Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Generierung von Sprachmustern für die Spracherkennung

Florian Deragisch

Semesterarbeit SA-2009-24

Herbstsemester 2009/2010

Institut für Technische Informatik
und Kommunikationsnetze

Betreuer: Dr. Beat Pfister und Michael Gerber

Verantwortlicher: Prof. Dr. L. Thiele

Inhaltsverzeichnis

1 Zusammenfassung

In folgender Arbeit geht es um das Generieren von Mustern für die Spracherkennung. Dabei sollten die erzeugten Muster möglichst robust gegen Störgeräusche und varierende Intensitäten, oder Aussprachen sein. Dazu wurden verschiedene Lösungsansätze entwickelt, die auf ihre Tauglichkeit untersucht wurden. Mit den entwickelten Mustergenerierungen und angepassten Parameter für den Detektor, konnte eine Verbesserung der Fehlerraten erreicht werden. Zusätzlich wurde ein Verfahren entwickelt, welches die Abweichung vom optimalen Schnitt verkleinern und so die Verschwendung von Daten verhindern konnte. Beim Mustervergleich kam der Ansatz der dynamischen Programmierung mittels DTW (Dynamic Time Warping) zum Einsatz. Die genaue Aufgabenstellung befindet sich im Anhang B.

2 Einleitung

Die meisten Menschen sind in ihrem täglichen Leben bereits mit Spracherkennung in Berührung gekommen. Sei dies durch Sprachsteuerung, oder sogenannte speech-to-text Systeme, wie sie in Diktiersoftware verwendet wird. Selbst kleinste Geräte wie Mobiltelefone können heute mittels Sprache gesteuert werden, was bis vor wenigen Jahrzehnten noch undenkbar war. Beim Spracherkenner muss zwischen verschiedenen Typen unterschieden werden. Dabei erstreckt sich der Rahmen vom einfachen Einzelworterkenner, bis hin zum komplexen kontinuierlichen Spracherkenner, die sich natürlich in Bezug auf das Hintergrundwissen einer Sprache, wie etwa Grammatik und Vokabular, immens unterscheiden. Weiter muss differenziert werden zwischen sprecherabhängigen, -unabhängigen, sowie -adaptiven Systemen. Je nach Anwendung kommt ein anderer Ansatz zum Zug. Nebst dem in dieser Arbeit verwendetem DTW Ansatz, gibt es andere Systeme, die mit Hidden Markov Modellen[2] , oder neuronalen Netzen[3] arbeiten.

Auf den ersten Blick scheint DTW in der heutigen Zeit ein wenig veraltet zu sein, doch gibt es weiterhin Anwendungen, die vorzugsweise mittels DTW implementiert werden. Als Beispiel könnte hier die Steuerung einer Maschine ins Feld geführt werden, wobei die Spracherkennung das Interface zwischen Mensch und Maschine überbrücken würde. Die Steuerung würde mittels einfachen Befehlen geschehen. Damit sind einzelne Wörter resp. Kommandos gemeint, die isoliert gesprochen werden. DTW hat gegenüber anderen Ansätzen den Vorteil, dass es sprachübergreifend ist. Das Befehlsvokabular kann vom Benutzer in seiner Muttersprache ausgewählt und aufgenommen werden. Es ist kein Hintergrundwissen zu dieser Sprache notwendig, wie etwa bei HMM oder NN, da DTW mittels Mustervergleich arbeitet. Die Befehle sind, falls keine Zustände innerhalb der Maschine definiert werden, unabhängig voneinander. Dieses Szenario entspricht dem sprecherabhängigen Fall, der in dieser Arbeit untersucht wurde. Da der Benutzer das Vokabular selber auswählen und entsprechend aufnehmen kann, wird das System bestmöglichst diesem Benutzer angepasst.

Besonders praktisch ist dies in Situationen, in denen man beschäftigt ist und seine Aufmerksamkeit nur beschränkt dem System widmen kann. Ein Autofahrer sollte mit beiden Händen das Lenkrad umfassen und auch die Augen auf die Strasse richten. Trotzdem werden die Möglichkeiten des Menschen eigentlich nur ungenügend ausgelastet, da er seine Stimme benutzen könnte, um einer anderen Tätigkeit nachzugehen. Ein Chirurg im OP Saal macht genau dies, doch sind es keine Maschinen, die ihm assistieren. Diese Abläufe könnten beschleunigt werden, wenn der Chirurg die Geräte direkt mit seiner Stimme steuern könnte. Wie man sieht sind die denkbaren Einsatzmöglichkeiten sehr vielfältig. Besonders in Verbindung mit dem Telefon kann die Spracherkennung ein breites Gebiet wie etwa automatisierte Auskunftsdienste, also Telefonbuch, Fahrplanauskunft oder andere automatisierte Dienste wie Teleshopping und -banking abdecken.

Trotz all diesen vielversprechenden Einsatzmöglichkeiten ist der grosse Durchbruch der Spracherkennung bis heute ausgeblieben. Zwar gibt es seit mehreren Jahren Diktiersysteme, die eine Umwandlung von speech-to-text realisieren und wohl jedem Menschen überlegen sind,

[2]HHM ist ein stochastischer Ansatz, der durch Hintergrundwissen einer Sprache genauere Entscheidungen ermöglicht. Durch Zustände und Graphen werden die Wörter miteinander verbunden.

[3]Model aus der künstlichen Intelligenz

doch die hohen Erkennungsfehlerraten sind dafür verantwortlich, dass sich Spracherkenner bis heute noch nicht richtig durchsetzen konnten. In dieser Arbeit wurde deswegen versucht diesem Problem nachzugehen.

3 Grundlagen

In den folgenden Abschnitten sollen nun die Grundlagen erarbeitet werden, um dem Leser das benötigte Wissen zu vermitteln, welches in der späteren Diskussion benötigt wird.

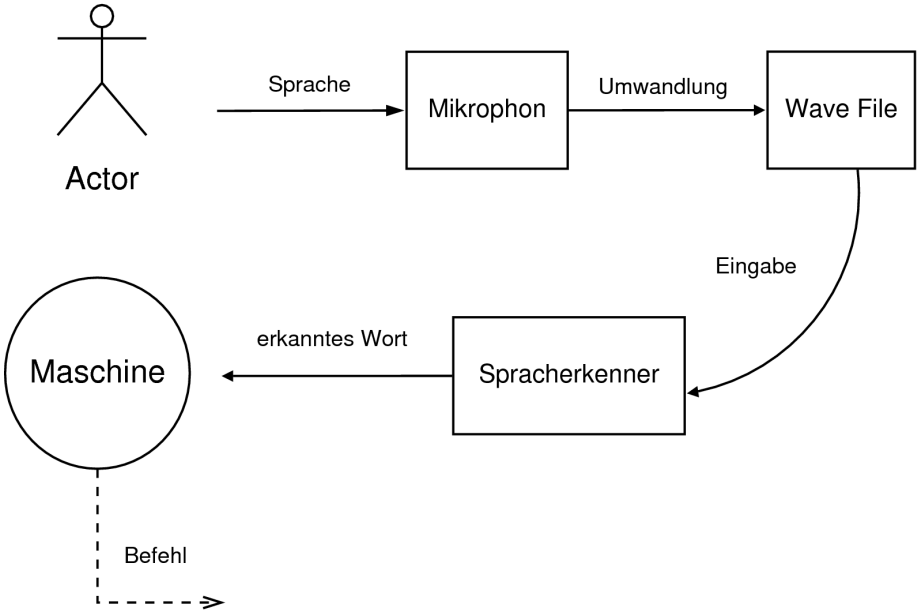

Abbildung 1: *Schmematischer Ablauf einer Sprachsteuerung*

3.1 Das Sprachsignal und dessen Merkmale

Beim Sprechen werden durch den Sprachapparat Schallwellen produziert, die wir als Sprache wahrnehmen. Durch einen elektroakustischen Wandler (Mikrophon) können diese in ein Sprachsignal umgewandelt werden.

3.1.1 RMS

Ein wichtiges Kriterium, um das Sprachsignal detektieren zu können ist die Intensität der Laute, oder der RMS-Wert (root mean square). Für uns hören sich die Laute innerhalb eines Wortes alle gleich laut an, doch die Intensitäten können grosse Schwankungen aufweisen. Um den RMS zu berechnen, werden die Amplituden des Signals innerhalb eines Fensters quadriert und aufsummiert. Dies entspricht der Leistung dieses Signalabschnitts. Zuletzt wird daraus die Wurzel gezogen, um die Energie des Abschnittes zu erhalten. Sei *Win* die Länge des Analysefensters und *n* der Index des Abschnittes, dann definiert sich der RMS zu

$$RMS(n) = \sqrt{\sum_{m=1}^{Win} s((n-1) * Win + m)^2}$$

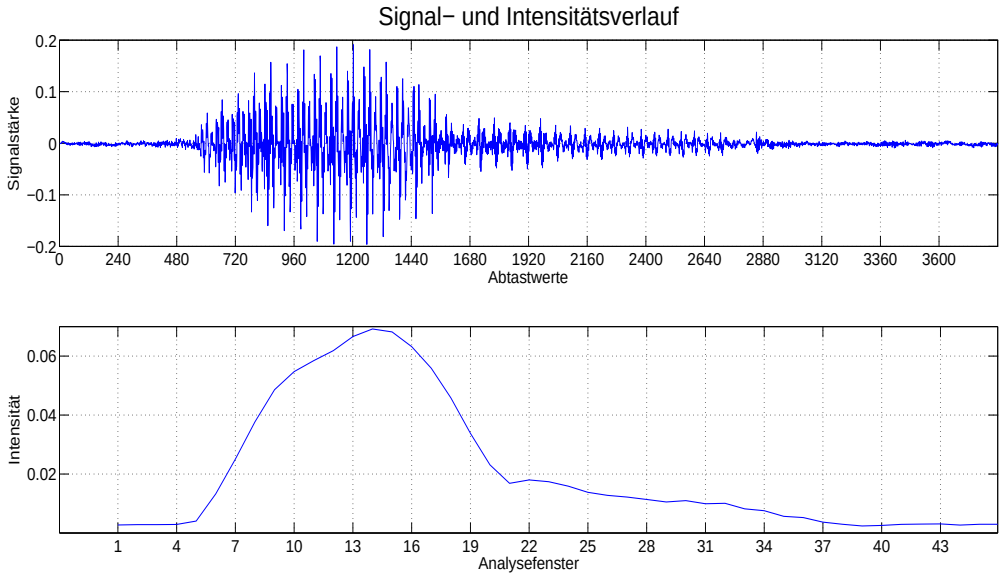

Abbildung 2: *Darstellung der Signalstärke und -intensität*

3.1.2 Mel-Cepstren

Die sogenannten MFCC (Mel frequency-cepstral coefficients) werden in der Spracherkennung oft als Merkmalsvektoren herangezogen. Sie haben ihre Grundlagen im Bereich, der Psychoakkustik, wo man mit Untersuchungen und Experimenten wichtige Erkenntnisse zur subjektiven Wahrnehmung gewinnen konnte.

Es konnte gezeigt werden, dass die empfundene Höhe eines Tones (Tonheit) nicht linear zur Frequenz, sondern logarithmisch über einen weiten Frequenzbereich verläuft. Aus diesem Grund wurde für die Tonheit die Mel-Skala eingeführt, welche die subjektive Tonhöhe angibt. Zusätzlich erkannte man, dass unser Gehör Schallintensitäten in einem schmalen Frequenzbereich, in sogenannte Frequenzgruppen, zusammenfasst. Diese Gruppen treten im gesamten Hörbereich auf. Deshalb transformiert man das Spektrum von einer linearen Skala in eine Mel-Skala. Folgende Skalen-Transformation ist dabei üblich

$$h(f) = 2595 * \log_{10}(1 + \frac{f}{700Hz}).$$

f entspricht dabei der Frequenz und *h* der Tonheit in Mel.Für gewöhnlich wird das Mel-Spektrum durch eine sogenannte Mel-Filterbank aus dem DFT-Spektrum gewonnen.

$$\breve{S}_j = \sum_k X(k) \cdot \breve{H}_j(k)\,, \qquad 1 \leq j \leq J.$$

Das Mel-Cepstrum berechnet sich nun noch wie folgt mit der Anzahl Filter in der Mel-Filterbank *J* und der Anzahl cepstraler Koeffizienten *D*.

$$\breve{c}(m) = \frac{1}{J} \sum_{j=1}^{J} (\log \breve{S}_j) \cos[m \cdot (j - \tfrac{1}{2})\frac{\pi}{J}]\,, \qquad 0 \leq m \leq D.$$

3.2 DTW

Damit ein Spracherkenner entscheiden kann, um welches Wort es sich handelt, müssen bereits Informationen in Form eines Vokabulars vorliegen. Dieses Vokabular steht in Mustern für jedes Wort zur Verfügung und wurde aus anderen Sprachsignalen erstellt, um eine möglichst gute Mittelung zu erreichen. Nun wird das zu bestimmende Wort eingelesen und umgewandelt, das heisst, die zu vergleichenden Merkmale werden extrahiert. In einem nächsten Schritt werden diese Merkmalsvektoren mit vorliegenden Mustern verglichen und auf ihre Ähnlichkeit hin überprüft. Das Muster, welches am ähnlichsten ist, also die geringste Abweichung aufweist, wird dann erkannt. Wie genau diese Abweichung definiert ist, wird im späteren Verlauf erläutert.

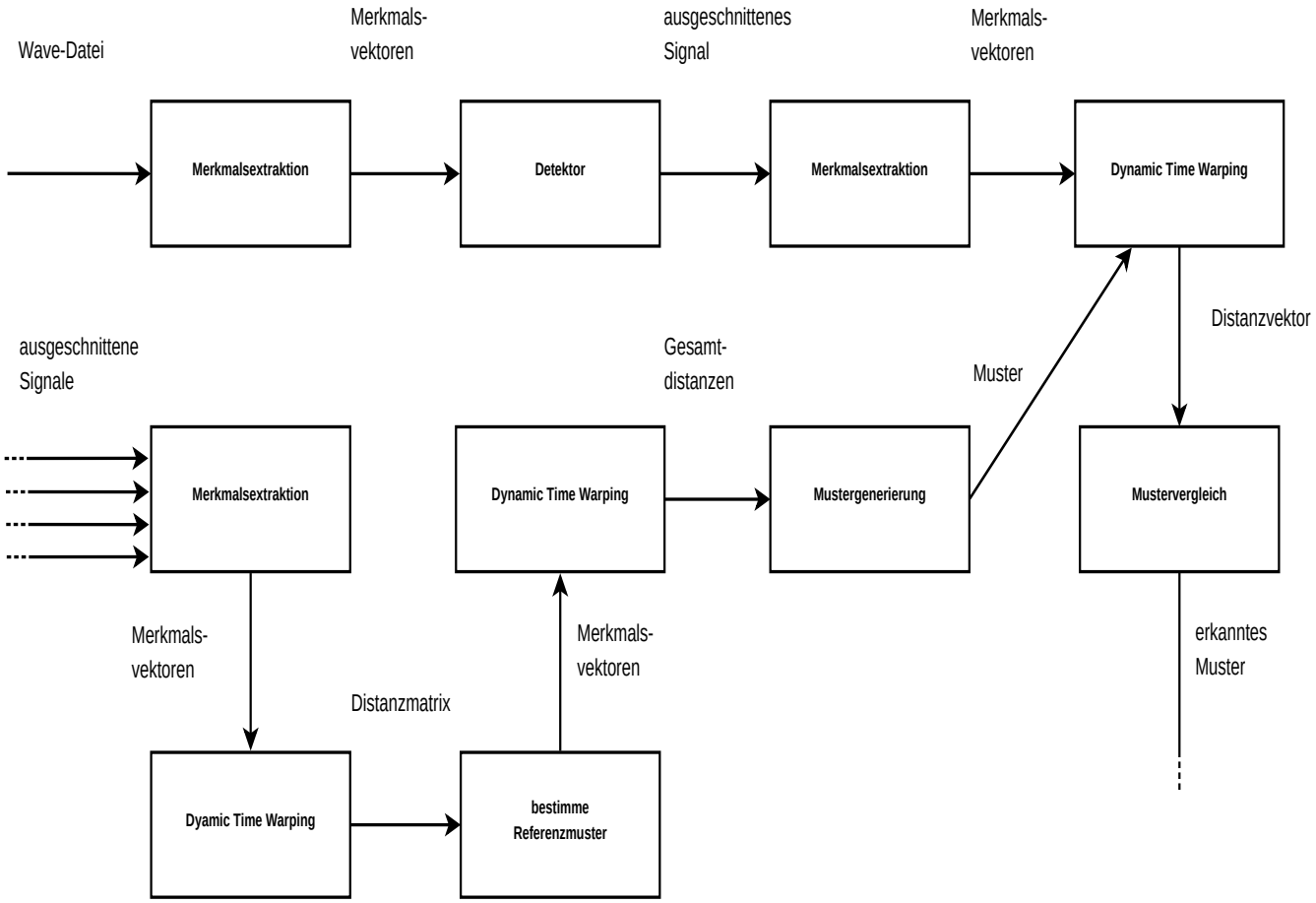

Abbildung 3: *Ein Spracherkenner im Überblick*

Wenn ein Wort mehrmals ausgesprochen wird, fällt einem vielleicht gar nicht auf, dass die Aussprache relativ grosse Differenzen aufweist. Professionelle Sprecher haben diese Variationen gut unter Kontrolle, doch ein ungeübter Sprecher wird das Wort jedes mal ein wenig anders betonen. Es sind hauptsächlich prosodische Unterschiede auszumachen. Die Signale weichen meist in Intensität, Dauer, Rhythmus, sowie Sprechmelodie voneinander ab. Da jedes Signal zum dazugehörigen Wort gemappt werden sollte, dürfen diese Unterschiede nicht ins Gewicht fallen.

Das Mel-Cepstrum wird intensitätsunabhängig, wenn der nullte Koeffizient weggelassen wird und die mel-cepstralen Koeffizienten mit tiefem Index beschreiben ungefähr das Spektrum, womit die Frequenzabhängigkeit umgangen werden kann. Durch eine Kurzzeitanalyse aus den zu vergleichenden Signalen kann nun je eine Sequenz von Mel-Cepstren ermittelt werden, womit Intensität und Grundfrequenz nicht als Kriterium für das Messen der Ähnlichkeit ins Gewicht fallen. Somit ist die berechnete Distanz unabhängig von diesen beiden Faktoren.

3.2.1 Warping-Kurve

Um den Rhythmus oder die unterschiedliche Sprechdauer neutralisieren zu können, muss das Signal zeitlich gemittelt werden, damit überhaupt ein Vergleich möglich ist. Von den zwei Signalen s_x und s_y werden die Sequenzen der Mel-Cepstren ermittelt.

$$\mathbf{X} = \mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_{T_x} \qquad \texttt{mit} \quad \mathbf{x}_i = \begin{pmatrix} \breve{c}_{x,i}(1) \\ \breve{c}_{x,i}(2) \\ \vdots \\ \breve{c}_{x,i}(D) \end{pmatrix}$$

und

$$\mathbf{Y} = \mathbf{y}_1\mathbf{y}_2 \ldots \mathbf{y}_{T_y} \qquad \texttt{mit} \quad \mathbf{y}_j = \begin{pmatrix} \breve{c}_{y,j}(1) \\ \breve{c}_{y,j}(2) \\ \vdots \\ \breve{c}_{y,j}(D) \end{pmatrix}$$

Die euklidische cepstrale Distanz berechnet sich nun zu

$$d(\mathbf{x}_i, \mathbf{y}_j) = \sqrt{\sum_{m=1}^{D} [\breve{c}_{x,i}(m) - \breve{c}_{y,j}(m)]^2} \doteq d(i,j).$$

Obige Gleichung drückt also den Unterschied zwischen dem *i*-ten Element von $\mathbf{X}$ und dem *j*-ten Element von $\mathbf{Y}$ aus. Diese lokale Distanz *d(i,j)* gibt Ausschluss darüber, ob in den beiden Mustern derselbe Laut vorliegt. Falls die lokale Distanz gross ist, heisst dies, dass verschiedene Laute vorliegen und falls die Distanz klein ist, kann man von gleichen Lauten ausgehen. Nehmen wir nun an, dass die beiden Signale die gleiche Länge ($T_x = T_y$) haben. So lässt sich durch Summation der lokalen Distanzen *d(k,k)* über alle $k = 1, \ldots, T_x$ die Gesamtdistanz zwischen den beiden Mustern $\mathbf{X}$ und $\mathbf{Y}$ berechnen.

Es kann jedoch vorkommen, dass sich Sprachsignale eines identischen Wortes in ihrer Länge deutlich unterscheiden. Somit varieren auch die Laute in ihrer Dauer. Um dieses Problem beheben zu können und die beiden Signale trotzdem miteinander vergleichen zu können, müssen die Signale zeitlich aneinander angepasst werden. Damit die lokalen Distanzen zwischen den beiden Signalen möglichst klein werden, ist eine nicht lineare Verzerrung der Zeitachsen nötig. Diese wird auch dynamische Zeitanpassung genannt. Natürlich wird diese Verzerrung nur bei Äusserungen desselben Wortes zu einer kleineren Gesamtdistanz führen.

Das Ziel der dynamischen zeitlichen Anpassung ist nun die Zeitachse lokal so zu strecken oder stauchen, dass eine optimale Übereinstimmung realisiert wird. Dies geschieht mittels DTW (dynamic time warping), wobei die beiden Sequenzen von Merkmalsvektoren einander in einem Koordinatensystem gegenüber gestellt werden. Es wird nun versucht einen Pfad zu finden, sodass die Gesamtdistanz minimiert wird. Die entstandene Kurve wird Warping-Kurve genannt.

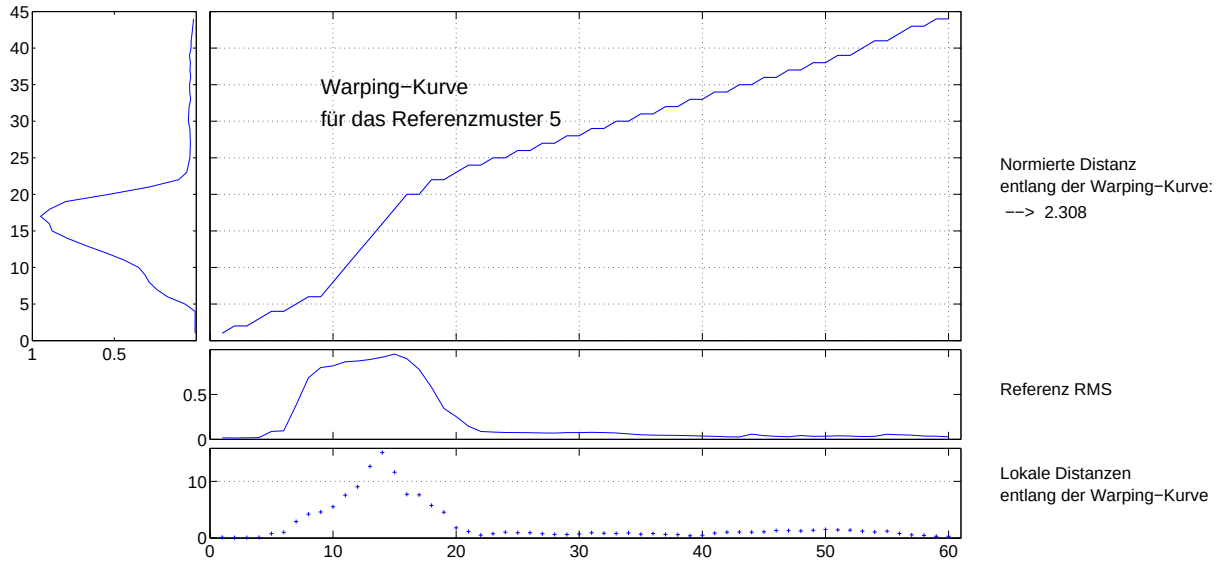

Abbildung 4: *Optimaler Pfad der Warping-Kurve*

Die Warping-Kurve an der Stelle *k* ist durch die Indizes der einander zugeordneten Vektoren gegeben.

$$\phi(k) = (i(k), j(k)), \qquad 1 \leq k \leq T.$$

Somit wird die lokale Distanz an der Stelle *k* der Warping-Kurve zu

$$d(\phi(k)) = d(i(k), j(k)).$$

Die Gesamtdistanz hängt ausserdem von der Länge der Warping-Kurve ab. Diese Gewichtung *w(k)* definiert sich wie folgt:

$$w(k) = i(k) - i(k-1) + j(k) - j(k-1), \qquad 1 < k \leq T$$
$$w(1) = 1$$

Somit ergibt sich die Gesamtdistanz zu

$$D_\phi(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{T} d(\phi(k)) w(k).$$

Nun wird aus allen möglichen Kurven die mit der kleinsten Gesamtdistanz ausgewählt

$$D(\mathbf{X}, \mathbf{Y}) = \min_{\phi} D_{\phi}(\mathbf{X}, \mathbf{Y}).$$

3.2.2 Randbedingungen

An die dynamische Zeitanpassung werden für den Fall des Mustervergleichs einige Anforderungen gestellt, damit keine Abschnitte übersprungen, oder der Verlauf umgedreht wird. Die zeitliche Abfolge des Signals muss, trotz zeitlicher Anpassung, gleich bleiben, da für unsere Anpassung Sprachsignale verwendet werden. Folgende Randbedingungen muss die Warping-Kurve erfüllen:

- **Monotonie:** Die Merkmalsvektoren müssen ihre Reihenfolge in der Sequenz beibehalten. Es dürften keine Vektoren vertauscht werden, da dies das Signal verändern würde, was nicht erlaubt ist.
- **Lokale Kontinuität:** Die Schrittgrössen auf der Kurve sind begrenzt, da nur kurze Abschnitte übersprungen werden dürfen, aber keine Laute.
- **Anfangs- und Endpunktbedingungen:** Man geht davon aus, dass Start- und Endpunkt in beiden Signalen korrekt sind. Somit müssen diese in der Warping-Ebene zwingend aufeinander fallen.

Obige Randbedingungen hängen von den erlaubten Pfaden, die man auf der Warping-Ebene definiert, ab. Zusammen mit einer Gewichtung für jeden möglichen Pfad werden sie zu einem Satz von Pfaderweiterungen zusammen gefasst. Pfaderweiterungen können so definiert werden, dass es erlaubt ist einzelne Merkmalsvektoren zu überspringen. Gewichtungen stellen sicher, dass ein Pfad nicht deswegen gewählt wird, weil er zu kürzeren Warping-Kurven führt. Eine Gewichtung von 2 würde verhindern, dass ein Pfad, der ein Merkmalsvektor überspringt, nur wegen der Kürze als optimal empfunden wird. Es wird unterschieden zwischen symmetrischem und asymmetrischem Warping. Falls **X** und **Y** vertauscht werden können, bei gleich bleibender Gesamtdistanz, spricht man von symmetrischem Warping. Bei asymmetrischem Warping führen die unterschiedlichen Pfade bei Vertauschung zu einer anderen Gesamtdistanz. Wir wollen nun klären wie der optimale Pfad entsteht.

3.2.3 Algorithmus

Für jeden Punkt innerhalb der Ebene wird die Gesamtdistanz berechnet. Diese berechnet sich relativ leicht, wenn man von den Vorgängerpunkten die akkumulierte Distanz kennt. Es wird derjenige Vorgänger gewählt, der in der geringsten Gesamtdistanz im Endpunkt resultiert. Folgende Gleichung drückt dies aus, wobei *p(m)* der Vektor der m-ten Pfaderweiterung ist.

$$D_A(i,j) = \min \left\{ \begin{array}{c} D_A((i,j) - p(1)) + wp(1)d(i,j) \\ D_A((i,j) - p(2)) + wp(2)d(i,j) \\ \vdots \\ D_A((i,j) - p(M)) + wp(M)d(i,j) \end{array} \right\}$$

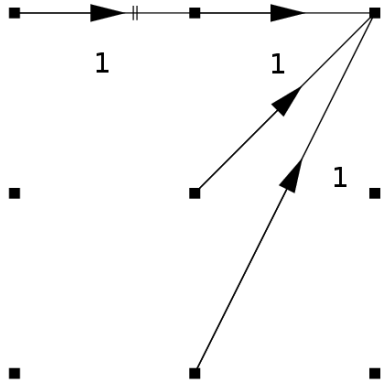

Abbildung 5: *Verwendete Pfaderweiterungen und Gewichtungen*

Der DTW Algorithmus lässt sich wie folgt beschreiben:

- Initialisierung: Es werden alle lokalen Distanzen $d(i, j)$ berechnet. Zudem wird $D_A(1, 1) = d(1, 1)$.
- Rekursion: Für alle bekannten Vorgängerpunkte werden die Gesamtdistanzen der Nachfolger berechnet. Weiter wird für jeden Punkt der Index der optimalen Pfaderweiterung in $\psi(i, j)$ abgespeichert, um schlussendlich die Warping-Kurve zu zeichnen. Dieser Schritt wird so lange wiederholt, bis der Algorithmus am Punkt (T_x, T_y) angekommen ist.
- Backtracking: Die optimale Kurve kann nun aus ψ mittels Backtracking vom Endpunkt extrahiert werden.
- Normierung: Nun muss die Distanz noch normiert werden, da entlang der Kurve die verschiedenen Pfade unterschiedlich gewichtet wurden, um gewisse Pfade nicht zu bevorzugen. Somit erlischt auch die Abhängigkeit der Gesamtdistanz von der Länge.

Der Spracherkenner muss nun das Eingangssignal mit allen vorhandenen Muster überprüfen. Die normierten Distanzen werden dann für jedes Muster gespeichert und am Schluss wird auf das Muster mit der minimalen Distanz entschieden:

$$i = \underset{j=1,\ldots,|V|}{\operatorname{argmin}} D_{\phi N}(\mathbf{X}, \mathbf{Y}^{(j)}) \; .$$

3.3 Stand der Technik

Recherchen in Fachliteratur, Journals und alten Arbeiten haben kaum brauchbare Informationen bzw. Ansätze geliefert, an denen man hätte anknüpfen können. Neuere Methoden der Spracherkennung wie etwa HMM oder NN scheinen in der aktuellen Forschung einen weitaus grösseren Stellenwert zu geniessen als etwa DTW. Es war so, dass in der gängigen Literatur vor allem neuartige Möglichkeiten zur VOD (voice activity detection) und Signalverstärkung thematisiert wurden, in Bezug auf mögliche Verbesserungen im Spracherkenner. Die Mustergenerierung wurde in diesen Ansätzen kaum gestreift und wenn, dann nur die Standardmethode, das Muster aus mehreren Signalen zu mitteln. Die Implementation eines solchen Ansatzes hätte den Rahmen dieser Arbeit, sowie auch die vorhandenen Vorkenntnisse gesprengt, da zuerst die

Grundlagen und Theorie der Spracherkennung erarbeitet werden mussten. Um nicht Gefahr zu laufen das eigentliche Ziel der Arbeit aus den Augen zu verlieren, wurde mit den zur Verfügung gestellten Funktionen gearbeitet. Vor diesem Hintergrund erscheint es sinnvoll die Möglichkeiten zur Verbesserung innerhalb eines Spracherkenners über die Generierung der Muster zu überprüfen, da auf diesem Gebiet wenig geforscht wurde. Zusätzlich ist DTW selbst heute noch ein interessanter Algorithmus, da er einerseits sprachunabhängig und andererseits gut für die Steuerung einer Maschine geeignet ist.

4 Aufbau des Spracherkenners

In diesem Abschnitt sollen die einzelnen Komponenten des Spracherkenners besprochen werden und die Anordnung der Versuche beschrieben werden.

4.1 Sprachaufnahmen in Rohfassung

Insgesamt wurden von 15 verschiedenen Sprechern die Ziffern $0, 1, \ldots, 9$, sowie die *2* als zusätzliche Variation, je 5 mal repetiert und als wave Dateien gespeichert. Die Ziffern wurden sowohl von Männern, als auch Frauen gesprochen. Die Signale liegen in Rohfassung vor, das heisst, sie wurden weder gefiltert, noch bearbeitet. Aus diesem Grund ist damit zu rechnen, dass einige Störgeräusche wie Klick-, Atem- und Schluckgeräusche das Signal überlagern. Weiter sind die Aufnahmen unterschiedlich stark verrauscht, sodass wir diese Faktoren berücksichtigen müssen. Es wurden keine professionellen Sprecher ausgewählt, um die Signale aufzunehmen, sodass man von realistischen Grundvoraussetzungen ausgehen darf. Entsprechend stark sind die Schwankungen in den Aussprachen, die man beobachten kann. Ein weiteres Phänomen, das auftaucht ist die Tatsache, dass innerhalb eines Repetitionsblocks der Dialekt teilweise stärker oder schwächer durchschlägt. So können mittels Clustering zwei Gruppen von Signalen gebildet werden innerhalb einer Repetition. Alle diese Voraussetzungen erschweren natürlich die Arbeit des Spracherkenners.

4.2 Merkmalsextraktion

Aus den Signalen wurden die Root Mean Square Verteilungen und Mel-Cepstren extrahiert, die wir in Kapitel 2 besprochen haben. Diese Merkmale haben sich bewährt für den sprecherabhängigen Fall, wo Signale vom gleichen Sprecher miteinander verarbeitet und verglichen werden. Für den sprecherunabhängigen Fall müsste man hier weitere Merkmale berechnen, da diese aufgrund der verschiedenen Frequenzen zwischen Frauen- und Männerstimmen schlechte Resultate liefern würden. Da die Signale kaum miteinander zu vergleichen wären, würde die Fehlerrate für solch einen Fall verhältnismässig hoch ausfallen.

4.3 Start- und Endpunkt Detektor

In einem nächsten Schritt werden die Signale zurecht geschnitten, indem versucht wird das reine Sprachsignal zu extrahieren. Um zu entscheiden wo das Sprachsignal beginnt, wurde der Intensitätsverlauf des Signals (RMS) beobachtet. Als Nächstes wurde ein Histogramm des Energieverlaufs erstellt, um den Threshold in Abhängigkeit davon zu definieren. Mit diesem Threshold vergleicht man nun das Signal. Man startet vorne und hinten im Signal und für die ersten Werte, die über dem Threshold liegen, definiert man die Start- und Endpunkte. Zusätzlich fügt man vorne und hinten je einen kleinen Puffer hinzu, um die Randbedingungen des DTW Algorithmus ein wenig zu lockern. Durch diesen Puffer müssen Start- und Endpunkte nicht mehr auf den gleichen Punkt zu liegen kommen, sondern haben noch ein wenig Spielraum. Dieser Detektor ist sehr anfällig für Störgeräusche, da ,aufgrund seiner Implementation, allfällige Störgeräusche am Anfang oder Ende des Signals, als Sprache ins Signal miteinfliessen. Selbst wenn die Äusserung schon lange abgeschlossen ist, aber der Sprecher zum Schluss ins Mikrofon atmet, detektieren wir den Endpunkt nach dem Atemgeräusch. Für diesen Fall erhalten wir einen hohen Rauschanteil im Signal und somit eine grosse Distanz zu den anderen Signalen dieses Wortes.

4.4 Vergleichsaufbau

Es wurde ein Testablauf für die Experimente vorgegeben, in welchem jeweils aus den 5 Repetitionen ein Signal als Input ausgewählt und die restlichen 4 zur Mustergenerierung verwendet wurden. Dies wurde über alle Ziffern iteriert, um die Muster für die einzelnen Ziffern zu erstellen. Anschliessend galt es die Inputs mit allen Mustern zu vergleichen und entsprechend auf das ähnlichste zu entscheiden. Dies wurde zuerst über alle möglichen Repetitionen und dann über alle Sprecher iteriert. Ein Block enthält für jedes Wort ein Muster. Das erste Muster vom erstem Block, bis hin zum letzten Muster des letzten Blocks sehen dabei wie folgt aus:

Input: [Sprecher1.Zahl1.Repetition1]	Muster: [Sprecher1.Zahl1.Repetition2-5]	1 Block
⋮	⋮	⋮
Input: [Sprecher15.Zahl11.Repetition5]	Muster: [Sprecher15.Zahl11.Repetition1-4]	75 Block

4.5 Mustergenerierung

Wird nun ein Signal als Input Signal ausgewählt, entsteht aus den vier verbleibenden Repetitionen ein Muster für die jeweilige Ziffer. Unter diesen 4 verbleibenden Signalen wird die Gesamtdistanz untereinander berechnet und in eine Distanzmatrix geschrieben. Anschliessend addiert man die erhaltenen Distanzen für jedes Signal auf. Das Signal mit der geringsten Summe weist die grösste Ähnlichkeit zu den verbleibenden drei Signalen auf und wird deshalb als Referenzsignal festgelegt. Nun werden die Merkmalsvektoren des Referenzsignals mit denen des zeitlich angepassten Signals addiert. Durch die Warping-Kurve entstehen die zeitlich angepassten Signale, deren RMS und MFCC addiert werden. Die erhaltenen Werte müssen nun noch normiert werden.

5 Methoden

Aufgrund der Variation der Äusserungen und der Fehler, die man mit dem Start- und Endpunkt Detektor macht, sind nicht alle Äusserungen gleich gut zu gebrauchen. Einige mögen so verrauscht oder gestört sein, dass sie gänzlich unbrauchbar sind. Die Muster sollten robust gegen Störgeräusche sein und möglichst alle vorhandenen Informationen miteinfliessen lassen. Die Verschwendung von vorhandener Information sollte, wenn möglich, vermieden werden. Dieses Kapitel dient dazu die Ansätze zu dokumentieren, welche die Muster verbessern sollten und ihre Weiterentwicklung begründen.

5.1 Verwendete Muster

5.1.1 Mittelung der Signale

Diese Art der Mustergenerierung stellt den Standard dar und wurde als Startpunkt der Aufgabenstellung gegeben. Dabei wird aus allen vier Signalen, die zur Mustergenerierung verwendet werden, ein Referenzsignal bestimmt. In einem nächsten Schritt werden die anderen Signale an das Referenzsignal zeitlich angepasst. Später addiert man die MFCC und RMS vom Referenzsignal mit den zeitlich gemittelten MFCC und RMS der anderen Signale. Zuletzt teilt man diese Summe durch die verwendete Anzahl Signale, um die neuen Merkmalsvektoren zu normieren. Die Muster stehen nun bereit und können mit den Eingangssignalen des Spracherkenners verglichen werden.

5.1.2 Gewichtung der Signale

Die erste mögliche Verbesserung könnte man erreichen, indem man nicht alle Signale gleich stark gewichtet, sondern dies in Abhängigkeit von der Distanz zum Referenzsignal setzt. Dahinter steckt die Überlegung, dass Signale untereinander stark varieren können. Fehlerhafte Signale sind ebenfalls nicht auszuschliessen, sodass man den Einfluss dieser Signale ein wenig abschwächen könnte. Beispielsweise gewichtet man das Referenzsignal mit dem Faktor *x* und die 3 anderen Signale fliessen mit einem Faktor *1-x* in die Rechnung mit ein. Nun berücksichtigt man die Distanz zum Referenzsignal. Je grösser diese ist, desto weniger Anteil nimmt das Signal am verbleibenden Faktor. Für *x* wurden die Faktoren $\frac{1}{4}$ und $\frac{1}{2}$ verwendet, da die Signale neben dem Referenzsignal auch einen gewissen Einfluss haben sollten.

5.1.3 Selektion der Signale

Für sehr verrauschte, oder fehlerhafte Signale empfiehlt es sich unter Umständen diese wegzulassen. Man kann beispielweise ein Qualitätskriterium festlegen, welches die Signale zu erfüllen haben, damit sie in die Mustergenerierung miteinfliessen. Alle anderen Signale werden nicht berücksichtigt. Dies bedeutet zwar gleichzeitig ein Verlust an Information, aber meist tragen diese Signale wenig Brauchbares zur Generierung bei. Die Idee dahinter wäre, dass schlechte Signale das Muster nur weiter verschlechtern. Als Kriterium sollten die Signale eine Distanz < *3* zum Referenzsignal aufweisen. Die *3* schliesst die meisten Symbole mit ein, sodass im Normalfall nur stark verrauschte Signale ausgeschlossen werden. Dies wurde durch Überprüfung der Distanzen zwischen den Signalen zur Mustergenerierung und anschliessender Kontrolle ersichtlich. Kontrolliert wurden die schlechten Signale, in dem man sich das Signal anhörte.

5.1.4 2 Cluster

In einigen Fällen können aus den vier Signalen 2 Gruppen gebildet werden, die untereinander sehr nahe beieinander liegen. Diese Clustergenerierung wird angewendet, weil es vorkommt, dass ein Sprecher nach zweimaliger Repetition einer Ziffer plötzlich in eine andere Betonung abgleitet. Gerade bei Dialekten kann dies öfters vorkommen, weswegen untersucht werden soll, ob der Vergleich mit mehreren Mustern für eine Ziffer möglicherweise die Fehlerrate senken kann.

5.1.5 Bestes Signal

Dieser Ansatz ist im eigentlichen Sinn keine Mustergenerierung, denn jedes Signal wird als Muster ausgewählt und mit dem Input verglichen. Das Muster mit der tiefsten Abweichung wird dann als erkannt ausgegeben. Die Idee dahinter geht davon aus, dass bei ähnlichen Wörtern, wie in unserem Fall die *Zwei* und *Drei*, das beste Muster von der richtigen Ziffer stammen müsste. Möglicherweise wird wegen anderen fehlerhaften Signalen das Muster verschlechtert und deswegen falsch bestimmt.

5.2 Vorgehen und Ansätze

Nachdem der Code für den Testdurchlauf implementiert war, wurde sogleich mit den ersten Experimenten begonnen. Es zeigte sich relativ schnell, dass der Spracherkenner offenbar Probleme mit einigen Ziffern hat und diese Wörter wiederholt Fehler produzierten. Durch Betrachten der Warping Kurven wurde klar, dass die Signale nicht sauber ausgeschnitten wurden. Das anschliessende Anhören der Signale bestätigte dies und zwar wurden vielfach Laute abgeschnitten. Ganz offensichtlich waren die Parameter des Detektors für einige Wörter im Vokabular zu hoch eingestellt. Bei den Plosivlauten am Ende der Äusserungen war dies besonders auffallend. Auch Frikative zu Beginn einer Äusserung wurden teilweise nicht erkannt, doch nicht in der Häufigkeit, wie etwa das *T* in der Ziffer *Acht* abgeschnitten wurde. Ein Problem vom Detektor ist die fehlende Differenzierbarkeit, da man eigentlich für die *Acht* den Endpunktthreshold, im Vergleich zum Startpunktthreshold, ein bisschen weiter runter setzen müsste. Dies sollte man generell so handhaben, da der Intensitätsverlauf meist relativ stark ansteigt und gegen Ende hin sanft abklingt. Ferner kann man nicht einfach die Parameter des Detektors wegen einem einzelnen Wort verändern, da dies möglicherweise für die anderen Wörter alles andere als optimal wäre. Aus diesem Grund wären 2 Thresholdwerte von Vorteil, um das Signal möglichst gut ausschneiden zu können. Insgesamt möchte man eine gute Erkennungsrate erreichen, sodass man wegen der fehlenden Differenzierbarkeit Parameter finden muss, die mit den meisten Wörter vernünftige Ergebnisse liefern. Die Komplexität des Spracherkenners erfordert diverse Testdurchläufe, um die optimalen Parameter für das entsprechende Vokabular zu finden. Dabei wurden nacheinander Start- und Endpunktthreshold verkleinert, bis eine Verschlechterung der Fehlerrate auftrat. Den besten Parameter behielt man und wechselte zum verbleibenden Parameter mit dem gleichen Vorgehen.

5.3 Filter

Bereits früh in der Arbeit wurde versucht, mittels Filter, Störgeräusche rauszufiltern, was sich als schwieriges und zeitaufwändiges Unterfangen herausstellte und nicht weitergeführt wurde.

Dabei wurde versucht mit Bandpässen das Rauschen zu filtern. Problematisch war vor allem, dass immer auch Anteile des Sprachsignals damit gefiltert wurden. Ebenfalls von Vorteil wäre es, gewisse Frequenzen innerhalb des Spektrums zu verstärken, um so möglicherweise intensitätsärmeren Lauten über die Grenzschwelle zu verhelfen. Hier gibt es verschiedene Filter, die oft in der Spracherkennung zum Einsatz kommen.

5.3.1 Präemphase-Filter

Unsere aufgezeichneten Signale weisen eine Tiefpasscharakteristik aufgrund des menschlichen Vokaltrakts auf. Das von den Stimmlippen erzeugte Signal weist viel stärkere Anteile der tiefen Frequenzkomponenten, als die der hohen auf. Diese unerwünschte Modifikation des Signals wird durch den Präemphase-Filter behoben. Dieses Filter entspricht einem Hochpass und soll die intensitätsärmeren Laute verstärken. Generell sind die Formanten der hohen Frequenzen tiefer, was problematisch ist für unseren Detektor. Deshalb wird dieses Filter verwendet, um so möglicherweise Fehler im Detektor zu verhindern. Die Übertragungsfunktion H_P definiert sich wie folgt:

$$H_P = 1 - 0.98z^{-1}$$

Zusätzlich wurde die Abtastfrequenz von *8kHz* auf *16kHz* erhöht, da sich die Mel-Spektren einiger Frikative erst oberhalb von *4kHz* unterscheiden.

5.3.2 Wiener Filter

Das Wiener Filter wird in der Signalverarbeitung verwendet, um Rauschen zu filtern. Mit anderen Worten: Das Wiener Filter verbessert das SNR (Signal to Noise ratio). Damit es angewendet werden kann, müssen Signal wie auch Rauschen stochastischen Prozessen mit bekannten Verteilungen ähneln. Der quadratische Fehler wird dabei versucht zu minimieren [2] . Da das Wiener Filter ein interessanter Ansatz wäre, um das Signal zu verstärken, aber gleichzeitig der Aufwand verhältnismässig gross ist selber einen zu implementieren, wurde auf eine bereits vorliegende Version[4] zurückgegriffen. Diese verwendete Version ist sicherlich nicht auf unsere Signale abgestimmt, aber so ungefähr sollte man erkennen können, ob dies ein lohnenswerter Ansatz wäre, den man weiter verfolgen könnte. Auf weitere Optimierungen des Wiener Filters wurde allerdings verzichtet, da das Experimentieren bereits überaus zeitaufwändig war.

5.4 Detektor

Die meisten Probleme des Spracherkenners liegen dem Detektor zugrunde. Aus diesem Grund liegt es nahe, direkt am Detektor etwas zu verändern. Nachdem auch Experimente mit einem anderen einfachen VAD [5] keine Verbesserungen gebracht hatten, siehe Anhang A, wurde nach einer moderneren Variante in aktuellen Papers Ausschau gehalten. Es wurden Detektoren in Erwägung gezogen, die zusätzlich zur Intensität weitere Kriterien berücksichtigen, wie etwa die Autokorrelationsfunktion, oder die Zero Crossing Rate. Leider war zu erwarten, dass diese Implementationen aufwändig sein würden und wenig Zusatznutzen liefern, da gerade die stimmlosen Laute aufgrund der niedrigen Intensität unserem Detektor Mühe bereiten. Da sowohl die

[4] Wiener Filter: http://www.mathworks.com/matlabcentral/fileexchange/7673-wiener-filter

[5] Voice activity detection nach Rabiner & Sambur (1975):http://www.mathworks.com/matlabcentral/fileexchange/19298-speechcore

ZCR, wie auch die AKF nur zwischen stimmlos und stimmhaft unterscheiden können, kann man weiterhin nicht genau zwischen Rauschen und stimmlosen Lauten unterscheiden. Andere Implementationen hätten den Rahmen dieser Arbeit gesprengt und würden wohl genug Stoff für eine weitere Semesterarbeit auf dem Bereich der Sprachverarbeitung liefern. Die Tatsache, dass der zu Testzwecken verwendete Detektor nach Rabiner & Sambur ebenfalls die ZCR als Kriterium verwendet, aber keine Verbesserung der Erkennungsrate brachte, hat diesen Verdacht noch bestärkt.

Stattdessen wurde am existierenden Detektor festgehalten und versucht Optimierungen für diese Variante zu finden. Wie oben bereits angetönt, wurden 2 Schwellwerte definiert, die für Start- und Endpunktdetektion verschieden waren. Diese Parameter wurden relativ hoch angesetzt, um lieber schwach betonte Plosivlaute oder Frikative abzuschneiden, statt Rauschen als Äusserung zu detektieren. Während der Mustergenerierung wurden die Abstände zwischen den Signalen ermittelt und daraufhin eine Entscheidung gefällt. Sind die Abstände sehr weit auseinander, kann man davon ausgehen, dass entweder beim Ausschneiden ein Fehler gemacht wurde, oder aber das eine Signal stark verrauscht ist und deshalb eine grosse Distanz zu den anderen Signalen aufweist.

Da häufig Plosivlaute am Ende der Äusserung schlecht detektiert wurden, setzte man in einem ersten Schritt noch einmal beim Ausschneiden an. Der Endpunkt-Threshold wurde in gewissen Abschnitten verkleinert und daraufhin wurde mit dem neuen Signal noch einmal die Distanz berechnet. Bei einer kleineren neuen Distanz wiederholte man diesen Schritt so lange, bis keine Verbesserung mehr auftrat. Man merke sich jeweils den besten Endpunkt und speichere diesen ab. Falls sich keine positive Änderung ergeben würde, änderte man das Vorzeichen der Thresholdänderung. Dieser erste Verbesserungsschritt zeigte ihre Wirkung: Die Distanzen zu den optimalen Schnittpunkten konnten verbessert werden. Den selben Verbesserungsschritt führe man auch für die Startpunkte aus und speichere diese ab, falls sich Verbesserungen der Distanzen aufzeigen.

Damit konnte ein erster Erfolg verbucht werden, doch gab es nach wie vor Distanzen zwischen Signalen, die so gross waren, dass sie nur durch die Differenz der Länge zustande kommen konnten. Übersteigt die Länge eines Signals, die Länge des anderen um einen Faktor *C*, wird die Distanz automatisch zu 999, da der DTW Algorithmus für diesen Fall nicht ausgelegt wurde und demnach nicht mehr funktioniert. Dieser Faktor *C* hängt von den verwendeten Pfaderweiterungen ab. Möchte man solche grosse Differenzen innerhalb zweier Signale erlauben, müsste man die Pfaderweiterungen anpassen. Wie man sieht kann für die Pfaderweiterung in Figur 5 nur zwei mal nacheinander der horizontale Pfad gewählt werden, sodass spätestens danach die Diagonale gewählt werden muss. Damit können Differenzen der Länge *C=3* zwischen den Signalen überbrückt werden. Man kann davon ausgehen, dass ein grober Schnittfehler vorliegt, ein Teil der Äusserung abgeschnitten worden ist, oder starkes Rauschen zusätzlich als Signal detektiert wurde, falls keine Warping Kurve erstellt werden kann. Für diesen Fall reicht es meist nicht aus in kleinen Schritten den Threshold Wert zu erhöhen bzw. verkleinern, da nach wie vor die Differenz zu gross wäre. Aus diesem Grund wurde eine Fallunterscheidung implementiert, die für diese Fälle grössere Sprünge beim Verändern des Thresholds macht. Durch diese Änderung konnte die Schnittmenge weiter optimiert werden.

5.5 Angepasstes Vokabular

Zusätzlich zu den vorgestellten Verbesserungen könnte das Vokabular angepasst werden, sodass Wörter, die innerhalb des Vokabulars eine hohe Ähnlichkeit aufweisen, nicht zugelassen werden. Der vorgestellte Spracherkenner hat Mühe damit gewisse Laute zu erkennen und weist wegen den verwendeten Merkmale bei einigen Frikativen grössere Eigen- als Kreuzdistanzen auf. Gerade Laute mit gleichen Vokalen, die generell eine hohe Intensität aufweisen, können aus diesem Grund einem anderen ähnlichen Wort zugewiesen werden. Für unser Vokabular stellen gerade die Ziffern *Eins,Zwei* und *Drei* ein grosses Problem dar. Diese drei Ziffern sind verantwortlich für den Grossteil der entstandenen Fehler. Innerhalb des Vokabulars ist eine zusätzliche Variation der Ziffer *2* vorhanden. Nun soll getestet werden, wie gross der Einfluss ähnlicher Wörter innerhalb des Vokabulars auf die Fehlerrate ist.

6 Experimente und Resultate

6.1 Resultate 1

Die erste Versuchsreihe wurde mit den vorgegebenen Funktionen und deren eingestellten Parameter durchgeführt.

Parameter	
Abtastfrequenz	8 kHz
Anzahl Cepstrale Koeff.	12
Anzahl Mel Filter	24
Länge Analysefenster	240
Länge Verschiebungsfenster	80
Detektor Threshold	3

Ergebnisse der Experimente		
Mustergenerierung	**Anzahl Fehler**	**Fehlerrate**
Gemittelt	71	8.61%
Gewichtet (1/2 zu 1/2)	76	9.21%
Gewichtet (1/4 zu 3/4)	77	9.33%
Beste Signale	74	8.97%
2 Cluster	66	8.00%
4 Muster	72	8.73%

1	13	14	19	30	31	37	40	43	44	45	46	50	60		
2	4	6	10	22	26	34	40	41	55	67	68	76			
3	7	11	12	14	20	23	28	29	39	58	60	67	69	70	72
4	24														
5	9	14	17	22	26	32	33								
6	8	24	25	48	53	55	73								
7	13	22	24	71											
8	2	4	6	28	33	38	39	40							
9	13	31	37												
0															
2’	73														

Tabelle 1: *Fehlerverteilung innerhalb der Blöcke*

Wort	1	2	3	4	5	6	7	8	9	0	2’
Anzahl Fehler	13	12	15	1	7	7	4	8	3	0	1

Tabelle 2: *Fehlerverteilung*

Da bei unseren detektierten Signalen die Ungewissheit mitschwingt, ob alle Signale korrekt ausgeschnitten wurden, wiederholte man obige Experimente mit den optimalen Start- und Endpunkten.

Ergebnisse der Experimente		
Mustergenerierung	**Anzahl Fehler**	**Fehlerrate**
Gemittelt	24	2.91%
Gewichtet (1/2 zu 1/2)	29	3.52%
Gewichtet (1/4 zu 3/4)	24	2.91%
Beste Signale	24	2.91%
2 Cluster	24	2.91%
4 Muster	33	4.00%

Leider wurden die erhofften Resultate in einigen Experimenten nicht erreicht, sodass man davon ausgehen muss, dass die Anordnung der Signale untereinander einen relativ geringen Einfluss hat. Die ursprüngliche Fehlerrate konnte verbessert werden, doch das Experiment mit den optimal ausgeschnittenen Signalen hat gezeigt, dass der Detektor weitaus wichtiger ist, als die Anordnung der Signale untereinander. Für suboptimal ausgeschnittene Signale bringt die Anordnung der Signale noch eine minimale Verbesserung, doch bei optimalen Start- und Endpunkten ist diese Verbesserung nicht mehr zu beobachten. Weiterhin wird deutlich, dass die Fehler meist nicht nur einzeln, sondern gestaffelt auftreten. Dies hängt mit der sprecherspezifischen Aussprache zusammen,da einige Sprecher die Laute gut und andere eher schlecht betonen. Zusätzlich fliessen fehlerhafte Signale 4 mal in das Muster mit ein, was die Staffelung der Fehler begründen könnte.

6.2 Schlussfolgerung 1

Leider führte dieser Ansatz nicht zu den gewünschten Verbesserungen der Erkennungsrate, aber immerhin zu einer anderen Erkenntnis. Der wichtigste Input im Spracherkenner sind die verwendeten Signale. Falls diese stark verrauscht, gestört, oder schlecht ausgeschnitten sind, wird man mit verschiedenen Anordnungen der Signale nur kleine Verbesserungen erzielen können. Weiterhin ist, wie im Kapitel Methoden beschrieben wurde, klar geworden, dass der verwendete Detektor fehleranfällig ist. Intensitätsarme Plosivlaute oder Frikative wie *f* oder *s* werden oft abgeschnitten, falls diese nicht besonders stark betont wurden vom Sprecher. Leider stellt sich das Problem, dass nicht einfach der Threshold für alle Signale runtergesetzt werden kann, da nicht alle Signale gleich verrauscht sind. Setzt man den Threshold zu tief an, schneidet man zwar keine Laute mehr ab, doch dafür ist die Chance gross, dass man starkes Rauschen ebenfalls als Sprache detektiert.

6.3 Resultate 2

Die Parameter wurden angepasst und die Signale neu ausgeschnitten, falls sie grosse Distanzen zum Referenzsignal aufweisen.

Parameter	
Abtastfrequenz	8 kHz
Anzahl Cepstrale Koeff.	12
Anzahl Mel Filter	24
Länge Analysefenster	240
Länge Verschiebungsfenster	80
Detektor Threshold	[2.5,2.5]

Ergebnisse der Experimente		
Mustergenerierung	**Anzahl Fehler**	**Fehlerrate**
Gemittelt	55	6.67%
Gewichtet (1/2 zu 1/2)	64	7.76%
Gewichtet (1/4 zu 3/4)	60	7.39%
Beste Signale	55	6.67%
2 Cluster	51	6.18%
4 Muster	64	7.76%

1	13	30	31	37	43	45	50	60						
2	6	10	22	29	34	40	41	55	67	68	76			
3	11	12	13	14	20	23	28	29	39	58	60	69	70	72
4														
5	14	55												
6	24	25	48	53	55	67	73							
7	13	22	24	31	71									
8	2	33	38	39										
9	13	31	37											
0														
2’	73													

Tabelle 3: *Fehlerverteilung innerhalb der Blöcke*

Wort	1	2	3	4	5	6	7	8	9	0	2’
Anzahl Fehler	8	11	14	0	2	7	5	4	3	0	1

Tabelle 4: *Fehlerverteilung*

Mittlerer Abstand zum optimalen Schnittpunkt	
Standard	1939 Abtastwerte
Verbessert	1908 Abtastwerte

Summe der verbesserten Distanzen innerhalb der Mustergenerierung
8999.4 Distanzeinheiten

Die optimierten Threshold-Parameter resultieren in einer beachtlichen Verbesserung der Erkennungsrate. Leider ergeben sich keine weiteren Verbesserungen in der Erkennungsrate für die angepassten Signale mit einer grossen Differenz zu den Referenzsignalen, obschon die Abstände zu den optimalen, manuellen Schnittpunkten verkleinert werden konnten. Dass die Differenz so klein ist, hängt damit zusammen, dass einige wenige Signale stark verbessert wurden, aber sich dies, über die minimal verbesserten Signale gemittelt, nicht mehr so stark andeutet. Die verkürzte Distanz hingegen zeigt den Effekt schon deutlicher. Für die getesteten Fälle konnten die neuen Schnittpunkte nicht die ausschlaggebenden Verbesserungen erzielen, die nötig wären, um einen Fehler zu verhindern. Nichts desto trotz ist die erzielte Änderung als Erfolg zu werten, auch wenn es ein wenig enttäuschend anmutet, dass sich die Fehlerrate deswegen nicht ändert. Der wesentliche Grund dafür liegt darin, dass nur die Muster verbessert wurden, aber nicht die zu erkennenden Signale. Somit gleiche ich meine Signale für die Mustergenerierung an, aber wenn der Input aus der Reihe tanzt, wird ein Fehler gemacht.

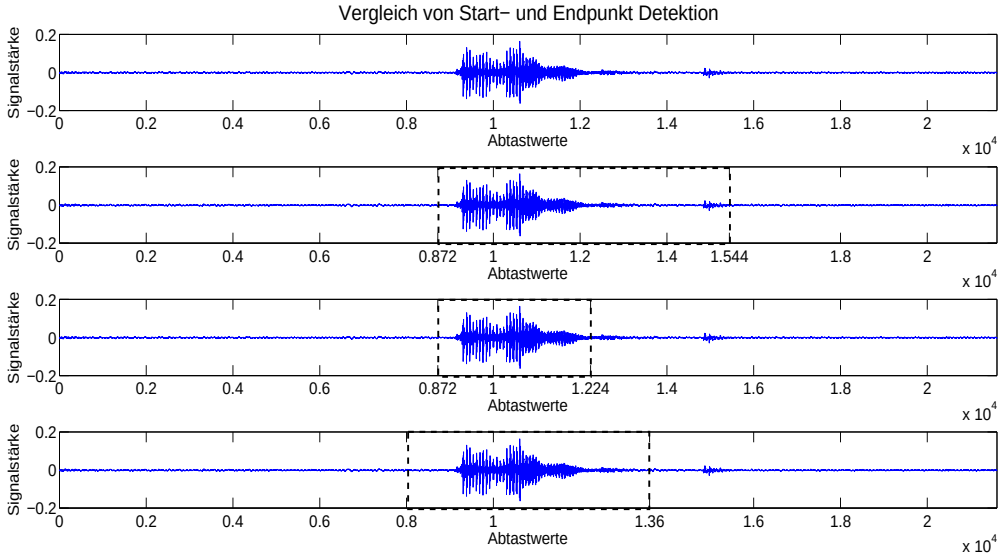

Abbildung 6: *Von oben nach unten:Signal, detektiertes Signal, verbessertes Signal, optimales Signal*

Nun steht noch das Resultat des neuen angepassten Vokabulars aus. Die *Zwei* wurde nun aus dem Vokabular entfernt und die *Zwo* nahm dessen Platz ein. Wie man im Resultat sehen kann, hat diese Änderungen einen massgeblichen Einfluss auf die Fehlerrate.

Mustergenerierung	Anzahl Fehler	Fehlerrate
Gemittelt (altes Vokabular)	55	6.79 %
Gemittelt (neues Vokabular)	31	3.75 %

Tabelle 5: *Vergleich der beiden Vokabulare*

Wort	1	2’	3	4	5	6	7	8	9	0
Anzahl Fehler	5	1	4	0	2	7	5	4	3	0

Tabelle 6: *Neue Fehlerverteilung*

Die getesteten Filter resultierten in einer Verschlechterung der Erkennungsrate. Die Resultate davon sind im Anhang A zu finden. Das Hauptproblem beider Filter sind Störgeräusche und Hintergrundrauschen. Die Frequenzanteile des Hintergrundrauschens werden durch das Präemphasefilter verstärkt, sodass der Detektor Probleme mit der Sprachsignaldetektion hat. Das erneute Ausschneiden der Signale konnte dieses Problem nicht beheben, da die anderen Signale ebenfalls verrauscht waren, sodass dieser Algorithmus nicht zuverlässig funktionierte. Das Wiener Filter erledigte seinen Job gut und die Rauschreduktion in den Signalen war zufriedenstellend. Die ersten Blöcke liessen auf eine Verbesserung der Resultate hoffen, doch einer der Sprecher hat starkes Hintergrundrauschen in seinen Signalen, sodass in diesem Block die Fehlerrate in die Höhe schnellt. Das Wiener Filter funktioniert anscheinend sehr schlecht mit stark verrauschten oder gestörten Signalen.

7 Fazit

Die Vorgabe dieser Arbeit war die Generierung von robusten Muster für die Spracherkennung. Dabei sollten die erstellten Muster resistent gegen Störgeräusche und varierende Intensitäten sein. Dieses Ziel konnte einerseits durch die implementierten Mustergenerierungen, wie Clustering, oder Optimierung der Threshold-Parameter erreicht werden. Weiterhin sollten alle Signale in die Mustergenerierung miteinfliessen. Dieser Punkt konnte ebenfalls grösstenteils erreicht werden, durch die erneute Detektion im Spracherkenner bei grossen Distanzen der Signale untereinander. Die Signale konnten mit dem entwickelten Vorgehen besser ausgeschnitten werden und somit trotzdem verwendet werden. Signale desselben Musters sind einander ähnlich und folglich müssten die Distanzen untereinander verhältnismässig gering ausfallen. Diese gegenseitigen Informationen lassen sich ausnutzen, indem die 'schwarzen Schafe' innerhalb einer Menge ausfindig gemacht und verbessert wurden.

Die vorhandenen Störgeräusche in den Aufnahmen waren grösstenteils sehr breitbandig, sodass sie nicht einfach gefiltert werden konnten und überlagerten sich stellenweise für längere Zeit mit dem Sprachsignal. Eine Detektion war sehr schwierig, da diese nicht isoliert vorkamen und der Zeitaufwand relativ gross war, sodass die Gefahr bestand mit den restlichen Punkten der Arbeit ins Hintertreffen zu geraten. Adaptive Filter wären meiner Ansicht nach ein sehr interessantes Thema für die Spracherkennung auf dem Gebiet der Signalverarbeitung. Doch ohne entsprechende Vorkenntnise, oder entsprechende Hilfestellung, ist die Störgeräuschfilterung schwierig anzugehen. Aus diesem Grund musste die Störgeräuschdetektion und anschliessende Filterung weggelassen werden.

Die Herausforderung dieser Arbeit war für mich einerseits das selbstständige Erarbeiten der Grundlagen und andererseits die Startschwierigkeiten mit der vorgegebenen Umgebung. Der Theorieteil hat sich in die Länge gezogen, da ich als Einsteiger gerade mit dynamic programming einsteigen musste, und demnach am Anfang ziemlich gefordert war. Mit den zur Verfügung gestellten Übungen des Betreuers und Recherchen in der Literatur konnte ich den Einstieg in das Thema jedoch gut finden. Das Vertrautmachen mit der Matlab Umgebung dauerte hingegen ein bisschen länger. Mit Matlab kannte ich mich zu Beginn der Übung kaum aus und die verschiedenen Funktionen, die zur Verfügung gestellt wurden, benötigten ihre Zeit bis ich mich eingearbeitet hatte. Nach dem Studieren der gestellten Funktionen und den ersten Programmierschritten, fand ich mich besser zurecht, sodass die auf dem Blatt entwickelten Implementationen ohne Probleme in Code übersetzt werden konnten.

Mein Hauptproblem dieser Arbeit waren teilweise die entmutigenden Ergebnisse. Da der Spracherkenner relativ komplex ist, haben mögliche Verbesserungen nicht immer den erwarteten Effekt. Teilweise würde man vermuten, dass mit dieser oder jener Überlegung die Erkennungsrate möglicherweise verbessert werden kann, doch nach dem Testdurchlauf folgt dann meist die Ernüchterung.

Ein weiteres Problem war die Tatsache, dass der Detektor das wichtigste Element innerhalb des Erkenners darstellt. Bei entsprechend schlechter Leistung, konnte auch mit den Mustern nicht mehr viel herausgeholt werden. Es war erstaunlich, wie wenig die Anordnung der Signale untereinander zur Mustergenerierung und deren Performance beigetragen hat. Falls ich nun beispielsweise von 4 Signalen zwei Signale habe, die einen Laut abschneiden und die anderen

zwei den Laut detektieren nützt es wenig, oder ist eher kontraproduktiv, auf die Erkennungsrate bezogen, wenn ich den Laut noch anhänge, falls dann das Input Signal den Laut ebenfalls abgeschnitten hat. Für das Input Signal hab ich keinerlei Vergleichspunkte, ausser den Mustern und kann deswegen nicht sagen, ob das Signal richtig geschnitten wurde, oder nicht. Falls mein Input Signal verrauscht, oder schlecht geschnitten ist, werde ich fast immer einen Fehler machen, selbst wenn meine Muster noch so gut sind.

Selbst optimal ausgeschnittene Signale produzieren noch Fehler. Dies liegt an den verwendeten Merkmalen und deren Eigenschaften. Wie bereits früher erwähnt weisen Frikative untereinander wesentlich kleinere Eigenheiten auf als etwa Vokale, was natürlich deren Erkennung schwieriger macht. Zusätzlich ist es so, dass bei einigen Frikativen die Kreuzdistanz kleiner ist als die Eigendistanz. Bei *f* und *s* heisst dies, dass die Distanz zwischen den beiden kleiner ist als die Distanz zwischen dem gleichen Frikativ. Darauf lassen sich die Schwierigkeiten bei der Unterscheidung zwischen den Ziffern *Eins,Zwei und Drei* zurück führen.

8 Ausblick

Grosses Verbesserungspotential sehe ich in der Anpassung des DTW Vergleichs und der lokalen Gewichtung der Distanzen. Möglicherweise lässt sich damit das Problem der Unterscheidung von ähnlichen Wörter beheben. Man könnte etwa die Besonderheiten oder Unterschiede der einzelnen Worte im Vergleich stärker gewichten. Denkbar wäre ein zweiter Durchlauf des Input Signals durch den Spracherkenner, doch diesmal mit angepassten Mustern, falls die Standardmuster in einem ersten Durchlauf auf eine *Eins, Zwei* oder *Drei* entschieden haben.

Mögliche Fortsetzungen dieser Arbeit sehe ich auch auf dem Gebiet der Signalverarbeitung. Ich könnte mir vorstellen, dass entsprechend angepasste, oder adaptive Filter die Signalqualität weiter verbessern könnten, sodass in einem nächsten Schritt die Detektion des Sprachsignals vereinfacht würde.

Einerseits müsste zuerst der Detektor weiter verbessert werden, resp. denke ich nicht, dass mit der existierenden Version noch grosse Sprünge möglich sind. Ein Detektor zu verbessern und implementieren ist eine anspruchsvolle Aufgabe und würde selber genug Stoff bieten für eine Semesterarbeit. Nicht nur die zu verarbeitenden Signale müssen präzise ausgeschnitten vorliegen, sondern noch viel mehr die Input Signale, da diese viel anfälliger auf Störgeräusche sind, als die Muster.

Vom Institut wurde bereits eine weiterführende Arbeit ausgeschrieben, die sich mit der intelligenten Äusserungsdetektion beschäftigt. Dabei könnte untersucht werden, ob sich die Erkennungsrate verbessert, wenn man für die Mustergenerierung die Parameter des Detektors in Abhängigkeit des Wortes setzt. Möglicherweise könnte man gleichzeitig untersuchen, ob durch partielle Vergleiche der Muster ein falsch geschnittenes Input Signal erkannt werden kann.

Der Mustervergleich für den sprecherunabhängigen Fall mittels neuronalen Netzen wurde in dieser Arbeit nicht behandelt und bietet sich allenfalls für eine fortführende Arbeit an, die möglicherweise einen der oben genannten Punkte aufgreifen würde.

9 Literaturverzeichnis

Literatur

[1] Beat Pfister und Tobias Kaufmann *Sprachverarbeitung:Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag, 2008.

[2] Institut für Signal- und Informationsverarbeitung *Fachpraktikum Signalverarbeitung:SV4.* ETH, www.isi.ee.ethz.ch/teaching/labs/SV4.pdf, [Stand 16.12.2009].

10 Anhang A:Weitere Experimente

Parameter	
Abtastfrequenz	8 kHz
Anzahl Cepstrale Koeff.	12
Anzahl Mel Filter	24
Länge Analysefenster	240
Länge Verschiebungsfenster	80
Detektor Threshold	[2.5,2.5]

Experiment	Anzahl Fehler	Fehlerrate
Wiener Filter	99	12.00%
alternativer Detektor	58	7.03%
Präemphase-Filter	116	14.06%

Tabelle 7: *Vergleich der Experimente*

10.1 Wiener Filter

1	15	23	24	34	37	40	44	45	50	54	55	60	72						
2	4	6	10	24	29	41	42	56	67	76									
3	7	11	12	14	20	22	23	28	29	33	39	40	55	57	58	60	69	70	72
4	23	24	25	33	36	54	55												
5	14	22	23	24	25	33	43	55											
6	13	15	23	24	33	48	49	53	55	71	73	76							
7	22	24	25	26	33	36	54												
8	2	23	33	37	38	39	55	72	76										
9	13	15	23	24															
0	23	24	33	54															
2’	23	24	34	41	55	73													

Tabelle 8: *Fehlerverteilung innerhalb der Blöcke*

10.2 Detektor nach Rabiner & Sambur (1975)

1	37	39	45											
2	10	13	19	24	38	42	46	55	62	67	75			
3	12	14	18	20	29	39	42	43	46	58	69	70	72	74
4	24	36	37	62										
5	14	37	45											
6	9	38	41	48	72	75	76							
7	22	32	59											
8	22	23	24	38	46	55								
9	13	22												
0	37	46												
2’	38	53	73											

Tabelle 9: *Fehlerverteilung innerhalb der Blöcke*

10.3 Präemphase-Filter

Parameter	
Abtastfrequenz	16 kHz
Anzahl Cepstrale Koeff.	12
Anzahl Mel Filter	24
Länge Analysefenster	240
Länge Verschiebungsfenster	80
Detektor Threshold	[2.5,2.5]

1	7	15	23	24	31	37	43	45	50	54	55	60	69			
2	10	23	24	34	41	42	45	53	54	56	67	68				
3	7	12	14	22	23	26	28	29	37	39	41	55	58	60	69	70
4	22	23	24	25	26	33	36	44	54	55						
5	23	24	25	26	43	54	55	69								
6	22	23	24	25	26	41	53	54	73	74	76					
7	13	22	23	24	25	26	31	33	54	55						
8	2	23	24	33	39	55										
9	13	22	23	24	25	26	37	54	55							
0	22	23	24	25	26	33	34	36	44	54	55					
2’	22	23	24	25	26	33	34	53	54	55						

Tabelle 10: *Fehlerverteilung innerhalb der Blöcke*

10.4 Kommentar

Es ist auf die hohe Fehlerrate der Blöcke *21-25* im Wiener Filter und dem Präemphase-Filter hinzuweisen. Es scheint ein Problem mit dem Rauschen dieser Signale aufzutreten. Zusätzlich ist es interessant, wie verschieden die Fehler der beiden Detektoren sind. Die fehlerhaften Blöcke sind grösstenteils verschieden, sodass ein Detektor mit weiteren Merkmalen durchaus interessant sein könnte, wenn er weiter optimiert würde.

11 Anhang B:Aufgabenstellung

Institut für
Technische Informatik und
Kommunikationsnetze

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

Herbstsemester 2009
(SA-2009-24)

Semesterarbeitsaufgabenstellung

für

Herrn Florian Deragisch

Betreuer: M. Gerber ETZ D97.4
Stellvertreter: Dr. B. Pfister ETZ D97.6

Ausgabe: 15. September 2009
Abgabe: 18. Dezember 2009

Generierung von Sprachmustern für die Spracherkennung

Einleitung

Erkennung isoliert gesprochener Wörter ist eine Anwendung der automatischen Spracherkennung, die zum Beispiel bei der Sprachsteuerung von Maschinen Anwendung findet. Einzelworterkenner können ausser mit statistischen Ansätzen wie zum Beispiel Hidden Markov Modellen auch mit Mustervergleich implementiert werden. Dabei wird die zu erkennende Äusserung mit Mustern aller Wörter des Vokabulars verglichen und das erkannte Wort entspricht jenem Wort, dessen Muster die kleinste Distanz zur Äusserung hat. Die Distanz zwischen der Äusserung und den Mustern wird mit Dynamic Time Warping (DTW) berechnet (siehe auch [1]). Der Einsatz vom Mustervergleich hat den Vorteil, dass die Muster aus wenigen oder sogar nur einer einzigen Äusserung generiert werden kann. Im Gegensatz dazu sind für statistische Erkenner entweder viele Äusserungen eines Wortes nötig, falls Wortmodelle zum Einsatz kommen, oder eine Transkription jedes Wortes, falls Laut-basierte Modelle zum Einsatz kommen (siehe auch [1]). Deshalb wird Einzelworterkennung oft mit Mustererkennung gelöst, insbesondere wenn es sich um einen sprecherabhängigen Erkenner handelt, d.h. wenn der Erkenner von derselben Person verwendet wird, die auch die Muster gesprochen hat.

Damit die Erkennung allerdings zufriedenstellende Resultate liefert, müssen die Wörter durch gute Muster repräsentiert sein.

Aufgabenstellung

In dieser Arbeit sollen verschiedene Methoden zum generieren von Sprachmustern implementiert und verglichen werden. Es ist nicht sinnvoll, die Sprachmuster aus einem Sprachsignal zu erzeugen, da mehrere Äusserungen desselben Wortes stark variieren können, sogar wenn sie vom selben Sprecher kommen. Eine erste mögliche Verbesserung ist deshalb, ein Sprachmuster aus der Mittelung mehrerer Äusserungen zu generieren. Einzelne Äusserungen können allerdings schlecht oder fehlerhaft sein. Es kann zum Beispiel sein, dass ein Störsignal vorhanden ist, oder dass die Äusserung nicht richtig ausgeschnitten wurde. Dies kann sich dadurch manifestieren, dass grosse Distanzen beim Vergleich der einzelnen Signale untereinander auftreten oder dass bei der zeitlichen Anpassung mit dem DTW-Algorithmus ungewöhnliche zeitliche Anpassungen vorkommen. Ein einfacher Ansatz ist, auffällige Äusserungen nicht zur Generierung des Musters zu verwenden. Dieser Ansatz ist allerdings eine Verschwendung des vorhandenen Datenmaterials, die unter Umständen vermieden werden kann, wenn eine auffällige Äusserung trotzdem verwendet werden kann, wenn Start- und Endpunkt der Äusserung korrigiert werden.

Es wird empfohlen, folgendermassen vorzugehen:

1. Es soll in der Literatur nach Methoden der robusten Mustererzeugung gesucht werden und die gefundenen Algorithmen sollen auf eine mögliche Verwendung hin untersucht werden.

2. In Zusammenarbeit mit den Betreuern sollen einige mögliche Methoden zur Generierung von Sprachmustern genau konzipiert werden.

3. Die eruierten Methoden sollen implementiert und mit den zur Verfügung gestellten Testdaten in einem geeigneten sprecherabhängigen Szenario getestet werden.

4. *optional:* Meistens wird im DTW-Algorithmus des Mustererkenners die euklidische Distanz verwendet. Da dies eine ziemlich grosse Sprecherabhängigkeit mit sich bringt, wurde am Institut eine Methode entwickelt, bei der die euklidische Distanz durch ein Distanzmass, welches auf einem neuronalen Netz (NN) basiert, ersetzt wird, um so mehr Sprecherunabhängigkeit zu erreichen. Unter Verwendung dieses NN-Distanzmasses sollen die implementierten Methoden auch für ein sprecherunabhängiges Szenario evaluiert werden.

Die ausgeführten Arbeiten und die erhaltenen Resultate sind in einem Bericht zu dokumentieren (siehe dazu [2]), der in gedruckter und in elektronischer Form abzugeben ist. Zusätzlich sind im Rahmen eines Kolloquiums zwei Präsentationen vorgesehen: etwa drei Wochen nach Beginn soll der Arbeitsplan und am Ende der Arbeit die Resultate vorgestellt werden. Die Termine werden später bekannt gegeben.

Literaturverzeichnis

[1] B. Pfister and T. Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag (ISBN: 978-3-540-75909-6), 2008. http://www.springer.com/978-3-540-75909-6.

[2] B. Pfister. *Richtlinien für das Verfassen des Berichtes zu einer Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, März 2004.
(http://www.tik.ee.ethz.ch/~spr/SADA/richtlinien_bericht.pdf).

[3] B. Pfister. *Hinweise für die Präsentation der Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, März 2004.
(http://www.tik.ee.ethz.ch/~spr/SADA/hinweise_praesentation.pdf).

Zürich, den 28. September 2009 Prof. Dr. L. Thiele