaltrakt wird durch einen linearen zeitinvarianten (englisch: *Linear Time Invariant (LTI)*) Filter modelliert .

Sprache wird ein Filter, dessen Impulsantwort der des Vokaltraktes entspricht, angeregt mit zwei möglichen Eingaben, simulierte. Die zwei Eingaben sind:

- Gaußsches Rauschen, das die stimmlosen Laute erzeugt.

- Ein periodischer Impulszug, der die stimmhaften Laute erzeugt.

Es gibt viele Möglichkeiten für die digitale Verarbeitung eines Sprachsignals. Jedes Methode versucht die maximale Information zu erhalten, während sie versucht die Fehler und das Rauschen zu reduzieren und das Signal effektiv zu speichern und zu verarbeiten. Natürlich stehen diese Kriterien in Konflikt miteinander, so wird die Methode abgestimmt nach gewünschter Gewichtung der Kriterien.

1.3 Spracherkennung

In Abbildung 1.2 sieht man die gesonderten Blöcke eines Spracherkennungsytems. Hier läuft das Sprachsignal $s(t)$ durch die Vorverarbeitung (hier findet man auch die A/D-Wandlung), dann kommt es in einem Vorgang von Merkmalextraktion/Erkennung, und danach mit Hilfe von einem Lexikon die Klassifikation des Wortes.

Die Aufgabe dieses Sracherkenners ist es, einen ganzen, gesprochenen Satz und möglichst fehlerfrei zu erkennen. Um überhaupt Wörter aus den empfangenen Lauten erkennen zu können, braucht man ein Wörterbuch, das diese Wörter enthält. Jedes Wort ist darin durch Phoneme, beschrieben. Hierin liegt eine Schwierigkeit, für den Erkenner, weil es Phoneme gibt, die sich nur in der Stimmhaftigkeit unterscheiden, die Artikulation aber gleich ist. Für diese Phoneme z.B. /t/ und /d/ könntet den unterschied schwierig zu machen werden, für Mensch und Computer.

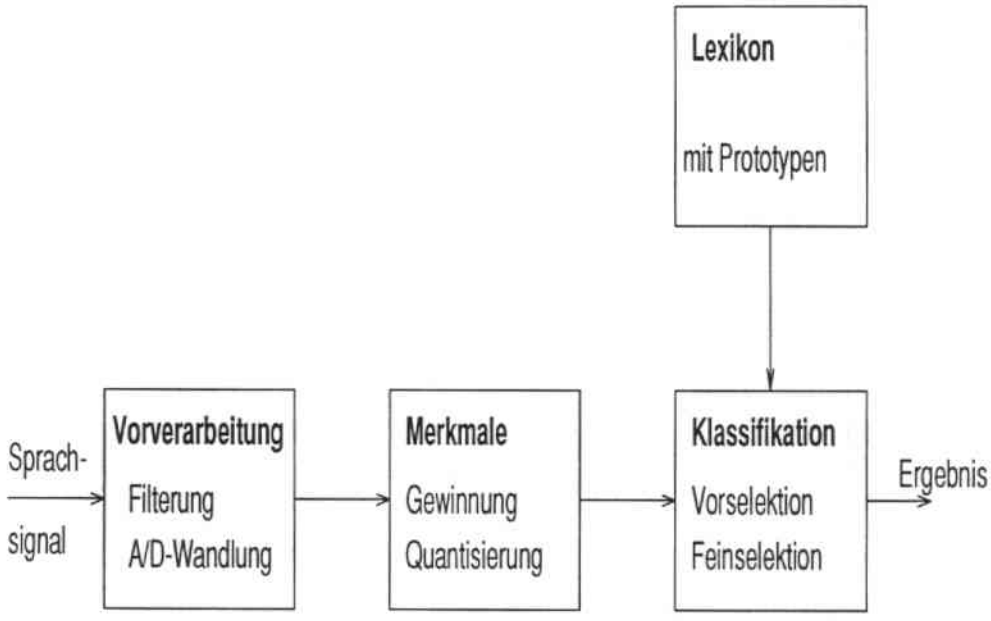

Abbildung 1.2: Eine Spracherkennungsysteme

Kapitel 2

Das Telephonsprachsignal

In diesem Kapitel werden die Hauptcharakteristika eines Telephon-Sprachsignals gegeben. Dann kommt die Probleme dazu die diese Charakteristika für einen automatischen Spracherkenner darstellen.

Es gibt verschiedene Störungen, die man in einem Telephonsignal finden kann, z.B.:

- Niederfrequente Laute
  (englisch: *Low frequency tones*)
- Additives stationäres Rauschen
  (englisch: *Additive stationary noise*)
- Impulsrauschen
  (englisch: *Impulse noise*)
- Amplituden- und Phasenverzerrung
  (englisch: *Amplitude and phase jitter*)
- Intermodulationsstörung
  (englisch: *Intermodulation distortion*)
- Unbekannte Amplitudenverstärung und Phasenverschiebung
  (englisch: *Unknown channel gain and phase response*)

Die wichstigsten Störungen (niederfrequente Laute und additives stationäres Rauschen [6]) kommen dabei von zwei verschiedenen Geräuschquellen. Das führt zu folgender Darstellung des geräuschhaltigen Sprachsignals:

$$y(t) = [s(t) + n(t)] \otimes h(t) \tag{2.1}$$

Hier ist $s(t)$ das saubere Signal, $n(t)$ das additives Rauschen, das von der Umwelt und dem Kanal kommt und $h(t)$ die Impulsantwort des Kanals.

Es gibt also zwei verschiedene Arten von Rauschen: Das erste, das additive Rauschen (englisch:*additive noise*), das im Zeitraum (und deshalb Frequenzraum) ist. Dieses besteht aus allen Rauschen und Störungen, die vom Sprecher und vom Telephonnetz ausgehen. Das zweite, bekannt als Faltungsrauschen (englisch: *convolutional noise*), ist das Ergebnis der Faltung des Sprachsignals mit der Impulsantwort des Telephonkanals. Diese Störung ergibt einen Bandpassfiltereffekt von bis zu 200 - 3300Hz, und eine sich langsam verändernde Störung [12]. Diese zwei Rauscharten werden getrennt betrachtet.

2.1 Additives Rauschen

Weil das Signal digital ist, stellt dieses Rauschen kein Problem dar wenn es weniger als die Hälfte des Quantizierungsniveau beträgt. Dann wird das Signal den richtigen Wert beibehalten. Sobald, jedoch, diese Störung größer wird als dieses Niveau, werden Fehler auftreten. Auch wird der Fehler mindestens die Größe dieses Quantizierungsniveau betragen, weil die A/D Wandlung den nächsten discreten Wert für dieses Muster ausgeben wird.

Diese Störung wird als feststehend angenommen. Wenn das so ist, könnte man sie eliminieren indem man einfach die Leistungs de Rauschens in einer Stilleperiode abschätzt, und dann diesen Wert vom Leistungspektrum des geräuschhaltigens Sprachsignals abziehen. Diese Methode wird in Kapitel 5 erklärt.

2.2 Faltungsrauschen

Aus Messungen von echten Telephondaten sieht man, daß diese Faltungsstörung eine sich langsam in der Zeit verändende und deshalb eine niederfrequente Komponent ist [1]. Da es eine Faltungsstörung ist, bedeutet das, daß man es wie eine additive Komponente im logarithmischen Spektrum von Sprache betrachten kann. Das führt zur Rauschunterdrückungsmethode *Mean Subtraction* die in Kapitel 3 eklärt wird.

Es gibt auch eine andere Lösung für das Problem der beiden Rauscharten; machen die Signal so die Erkenner weniger empfindlich an dieser Rauschen ist. Dieser hat mehrere Methoden und weil es die beiden Rauschen zu eliminieren versucht ist es warscheinlich besser. Die andere Methoden, die in dieser Diplomarbei getestet werden basieren auf dieses Gründ.

Kapitel 3

Signalverarbeitung

Dieser Kapiteln gibt die Gründ hinter die Vorverarbeitung eines Sprachsignals, das zu erkennen ist. Danach kommt die Detailen des front-end Voverarbeitung (A/D Wandlung, FFT) des Sprachsignal.

3.1 Sprachverarbeitung

Nacher wird der Ausdruck "Worte" eine Buchstabe berichten. Der Ausdruck "Satz" berichtet ein Schnur des Wortes.

Die Vorverarbeitungsphase versucht alle nutzlos Information (Rauschen, Falsche information) von den ankommend Sprachsignal zu beseitigen. In diese Phase man macht der signal so erkennbar wie möglich. Das heißt alle die Merkmalen von Worten mussen so unterschiedlich wie möglich sein und so gut definiert wie möglich werden. Die Auffassung von was gut erkennbar ist, kommt natürlich von die Spracherkenner.

Zum Gründvorerarbeitung wird das Signal am 8kHz abgetastet (das gibt ein Effektivesbandbreits von 4kHz) der ist genug, alle das Signalsinformations zu bewahren, gegeben der Bandbreit des Telephonkanal von ungefähr 3.3kHz. Jedes Abzeitpunkt wird dann mit einem 16-Bit-Wert quantiziert.

Nach der A/D-Wandlung wird das Signal in Frequenzraum konvertiert. Das Sprachsignal wird zuerst durch eine 16ms breit Hammimg-Fenster geführt. Dies gibt das Signal ein glatt Zeitdurchschnitt. Die Hammingfenster ist glatt und zugespitzt, um kein Effekte der Sprachparameter zu haben wenn es periodische entlang das Sprachsignal bewegt wird. Dann, durch eine FFT (englisch:*Fast Fourier Transform*) erhält man die 256 Spektralwerte. Man verzichtet auf die darin enthaltene Phaseninformation und bildet das Leistungsspektrum, also das jeweilige Betragsquadrat der Spektralwerte. Das gibt 129Koeffizienten des Leistungsspektrum.

Kapitel 4

Filterbankanalyse

In dieser Kapiteln werden die Melscale und Auditory Filterbank methoden, als versuchen der Errkennung des Sprache in Telephonqualität zu verbessern, dargestellt.

4.1 Einführung

Der Prizip der Filterbanksprozeß, ist das *Envelope* des Sprachleistungsspektrums zu modelieren um dieser die Vokaltractstransferfunktion zu identifizieren. Da dieses ein nützliche Element, um das Spache zu idendifizieren ist, erwartet man daß dieser Modelisierung ein besser darstellung die was gesagt war geben wird. Außerdem, weil das mehr detailiert Sprachsinformation beseitigen wird, wird es ein bezeichnende Dimensionsreduktion der Signal geben. Dieser gibt ein Signal das schneller und wirkungsvollste zu Arbeiten und zu Speichern ist.

Die Schwerpunkte liegt in der Verteilung und Position des Filters. Zwei mögliche Methoden werden hier dargestellt.

4.2 Melscale Filterbank

Im Melscale Prozeß benutzt man eine Filterbank mit 16 trapezförmigen Bandpässen, um die sogennanten *melscale*-Koeffizienten (MSC, *englisch: Melscale coefficients*) aus dem Leistungsspektrum zu gewinnen. Die Anordnung der Bandpässe über die Frequenz erfolgt dabei über die logarithmisch skalierte *Melscale*. Diese ist der frequenzabhängigen Empfindlichkeit des menschlichen Gehörs nachvollzogen (siehe Abbildung 4.1). Diese Koeffizienten werden noch weiter durch ein Logarithmischeskalierung, in Amplitude transformiert. Die Gewinnung der 16 MSC ist eine Form der Dimensionre-

duktion (von 129 bis 16 in diesem Fall), die wichtigen Merkmalen in dem Sprachsignal sind auch erweitert.

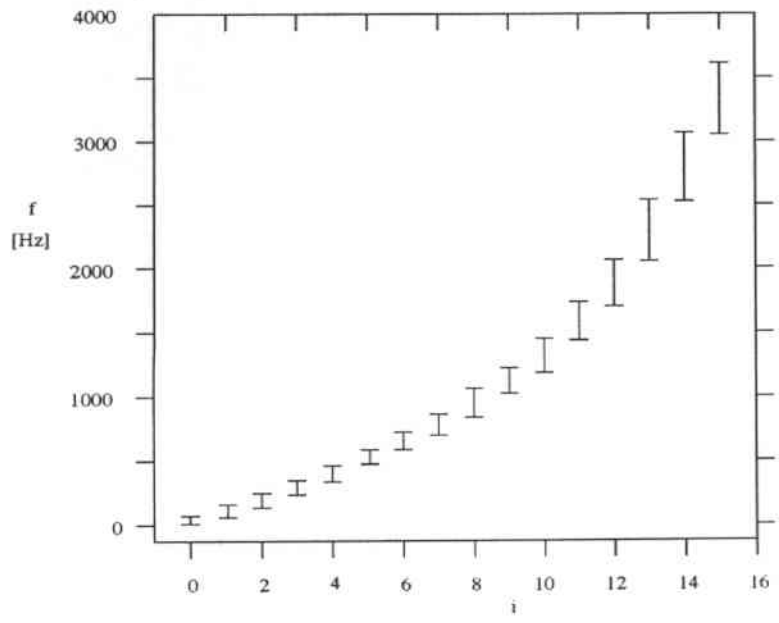

Abbildung 4.1: Verteilung des Melscale Koeffizienten

4.2.1 Realisierung

In Abbildung 4.2 sieht man diese Prozeß und die resultierenden Signalen. Das Beispiel ist die Buchstabierung von "s a b i n e"[1] die einige problematische phoneme enthält z.B. /s/ oft mit /f/ verwechselt, /b/ oft mit /d/ oder /t/ verwechselt und /i/ und /e/, die oft ausgetauscht werden.

[1] Dieser Beispiel wird durchweg dieser Diplomarbeit benutzt, die Ausgabe des jedes Vorverarbeitungsmethode zu demonstrieren.

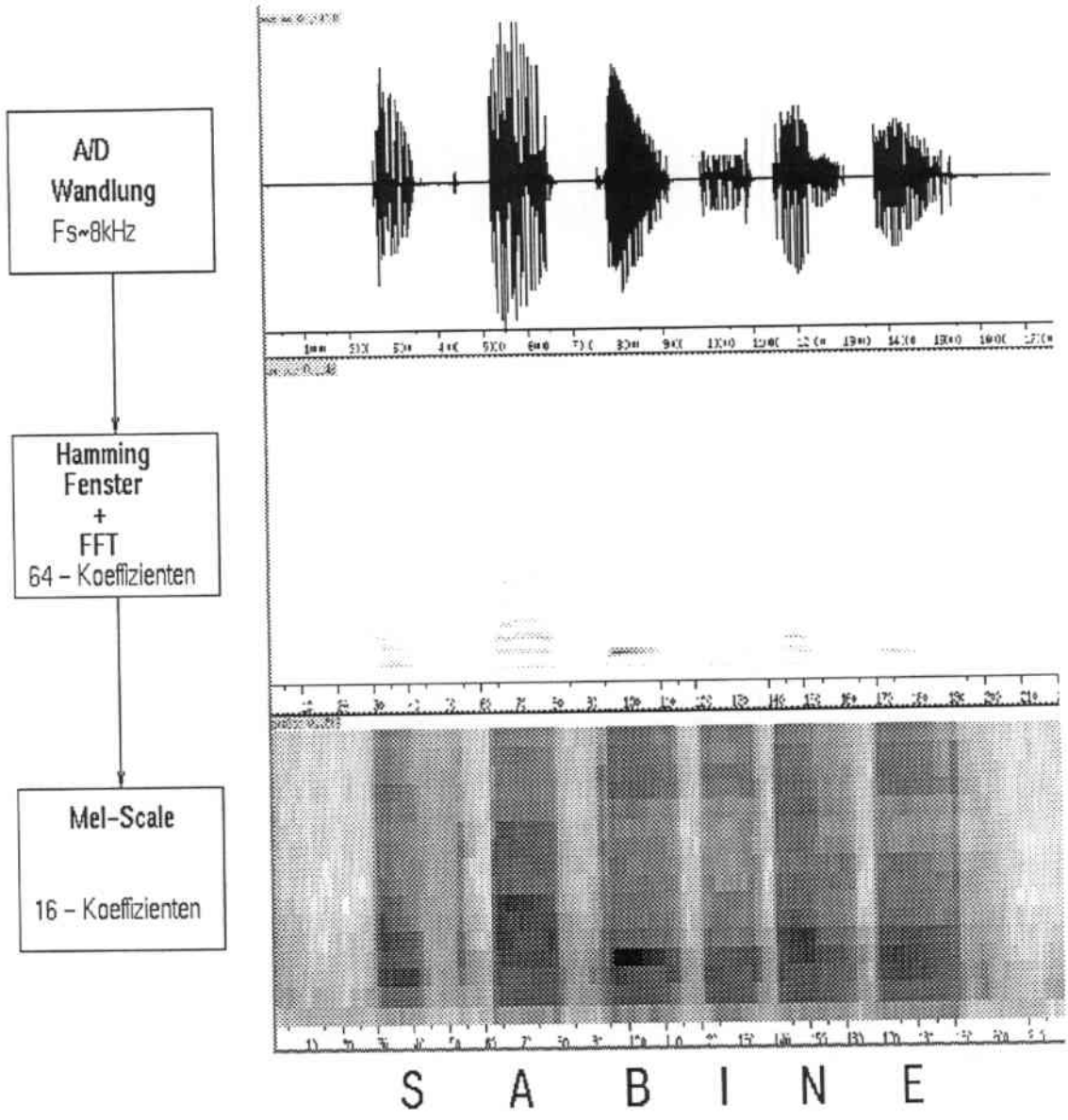

Abbildung 4.2: Melscale Vorverarbeitung

Wie für alle die Vorverarbeitungsmethoden die in dieser Diplomarbeit implimentiert werden, wird das Melscale *Feature*[2] durch den *Featuremaker* Programm erzeugt. Man findet die Kommands davon, und alle die *description files* die in dieser Diplomarbeit benutzt wurden, in Anhang A.

[2]Alle Signalen die durch ein Vorverarbeitungsmethod erzeugt werden, werden wie *features* berichtet. Der Ausdruck "Signal" bezeichnet das original Sprachsignal zum verarbeiten.

4.3 Auditory Filterbank

Das Auditory Filterbanksmethode versucht, die Eigencharakteristika des menschlichen Gehörs zu modellieren. Es ist offensichtlich, daß der Mensch das Rauschen eines Sprachsignals gut ausfiltern kann. Deshalb ist es interessant die Effekte auf dem Computer basierten Spracherkenner zu sehen, wenn hier ein Sprachsignal so verarbeitet wird, wie es im menschlichen Gehör geschehen würde.

Die Wirkungen des menschlichen Gehörsystemes sind sehr kompliziert. Zuerst muß man der Unterschied zwischen perzeptuellen Charakteristika, infolge hauptsächlich wie das Verstand die ankommend Signal legt, und Gehöhrscharakteristika, infolge die physiska Verfahren des Ohr ausgeführt an das ankommend Signal bevor es die Verstand reicht. Wenn man der Verstand wie das Erkenner denkt, wird der Ohr die Vorverarbeiter. Was jetz interessant ist, ist wenn der Vorverarbeitung des Ohr, die Erkenner das Geräuschhaltigen Sprachsignal zu erkennen, helfen wird.

4.3.1 Charakteristika des menschlichen Gehörs

Es gibt innerhald der Ohres ein Lowpassfiltereffekt von 15dB/octave für Frequenzen über 1kHz. Diese Effekte, obwohl wichtig, ist nicht so bezeichnende als die Hauptphänomen von *masking*. Diese Effekt kommt von extrem nichtlinear Schwingen die innerhalb den Ohr passieren. Sie machen die Auffassung von Lautenergie an einem Frequenz, abhängig von die Verteilung des Lautenergie an andere Frequenzen und auch abhängig von der Zeitlauf der Energie bevor und nach die Laute. Die bezeichnedete Ergebnisse dieses Effekts ist, die Gegenwart von eine Laute erhebt die Hörendschwelle einer anderer Lauter die gleichzeitig oder nach eine kurze Verzögerung gehört werden.

4.3.2 Gehörverarbeitung

Das Antwort den Ohr verschiedenes Frequenzes kann, durch eine Satz von Kritischebandcharacteristika (englisch:*critical band characteristics*) definiert werden. Jedes Kritischen-Band definiert eine Frequenzreihe. Die Auffasung des Lauten an diese Frequenzen ist ähnlich. Die Änderung eines Kritisches-Bands an ein andere präsentiert ein Änderung der Frequenzen. Die Auffpassung dieses Frequenzes ändert plötzlich. Man präsentiert deshalb, ein Kritische-Band wie eine Bandpassfilter deren Antwort korrespondiert ungefähr mit die Abstimmungskurven des Gehörsneuronalen. Die Kritische-Band Spektrum wird durch die sogennante Bark-Scale definiert (siehe Abbildung 4.3). Hier ein Bark präsentiert ein Kritischen-Bandbreit. Die Transformationsfunktionen sind deshalb:

$$T_\Omega(\omega) = 6 \log \left(\frac{\omega}{1200\pi} + \sqrt{1 + (\frac{\omega}{1200\pi})^2} \right) \tag{4.1}$$

where $\omega = 2\pi f$.

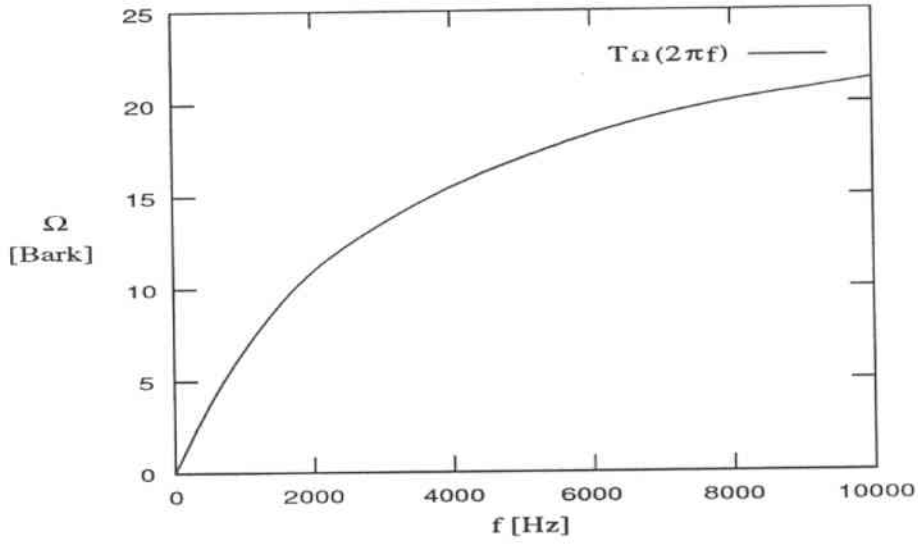

Figure 4.3: Bark-Scale Transformationsfunction

Für die *auditory* Filterbank, werden die Filters nach dem Bark-Scale verteilt (siehe Abbildung 4.4), um die perzeptuellen Effekte des menschlichen Gehörs zu aufnehmen, . Man sieht der Form des Filters in Abbildung 4.5. Das Ergebnis der Faltung des Spektrum mit diesem Filter ist das auditive Spektrum des Sprachsignals:

$$\psi(\Omega) = \begin{cases} 10^{2.5(\Omega+0.5)} & -1.3 \leq \Omega \leq -0.5 \\ 1 & -0.5 \leq \Omega \leq 0.5 \\ 10^{-1.0(\Omega-0.5)} & 0.5 \leq \Omega \leq 2.5 \\ 0 & \text{otherwise} \end{cases}$$

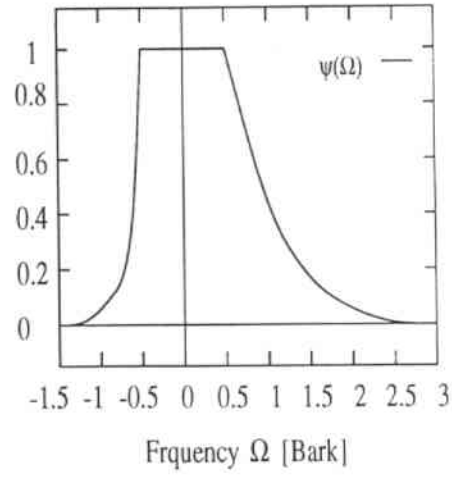

Figure 4.4: Kritische-Band Funktion der Bark-Scale

$$\Theta(\Omega) = \int_{-1.3}^{2.5} P(\tilde{\Omega} - \Omega)\Psi(\tilde{\Omega})d\tilde{\Omega} \tag{4.2}$$

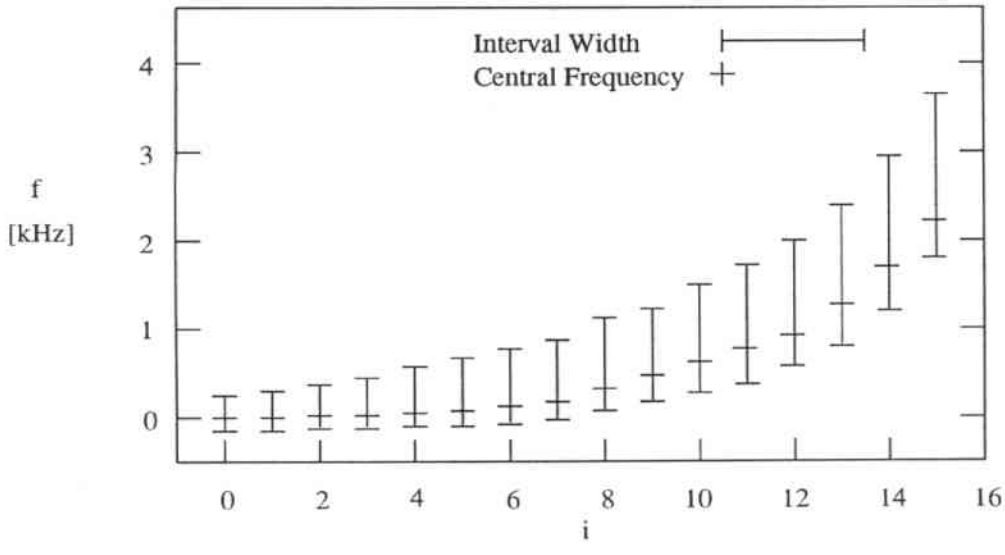

Figure 4.5: Verteilung der Frequenz nach den Bark-Scale

Man kann eine weitere Annäherung an die Wirkung des menschlichen Gehörs durchführen. Das Bark-Scale transformierte Signal wird mit der Equal-Loudness-Kurve multipliziert (siehe Abbildung 4.6). Diese Kurven stellen die Allgemeinempfindlichkeit des Ohr bei verschiedenen Frequenzen dar (die vorgesagt Filtereffekt).

Schließlich werden die Koeffizienten durch die Empfindlichkeit des menschlichen Gehörs gegenüber Tönen unterschiedlicher Lautstärke skaliert. Diese Tranformation wird durch das 3 Wurzel des Signal gemacht.

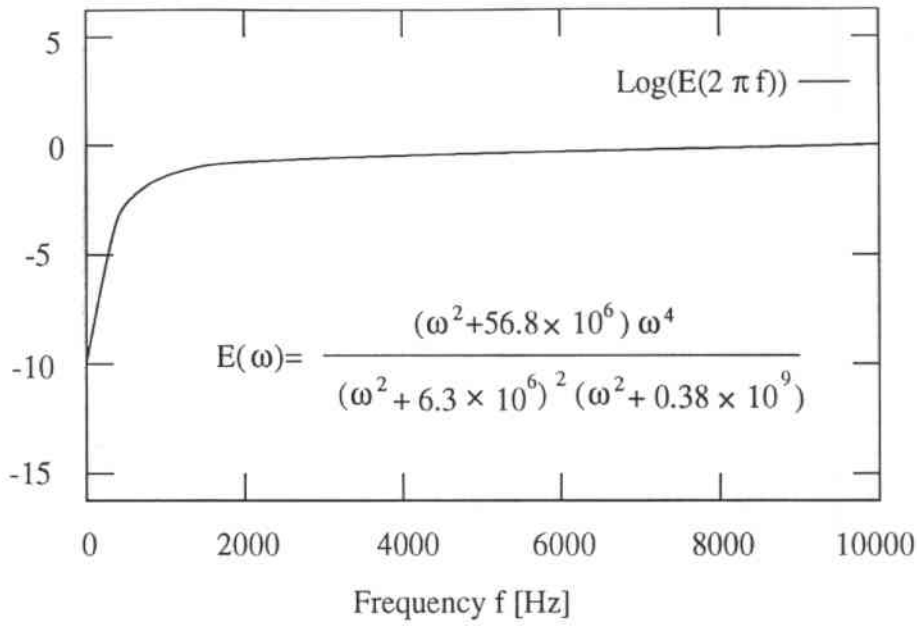

Figure 4.6: Equal-Loudness Kurve

4.3.3 Realisierung

Man sieht in Abbildung 4.7 die Ausgabe eines Auditory Filterbanks. Dieses wird durch einem Bank von 16 Filters realisiert. Das Auditivesspektrum wird dann *mean subtracted*[3] wie in Kapiteln 5 eklärt wird.

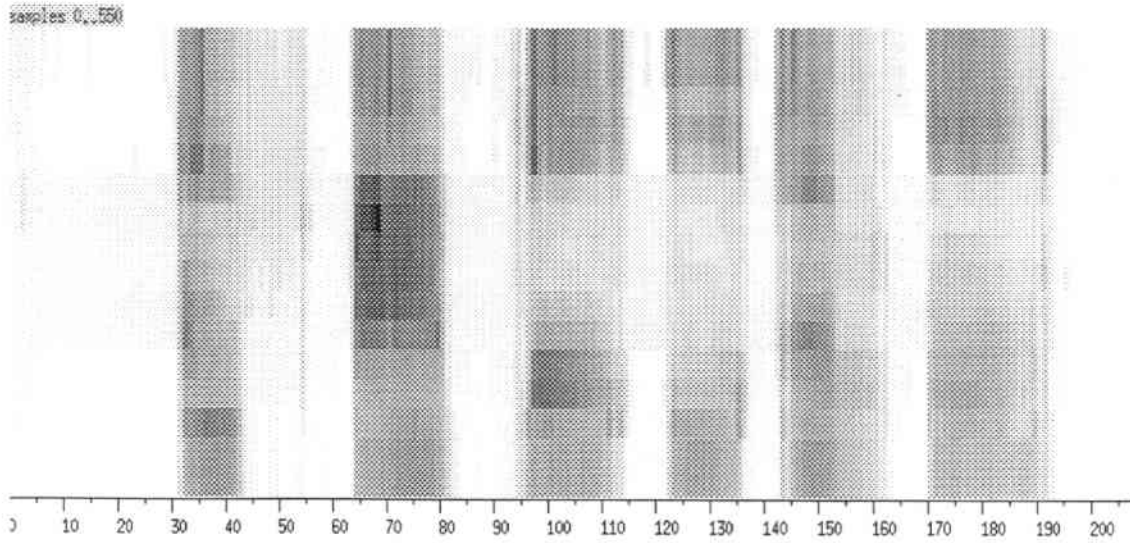

Abbildung 4.7: Die Ausgabe des Auditory Filterbank

[3]Alle die *features* die hier gezeigt werden, werden *mean subtracted* und den Standard Abweichung normalisiert, wie in Kapiteln 5 eklärt wird

Kapitel 5

Rauschunterdrückung

Dieser kapitel führt drei Methoden der Rauschen zu unterdrücken ein: Spectral Subtraction, das der additive Rauschen zu reduzieren versucht, Mean Subtraction das der Faltungsrauschen zu reduzieren versucht und RASTA das die beidene Störunge zu reduzieren versucht. Zuerst ist Die grunde jedes Methodes gegeben und dann werden das Theorie und Realisierung im detail eklärt.

5.1 Spectral Subtraction

Spectral Subtraction ist vielleicht der offensichtlichsten Lösung des Problemes des additive Rauschens in einem Sprachsignal. Es versucht alles additive Rauschen zu eliminieren durch Subtraction einer Schätzung des Rauschens, während der Sprachpausen, vom ganzen Signal. Dieser Schätzung wird im Zeitraum gemacht, und der Abzug findet im Frequenzraum statt. Die Abzug wird auf dem Leistungsspektrum durchgeführt, und mann nimmt an daß das ausreichend ist, um das Rauschen zu reduzieren [3]. Dieser Methode könnte als ein Vorverarbeitung an irgend andere Kodungssysteme, Fehlerkorrektungssysteme, Erkenner etc. implimentiert werden und so hat es der Vorteil daß man nicht die ganze system zu ändern braucht, wenn er dieser Methode einbeziehen möchtet. In der Tat hat mann gefunden daß dieser method besser ist wenn er es als Vorverarbetung eines anderes Verarbeitungsmethodes benutzt [3].

Man nimmt an, daß die durchschnittliche Signalamplitudes während der Sprachpausen eine gute Annäherung des additives Rauschens ist. Deshalb muß man, um die *Spectral Subtraction* zu realisieren, das Duchschnitt aller Signal Vektoren der Stilleperioder vom gezampten Signal abziehen. Es gibt, deshalb, zwei Probleme:

1) Wie erkennt man die Sprachpausen des Signals.

Stille kann durch die Leistung des Signals erkannt werden. Die Leistung jedes *Frame* wird berechnet und dann wird aufgrund einer bestimmten Schwelle entscheiden ob es Sprache gibt oder nicht. Wenn die Leistung niedriger als die Schwelle ist, wird angenommen daß Stille herrscht.

Bevor die Schwelle, wird die Leistungssignal durch ein Logarithmischestransformation Skaliert. Dieser besser trennt die niedrige Lesitungsperioden von den Sprache. Das tansformiert Funktion wird dann durch ein Lowpassfilter geführt, um die kurze Hochleistungsperioden (die wahrscheinlich Rauschenstifte sind) zu beseitigen. Nach dem Schwelle wird die "Stille" Signal discretisiert, damit es nur die Werte 0 die die Sprachperioden präsentiert oder 1 die, die Stilleperioden präsentiert zu habt (siehe Abbildung 5.1). Das *Mean* Werte des allen *Frames*, die Stilleperioden darstellen, werden jetz von die ganze Signal abgezogen.

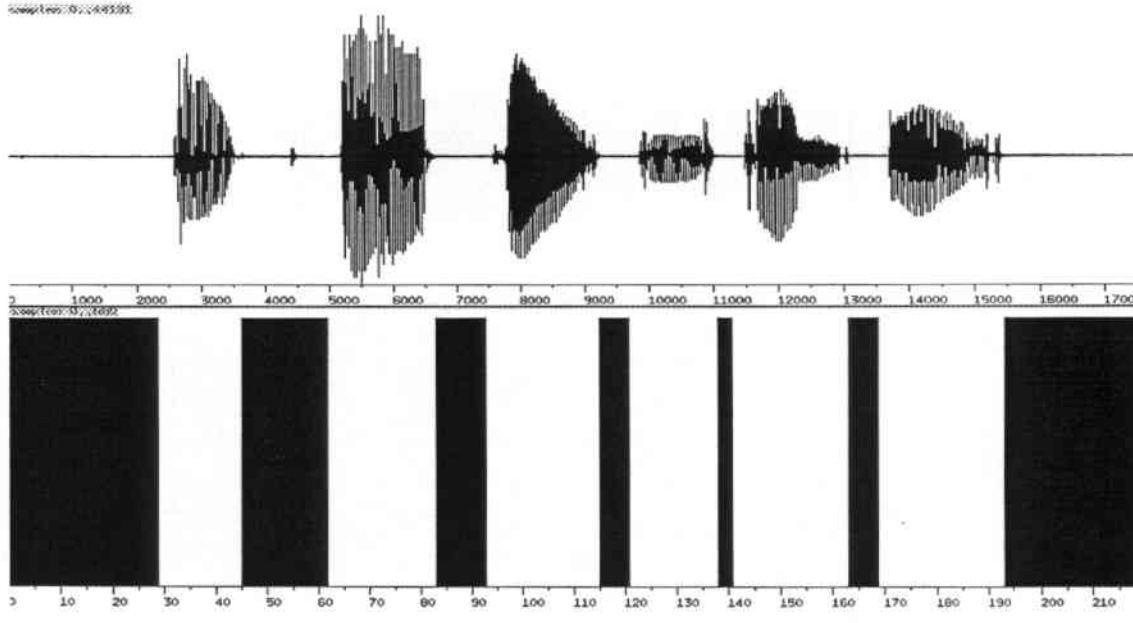

Figure 5.1: Sprachpausendetektion

2) Wie die negative Ergebnisse den Abzug zu Verarbeiten.

Da die *Spectral Subtraction* als ein *front-end* der anderen Vorverarbeitungsmethoden benutzt wird, und da der Großteil der Methoden einen logarithmischen Prozeß erfordert, ist es wichtig, daß das Signal, das von der *Spectral Subtraction* kommt keine negativen Werte besitzt.

Die offensichtliche Lösung, ist alle die negativ Werte weg von den Signal zu machen. Aber wahrscheinlich wird es viel Nulldurchgänge, in die abgezieht Spektrum geben geben. Wenn man dieser einfach raus schneidet, würde das Spektrum viel Diskontinuitäte haben. Jedes weitere verarbeirtung dieses Spektrums würde kein gut Ergebnisse geben. Ein besser Alternativ ist die

negativ Werte im Spektrum zu lassen, so lang wie möglich, bis zum logarithmische Prozeß. Mit dem Dimensionsreduktion des Signal, durch die Filterbanks und die Kodierend Methoden, wird es weninge Nulldurchgänge geben. Wenn jetz das Signal geschneidet wird, wird es viel wenige Diskontinuitäte geben.

5.1.1 Realisierung

Da *Spectral Subtraction* wird auf dem Melscale Prozeß getestet. Um die negativ Werte so lange wie möglich zu beibehalten, werden die 16 Melscale Koeffizienten aus dem *Silence Subtracted* Spektrum berechnet. Dann werden alle negativen Werten ausgeschnitten bevor die logarithmisches Transfomation durchgeführt wird. In Abbildung 5.2 sieht man das Melscale das von ein Spektrum mit abgezeigt Stille genommen wird.

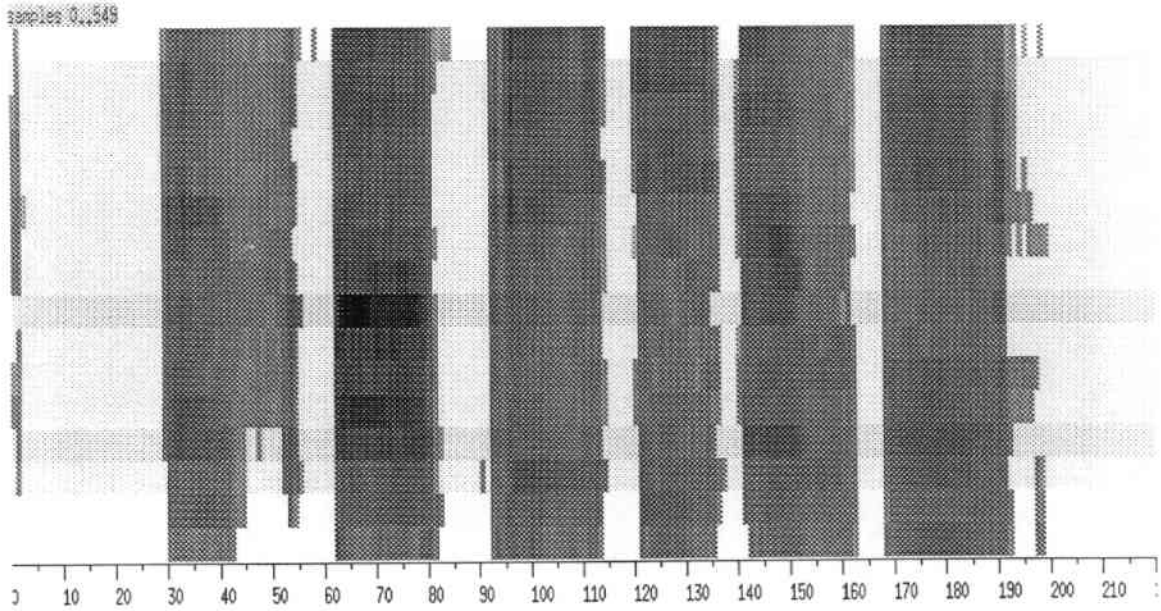

Abbildung 5.2: Ausgabe der Melscale Prozeß mit Spectral Subtraction front-end

5.2 Mean Subtraction

Nach der Berechnung des *Feature* Koeffizienten braucht man einige Normalization, um kein groß Eingabesignalbereich für den Erkenner zu haben. Ein Anfangsnormalization ist, die Koeffizienten innerhalb ein festgelegt Bereich zu normalisieren (von 0 bis 1 wird in dieser Diplomarbeit getestet).

Ein andere Method ist die Standard Abweichung des *Feature* zu normalisieren um die Signalen mit Verschiedene dynamische Angebot, und Amplitudesverteilungen zu in Betracht ziehen. Noch weiter, gibt es die möglichkeit ein möglicherweiser Störungreduzierendmethode innerhalb die Normalization zu aufnehmen:

Mann nimmt die Geraüschhaltigen Sprachsignal $y(t)$ die in Ausgleich 2.1 gegeben wird, und er berechnet die Melscale Koeffizienten wie gerade eklärt würde, die das folgende gibt:

Nach den A/D Wandlung die digital Signal ist:

$$Y(\omega, k) = [S(\omega, k) + N(\omega, k)] \times H(\omega, k) \tag{5.1}$$

wo k die Abtasten präsentiert. Dann durch ein FFT, berechnet man die Leistungsspektrum:

$$Y(\omega, k) = [S(\omega, k) + N(\omega, k)] \times H(\omega, k) \tag{5.2}$$

Angenommen daß der Prozeß logarithmisch ist, sollte das Endsignal ein Form wie die Folgende haben (es wird nicht so einfach sein weil gibt es auch der Dimensionsreduktion):

$$\log(Y_{\omega,k}) = \log(S_{\omega,k} + N_{\omega,k}) + \log(H_{\omega,k}) \tag{5.3}$$

Jedes Sprach Vorverarbeitungsmethode, die etwas ähnlich wie eine logarithmische Skalierung anwendet, macht die Faltungsstörung additiv. Obwohl, diese Störung immer zufällig ist, kann man zu Begginenversuchen sie zu reduzieren, der durchschnittliche *Feature* Vektor vom gezamten *Feature* abziehen.

Die Transferfunktion des Kanals bleibt konstant über dem ganzen Sprachsignal, und deshalb bleibt auch $H(\omega, k)$ konstant. Das heißt der Durchschnitt des Faltungsrauschens, in diesem logarithmische Spektralraum, ist gleich als $H(\omega, k)$. Wenn man das *mean* signal, im logarithmische Spektrum abzieht, wird der Werte des Faltungsrauschen plus das Durchschnitt des Rests des Signals unterdrückt. Man erwartet daß dies ein allumfassend Reduktion des Faltungsrauschens ergibt.

5.2.1 Realisierung

Ein *mean subtraction* Methode wird an alle die Vorverarbeitungsprozeßen getestet. In dieser Methode, zieht man die *mean feature* Vektor von den *Feature* ab. Zweimal seine Standard Abweichung wird dann an eins normalisiert. Das gibt ein Signal mit null *mean* und ein Standard Abweichung gleich als 0.5.

5.3 RASTA

Wie frühe eklärt wurde, ist den Faltungstörung ein sehr groß problem wenn man ein Signal durch ein Kommunications Kanal, so wie den Telephon, lauft.

Die Darstellung des Sprachsignals ist die sprachlich Meldung die als Veränderung des Vokaltract kodiert wird. Der Spachsignal reflektiert deshald diese Veränderung. Rasta basiert darauf, daß die Veränderungsrate der Komponenten jeder nicht-sprachlichen Information, außerhalb (sich schneller oder langsamer ändernd) des Vokaltracts liegt[1].

5.3.1 Realisierung

Der RASTA process erfodert dieser Prozeßen:

1)Rechnen die Kritikal Band Spektrum - wie für der auditory Filterbank.

2)Transformieren den Spektralamplituden durch eine komprimierende statische nicht lineare Funktion. Diese versucht alle die umgebungsmäßig Störung an einem zusätzliche Komponent zu konvertieren [13].

3)Filtern die Zeitflug von jedes transformiert Spektralkomponent.

4)Transformieren dieser gefiltert Sprachereprăsentation durch eine erweiternd nicht linear Transformation.

Die Schritten zwei, drei und vier sind dann was eigentlich RASTA ist. Schritt vier ist dei invertierte von Schritt drei, so interessiert man sich mit Schritten zwei und drei.

Die Method die heir realisiert wird, würde von [1] genommen. Die gründsatzlich Prizip sind gegeben, aber man sollte [1] sehen, um mehr Detail zu haben.

Die Filter die benutzt wird ist ein IIR (englisch:*Infinite Impulse Response*) mit einem 160ms Zeitkonstant, und einem 0.26Hz niedergrenzfrequenz und einem Gefälle von 6db/oct nach 12.8Hz. Es gibt *sharp zeros* an 28.9Hz und 50Hz (siehe Abbildung 5.3).

$$H(z) = 0.1z^4 \times \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.94z^{-1}} \tag{5.4}$$

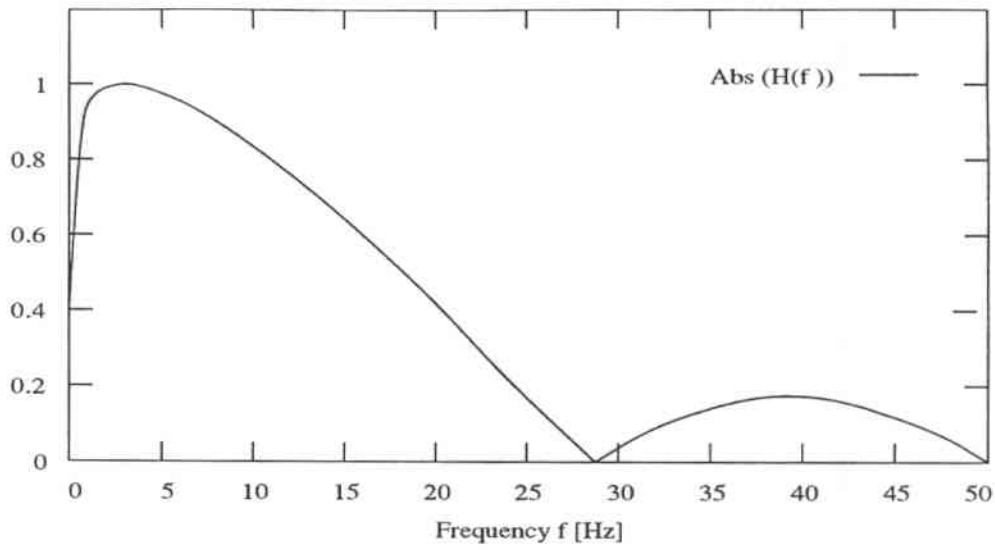

Figure 5.3: Das RASTA Filter

Die Transformation für Schritt zwei ist entwickelt geworden um das Faltungsrauschen und das additive Rauschen wie additive Rauschen im RASTA Prozeß additiv zu lassen. Beide Rauschen können danach unterdrückt werden. Obwohl eine logarithmische Transformation ideal für des Faltungsrauschen ist, machtsie die additive Rauschen Komponenten abhängig vom Signal. Man benutzt deshalb das Funktion gegeben in [1][13]:

$$Y_i = \log(1 + JX_i) \tag{5.5}$$

Diese Funktion ist ungefähr linear für kleine Werte von J (für additive Rauschen), und ungefähr logarithmisch für große Werte von J (für Faltungsrauschen). Es kann so für verschiedene Werte J getestet werden. In Abbildung 5.4 sieht man die Funktion für mehrere Werte von J.

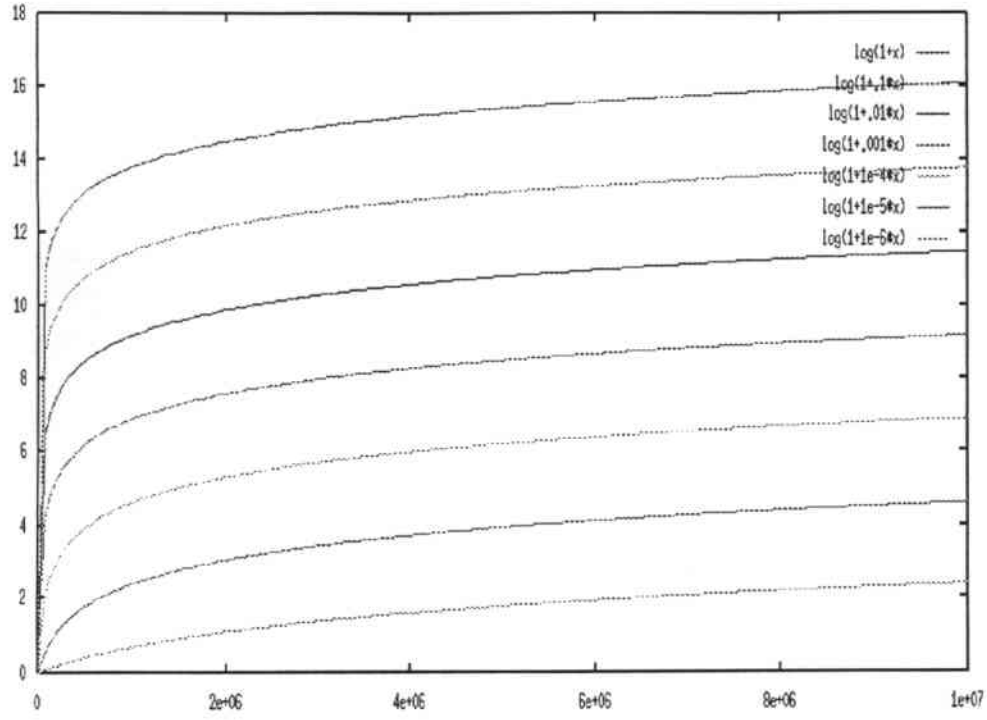

Abbildung 5.4: Die Transformfunktion des RASTA Prozeß

In Abbildung 5.5 sieht die Ausgabe den RASTA Prozeß, mit J gleich als 2×10^{-6}, und 16 Koeffizienten.

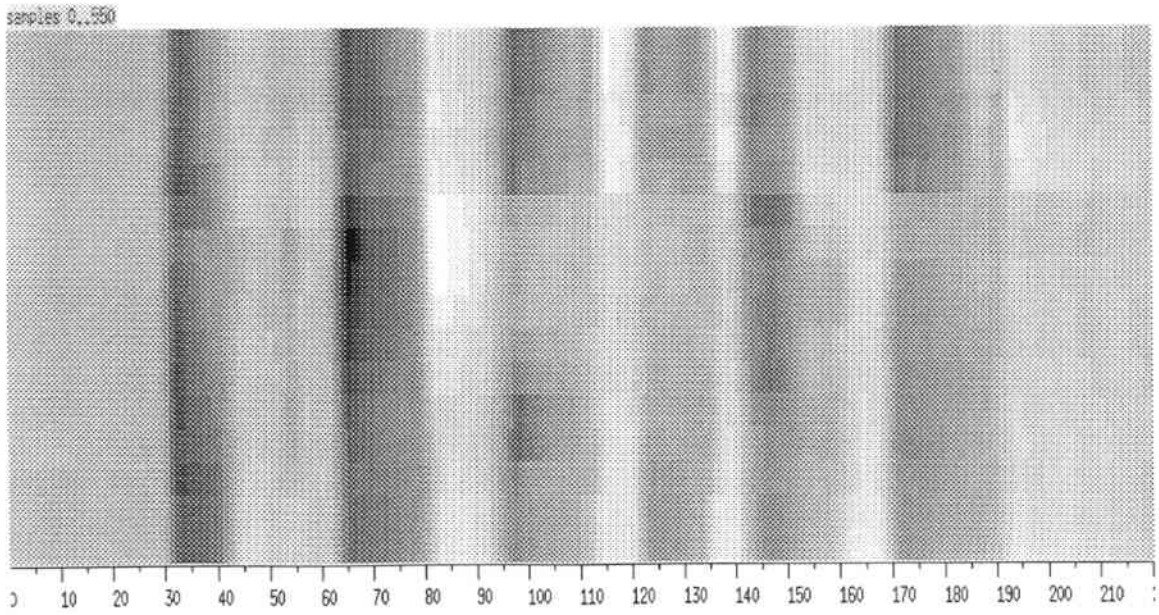

Abbildung 5.5: Die Ausgabe des RASTA Prozeß

Kapitel 6

Peceptuell Linear Prediction (PLP)

Dieser Kapiteln wird die Prizip von PLP eklärt. Nach der Gründsätzlich Theorie von LPC-Analyse (englisch:Linear Predictive Coding) fidet man die Detailen des spezifische realisierung durch Featuremaker.

6.1 LPC-Analyse

Linear Predictive Codeing (LPC) ist einen Sprachcodierend methode daß im Rechenbedarf schnell und genau ist. Es beutet die Tatsache aus daß man eine Sprachmuster mit die lineare Kombination von die letze p Sprachmuster nähern kann [7]. Durch die Minimisation von den Unterschied zwischen die eckten Sprachmuster und die vorhergesagten vorraussichtlich Muster, kann man die Prediktor Koefficienten beenden. Diese Koefficienten (oder Gewichten) sind fogesetzt angepaßte, die zeitverändernd Eigentrum von der Sprachsignal zu anpaßsen.

Die Transferfunktion der lineare Systeme (im kurze zeit) ist [20]:

$$H(z) = \frac{A}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{6.1}$$

Man definiert die Anregung (ob stimmhafte oder stimmlose Sprache) wie $\delta(n)$. Die Srachmuster $x(n)$ sind an der Anregung berichtet durch:

$$x(n) = \sum_{k=1}^{p} a_k x(n-k) + \delta(n) \tag{6.2}$$

Wo a_k sind die coefficienten die der Vokaltrakttransferfunktion representieren. Wenn Man die Sprache mit einem Linearepredikтor verarbeitet, ist die vorhergesagt Signal:

$$\tilde{x}(n) = \sum_{k=1}^{p} \alpha_k x(n-k) \tag{6.3}$$

Wo α_k sind die Koeffizienten.

Die Prediktorfehler ist dann:

$$e(n) = x(n) - \tilde{x}(n) = x(n) - \sum_{k=1}^{p} \alpha_k x(n-k) \tag{6.4}$$

Wenn $\alpha_k = a_k$ wird die Prediktorfehler gleich als $\delta(n)$ sein. Diese Fehler ist sehr klein zwischen die Anregungimpules von stimmhafte Sprache. Davon ist die Prediktorpolynom:

$$P(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k} \tag{6.5}$$

ein gute Annäherung an der Vokaltrakttransferfunktion.

Die Prediktorkoefizienten α_k werden durch die Minimization von den mittlerer quadratischer Prediktorfehler für ein klein Sprachsegment berechnet:

$$MIN[E_l] = MIN\left[\sum_{n=0}^{N-1}\left(x_l(n) - \sum_{k=1}^{p} \alpha_k x_l(n-k)\right)^2\right] \tag{6.6}$$

Wo $x_l(n)$ ist die Sprachsegment die für den Berechnung ausgewählte war.

$$x_l(n) = x(n+l) \tag{6.7}$$

Man findet die Werten für α_k durch $\partial E/\partial\alpha_i = 0, i = 1, 2, .., p$ das gibt:

$$\sum_{k=1}^{p} \alpha_k \varphi_l(i,k) = \varphi_l(i,0) \tag{6.8}$$

Wo

$$\varphi_l(i,k) = \sum_{n=1}^{N-1} x_l(n-i)x_l(n-k) \tag{6.9}$$

Diese ist ein Satz von p Ausgleichen mit p Unbekannten (p ist die Ordnung von den Prediktor), das um die Prediktor Koeffizienten zu finden, gelöst werden können. Dieser Koeffizienten miniminieren die mittlerer quadratische Fehler.

6.2 PLP Realisierung

In PLP - einem Allpol (LPC) model wird benutzt, um eine kurze Zeit Leistungsspektrum zu Kodieren. Dieser Spektrum ist zuerst durch dem Auditory Filterbank, die im letzer Kapitel eklärt wurde, geführt.

Dieser method hat zwei vorteil; diese Glaube von den Pezeptuell gewichtete Sprachsignal braucht weniger koeffizienten und so gibt es noch ein Dimensionreduktion. Das erkennung konnte auch schnelle sein. Naturlich braucht die Pradiktor auch zeit. Seit ein nth ordnung LPC-Analyse kann nur ein Spektrum mit am vielstens $n/2$ Spitzen genau representieren, wird ein niedrige ordnung Analyse das spektrum gleiten. Das konnte dann das additive Rauschen vermindern, weil ist es normaleweiser an hohefrequenz.

So was ziemlich wichtig ist, ist die ordnung von den LPC-Model. Die optimale ordnung ist veileicht besser mit test bekommt, weil gibt es zur zeit verschiedene meinung [2]. Das konntest abhängig von den Erkenner sein.

In Abbildung 5.1 sieht man die Ausgabe des PLP Prozeß. Diese ist ein 8 ordnung Mean Variance Normalisiert PLP Modele.

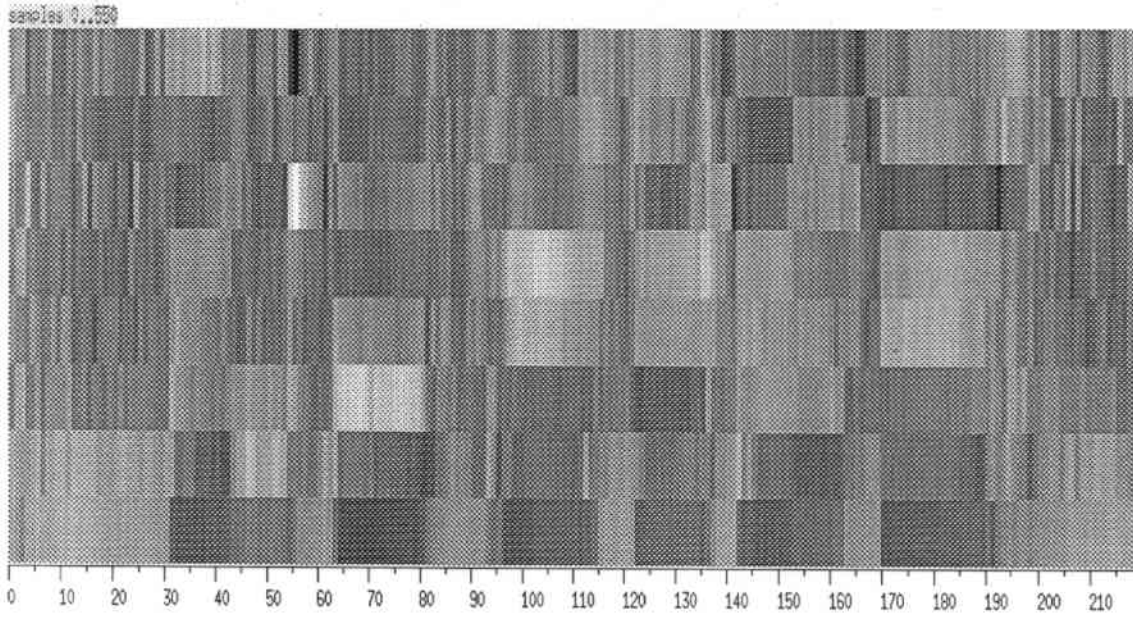

Abbildung 4.1: Die Ausgabe des PLP Prozeß

6.3 RASTA-PLP

Wahrend benutzen die zwei Prozeß PLP und RASTA, verschiedene Methode, um die Erkenungsfehlers infolge Rauschen zu reduzieren, ist es interessant zu

sehen, die Ergebnisse, wenn man die beiden kombiniert und so vielleicht kombiniert er seinem Attributen. Das erfordert nur ein PLP Modelle des RASTA *Feature*.

In Abbildung 6.2 sieht man die Ausgabe den RASTA-PLP Prozeß. Es wird mit einem 8 ordnung PLP Model von ein 16 Koeffizienten RASTA Prozeß erzeugt.

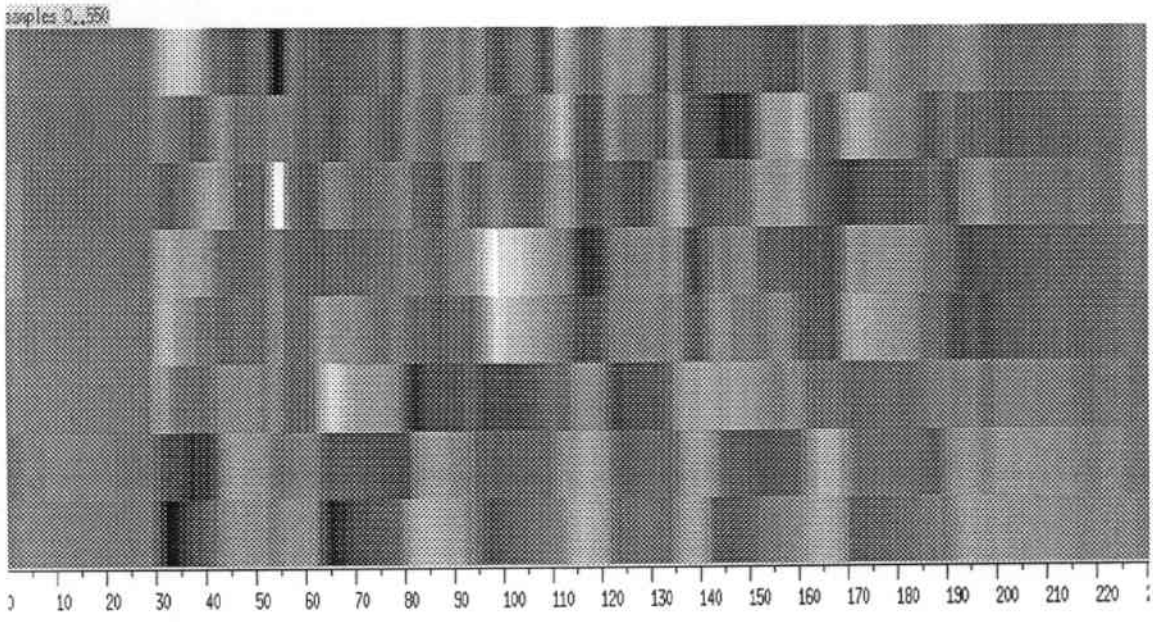

Abbildung 6.2: Ausgabe der RASTA-PLP Process

Kapitel 7

Die Lineare Diskriminanzanalyse (LDA)

In dieser Kapiteln hat man ein Überblick von der LDA wie ein method von Muster Klassifikation. Zuerst kommpt die gründsätzliche Prizip von Muster Klassifikation und ein paar wichtigen definitionen. Dann wird die LDA method selbe und seine Implimentation wie ein BACK END für die anderen vorverarbeitung methoden, in die zwei Erkenners vorgestellt.

7.1 Musterklassifikation

In jedes Muster Klassifikationmethode, versucht man alle die wichtigsten Charakteristichen den jedes Musterklass zu identifizieren. Dies erlaubt viel nutzlos Information abgelegt zu werden und so kommt ein Dimensionsreduktion von den signal zu verarbeiten und auch ein verbesserung des erkennung.

Die einfachste Methode von Dimensionsreduktion ist ein Linearestransformation durch den Multiplikation der signal mit irgendein Transformationmatrix. Mit dem eingabevektor x mit n Elemente $y = A\dot{x}$, wobei A eine m $\times$ n Matrix (m $<$ n) ist, gibt ein ausgabvektor y mit wenige Elemente (m) als x (n).

Die folgeneden Matrizen sind jetz definiert um besser die Klassen zu representieren und so die optimal Transformmatrizen zu finden:

Total covariance matrix (Kovarianzmatrix):

$$T = E\{(X - M_0)(X - M_0)^T\} \tag{7.1}$$

Diese definiert die Gesmatverteilung des alle die Mustern .

Within-class covariance matrix (mittlere Klassenkovarianz, Klasse ω_i):

$$W = \sum_i P(\omega_i)\dot{E}\{(X - M_0)(X - M_0) \mid \omega_i\} \tag{7.2}$$

Diese definiert die Verteilung des Mustren innerhald ein Klasse.

Between-class covariance matrix (Kovarianz der Mittelwerte):

$$B = \sum_i P(\omega_i)\dot{(}M_i - M_0)(M_i - M_0)^T \tag{7.3}$$

Diese definiert die Verteilung des Klassen selbe (Verteilung des Mittelwerte).

Dabei sind:

- X Zufallsvariable (Datenvektoren oder Merkmalsvektoren)
- M_0 Gesamtmittelwert für alle die Verteilungen
- M_i Mittelwert der Klasse ω_i

7.2 LDA

Die Lineare Diskriminanzanalyse versucht die Varianz der Koeffizienten der verschiedene Eingabeklasse zu maximisieren, während sichert es daß die Vektoren die der gleich Klasse representieren beieinander bleiben. Dies erreicht man dadurch, daßman die Varianz innerhalb einer Klasse minimiert. Beide Ziele vereinigt man in dem folgenden Gütekriterium:

$$J_1(m) = \frac{det(T_m)}{det(W_m)} = det(W_m^{-1}\dot{T}_m) \tag{7.4}$$

7.2.1 Realisierung

Um die optimal LDA-Matrix A zu berechnen, muß man zuerst die andere Kovarianz Matizen erlangen. Das ist durch der Janus Spracherkenner gemacht. Für die Tests in dieser Diplomarbeit, sind die Datenvektoren in Subphonemen Klassen Zuordnet. Jedes Phonem ist duch drei Subphonemen definiert wie war fr"he eklärt. Das heißt die *within-class* Kovarianz Matrix definiert die Verteilung der Muster die der gleiches Subphoneme representieren, und die *between-class* Matrix representiert die Verteilung des alle mögliche Subphoneme in den Musterraum. Mit den Trainingssätzen und der Klassendatei kann nun die Transformationsmatrix bestimmt werden. Dies geschiet in drei Schritten:

- Die relative Häufigkeit der Klassen und ihre Mittelwerte werden bestimmt.
- Mit hilfe der Mittelwerte werden die Kovarianzmatrizen ermittelt.

- Mit der zwei Matritzen (T und W) wird eine simultane Diagonalisierung [19] durchgeführt, deren Ergebnis schließlich die Transformationsmatrix A ist.

7.3 Delta-Koeffizienten

Die Verbindung des Delta-Koeffizienten, in einer Sprachverarbeitungsmethode ist ein möglicherweise Methode, die Erkennungsgenauigkeit zu verbessern. Tatsächlich für große Vokabular Spracherkennung sind dynamische Delta-Koeffizienten oft unbedingt, die gute Ergebnisse zu erreichen [16]. In den Delta-Koeffizientsprozeß, ziegt man von ein vorwärts verschiebt Version eines *Features* ein rückwärts verschiebt (vercheibt der gleiches Länge) Version des gleiches *Features*. Die delta-Koeffizienten präsentiert deshalb, der Unterschied zwischen die zwei Signalen. Für ein zwei Zeitfenster *frame* Verschiebung, ist die Deltafunktion:

$$\delta(2) = [X(t+2) - X(t-2)] \tag{7.5}$$

Man hängt die berechnete Deltasignalen (mehrere deltas, mit verschiedene Verschiebungen, könnten benutzt werden) auf das *Feature* das er verarbeitet an.

Man nimmt ein Länge des Verschiebung, viel kleine als ein Häftewortelänge. Das heißt die Abzug wird nich von verschiedene Worte gemacht. Das Delta wird ein relativ groß positiv Werte an die Ende eines Wortegrenz haben (der Anfang eines Ruheperiodes der erweiterte *Feature* wird von das Ende eines Lautperiodes der verzögerte *Feature* abgezieht), es wird ein relativ groß negativ Werte am Anfang eines Wortgrenz haben (der Anfang eines Lautesperiod der erweiterte *Feature* wird von das Ende eines Ruhesperiod der verzögerte *Feature* abgezieht) und er wird klein Werte anderswo haben (die ähnliche Koeffizienten der *Feature* - Ruhe und Laut - wird von einander abgezieht).

Die adschlieënd verbunden *Feature*, wenn die Verschiebungen gut positioniert wurden, wird das originales *Feature* Merkmalen haben, die Worte zu modelieren, und es wird Deltas haben, die Übergänge zwischen Worte zu modelisieren. Dieser Methode liefert deshalb, wichtig Extrainformation. Er erzeugt mehrere Merkmalen die der Erkenner benutzen kann, die Worte zu identifizieren und differenzieren. Es ist auf dieser Gründ daß dieser Methode nütlich mit dem groß Vokabular ist, weil werden die Anzahl des —'ahnliche Worte steigen während steigen die nunner der verschidene Worte innerhalb den Vokabular.

Der Nachteil dieses Methode ist der Wachstum des Koeffizienten die ein *Feature* modelisieren. Für jedes Delta benutzt, erffordert man die gleiche Anzahl des Koeffizienten die in das original *Feature* waren, mehr.

Man versucht, die Muster Klassifikation zu verbessern durch die Delta-Koeffizien. Zwei Paar Delta-Koeffizienten werden, mit Verschiebungen von 1 *frames* und 3 *frames*, berechnet. Die zwei Delta Funktionen kommen zusammenmit dem *Feature* (siehe Abbildung 7.1). Das LDA matrix kann jetz berechnet werden und das *Feature* damit transfomiert wird. Wie ein Spracherkenner selbs, nützt der LDA Prozeß des Delta Koeffizienten weil hat es mehr verschiedene Merkmale die Muster zu Klassifizieren.

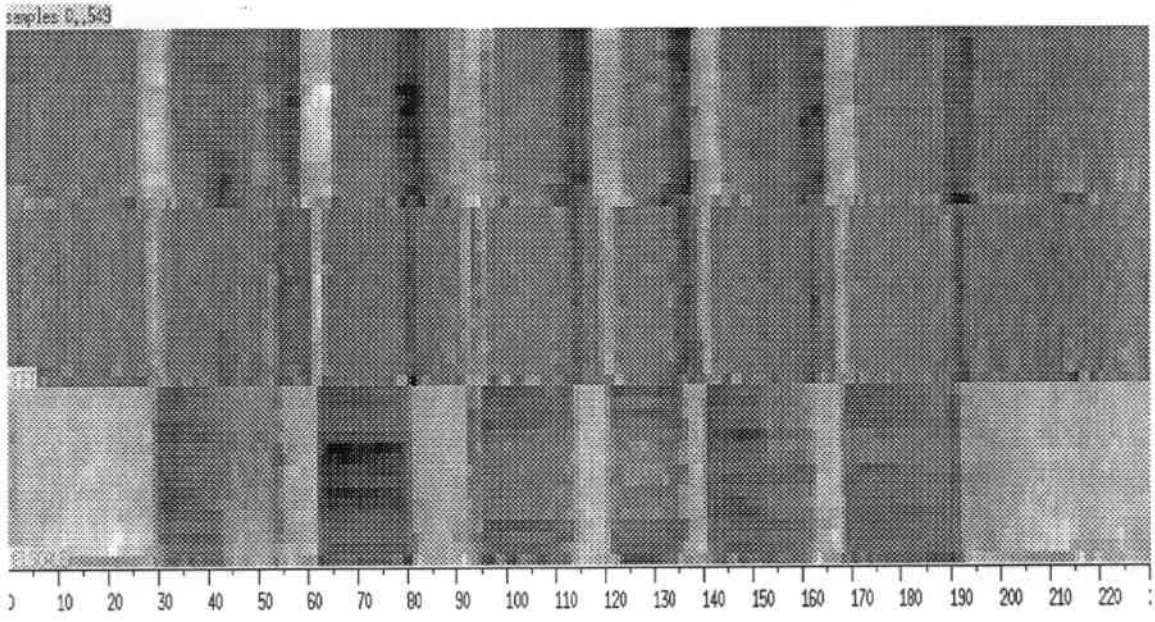

Figure 7.1: Melscale plus Deltas for LDA Processing

Obwohl es viel mehr Koeffizienten gibt (infolge des Deltas), braucht man nach den LDA Transformation nicht alle zu benutzen. Diese begründet auf den Sortierung des Koeffizienten zufolge seinem Klassdiscriminanzfähigkeit. In Abbildung 7.2 sieht man die Ausgabe der LDA Prozeß wie ein *back-end* an der Melscale Verarbeitung. Obwohl, es 48 Koeffizienten gibt, sieht man hier nur 16. Die optimale Anzahl des Koeffizienten wird erforschen werden.

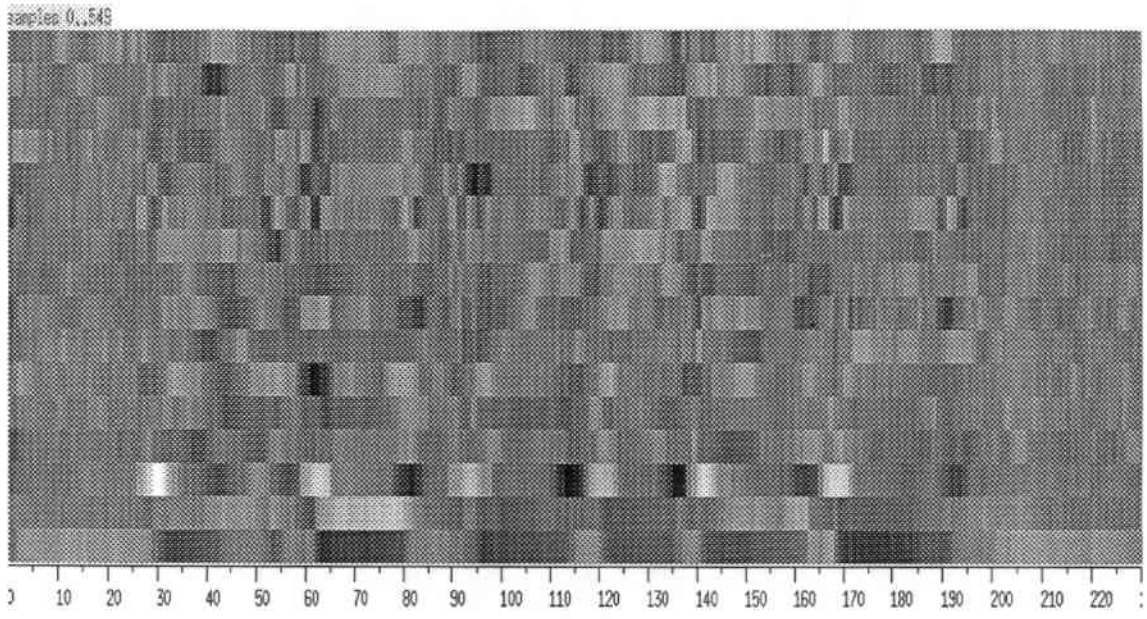

Figure 7.2: Ausgabe den Melscale Prozeß mit LDA back-end

7.3.1 PLP+Delta-Koeffizienten

Dieser Methode wird mit dem PLP Prozeß der frühe in dieser Diplomarbeit, eklärt wurde. Nur ein Paar Deltas wird genommen und es hat ein Verschiebung des vier *frames*. In Abbildung 7.3 sieht man den Ausgabe des PLP mit deltas Prozeß. Diese ist ein 8 ordnung Mean Variance Normalisiert PLP Modele mit 8 Delta-Koeffizienten das gibt so 16 Koeffizienten.

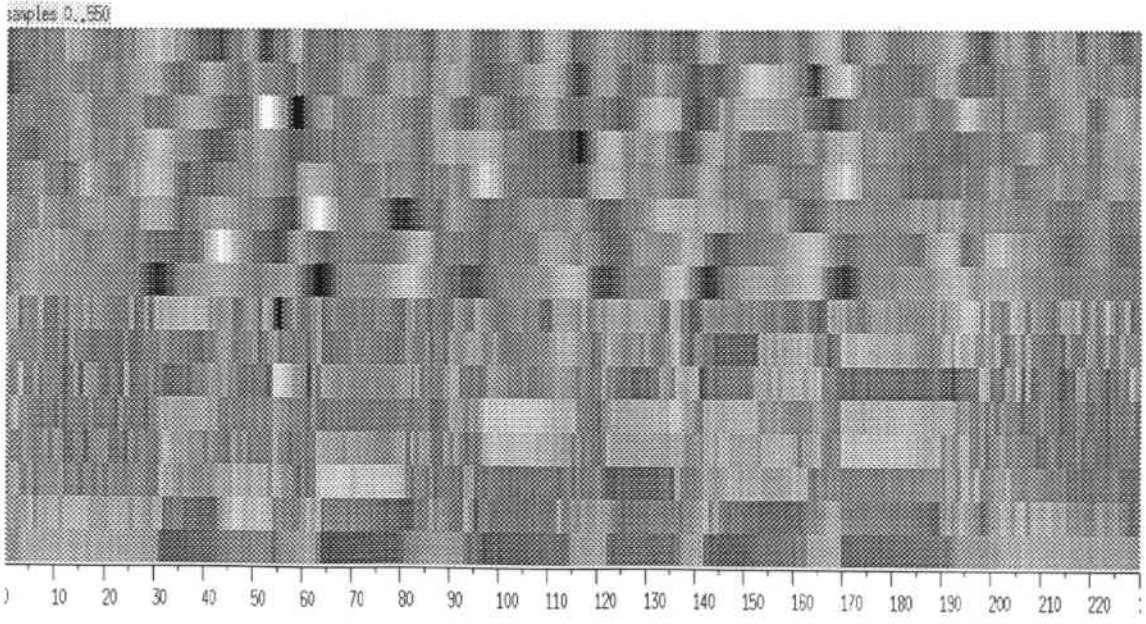

Abbildung 7.3:Die Ausgabe Des PLP+deltas Prozeß

Kapitel 8

Multi-State Time Delay Neural Network

In dieser Kapteln kommt eine Hintergrund des Entwicklung von Neronalen Netz, und seine Anwendung in Spracherkennung. Dann ist der MSTDNN daß in die test benutzt ist [4][5], in mehr detail eklärt.

8.1 Das Multi Layer Perceptron

Ein erste nützliche Neuronalen Netz waren die sogenannte *Multi Layer Perceptron* (siehe abbildung 8.1). Dieser systeme ist von mehrere *perceptrons* hergestellt. Dieser *perceptrons* sind in abgesondert Schichten eigerichtet. Es gibt ein Eingabeschicht, ein Ausgabeschicht und dann mehrere versteckten Schichten zwischen die beiden. Jedes *perceptron* ist an alle die andere in den nächsten Schicht verbunden. Jedes Verbindung ist gewichtet, so wenn ein signal von ein *perceptron* an ein andere durchlauft, ist diese Signal mit den Gewicht multipliziert. Jedes *perceptron* hat auch eine Funktion das der Ausgabe controliert. Dieser Funktion sagt man ist ein Entscheidungfunktion und ist es entweder Sigmoid oder Schwelle (englisch: *step*) (siehe Abbildung 8.2). Wenn einem Eingabe durch das Modelle geführt wird, wird die *perceptron* der Ausgabeschicht bestimmten Werte haben. Man kann die Zustand den ganzen Netz durch die Zustand von jedes *perceptron* in den Ausgabeschicht identifizieren.

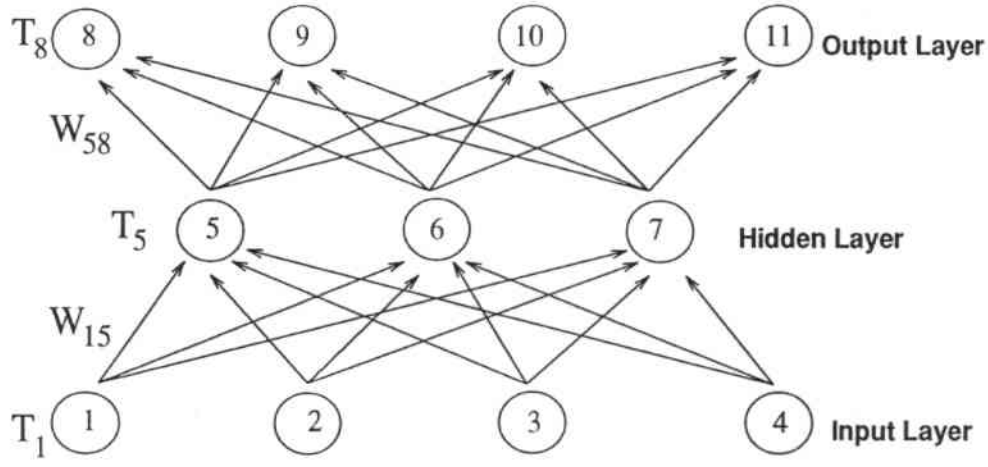

Abbildung 8.1: Die Multi Layer Perceptron

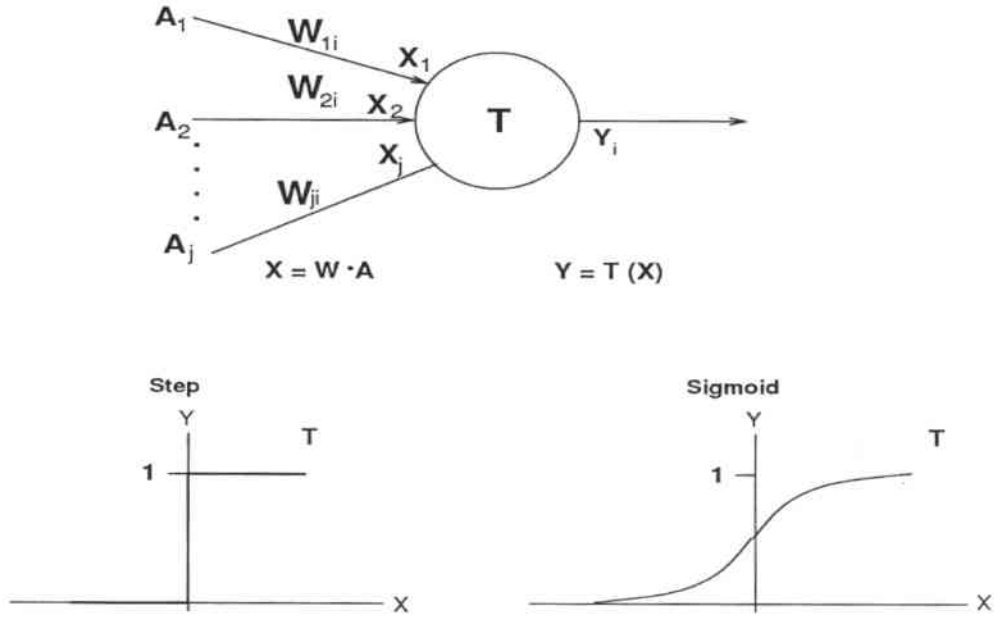

Abbildung 8.2: Ein Perceptron ud Die Zwei Mögliche Funktionen

Dieser systeme hat ziemliche gut Erkennungeigentrum. Es wird mit änliche abgabe die gleice Werte im Ausgabeschicht haben. Es kann auch trainiert wird, die maximale von unterschiedlich Zustanden zu haben, und auch so besser zwischen änlich aber unterschiedlich abgaben zu differenzieren.

In dieser Trainingsphase wird ein Abgabe durch den netz gefürt, bis zum einem Ausgabe der Ausgabeschicht bekommt wird. Die Zustand den Ausgabeschicht wird dann an was mochtete war verglichen. Dieser gibt ein fehler wer ist zurück durchgeführt, die gewichtet von alle die Verbindung zu anpaßen. Dieser Prozeß ist für alle mögliche Abgaben gemacht.

8.2 Das MLP in Spracherkennung

In einem Spracherkennungssysteme ist die Auswahl den Abgabe sehr wichtig. Es könnte ein buchstabe, ein phoneme oder eine sub-phoneme sein (im moment sagen wir immer daß den Abgabe wie Phoneme ist). Das Erkenner ist zuerst mit viel beispielen des Abgaben trainiert. Das heißt mehrere beispielen des jedes phoneme sind durch die Netz geführt, und die gewichten in den Netz angepaßte immer die gleichen Abgabezustanden zu haben. Das ist nicht naturlich immer möglich, weil eine unterschiedliche phoneme könnte zu ännlich sein, aber der system versucht die Fehler zu minimisieren. Im Test sind dann andere beispeilen durch das Erkenner geführt. Das Erkenner nimmt, so die Abgabezustand wie was gesagt war.

8.3 Das MSTDNN

Eine *Time Delay Neural Network* (TDNN) vereinigt in die gründsatzlich Stuktur des Neuronalen Netz ein zeitliche Verscheibungsinvarianz [5]. Das heißt, anstatt die abgabe und Signalen zwischen jedes Schichten an ein Zeitintervalle festgellegt sind, Sie passieren durch ein Satz des Zeitintervallen die vordefiniert werden. Jedes Schicht ist deshalb nicht mit ein verbinen aber mit so viel wie es Zeitintervallen gibt. Jedes Verbinden representiert die Zustand des verbindene Neurone für diese Zeitintervalle.

Die Multi-State TDNN, weiter integriert die nicht-lineares *time-alignment* Methode *dynamic time warping (DTW)*, in die TDNN Architektur [5]. Dies gibt die fünf Schichten Modele die man in Abbildung 8.3 sieht.

Die erste drei Schichten sind die Standard TDNN Architektur. Sie functionnen wie ein normale Neuronalen Netz, aber es gibt zwischen die erste und zweite Schichten drei verbindungen pro Neurone die der Zustand des Neurone für t, $t-1$*frame* und $t-2$*frames* präsentieren. Diese ist gleich zwischen die Schichten zwei und drei aber hier gibt es fünf Zeitintervallen. Das *score* für jedes Zustand, wird mit ein Gleitfenster durchschnitt (englisch: *sliding window average*), über die Zeitintervallen berechnet. In dem *DTW* Schicht werden jedes Worte zum Erkennen, durch ein Sequenz des Phoneme der von das ẅordS̈chicht Kopiert wurde, modelisiert. Hier das Erkenner sucht ein optimale *time alignment* pfad für jedes Worte. Das heißt er sucht die warscheinlichste Position des Phonemen innerhald der Worte *frame*, um es besser mit die Richtige Model zu anpaßen.

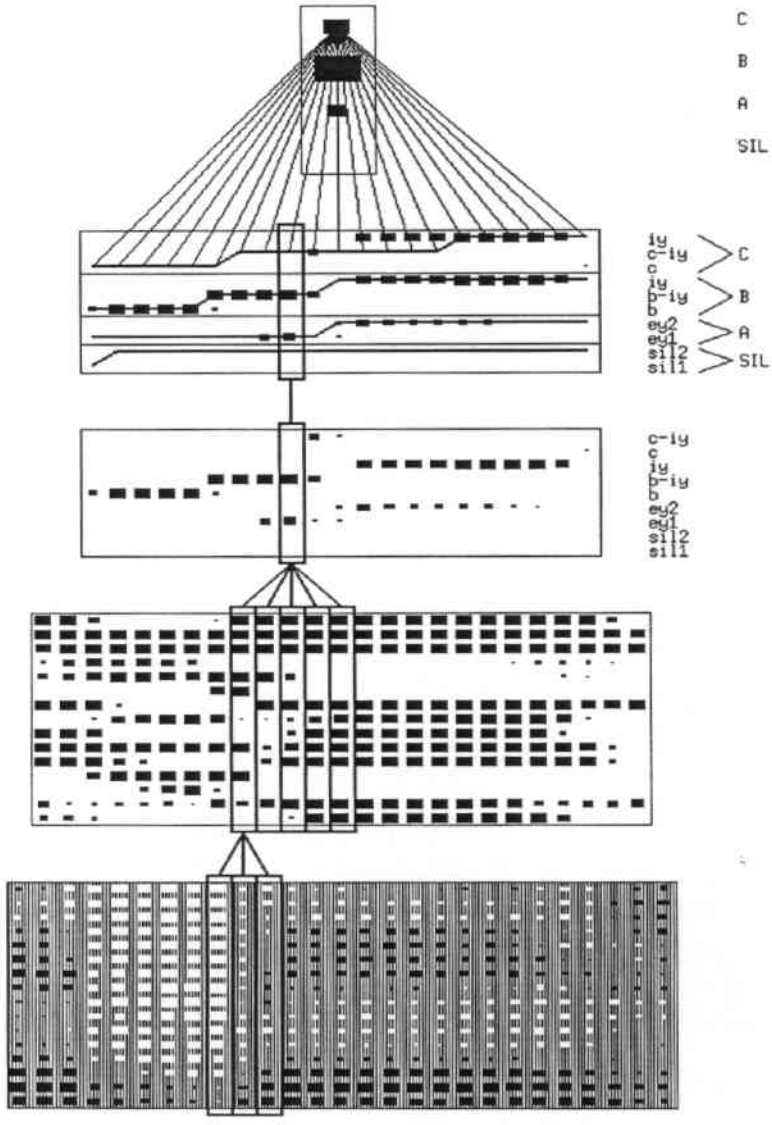

Abbildung 8.3: Das MSTDNN

8.4 Training

Es gibt zwei Trainingsphasen, die Ertste ist die sogennante bootstrappingphase. Hier nur das *front-end* TDNN wird benutzt. Die Phonemegrenzen sind zum Erkennungszielen festgelegt. Die Fehlerderivativen werden zurück durch den TDNN geführt und die Verbindunggewichten werden nach die Objectivefunktionen gegeben in [4][5] angepaßte. In der zweite Phase ein Wortstufetraining wird gemacht. Die Phonemegrenzen werden frei (innerhalb die Wortgrenzen) in der *DTW* Schicht zeitausgerichtet. Die Fehlerderivativen werden von die Wort Elemente, durch die *DTW* Schicht und die *front end*

TDNN geführt. Noch einmal wierden die Verbindungsgewichten nach die Objektivfunktionen angepaßte. Das Netze fangt die zweite Trainingsphase mit die Verbindunggewichten die am Ende des b̈ootstrappingP̈hase berechnet wurden.

Kapitel 9

HMM Spracherkenner

In dieser Kapitel findet man zuerst ein kurz Umrisse von dem Prinzip hinter HMMs, und auch sein Anwendung in Spracherkennung. Für eine detailiert führer an HMM siehen [15]. Dann kompt die Details über den Janus HMM, die Charakteristika von den erkenner zum besser eklären.

9.1 Die statische Betrachtungsweise

Die Systeme ist die folgenede:

- W ist der Texte zu sprechen werden.
- Y ist der erhielte Signal
- O ist der Beobachtung von der Untersuchungen von Y
- $\tilde{W}$ ist die bekannte Sprache

Der Statische Zutritt sucht das Wahrscheilichste Texte zu gesprochen worden sein, gegeben die Untersuchungen O. Das Erkenner sollte deshalb die Antwort $\hat{W}$ geben Solche:

$$P(\tilde{W} \mid O) = max_W P(W \mid O) \tag{9.1}$$

Von Bayes Theorem:

$$P(W \mid O) = \frac{(P(O \mid W) \cdot P(W)}{P(O)} \tag{9.2}$$

und weil $P(O)$ unabhängig von die Texte W ist, ist die maximierend des Wahrscheinlichkeit $P(W \mid O)$ gleich als maximierend den Wahrscheinlichkeit:

$$P(W \mid O) = P(O) \cdot P(O \mid W) \tag{9.3}$$

Man kann die Wahrscheinlichkeit von jedes Meldung berechnen und dann die Wahrscheinlichsten selektieren.

9.2 Definition eines HMMs

Der Systeme ist ein endlicher Zustandsautomat (englisch: *finite state machine*), oder gleich eine Markovmachine, der ist von Zustanden gennant $Q = q_1....q_T$ und Übergängen zwischenn die Zustanden (siehe Abbildung 9.1). Jedes Übergänge ist durch die Übergangswahrscheinlichkiet zwischenn die ausgebend Zustand und die nächste rperäsentiert. Dieser Übergangswahrscheinlichkeit wird durch ein kontinuierliche Wahrscheinlichkeitsdichte definiert (multivariate Gauss Funktion). Die dichte Funktion wird durch eine (normaleweise diagonal) Kovarianzmatrix und Verteilungsvektor präsentiert. Dieser versteckt Marcov Modele (versteckt weil jedes Ursprung korrespondiert an die Produktion von einige gesund nicht direkt bemerkenswert) ist so definiert:

- Die N Zustanden $q_1, q_2...q_N$
- Die Verteilungsvektor von den Anfangszustand:

  $\prod^T = (\prod_1, ..., \prod_N)$ mit

  $\prod = Wahrsheinlichskeit(X(1) = q_i) i = 1, ..., N$

  In Spracherkennung die Anfangszustanden bleiben Konstant.
- Ein Übergangswahrscheilichkeitsmatrix zwischen Zustand:

  $A = [a_{ij}] 1 \leq i \leq N, l \leq j \leq N$ mit

  $a_{ij} = Wahrscheinlichkeit(X(t) = q_j \mid X(t-1) = q_i)$

  Im Janus wird die Übergangswahrscheinlichskeiten am Am=nfangs des Trainings fetsgelegt.
- Und am wichstigstens, die Emissions Wahrscheinlichskeiten:

  $Probability(X \mid q_i) = \sum_k d_k N(\mu_i, \Sigma_k)$

  Wo $N(\mu_k, \Sigma_k)$ ist eines Guaßches Wahrscheinlichkeitsdichte Funktion und d_k ist die Gewicht dieses Funktion nach sein Positsion (Positsion der Mittelwerte) innerhalb die Verteilung alle die andere Wahrscheinlichkeitsdichte Funktionen (Verteilungen des Mittelwerte).

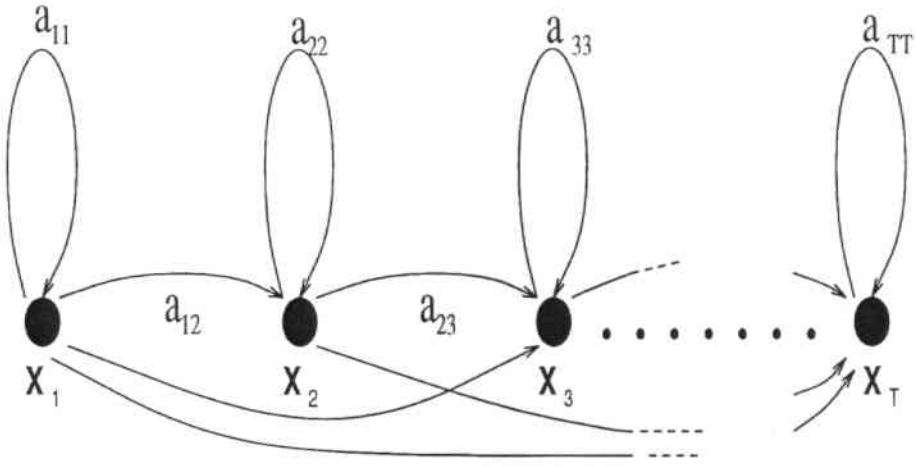

Abbildung 9.1: Ein allgemein Hidden Markov Model

Gegeben den Beobachtung $O = O_1O_2...O_T$ für ein Sprachsignal Der Beziehung von den Kette X und den Beobachtsequenz O ist durch die definition, für jedes q_i in den Modele, von ein Wahrsceinlichkeitsdichte Funktion b_i Solche:

$$N_i(v) = Wahrscheinlichkeit(O(t) = v \mid q_i) für i = 1, ..., N \tag{9.4}$$

9.3 Training des JANUS HMM

Man findet in Abbildung 9.2 Das Trainings Prozedure der Janus Spracherkenner. Die erste Operation ist die Worte *labels* zu machen. Diese identifizieren die Anfangspnonemesgrenzen. Um die gute Grenzen für die erste Trainings Phase zu Liefern, braucht man die Codebuche und Verteilungen von eine schon gut Trainiert Janus Modelle. Diese könnte von ein anderes *Feature*, oder ein *Feature* mit ein verschiedene Trainingsmenge kommen, weil erwartet man daß obwohl die Merkmale der ein andere *Feature* sehr unterschiedlich sind, die Positionen des Wortegrenzen ähnlich sein sollten. Die Anfangs *labels* für den Tests die in dieser Diplomarbeit gemacht werden, kommen von frühe Trainiert Melscale Daten.

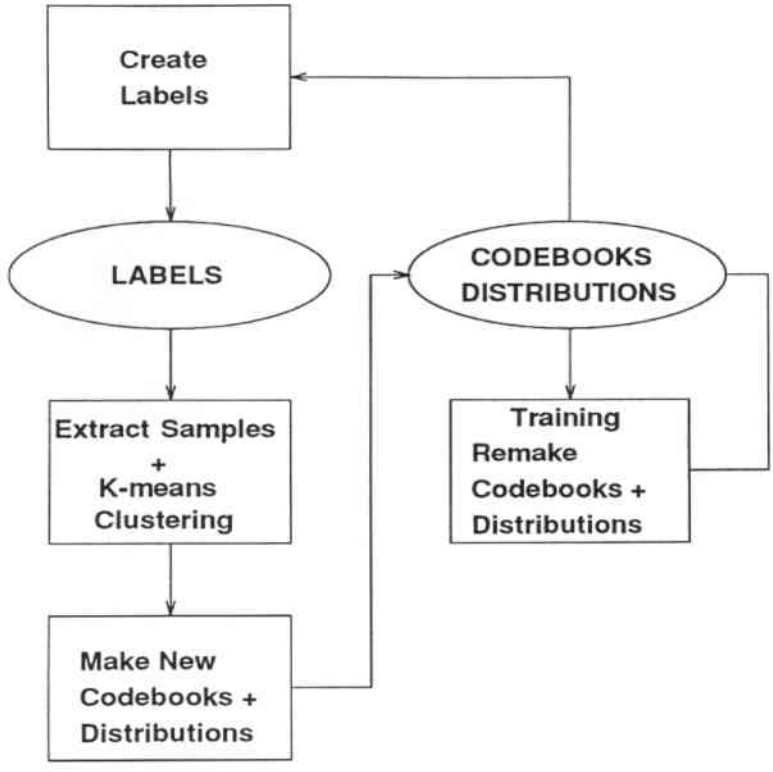

Figure 9.2: JANUS Training Procedure

Das Systeme benutzt die Phonemesgrenzen, jedes Worte in den Trainingsmenge zu identifizieren, und davon erzeugt es ein Modelle für jedes mögliche Phoneme. Dieser so-gennante *k-means clustering*, wohin wird alle die möglichen Phonemegruppen fortschrittlich zusammen gestellt, biszum es ein Modelle pro Phoneme gibt, wird wie die erste Trainings Phase betrachtet. Die Merkmale Modellen für jede Worte werden in einem Codebuch wie ein Vektor die der Zustand der Jedes Phoneme präsentiert gespeichert. Der große der Vektor ist Variable , aber für alle die Tests die hier gemacht werden, wird es am 20 festgelegt. Die gaußches Verteilung des Zustanden innerhalb jedes Codebuch wird auch definiert. Das Covarianzmatrix das die gaußche Wahrscheinlichkeitsdichte des emissions präsentiert, wird auch berechnet, aber es gibt die auswahl daß eine Vollmatrix, nur ein Diagonalmatrix oder kein Matrix definiert wird. Wenn kein Matrix definiert wird, prädentiert man die Verteilungen durch das Mittelwerte. Die Covariazmatrizen wird in der Codebuch gespeichert. Um die Trainingsumfang zu reduzieren , wird Diagonalcovarianzmatrizen benutzt.

Für das Rest des Trainings gibt es jetzt ein Vollständigmodelle, so man braucht kein *labels* mehr. Von der Anfangszustand, wird das Trainingsmenge durch den Modelle geführt, und die Codebuche, Verteilungen und Covarianzmatrizen werden aktulisiert entsprechend die Leistung der Modelle, die Worte zu klassifizieren. Dieser Prozeß wird dann mit dem aktulisierte Modelle für den gewünscht Anzahl des iterationen wierderholt.

Kapitel 10

Ergebnisse

In dieser Kapitel werden die Testprozeduren (Details des Test/Trainings Daten, Ergebnisse rechnen/verglichen Prozeduren ETC.) für die zwei Erkenners eklärt. Danach kommt die Ergebnisse von jedes vorverarbeitung methoden und die Kommentar darüber.

Die beiden Erkenners werden mit 491 buchstabierte Deutsche Namen trainiert, und mit 100 getestet. Dies sind eckte aufgezeichnete Telephon Daten, die wurden von mehrere Deutsche Sprecher gesprochen. Alle die Teste Können dafür wie sprecher unabhängig (englisch *speaker independent*) erwägt werden. Das Telephon Daten wurden von Siemens AG lieferte.

Innerhalb die die Deutsche Sprache, gibt es im Total 32 mögliche Buchstaben Außprachen. Diese schließt die ä, ö, ü und die drei Außprachen des ß ein. Noch weiter gibt es die möglichkeit der "doppel" ausgesprochen werden würde, die doppel Buchstaben zu ausdrücken. Dann kommen die zwei mögliche Aüsprachen "Strich" oder "Bindestrich", die Namen die mit Bindestrich geschrieben werden zu ausdrücken. Und schließlich, muß es ein Stillemodelle für alle die Sprachpause geben. Das gibt ein Buchstabe Niveau Vokabular von 36 worte. Die Buchstabe Vokabular ist auch durch ein Phoneme Vokabular von 70 worte präsentiert (siehe Abhängen B).

Note: Noch ein sehr wichtig probleme für eine automatische Spracherkenner ist der unterschied zwischen die Trainings Umwelt und die Test Umwelt. Obwohl, die gründsätzlich Charakteristische des Telephon Kanal änlich sind, kann das Telephon Daten durch mehrere systeme und Störung möglichkeit laufen. Es ist wichtig, deshalb, zu sehen, daß die Trainings Umwelt und die Test Umwelt nicht zusammen paßen. Obwohl erwartet man daß die zufälligkeit alle die mögliche Störungen, die zwei Umwelt angepaßt machen werden würde, ist es ein wichtig Hinweis, besonders wenn man die Ergebnisse mit andere änliche Tests die mit gut zusammengepaßt Trainings und Test Umwelten gemacht wurden, verglichen möchte.

10.1 Testen des MSTDNN

In der MSTDNN nach jedes Trainingsiteration wird ein Test gemacht. Während die beiden Trainings Phase (Phoneme Niveau Training und Buchstabe Niveau Training) wird ein Buchstabe Niveau Test gemacht. Hier sucht den Systeme für die Buchstabegrenzen, bevor er die an die Netz für erkennung gibt. Die erkennte Buchstabe wird dann mit dem eingabe verglichen und die Anzahl der richtigen Buchstaben wird wie ein Prozent des Total gegeben. Ein *Scatter Matrix*, das welches Buchstaben mit welche Andere vewwirrt werden, wird erzeugt (siehe Abbildung 10.1). Es ist wichtig zu sehen daß hauptsächlich die Erkenner ein sehr gut Stilleerkennungsgenauigkeit habe wird, weil ist ein Stille *frame* sehr verschieden. Gegeben daß ein Große Proportion der Sprache Stille ist (ungefähr 50 % für diese Daten), kann der Erkennungsgenauigkeit oft irreführend sein. Das MSTDNN liefert, deshalb, zwei Wort Erkennungsgenauigkeiten; ein für alle die Worte, und ein für alle die gesprochene Worte, ohne Stille. Die zweite ist ein viel besser Repräsentation der Syteme Leistung, und ist deshalb das das in alle die Ergebnisse gegeben wird.

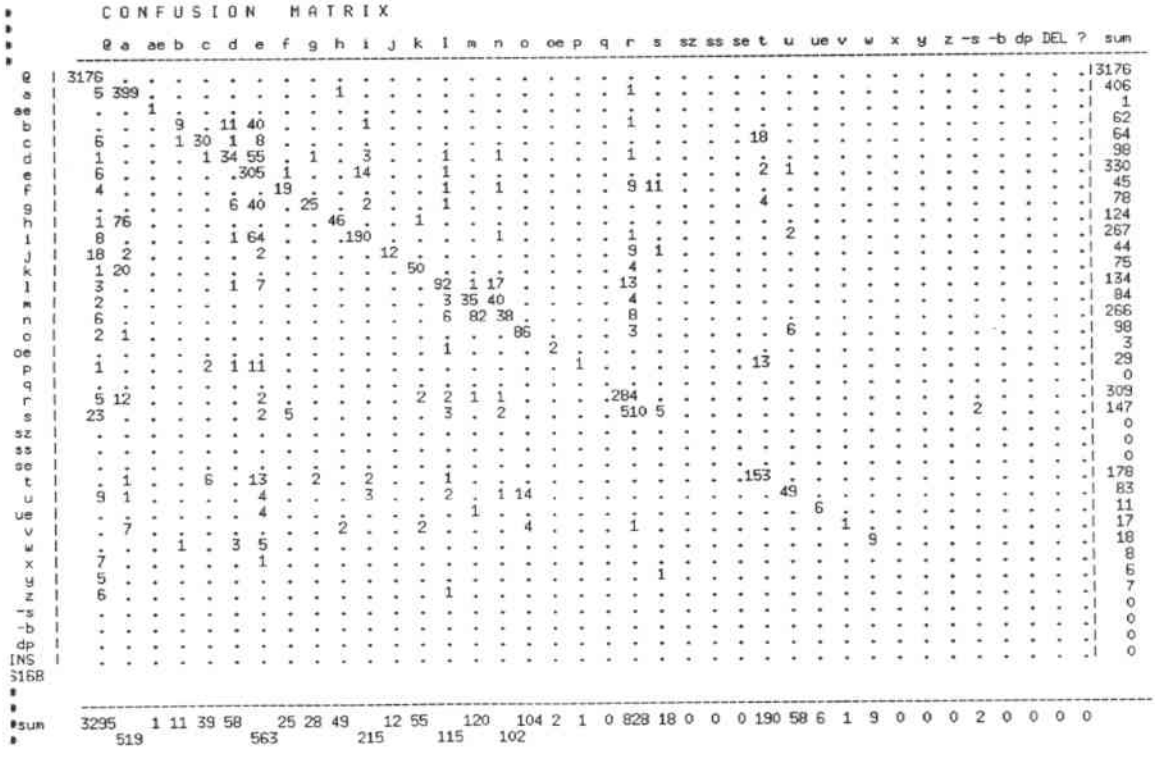

Figure 10.1: Beispiel ein Scatter Matrix

Wie das MSTDNN Systeme, trennt die Satz in Buchstabe, bevor das Test. Und angenommen, kein Buchstabegrenz wird vepaßt, die Erkenner wird immer die Richtige Anzahl der Buchstabe in den Satz identifizieren. Es wird,

deshalb, kein *inserted* oder *deleted* Buchstaben, nur *substitutions* von richtige Buchstaben mit falsche. Das Wort genauigkeit (englisch *word accuracy*) ist deshald:

$$WA_{single} = \frac{correct\_words}{total\_words} \tag{10.1}$$

Dieser ist kein kontinuerliche Erkennung (wohingegen der Janus Erkenner ist) und obwohl, die möglichkeit von kontinuerliche Erkennug durch den MSTDNN verfügbar ist, wird es nicht in dieser Diplomarbeit getestet. Dieser ist hauptsächlich weil erwartet man daß die Buchstabe Niveau Erkennung, mit individuell buchstabierte Buchstaben besser ergebnisse geben werden würde.

Die Structur der MSTDNN erlaubt das Leistung des Training Phase zu getestet werden. Diese testet die Fähigkeit der Erkenner, die Trainingssetz zu sich erinnen. Nach jedes, Trainingsiteration wird die Anzahl des Worte die richtig erkennt wurden (diese die kein Verbindung Gewicht zu wechseln machen I.E. kein Fehler geben) wie ein Prozent der Total, immer ohne Stille, gegeben. Obwohl, der Test Genauigkeit die wichtigstens ist, gibt das Trainings genauigkeit auch ein paar information über den Leistung der Systeme, wie später eklärt wird.

10.2 Test des Janus Erkenners

Der Janus Erkenner is ein Kontinuerliche Spracherkenner. Das heißt, er nimmt die ganzen gesprochene Satz, und davon versucht er zu erkennen was gesagt wurde. Es gibt, eshalb, kein Anfangstrennung des Satz in die Buchstaben. Diese wird in die erkennungs Prozeß selbst gemacht. Die Systeme nimmt jedes Satz, ein nach ein andere, und für jedes *frame* er sucht die wahrscheinlichste Wortpfad das zu anpaßen (er benuztz für dieser versucht die Viterbi Algorithmus). Jede mögliche Pfad bekommt ein *score* und die beste wird genommen, bis er an einem ferige Worte kommt. Dieser wird wie die gesprochene Worte betrachtet, und es bekommt wie ein Totale *score* die Summe des Pfad *scores*.

Dieser Methode der Kontinuerlich Spracherkennung führt an die Gegenwart der *inserted* Worte (wenn ein wort zu früh erkennt wird, gibt es extra *frames* die wie etwas anderes erkennt werden) und *deleted* Worte (wenn ein Worte zu späte erkennt wird, wird die folgende Worte verpaßt werden). Das berechnung des *word accuracy* ist deshalb mehr kompliziert als für den MSTDNN, weil werden die richtige erkrnnt Worte nicht unbedingt gut mit die eingabe Worte angepaßt. Man mach so alle die Falsche Worte (*insertions*) weg und dann paßt die andere Worte mit die eingabe Worte zusammen, um für die

fehlende Worte (*deletions*) zu Kompensieren. Die *word accuracy* wird dann mit dem Wort fehler Teil (*word error rate*) berechnet:

$$WER = \frac{substitutions + deletions + insertions}{correct\_words + substitutions + deletions} \tag{10.2}$$

$$WA_{continuous} = 100\% - WER = \frac{correct\_words - insertions}{correct\_words + substitutions + deletions} \tag{10.3}$$

Janus wird nur mit der *features* die das LDA Prozeß durchgeführt worden. Weil findet mann die schlecht Ergebnisse ohne diese Prozeß.

10.3 Ergebnisse des MSTDNN

10.3.1 Melscale Ergebnisse

Zuerst sieht man in Abbildung 10.2a und b, die Test und Trainings Ergebnisse der MSTDNN, für den Melscale mit Meansubtraction *Feature*. Die erste ist für den Phoneme Niveau Training und die zweite ist für den Worte Niveau Training. Die Forme des Kurven sind typische des Leistung der MSTDNN für alle *features*. Die Trainings genauigkeit wächst ziemlich zuverlässig an, wie mann erwarten werden würde, weil wird das Nets immer besser das Trainings setz sich merken. Der wanderung kommt wahrscheinlich infolge die Änderung des Gewichten die ein Änderung in ein Richtung erzeugt, und dann in der nächste iteration versucht für das zu Kompensieren ind so ein Änderung in die andere Richtung erzeugt. Das stark abweichend, aber langsam zuhnehmend Form des Test Ergebnisse, shaut daß der Netz richtig zufällig ist wenn er ein unbekannt Eingabe hat.

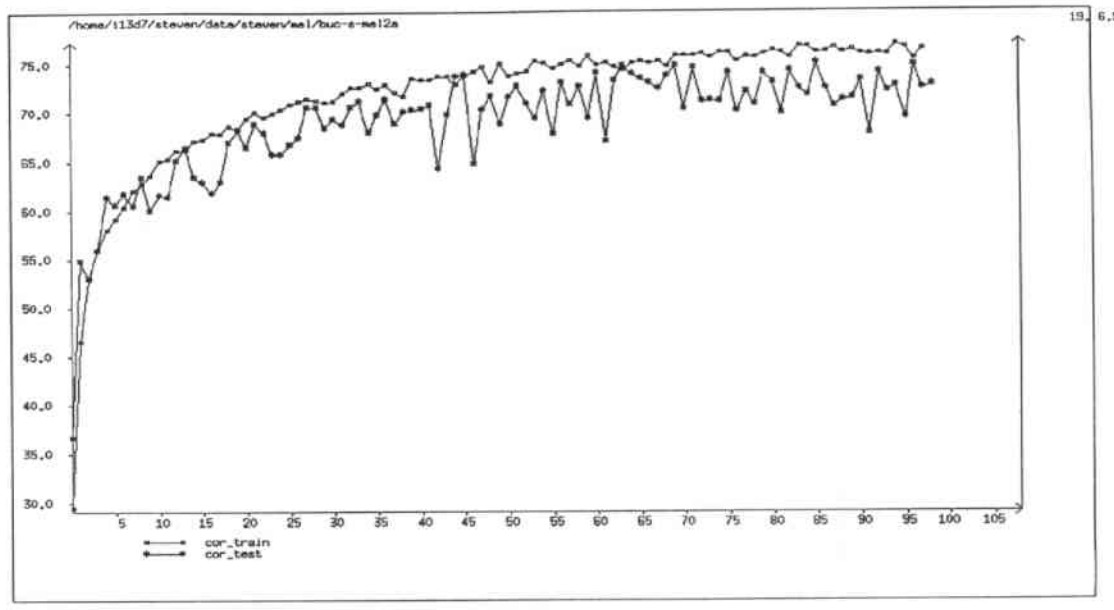

Figure 10.2a: Test und Training Ergebnisse für Phoneme Niveau Training

Figure 10.2b: Test und Training Ergebnisse für Worte Niveau Training

Die beiden Ergebnisse werden für 100 Trainings iterationen gezeigt. Für den Rest Ergebnisse die in dieser Lektion discutiert werden, wird der beste Leistung der nach 50 iterationen ausgeführt wird betrachtet. Obwohl wird das Leistung wahrscheinlich mit mehr iterationen verbessern, erwartet man daß diese Anzahl der iterationen ist genug, vergleichend Ergenbnisse zu geben.

Tabelle 10.1 gibt die Ergebnisse für den Melscale Teste mit Mean Subtaction und ein Normalisation von 0 bis 1.

		No. of Iterations		Test Word Accuracy		Train Word Accuracy	
Pre-Pocessing	No. Coeffs.	P	W	Phoneme Level	Word Level	Phoneme Level	Word Level
Melscale Normalised from 0 to 1	16	50	50	74.83	83.28	73.20	89.84
Melscale With Mean-Subtraction	16	50	50	73.68	83.61	74.93	90.74

Table 10.1: Melscale Test und Training Ergebnisse für den MSTDNN

Wie man sieht, der erste Test nach dem Phonem Niveau Training, die Melscale mit Mean Subtraction gibt ein Leistungs verbesserung über die Melscale die normalisiert von 0 bis 1 ist. Auf diese Gründ werden alle die folgende Teste mit *features* die mit Mean Subtraction sind gemacht.

10.3.2 PLP Ergebnisse

In Abbildung 10.3 sieht man die Leistung des PLP Prozeß für verschiedene Ordnungen der PLP Modelle. Diese sind ohne die Delta-Koeffizienten. Man sieht daß in diesem Test ein Modell der Ordung von 13 optimal ist, und höhere Ordnung keine Verbesserung geben oder sogar verschlechternd wirken. Ein Optimum der *word accuracy* von 81 % wird mit dem Modell der Ordnung 13 erreicht. Der ist mehr als 2 % wenige als der die Melscale Prozeß.

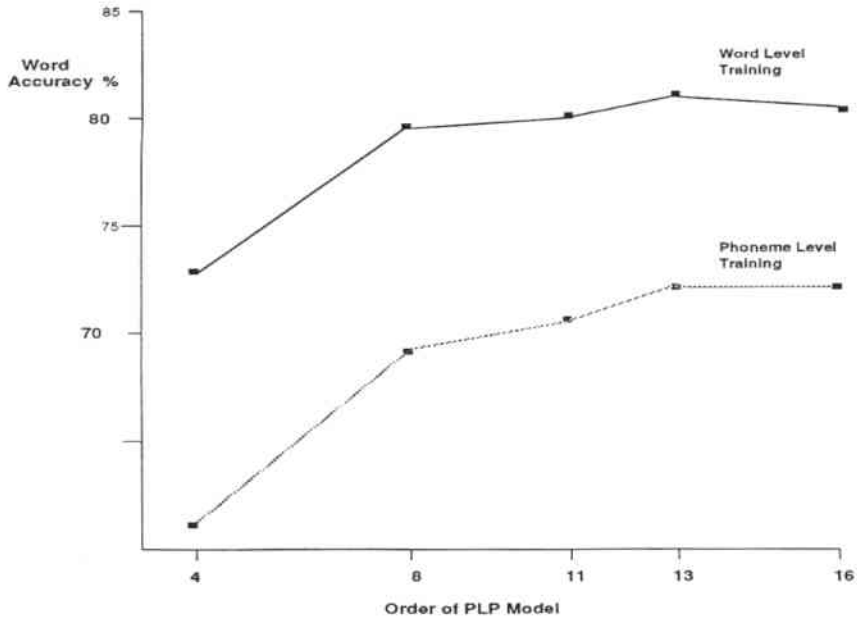

Figure 10.3: Effekt der Ordnung der PLP Modelle

Der PLP Prozeß mit den Delta-Koeffizienten gibt die Ergebnisse die man in Abbildung 10.4 sehen kann. Hier wird ein Model der Ordnung 11 als optimal

gefunden. Die Form des Kurve ist änlich wie die ohne den Delta-Koeffizienten Prozeß, mit einer sich sehr verschlechternden in Erkennungsgenauigkeit für eine Ordnung kleiner 8, und einer sich langsam verschlechternden für eine Ordung die größer als der optimal Wert ist. Auch sieht man daß, die maximale *word accuracy* hier bei 84.11 % erreicht wird. Das ist eine kleine (0.5 %) Verbesserung gegenüber dem Melscale Prozeß.

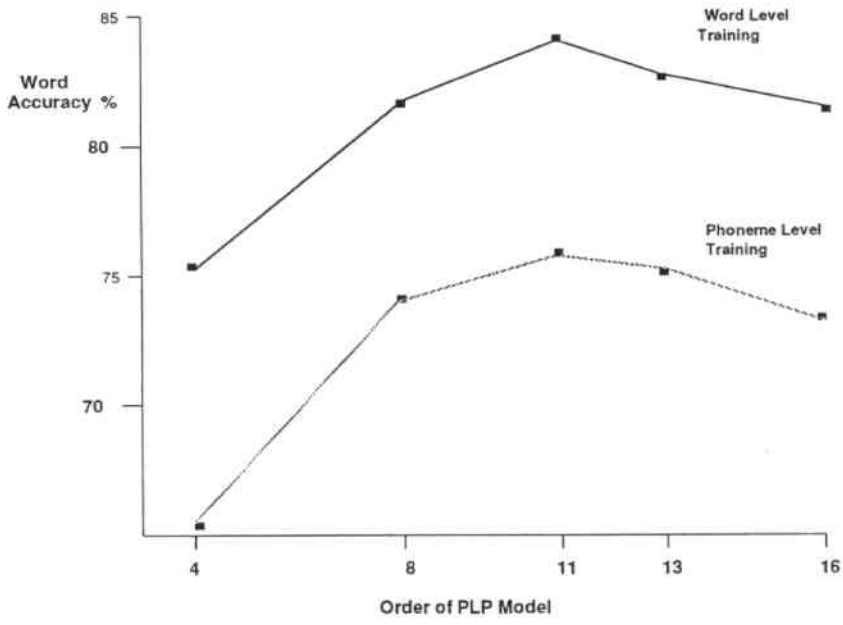

Figure 10.4: Ergebnisse für Verschiedene Ordnung der PLP Prozeß

10.3.3 RASTA Ergebnisse

Man sieht in Abbildung 10.5 die Ergebnisse für den RASTA Prozeß mit verschiedenen Werten von J. Diese sind nur für die Phoneme Trainings Phase bestimmt. Der RASTA Prozeß scheint besser zu funktionieren mit einem J kleiner 1×10^{-4}. Die Systeme wurden nicht mit genügend kleinen Werten von J getestet, um sehen ob die Leistung mit noch kleiner werten den J wieder schlechte werden würde. In Kapitel 6 wurde aufgezeigt, daß kleine Werte von J besser das additive Rauschen Kompensieren würden. Aber es ist nicht klar wie klein solche Werte sein müssen. Ob eine verschlechterung der Erkennungsgenauigkeit für J größer als 1×10^{-4} infolge einer Reduktion der Verarbeitung des additive Rauschen auftritt, ist nicht Klar.

Mit $J = 1 \times 10^{-6}$ wurde ein Wort Niveau Test auch gemacht. Die besten *word accuracies* die nach die beiden Trainingsphasen erreicht wurden sind, wahrend wenige als dieser des Melscale oder PLP+delta Prozeß (siehe Tabelle 10.2).

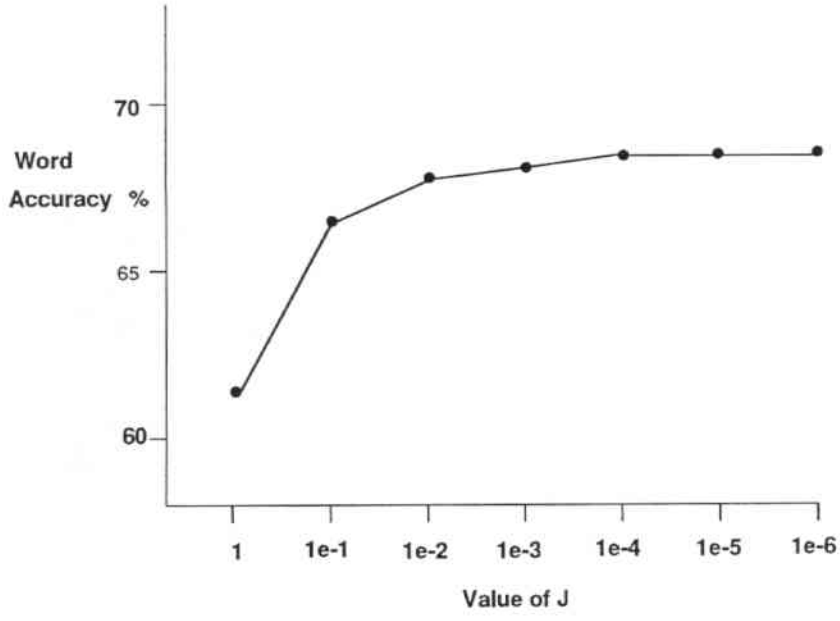

Figure 10.5: Effekte verschiedene J bei RASTA

10.3.4 Zusammenfassung der Ergebnisse

Tabelle 10.2 gibt ein Zusammenfassung der besten Ergebnisse die durch der MSTDNN, nach 50 iterationen, für jede der Vorverarbeitungsmethoden, erreicht wurden. Hier sieht man, daß die Auditory Filterbank keine gute Leistung gibt. Der RASTA-PLP Prozeß wurde mit dem optimalen J für den RASTA Prozeß und der Ordnung 11 PLP+delta Prozeß die auch optimal war implementiert. Trotzdem die Ergebnisse sind nicht befriedigend. Die *Spectral Subtraction*, die mit dem Melscale kombiniert wurde, hat sehr schlechte Ergebnisse. Wenn man die Ausgabe des dieses Prozeß betrachtet (abbildung 7.2), gibt es einen Informationsverlust in der nähe der Wortgrenzen, es könnte sein daß das die Erkennung erschwert.

Ein Verbesserung des Melscale Prozeßes wurde mit dem LDA Methode, mit 16 Koeffizienten erreicht. Eine Suche nach der optimalen Anzahl der Koeffizienten wurde durch den Janus Erkenner geführt (siehe die nächste Lektion). Das Addition der LDA Prozeß an den PLP+delta Prozeß setzt die Leistung herab. Infolge ein Problem mit der Berechnunge der LDA Matrizen (siehe nächste Lektion) wurden die andere Prozeße nicht mit LDAs getestet.

	No. of Coefficients			Test Word Accuracy		Train Word Accuracy	
Pre-Pocessing	Basic	Deltas	Used	Phoneme Level	Word Level	Phoneme Level	Word Level
PLP+Deltas	11	11	22	75.33	84.11	77.21	92.38
Melscale + LDA	16	16+16	48 to 16	72.54	84.11	74.87	92.55
Melscale With Mean-Subtraction	16	0	16	73.68	83.61	74.93	90.74
RASTA	26	0	26	68.54	82.75	64.24	87.13
Auditory Filterbank	16	0	16	69.04	81.46	73.16	90.54
PLP +Delta+LDA	11	11+22+22	66 to 16	71.52	81.46	71.69	89.64
PLP	13	0	13	72.35	81.00	72.03	90.44
RASTA - PLP	11	0	11	67.22	79.64	66.88	88.03
Melscale + with Silence Subtraction	16	0	16	69.87	76.32	79.12	80.45

Tablle 10.2: Zusammenfassung Der MSTDNN Leistung

10.4 Janus Ergebnisse

10.4.1 Melscale LDA Ergebnisse

In Abildung 10.6 sieht man die Ergebnisse des Janus Spracherkenners der mit dem Melscale LDA Prozeß getestet wurde. Man sieht das Effekt verschiedener Anzahlen von Koeffizienten, und daß 16 optimal war. Obwohl man eine Verschlechterung mit sehr wenig Koeffizienten erwarten würde, ist es interessant daß mit mehr Koeffizienten ein Abnahme der Spracherkennungsfähigkeit eintritt. Bei 28 Koeffizienten gibte es wieder ein teilweise Verbesserung. Diese *word accuracies* werden in das optimal Training übergenommen.

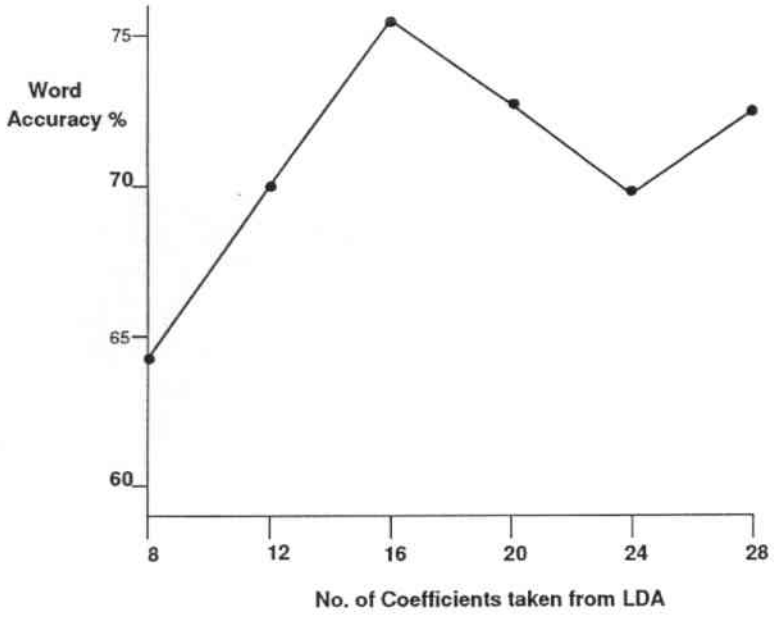

Figure 10.6: Effekte der Koeffizientenanzahl bei Melscale-LDA

10.4.2 PLP-LDA Ergebnisse

Wie mit dem MSTDNN werden Janus mit verschiedene ordnung der PLP Modelle getestet (siehe Abbildung 10.7). Hier die Anzahl der LDA Koeffizienten ist immer 16. In dieser Test wurde ein Modelle Ordnung von 8 wie optimale gefunden. Es ist, wahrend, interessant daß wie mit dem MSTDNN, von eine schlecht *word accuacy* für kleine ordnung, steigt Die Leistung an ein optimale Werte und dann herabsetzt langsam für höhere ordnung.

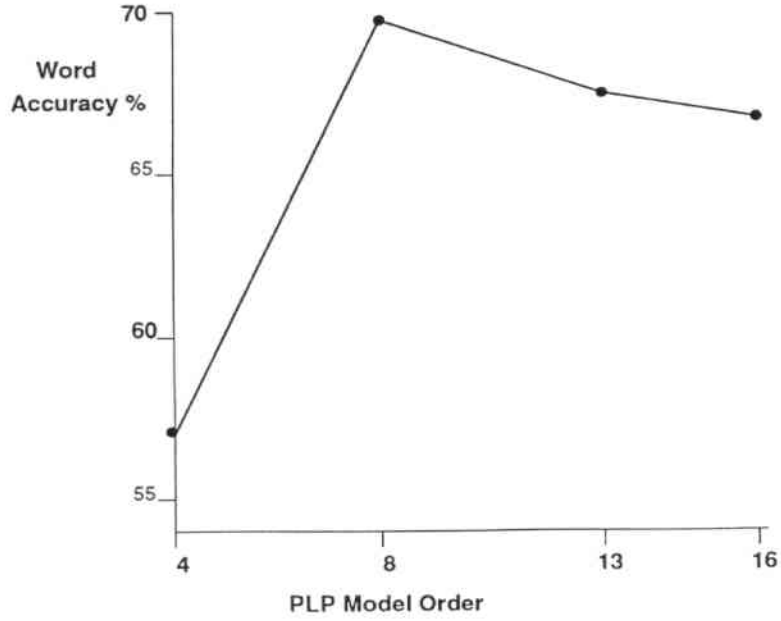

Figure 10.7: Effekte der Ordnung der PLP Modelle bei Janus

10.4.3 Zusammenfassung des Ergebnisse

Abbildung 10.8 gibt eine Zusammenfassung der Ergebnisse die durch Janus mit dem Melscale und dem PLP Prozeß erreicht wurden. Die Iteration 0 bezieht sich auf die anfanglich Erzeugung des Codebuchens und der Verteilungen. Weil die Labels schon durch einen Melscale Prozeß erzeugt wurden, wurden keine neue Labels mehr für den Melscale Prozeß benötigt. Man sieht daß von der schon optimalen 0 Iteration (mit den bestimmten Labels), keine Verbesserung des *word accuracy* durch weiteres Training entsteht. Es gibt eine Verbesserung des PLP Prozeßes mit der bestimmten Labels (siehe Tabelle 10.3).

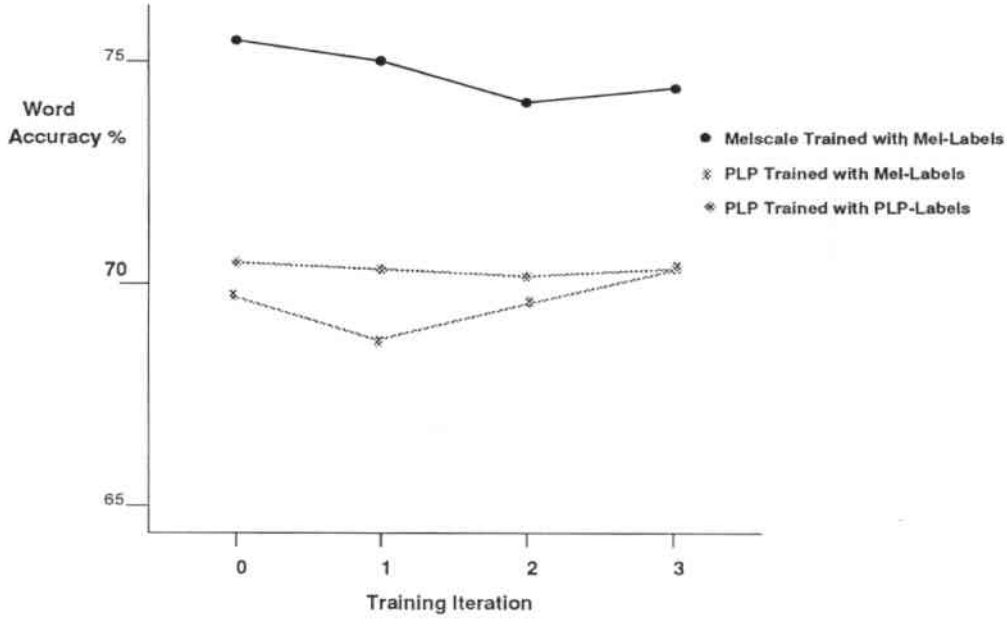

Figure 10.8: Ergebnisse des Janus Erkenner

Pre-Pocessing	No. Coeffs.	No. of Iterations	Test Word Accuracy
Melscale + LDA	16	0	75.5
PLP+LDA (Dedicated Labels)	16	3	70.5
PLP+LDA	16	0	71.0

Table 10.3: Zusammenfassung des Janus Ergebnisse

Der Berechnen der LDA Matrizen für die anderen *Features*, die in dieser Diplomarbeit erklärt werden, ist problematischer gewesen. Die Ergebnisse

waren unbrauchbar. Der Grunde dafür ist nicht erforscht worden, aber es ist der Gründ dafür, daß es kein Ergebnisse für den LDA mit RASTA, Auditory Filterbank und *Spectral Subtraction* gibt.

Kapitel 11

Zusammenfassung

Dieser Kapiteln findet man alle die wichtige Punkte, die in dieser Diplomarbeit vorgestellt wurden. Die bezeichnendete Ergebnisse der Arbeit werden erweitert und Vorschläge für weiterführende Arbeiten werden gegeben.

Verschiedene Sprachsignale Vorverarbeitungsmethoden wurden wie die *front-end* von zwei verschiedenen Spracherkennern getestet, um die Erkennung von Buchstabierte Namen über das Telephon zu verbessern. Die Probleme, die aus der Benutzung des Telephonkanals entstehen wie eine Methode der Kommunikation mit einem Computer sind diskutiert worden.

Die Benutzung der Mean Subtraction mit dem gründlegenden Melscale Prozeß gab eine kleine Verbesserung der Erkennungsgenauigkeit, wenn es mit ein Normalisierung von 0 bis 1 verglichen wird.

Die erfolgreichsten Methoden, waren für den MSTDNN, die PLP Modelle 11 Ordnung mit 11 Delta-Koeffizienten und die Melscale mit einem 16 Koeffizienten LDA Prozeß. Und für den HMM des Janus Projekts (der nicht vollständig getestet wurde) die Melscale mit einer 16 Koeffizienten LDA.

Man hat ein bestimmtes Konsistenz Niveau für die Effekte der Leistung der verschiedene Modellordnungen beim PLP Prozeß gesehen. Die optimalen Werte lagen zwischen 8 und 13 mit einer große Veschlechterung für Werte unter 8 und eine langsamen Abnahme für Werte die oberhalb des Optimums lagen.

Die Große des Unterschieds zwischen den Ergebnissen der Trainingsphase und den Ergebnisse der Test Phase des MSTDNN, für einige Methoden (ungefähr 8% größenfalls) ist interessant zu beobachten. Diese Unterschied wird erwartet, weil mit einer klein genugen Trainingsmenge, das MSTDNN sich alle die Muster merken könnte, und es so eine Erkennungsgenauigkeit von 100% gabe. Aber die Erkennungs genauigkeit in der selbständigen Test Phase wäre sehr schlecht, wenn die Erkenner zu wenige Beispielen der möglichen Eingabe Daten hätte, somit gäbe es einen großten Unterschied zwischen den Trainings- und den Testergebnissen. Wenn die Trainingsmenge immer Größer

wird, wird der Erkenner mehr Probleme haben, sich alle möglichen Trainingsmuster zu merken, und damit werden die Trainingsergebnisse sich verschlechtern. Aber wenn der Erkenner mit mehr Beispeilen trainert wird, sollte er die neuen Testdaten besser Erkennen. Die beiden Ergebnisse (Test und Training) sollte deshalb zusammen kommen. Es wird dann sinvoll, die *features* die einen großen Unterschied zwischen Trainings- und Testergebnissen aufweisen, mit einer viel größeren Trainingsmenge zu Trainieren.

Es gibt auch mehr Arbeitsmöglichkeiten mit dem Delta-Koeffizienten, die hier nur mit der PLP und innerhalb der LDA getestet worden sind, mit anderen *features*. Zusätzlich wäre eine Erforschung welche und wieviel Delta-Koeffizienten optimale sind erforderlich.

Schließlich, der Leistungsunterschied zwischen der Kontinuerlichen Erkennung des Janus HMM Erkenners, und der einzelnen Worterkennung der MSTDNN. Der Unterschied ist logisch, da die Beseitigung der *insertions* und *deletions* automatisch die *word accuracy* verbessert. Es ist auch logisch daß die Erkennung getrennter Buchstaben leichter als die Erkennung eines Ganzen Satzes ist. Die Leistung eines HMM Spracherkenners, mit getrennten Buchstaben würde deshalb einen interessant Verglich mit dem der MSTDNN machen.

Dieser Diplomarbeit hat viele Methoden gesehen, ihre Dauer war allerdings nicht lang genug, um erschöpfende Tests zu machen. Einige solide Schlusse sind, jedoch, gezogen worden, sowie viele wietere Betätigungsfelder.

Anhang A

Featuremaker

Das Featuremaker Signalverabeitungs Program, hat viel funktionen. Die die in dieser Diplomarbeit benutzt werden sind:

READ_ADC	:	Reads in adc data to destination.
FFT	:	Calculates FFT of source; param1 = no. of points in fft; param2 = shift between points (ms).
ADC_TO_MEL	:	Melscales from adc source; param1 = no. of points in fft; param2 = shift between points (ms).
ADC_POWER	:	Calculates power of source; param1 = power window size; param2 = shift.
MERGE	:	Merges two features (source1 + source2).
CUT	:	Takes a section of source; param1 = start coefficient; param2 = end coefficient.
DELTA	:	Calculates delta coefficients; param1 = Size of delta shift.
LIN	:	Calculates param1 $\times$source + param2.
LOG	:	Calculates param1 $\times$log10(source + param2).
EXP	:	Calculates param1 $\times$exp(param2$\times$source).
ALOG	:	Calculates param1 $\times$log10(source+1e-param2$\times$max); max = maximum source value.
SMEANSUB	:	Subtracts signal mean of source1 weighted by source2 variance normalisation to param1.
MELSCALE	:	16 Melscales from FFT.
NORMALIZE	:	Normalises source between param1 and param2.
TRANSFORM	:	Multiply source vector my matrix.
FILTER	:	Filter source with specified filter.
AUDITORY	:	Calculation of auditory spectrum from FFT.
POST_AUD	:	Multiplies source by equal loudness curve.
PLP	:	Calculates nth order LPC model.

THRESH	:	destin = param2	if (source <= param1 && param3 == -1);
			if (source >= param1 && param3 == 1,0);
		destin = - param2	if (source <= - param1 && param3 == 0);
		destin = source	else
MEANSUB	:	Subtracts the source mean vector, normalises source to param1 × standard deviation.	

Featuremaker description file for Melscale processing

```
;-------------------------------------------
; read ADC
;-------------------------------------------
ADC         1     1              read_adc      ADC
;-------------------------------------------
; 16 Melscale coefficients. 256 FFT, 10ms shift.
;-------------------------------------------
MELSCALE 16       2              adc_to_mel    ADC       256       10
;-------------------------------------------
; Normalisation: Meansub or 0 to 1
;-------------------------------------------
NMEL1       16    3              meansub       2
NMEL2       16    4              normalize     0         1
```

Auditory Filterbank Processing

```
;-------------------------------------------
; read ADC
;-------------------------------------------
ADC         1     1              read_adc      ADC
;-------------------------------------------
; Fourier Transform
;-------------------------------------------
FFT         65    2 fft          ADC           256       10
;-------------------------------------------
; Auditory Filterbank
;-------------------------------------------
AUD         16    3 auditory     FFT
PAUD        16    4 post_aud     AUD
;-------------------------------------------
; Mean Subtraction
;-------------------------------------------
FEAT        16    5              meansub       PAUD      2
```

Auditory Filterbank +PLP Processing

```
;-------------------------------------------------
; read ADC
;-------------------------------------------------
ADC         1       1               read_adc        ADC
;-------------------------------------------------
; Fourier Transform
;-------------------------------------------------
FFT         129     0 fft           ADC             256         10
;-------------------------------------------------
; Auditory Filterbank
;-------------------------------------------------
AUD         16      2 auditory      FFT
PAUD        16      3 post_aud      AUD
;-------------------------------------------------
; Perceptual Linear Prediction
;-------------------------------------------------
PLP         8       4 plp           PAUD            20
;-------------------------------------------------
; Delta Coefficients, 4 frames shift
;-------------------------------------------------
delta4      8       5               delta           PLP         4
PLPdelta    16      6               merge           PLP         delta4
;-------------------------------------------------
; Mean Subtraction
;-------------------------------------------------
PLPN        16      7               meansub         PLPdelta    2
```

RASTA-PLP Processing

```
;-------------------------------------------------
; read ADC
;-------------------------------------------------
ADC1        1       1           read_adc    ADC
;-------------------------------------------------
; Fourier Transform
;-------------------------------------------------
FFT         129     0 fft       ADC1        256         10
;-------------------------------------------------
; Auditory Filterbank
;-------------------------------------------------
AUD         16      3           auditory    FFT
;-------------------------------------------------
; Nonlinear Compressing Function
;-------------------------------------------------
AUD_2       16      4 lin       AUD         1e-6        0
lAUD        16      5 log       AUD_2       1           1
;-------------------------------------------------
; RASTA Filter
; IIR
; -2
; 5
; 0.2 0.1 0.0 -0.1 -0.2
; 1
; -0.94
;-------------------------------------------------
RASTA       16      6 filter    lAUD        filter.rasta1
;-------------------------------------------------
; Inverse Function
;-------------------------------------------------
eRASTA      16      7           exp         RASTA       1           1
;-------------------------------------------------
; PostAuditory Processing
;-------------------------------------------------
PAUD        16      8           post_aud    eRASTA
;-------------------------------------------------
; Wiht Perceptual Linear Prediction
;-------------------------------------------------
PLP         11      9 plp       PAUD        12
;-------------------------------------------------
; Mean Subtraction
;-------------------------------------------------
RPLP        11      10          meansub     PLP         2
```

Spectral Subtraction with Melscale

```
;-------------------------------------------
; read ADC
;-------------------------------------------
ADC1        1       1               read_adc        ADC
FFT1        129     0               fft             ADC1        256         10
;-------------------------------------------
; Calculate Power, Transform to log scale, and normalise
;-------------------------------------------
POWER       1       2 adc_power     ADC1            160         10
ALOG0       1       3 alog          POWER           1           4
ALOG        1       4 normalize     ALOG0           0           1
;-------------------------------------------
; Smoothing filtering
;FIR
;-2
;5
;1 2 3 2 1
;-------------------------------------------
SLOG        1       5 filter        ALOG            filter.smooth
SSLOG       1       6 filter        SLOG            filter.smooth
;-------------------------------------------
; Normalisation, thresholding and discretisation
;-------------------------------------------
N-P         1       7               normalize       SSLOG       -0.1        0.5
SPEECH0     1       8               thresh          N-P         0           1.0         0
SPEECH      1       9               lin             SPEECH0     -0.5        0.5
;-------------------------------------------
; smeansub subtracts the power average for the periods
; where SPEECH is 1 ie. silence
;-------------------------------------------
SIL         129     10              smeansub        FFT1        SPEECH      0.0
;-------------------------------------------
; Melscale calculation, Silence is subtracted before
; the logarithmic amplitude scaling.
;-------------------------------------------
MATRIX      16                      13              transform   SIL         mel.matrix
TRUNC       16      14              thresh          MATRIX      0           0           -1
MELS        16      15              log             TRUNC       1.0         1.0
MELSN       16      16              meansub         MEL 2
```

Melscale with LDA Processing

```
;-------------------------------------
; Basic Melscale Calculation
;-------------------------------------
ADC1       1    1    read_adc     ADC
MEL        16   0    adc_to_mel   ADC1      256      10
MELSCALE   16   2    meansub      MEL       2
;-------------------------------------
; Deltas, shifts 1 and 3 frames.
;-------------------------------------
d1FFT      16   3    delta        MELSCALE 1
d3FFT      16   4    delta        MELSCALE 3
dFFT       32   5    merge        d1FFT d3FFT
LDAS       48   6    merge        MELSCALE dFFT
;-------------------------------------
; LDA matrix multiplication
;-------------------------------------
LDAST      48   7    transform    LDAS      matrix.mel
;-------------------------------------
; Truncation to first 16 cefficients
;-------------------------------------
LDAT       16   8    cut          LDAST     0        15
```

Anhang B

Word/Phoneme Vocabulary

Die Vokabular der deutsche "Buchstaben" und seinem wesentliche Bestandteile Phonemen sind die Folgenden:

@	si1	si2							
a	?	ahI	ahF						
ae	?	aeI	aeF						
b	bI	b-eh	ehF						
c	tI	s	s-eh	ehF					
d	dI	d-eh	ehF						
e	?	ehI	ehF						
f	?	aeI	ae-f	fF					
g	gI	g-eh	ehF						
h	hI	h-ah	ahF						
i	?	ieI	ieF						
j	jI	j-o	o-t	tF					
k	kI	k-ah	ahF						
l	?	aeI	ae-l	lF					
m	?	aeI	ae-m	mF					
n	?	aeI	ae-n	nF					
o	?	ohI	ohF						
oe	?	oeI	oeF						
p	pI	p-eh	ehF						
q	kI	k-uh	uhF						
r	?	aeI	ae-r	rF					
s	?	aeI	ae-s	sF					
sz	?	aeI	ae-s	tI	s-t	t-ae	ae-t	tF	
ss	sch	a	ae-r	fF	?	aeI	ae-s	sF	
se	sch	a	ae-r	fF	ae-s	sF	?	aeI	ae-s	sF
t	tI	t-eh	ehF						
u	?	uhI	uhF						
ue	?	ueI	ueF						

v	fI	f-au	auF							
w	vI	v-eh	ehF							
x	?	iI	i-k	k-	sF					
y	?	ueI	p	s	i	l	o	nF		
z	tI	s-t	t-ae	ae-t	tF					
-s	sch	tI	rF	ieI	ch					
-b	bI	ieF	nF	dI	d-eh	sch	tI	rF	ieI	ch
dp	dI	ohI	p	p-eh	lF					

Literaturverzeichnis

[1] H. Hermansky and N. Morgan. RASTA Processing of Speech. In *IEEE Transactions on Speech and Audio Processing* October 1994.

[2] M.J. Hunt, S.M. Richardson, D.C. Bateman and A. Piau. An Investigation of PLP and IMELDA Acoustic Representations and of their Potential for Combination.

[3] S.F. Boll. Suppression of Acoustic Noise in Speech Using Spectral Subtraction. In *IEEE Transactions on Acoustic, Speech and Signal Processing* April 1979.

[4] H. Hild and A. Waibel. Speaker-Independent Connected Letter Recognition With a Multi-State Time Delay Neural Network. In *European Conference on Speech, Communication and Technology* September 1993.

[5] H.Hild and A.Waibel. Connected Letter Recognition with a Multi-State Time Delay Neural Network In *Advances in Neural Information Processing Systems 5* 1993.

[6] P.J. Moreno and R.M. Stern. Sources of Degradation of Speech Recognition in the Telephone Network. *ARPA Workshop* March 1994.

[7] R.W.Schafer and L.R. Rabiner. Digital Representations of Speech Signals. In *Proceedings of the IEEE*[9] .

[8] W.C. Treurinet and Y.Gong. Noise Independent Speech Recognition For a Variety of Noise Types. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* IEEE, March 1994. P I437-440.

[9] Waibel and Lee (editors). Readings in Speech Recognition. Published by Morgan Kaufmann 1990.

[10] M.G.Rahim and B-H Juang. Signal Bias Removal For Robust Teelphone based Speech Recognition in Adverse Environments. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* IEEE, March 1994. P I445-448.

[11] J.P. Openshaw and Z.P. Sun and J.S. Mason. A Comparison of Composite Features Under Degraded Speech in Speaker Recognition. In *Proceedings of the IEEE* 1993.

[12] C.Mokbel, P.Paches-Leal, D.Jouvet and J.Monne. Compensation of Telephone Line Effects For Robust Speech Recognition. In *International Conference on Speech and Language Processing*.Sept. 1994.

[13] J. Koehler, N. Morgan, H. Hermansky, H. Guenter Hirsch, G. Tong. Integrating RASTA-PLP Into Speech Recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* IEEE, March 1994. P I421-424.

[14] K.K.Paliwal and B.S. Atal. A Comparative Study of Feature Representations for Robust Speech Recognition In Adverse Environments. In International Conference on Speech and Language Processing.Sept. 1994.

[15] Douglas O'Shaughnessy Speech Communication Human and Machine. Published by *Addison Wesley.*

[16] M.J.F.Gales and S.J.Young Robust Continuous Speech Recognition Using Parallel Model Combination. *Cambridge University Engineering Department* March 1994.

[17] Ekkhart Bolten. PLP und RASTA-PLP Zwei Verfahren zur Vorverarbeitung von Sprachsignalen. *Studienarbeit for Institüt für Logik Komplexität und Deduktionssysteme, Universität Karlsruhe.* 22 December 1994.

[18] Martin Maier. Dimensionsionalitätsreduktion von Sprachsignalen mit statistischen und neuronalen Methoden. *Diplomarbeit for Deutsch-Französisches Institut für Automation und Robotik, Universität Karlsruhe.* 10 January 1994.

[19] K. Fukunaga Introduction to Statistical Pattern Recognition. Academic Press, NewYork and London, 1972.

[20] P.Jardin Codages Moyen Debit - Issus de l'Analse LPC. *Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique.* 1993-1994.