“Navigation - ich möchte nach Calw”

Spracherkennung für große Lexika im Auto

Christoph Schillo

Dipl.-Inform. Christoph Schillo
AG Angewandte Informatik
Technische Fakultät
Universität Bielefeld
email: schillo@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor-Ingenieur (Dr.-Ing.).
Der Technischen Fakultät der Universität Bielefeld
am 19.08.2002 vorgelegt von Christoph Schillo.

Gutachter:

Prof. Dr. Franz Kummert
Prof. Dr. Henning Lobin

Prüfungsausschuss:

Prof. Dr. Ipke Wachsmuth
Prof. Dr. Franz Kummert
Prof. Dr. Henning Lobin
Dr. Christian Bauckhage

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

“Navigation - ich möchte nach Calw” Spracherkennung für große Lexika im Auto

Der Technischen Fakultät der Universität Bielefeld

zur Erlangung des Grades

Doktor-Ingenieur

vorgelegt von

Christoph Schillo

Bielefeld – August 2002

Danksagung

Bei der Erstellung der vorliegenden Arbeit wurde ich von vielen Menschen unterstützt, denen ich hiermit meinen tiefen Dank ausdrücken möchte.

Allen voran gilt mein Dank meinem Betreuer und Erstgutachter Franz Kummert, der jederzeit mit konstruktiver Kritik aufwartete und immer die richtigen Worte fand, um die Weiterführung der Arbeit auch nach meinem Übergang in die nicht universitäre Arbeitswelt zu motivieren. Weiterhin danke ich dem Zweitgutachter Henning Lobin und den Mitgliedern des Prüfungsausschusses Ipke Wachsmuth und Christian Bauckhage für ihre Rezension und anregenden Fragen während der Disputation.

Dank gebührt selbstverständlich auch den Mitgliedern der Arbeitsgruppe "Angewandte Informatik", mit denen in angenehmer Arbeitsatmosphäre viele inspirierende und fachübergreifende Gespräche stattfanden. Hervorheben möchte ich meine Bürokolleginnen Katrin Kirchhoff und Britta Wrede, sowie Gernot A. Fink, die jederzeit mit linguistischen und informatischen Fachwissen Beistand leisteten und als gute Freunde auch außerhalb der Universität die nötige Distanz zur Arbeit schufen. Weiterhin gilt mein Dank auch Gerhard Sagerer und Lisbeth van Iersel, dem Rückgrat der Arbeitsgruppe, die immer ein offenes Ohr für Belange aller Art hatten.

Schließlich danke ich noch meinen Eltern, die mir während meines Werdegangs durch Ihre moralische und tatkräftige Unterstützung die Freiheit gaben, meinen eigenen Weg zu finden.

Inhaltsverzeichnis

1 Einleitung

Im Dasein des Menschen ist die Interaktion mit seiner Umwelt ist ein wesentlicher Aspekt. Gegenstand der vorliegenden Arbeit ist die Betrachtung eines Ausschnittes dieser Interaktion, der Kommunikation, und ihres Einsatzes als Mensch-Maschine Schnittstelle.

In der direkten Kommunikation untereinander benutzen Menschen in erster Linie Mimik, Gestik und Sprache. Abgesehen von subtilen Abstufungen spiegelt die Mimik lediglich den emotionalen Zustand einer Person wider. Die Gestik hingegen kann schon komplexere Informationen transportieren, sie wird u. a. benutzt, um auf etwas hinzuweisen (deiktisch), um Gegenstände zu beschreiben (pantomimisch) oder auch um Relationen zwischen Objekten darzustellen (ikonisch)[1]. Die Sprache schließlich ist in vielen Situationen das Medium der Wahl, sie hat gegenüber den anderen Modalitäten eine Reihe von Vorteilen: Zum Sprechen werden keine Hände benötigt, es können während des Informationsaustauschs weiterhin manuelle Arbeiten wie beispielsweise das Lenken von Fahrzeugen oder das Operieren von Patienten durchgeführt werden. Ähnlich vorteilhaft wirkt sich aus, dass Sprache keinen Sichtkontakt erfordert. Sie kann sowohl unter eingeschränkten bis fehlenden Sichtverhältnissen eingesetzt werden als auch ohne dass der Blick speziell auf etwas gerichtet werden muss und somit von anderen Aufgaben abgelenkt wird. Weiterhin erlaubt Sprache, komplexe Sachverhalte und exakte Angaben effizient zu vermitteln.

Zusätzlich zu den bereits genannten vorteilhaften Eigenschaften ist auch die technische Übermittlung der Sprache leicht zu realisieren. Auf der Eingabeseite benötigt man lediglich ein Mikrofon, auf der Ausgabeseite einen Lautsprecher. Diese Hardware läßt sich zu geringen Kosten fertigen und wird heutzutage bereits von Vielen in Form von portablen Telefonen ständig mitgeführt.

Darüber hinaus gibt es mit der Schrift ein etabliertes Mittel, um Sprache zu fixieren und zu transportieren, ein Aspekt, der in der Erweiterung eines spracherkennenden Systemes (s. Kapitel 5.3) eine wichtige Rolle spielt.

Mit der Sprache steht ein Medium zur Verfügung, das vom Menschen intuitiv, alltäglich und in den verschiedensten Situation zur Kommunikation benutzt wird. Im Kontrast dazu steht die Ansteuerung von Maschinen, die fast immer noch auf herkömmliche Art über Schalter und Regler oder ber etwas ausgefeiltere, aber immer noch künstliche Eingabegeräte wie Tastatur und Maus durchgeführt wird.

[1] Für eine ausführliche Beschreibung von Gestik s. beispielsweise [37]

Abgesehen vom Aspekt der Benutzerführung (die Programmierung älterer Videorekorder ist ein mittlerweile legendäres Negativbeispiel für verwirrenden und schlecht gestalteten Benutzerdialog), der auf alle Interfaces zutrifft, wäre es also wünschenswert, die Sprache auch zum Ansteuern diverser Geräte zu benutzen, schon allein um eine höhere Benutzerakzeptanz zu erreichen. Leider hat sich der Einsatz der Sprache zur Mensch-Maschine Interaktion bis auf einige Ausnahmefälle, die im Weiteren betrachtet werden, noch nicht durchgesetzt. Dies gilt in beide Richtungen, für die Sprache als Ein- und auch als Ausgabemedium[2]. Der Schwerpunkt der vorliegenden Arbeit ist die Sprache als Eingabemedium.

Eine wichtige Unterscheidung in Bereich der Sprachverarbeitung ist die Einteilung in die Spracherkennung und das Sprachverstehen. Während das Ziel der Spracherkennung die Überführung von Sprachsignalen in ihre entsprechende Transkription, ihre textuelle Repräsentation, ist, geht das Sprachverstehen auf verschiedenen Abstraktionsebenen darüber hinaus. Es reicht von der durch Grammatiken vorgeschriebenen Syntax, über die Bedeutungszuordnung der Semantik bis zur situationsbezogenen Interpretation in der Pragmatik und im Dialog. Die Spracherkennung ist eine notwendige Voraussetzung zum Sprachverstehen, während das Sprachverstehen seinerseits die Spracherkennung unterstützen und verbessern kann, dabei aber auch den gesamten Erkennungsprozess komplexer im Zeit- und Rechenaufwand macht.

Zu den schon verfügbaren Erkennungssystemen gehören Einzelworterkenner, Diktieranwendungen und Auskunftsysteme[3]:

Einzelworterkenner sind die einfachste Form der Spracherkenner. Durch die Information, dass ein gegebenes Signal genau auf ein Wort aus einem festen Repertoire von Wörtern abgebildet wird, vereinfacht sich die Aufgabe derart, dass potentiell ein einfacher Mustervergleich über eine Zeitverzerrungsfunktion (s. Kapitel 2.1) für ein zufriedenstellendes Resultat ausreicht. Solche Systeme befinden sich beispielsweise in Telefonauskunftsystemen, die einfache Entscheidungsbäume verarbeiten ("`Wenn Sie eine Beschwerde haben, drücken Sie die '5' oder sagen Sie 'Beschwerde', wenn Sie eine Information ...`"). oder in etwas anderer Form in Handys. Die verbale Handysteuerung beschränkt sich momentan noch auf das Wiedererkennen zuvor gesprochener Namensschablonen zur Schnellwahl von manuell eingegebenen Telefonnummern, nicht zur Ansteuerung der kompletten Funktionalität des Telefons. Sie ist lediglich ein Versuch, ein Produkt von der Vielzahl der Konkurrenzprodukte abzusetzen (s. [44], S.5f). Die ursprüngliche Motivation der alternativen Sprachansteuerung für die Auskunftsysteme dagegen war die zur Verfügungstellung diverser tastensteuerbarer, telefonbasierter Systeme (wie etwa Kontoführung via Phone),

[2]Für ausführliche Informationen zur Technik und zum Stand der Sprachausgabe und Sprachsynthese s. [1], [52] oder auch [65]

[3]Eine detailliertere Einteilung der Komplexität der verschiedenen Anwendungen befindet sich in Kapitel 2.1 in der Grafik 2.1.

auch für Besitzer von nicht *DTMF*[4] fähigen Telefonen. Andere Einsatzgebiete für Einzelworterkenner sind Sprachinterfaces, die lediglich wenige Dutzend Wörter benötigen, aber sehr robust arbeiten sollen. Diese Interfaces dienen zur Steuerung von Maschinen, etwa die Manipulation von mobilen Robotern ohne Fernbedienung oder die Einstellung von komplexen Mikroskopen, aber auch zur Navigation innerhalb von Programmen und Webseiten.

Diktieranwendungen sind Erkennungssysteme wie beispielsweise *IBM ViaVoice* oder *Philips FreeSpeech*, die sprecherabhängig in ruhiger Umgebung mit Vokabularen von 50.000 Wörtern und mehr umgehen können. Die Sprecherabängigkeit bedeutet, daß sich ein sprecherunabhängiges, also ein unspezialisiertes, Grundsystem durch Vorlesen definierter Sätze an einen Sprecher adaptiert, was eine enorme Verbesserung der Erkennungsrate für genau diese Person mit sich bringt. Neben der geforderten ruhigen Umgebung werden diese Systeme in der Regel mit einem mitgelieferten, normierten Mikrofon betrieben, woraus eine weitere Reduktion der Abweichungen von für dieses System optimal gestalteten Signalen folgt. Des Weiteren wird die Aufgabe dadurch erleichtert, dass die Sprecher kooperativ und ruhig sind sowie in der Regel wohlgeformte Sätze äußern. Der Anwendungszweck dieser Systeme liegt im automatischen Transkribieren von gesprochenen Texten, um ungeübten Typisten die Arbeit zu erleichtern bzw. den Umweg vom aufgenommenen Diktat zur Verschriftlichung zu sparen oder wenigstens zu vereinfachen.

Auskunftsysteme werden eingesetzt, um auf Anfrage eines Benutzers Informationen aus einer bestimmten Domäne zur Verfügung zu stellen, beispielsweise über Zugfahrpläne wie beim TABA-System von Philips [6] oder über das Wetter wie im Jupiter-System des MIT [48]. Im Allgemeinen dialoggesteuert wird der Benutzer so lange befragt, bis alle für eine Anfragebearbeitung benötigten Informationen angesammelt werden konnten. Diese werden dann in eine Datenbankanfrage umgesetzt und die gefundenen Lösungen verbal oder textuell präsentiert. Die dafür benötigte Sprecherunabhängigkeit sowie die spontansprachlichen Phänomene komplizieren die Aufgabe. Die eingeschränkte Domäne und die Möglichkeit, den Benutzer über einen Dialog zu 'steuern' vereinfachen sie allerdings wieder.

Im Gegensatz zu den geschilderten Systemen wird in dieser Arbeit ein System entwickelt, das nicht in normierten (Büro-) Umgebungen, sondern in der variablen Umgebung eines Fahrzeugs eingesetzt werden soll. Es werden potentielle Einsatzmöglichkeiten von Sprache als Interface für die Ansteuerung komplexer Geräte aufgezeigt. Zu diesen Geräten zählen beispielsweise Navigationssysteme, Stereoanlagen, Handys sowie das Fahrzeug selbst. Dabei werden allerdings nicht alle möglichen Funktionen berücksichtigt: die Sprache zur Ausgabe von kontinu-

[4] *DTFM*=**D**ual **T**one **M**ultiple **F**requency. Jeder Telefontaste werden zwei simultan abgespielte Töne zugeordnet, die auch analog sicher übertragen und auf der Gegenseite automatisch erkannt werden können.

ierlichen Statusreports wie der momentanen Geschwindigkeit oder der Aktivität eines Blinkers ist überflüssig oder gar störend. Dies kann besser durch die gewohnten analogen Anzeigen wie Tachometer und Kontrolllampen geschehen. Weiterhin ist der Einsatz einer Sprachsteuerung von fahrkritischen Funktionen wie beispielsweise Handbremse, Lenkung und Aufblendlicht zu risikobehaftet, da keine 100%ige Erkennung garantiert werden kann. Dem gegenüber ist die Spracherkennung hervorragend geeignet, um die Ansteuerung eines Handys, ansonsten eine gefährliche und mittlerweile unter Strafe gestellte Ablenkung für den Fahrer, zu realisieren, oder auch, um die Kommunikation zwischen Benutzer und Navigationssystem herzustellen ("*Ich möchte ins Theater*" "`an der nächsten Kreuzung links abbiegen`").

Die Zielsetzung dieser Arbeit ist die Entwicklung eines Spracherkenners, der

- speziell für die Anforderungen im Fahrzeugeinsatz zugeschnitten ist,
- sowohl sprecherabhängig und sprecherunabhängig arbeiten kann,
- fähig ist, große Lexika zu verarbeiten,
- in ein Fahrzeug mit verschiedenen anzusteuernden Geräten eingebunden werden kann und
- auf einen integrierten Baustein portiert werden kann, sowohl in Hinsicht auf Verarbeitungsgeschwindigkeit als auch auf Speicherbedarf.

Im folgenden Kapitel werden zunächst neben der Geschichte der Spracherkennung die allgemeinen Grundlagen für einen statistischen Spracherkenner zusammengefasst. Das dritte Kapitel beleuchtet zuerst zwei allgemeine Probleme der Spracherkennung, die Anpassung an neue Benutzer und die Behandlung unbekannter Wörter. Anschließend werden verschiedene für die Anwendung im Fahrzeug relevante Aspekte behandelt. Das vierte Kapitel zeigt interessante Anwendungen speziell für die automobile Spracherkennung mit den damit einhergehenden Ansprüchen an Vokabulargrößen und Lexikonvariabilität. Anschließend werden bestehende Systeme vorgestellt. Das fünfte Kapitel beschreibt detailliert das im Rahmen dieser Arbeit entwickelte Sprachererkennungssystem einschließlich der Arbeitsweise und Ansteuerung der unterschiedlichen Geräte sowie die einzelnen Teilkomponenten des Systems. Nach der Beschreibung der verwendeten Entwicklungsumgebung und der benutzten Sprachkorpora präsentiert das sechste Kapitel die Evaluierungen und Diskussion der Leistungsfähigkeit der entstandenen Module und untersuchten Verfahren. Das siebte Kapitel faßt schließlich die geleisteten Arbeiten noch einmal zusammen und stellt kurz dar, wofür sie eine Grundlage bieten.

2 Grundlagen der Spracherkennung

Das allgemeine Ziel eines Spracherkennungssystems ist die Umsetzung von Sprachsignalen in eine textuelle Repräsentation. Wie andere für den Menschen scheinbar einfache Aufgaben, etwa das Greifen von Gegenständen oder Erkennen von Objekten aus beliebigen Perspektiven, ist dies nicht trivial mit Maschinen und Programmen nachzubilden. Der menschliche Hörapparat[1] ist besonders gut geeignet, die in der Sprache benutzten Frequenzen aufzulösen und durch aktive Bewegung des Hörers gezielt auf Geräuschquellen ausgerichtet zu werden, um damit eine Hervorhebung momentan interessanter gegen irrelevante Signale zu ermöglichen. Darüber hinaus besitzt der Mensch noch abstraktere Verarbeitungsebenen. Während er zur Erkennung eines Sprachsignals sein gesamtes Weltwissen einsetzen kann und somit Kontext in Betracht zieht, momentan sinnvolle und sinnlose Interpretationen berücksichtigt und eine Vorstellung für die Wohlgeformtheit von Äußerungen besitzt, wird ein einfacher maschineller Spracherkenner lediglich auf das bereits diskretisierte Signal und sein Lexikon mit den ihm bekannten Wörtern zurückgreifen können (wenn es eines gibt, s. Abschnitt 5.3). Das technische System versucht in seiner eingeschränkten Welt zu erkennen, kann aber eine Äußerung nicht *verstehen* und damit auch nicht den Sinngehalt als Maß einer korrekten Erkennung heranziehen. Durch den Einsatz von Grammatiken und Sprachmodellen kann bedingt weiterführendes Wissen in ein System eingebracht werden. Mit nachgeschalteten bzw. integrierten Verstehenskomponenten, die auf ein definiertes Weltwissen zurückgreifen können, besteht die Möglichkeit, unsinnige Alternativen in der Erkennung rechtzeitig zu identifizieren und nicht mehr weiter zu verfolgen bzw. dedizierte Dialogkomponenten zur Klärung einzusetzen.

Im Folgenden wird ein kurzer zeitlicher Abriss der Entwicklung von Spracherkennungssystemen gegeben, bevor der gegenwärtige Status Quo prototypisch näher betrachtet wird. Anschließend folgt ein ausführlicher Blick auf die Theorie der *Hidden-Markov-Modelle*, die ein wesentlicher Bestandteil praktisch aller aktuellen Systeme sind. Zusätzlich werden noch verschiedene aus einem Sprachsignal extrahierte Merkmale behandelt und Kriterien der Evaluierung vorgestellt.

[1] Der menschliche Hörapparat wird beispielsweise in [60] ausführlich beschrieben und im Rahmen der Arbeit nicht näher betrachtet.

2.1 Eine kurze Geschichte der Spracherkennung

Eine kurze historische Betrachtung der Spracherkennung zeigt die Abstufung der verschiedenen Schwierigkeitsgrade für ein Spracherkennungssystem. Stets beschränkt durch die verfügbare Hardware entwickelten sich die Programme und Verfahren von simplen sprecherabhängigen Einzelworterkennern für Lexika mit nur zehn oder weniger Wörtern nach und nach zu sprecherunabhängigen Verbundworterkennern, die Lexika von mehr als 50.000 Wörtern verarbeiten können.

Der erste bekannt gewordene, allerdings fruchtlose, Schritt in Richtung maschineller Spracherkennung wurde von dem Ungarn Tihamér Nemes im Jahre 1930 unternommen. Er plante, die optisch kodierte Tonspur von Filmen zur automatischen Transkription des Gesprochenen zu nutzen, und somit das Problem der Spracherkennung auf die Bilderkennung zu transferieren. Die Idee geriet durch Ablehnung seines Patentantrags mit der Begründung "das Verfahren sei zu unrealistisch" in Vergessenheit ([44],S.12).

Im Jahre 1939 wurde auf der Weltausstellung in New York das *Voder*-System von *Bell Labs* vorgestellt. Es war ein Sprachsynthesegerät, das elektrisch[2] den Stimmapparat des Menschen nachbildetete. Die Stimmbänder wurden durch einen Summer ersetzt und der Vokaltrakt, die veränderlichen Bereiche im Mund, Rachen und Nasenraum durch Hintereinanderschaltung verschiedener Filterbänke. Die Bedienung des Gerätes wurde von speziell ausgebildetem Personal vorgenommen, das über 14 Tasten sowie einen Hebel und ein Pedal die Ansteuerung der Filter vornahm. Mit Ersetzung der BedienerIn durch eine automatische Steuerungskomponente wurde das gleiche System später unter dem Namen *Vocoder* eingesetzt. Das Ziel dieses Systems war aber nicht die Synthese, sondern die Komprimierung von Daten: Wenn es gelingt, ein Sprachsignal auf eine kleine Anzahl von Merkmalen (hier Filterbankkoeffizienten) zu reduzieren, die zur Rekonstruktion des ursprünglichen Signals ausreichen, genügt es, anstatt des gesamten Signals lediglich die Merkmale zu übertragen. Der *Vocoder* erreichte eine Datenratenreduktion von 3000 Hz auf 300 Hz[3]. Für die Spracherkennung in den Kinderschuhen bedeutete diese Erkenntnis, dass offensichtlich wenige Merkmale ein Sprachsignal nahezu vollständig beschreiben können. Somit kann eine Spracherkennung auf diesen handlicheren Daten das gleiche Ergebnis erreichen, wie auf dem kompletten Sprachsignal.

Ebenfalls von *Bell Labs* wurde 13 Jahre später der erste Spracherkenner entwickelt, der damals im Gegensatz zu heutigen Systemen eine reine Hardwarerea-

[2] Mechanische Nachbildungen gab es schon Ende des 18. Jahrhunderts, beispielsweise die von von Kempelen[67].

[3] Eine Angabe in Bit/Sekunde [bps] ist nicht möglich, da die Signale analog übertragen wurden. Das Verfahren entspricht aber der Kodierung über lineare Vorhersage (LPC, s. 2.2), die auch noch heutzutage auf digitalen Daten eingesetzt wird und eine Reduktion von 64 kbps auf etwa 2.4 kbps für Telefonqualität erreicht.

lisierung war. Er konnte sprecherabhängig einzeln gesprochene Zahlen von 0 bis 9 erkennen. Das System arbeitete mit *Template Matching*: von jedem zu erkennenden Wort war eine Referenz auf Basis von spektralen Merkmalen, spezieller den ersten beiden Formanten[4], gespeichert. Das Wort mit dem ähnlichsten Muster wurde ausgewählt, wobei es keine Rückweisung von unbekannten Wörtern gab. Eine Anpassung auf andere Sprecher konnte nur manuell und mit hohem Aufwand erfolgen, danach erreichte der Erkenner aber bis zu 99% Wortakkuratheit [10].

Ein weiterer Meilenstein in der Genese der Spracherkennung war die Verwendung von Wortuntereinheiten: Anstelle kompletter Wörter werden Teile von Wörtern als Einheiten erkannt. 1959 wurden gleich mehrere Phonemerkenner entwickelt. Phoneme sind die kleinsten bedeutungsunterscheidenden Einheiten einer Sprache. Das System des *MIT (Massachusetts Institute of Technology)* erlaubte das sprecherunabhängige Erkennen von zehn Vokalen[5] im Kontext */b/vokal/t/* [17]. Ein zur gleichen Zeit vom *University College*, London, entwickeltes Verfahren konnte vier Vokale und neun Konsonanten erkennen. Dabei war eine zusätzliche Neuerung der Einsatz statistischer Informationen über erlaubte Phonemsequenzen zur Steuerung der Erkennung.

Erste Schritte in Richtung Miniaturisierung und damit Verbreitung der Spracherkennung in verschiedenen Produkten wurde Anfang der 60er Jahre in Japan mit der Entwicklung mehrerer Chips, auf denen Programme zur Erkennung von Phonemen integriert waren, unternommen. Wichtiger war aber die dortige Entwicklung sprecherabhängiger Systeme, die sich in einer Lernphase von einem einheitlichen Grundsystem auf ein sprecherspezifisches System umstellen konnten. Dazu wurden Methoden der dynamischen Programmierung eingeführt, die die zeitliche Verzerrung eines Testmusters zum Vergleich mit Referenzmustern erlaubten[6].

Das erste kommerzielle Produkt wurde 1970 von *Threshold Technology Inc.* auf den Markt gebracht. Das *VIP 100* war sprecherabhängig, hatte ein kleines Vokabular und erkannte nur Einzelwörter. Das System reichte dennoch aus, die *Advance Research Project Agency (ARPA)* für Spracherkennung zu interessieren und ließ sie schließlich das *ARPA SUR* (**S**peech **U**nderstanding **R**esearch project) ausschreiben. Die erklärte Zielsetzung des Projektes war die Entwicklung eines Erkenners

- mit einem Vokabular von mindestens 1000 Wörtern
- für kontinuierliche Sprache

[4] Vor allem im Spektrum von Vokalen auftretende Energiespitzen, die von niedrigen zu den hohen Frequenzen durchnummeriert werden.

[5] Es gibt mehr phonetische als ortografische Vokale.

[6] Das als 'dynamic time warping' (DTW) bekannte Verfahren wird im Kapitel 5.3.2 in etwas anderem Zusammenhang ausführlich vorgestellt.

- von mehreren bekannten, kooperativen Sprechern
- mit einer künstlichen Syntax zur Einschränkung des Suchaufwandes
- mit einem Laufzeitverhalten von höchstens n-facher Echtzeit[7]
- der eine Fehlerrate von weniger als 10% hat
- und auf einer definierten, realen Domäne arbeitet.

Insgesamt beteiligten sich drei Institute mit sechs verschiedenen Systemen an dem Projekt. Nach 6-jähriger Projektlaufzeit genügte 1976 *Harpy* ([43]) von der *Carnegie Mellon University* schließlich als einziges System allen Anforderungen. Erwähnenswerter Mitstreiter war das *Dragon*-System, der Urahn der auch heute noch verbreiteten Diktieranwendung *Dragon Dictate*. In den 80er Jahren wurde die Arbeit fortgeführt und der Übergang vom Einzelworterkenner zum Erkenner für kontinuierliche Sprache abgeschlossen. Damit einher ging der Einsatz von statistischen Modellen, einerseits in Form der *Hidden-Markov-Modelle* zur Reproduktion der akustischen Anteile im Sprachprozess (s. nächster Abschnitt) und andererseits der Sprachmodelle, die durch verfeinerte Verfahren bessere Nachbildungseigenschaften von Wortfolgen erreichten ([28]). Gleichzeitig wurden die ersten *Homecomputer* auf den Markt gebracht, womit die dramatische Entwicklung immer schnellerer und günstigerer und somit allgegenwärtigerer Computerplattformen gestartet wurde. Damit gab es einen neuen, großen Markt für Spracherkennungssoftware, die von simplen Steuersystemen schnell zu großen Diktieranwendungen wie *Dragon Dictate*, *IBM Easy Speak* oder *Philips FreeSpeech* wurde.

Der vorrangige Aspekt der 90er war die Vergrößerung der Vokabulare und damit die stärkere Forschung im Bereich von Wortuntereinheiten. Mit einem gewissen Vorrat an solchen Bausteinen können potentiell alle Wörter konstruiert werden. Zusätzlich wurden immer mehr Anwendungen entwickelt und komplette Systeme mit integrierten Verstehenskomponenten realisiert, etwa das Zugauskunftssystem von Philips [6], das rund um die Uhr telefonisch Fahrplananfragen beantworten kann, oder das *VRCP* (Voice Recognition Call Processing) System von AT&T, das eine automatische Gesprächsartwahl für Telefonabrechnungen zulässt [22].

Die Grafik 2.1 zeigt einen relativen Komplexitätsvergleich mehrerer Disziplinen in der Spracherkennung. Die einfachen Aufgaben sind nahe dem Ursprung aufgeführt. Entlang der Achse *Sprechart* und der logarithmischen Achse *Lexikongröße* steigt der Schwierigkeitsgrad. Weitere Parameter, wie etwa die Sprecherabhängigkeit/-adaption oder die Qualität und Quantität von Hintergrundgeräuschen hätten noch zusätzlich eingetragen werden können, wurden aber aus Gründen der Übersichtlichkeit weggelassen. Der graue Balken markiert ungefähr den heutigen Stand der Spracherkennung (in Anlehnung an [29]).

[7] Damit es auf zukünftigen, n-fach schnelleren Systemen in Echtzeit laufen kann. Eine gängige Annahme bei der Softwareentwicklung.

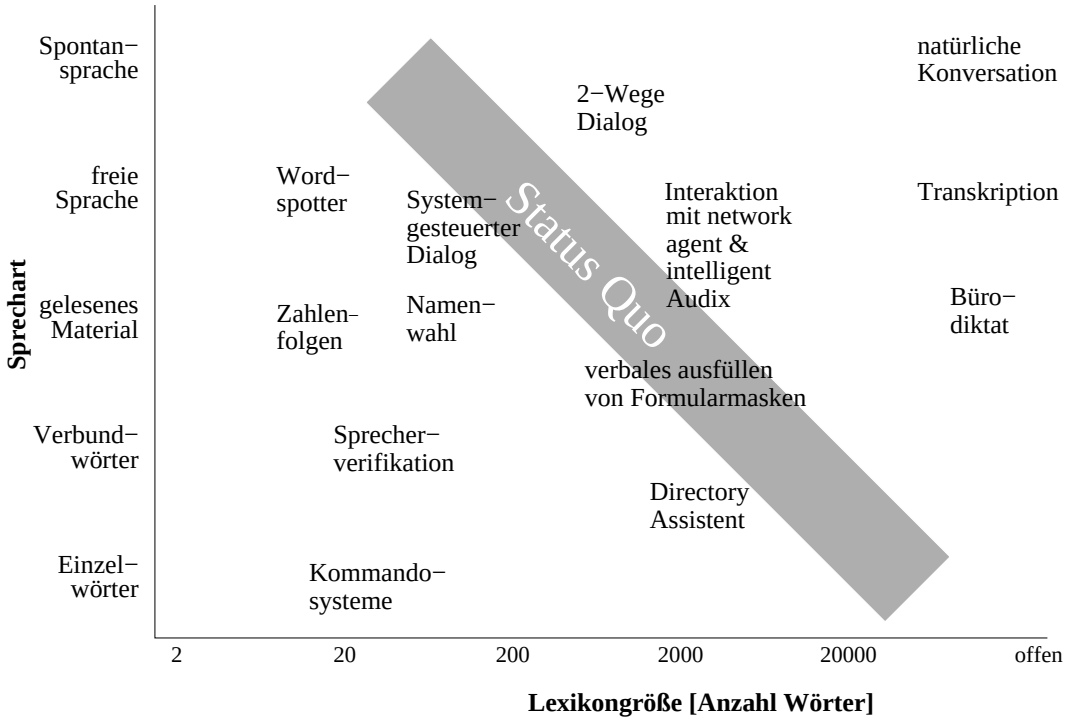


Abbildung 2.1: Komplexität verschiedener Anwendungen der Spracherkennung, der graue Balken zeigt den Stand heutiger Systeme (nach [29]).

2.2 Prinzipien der Spracherkennung

Die einzelnen Schritte im Prozess der Spracherkennung werden im Weiteren anhand der Abbildung 2.2 dargestellt. Formal betrachtet handelt es sich bei der Spracherkennung um das Problem, für ein gegebenes (kontinuierliches) Sprachsignal X (Abb. 2.2, (b)) die zugrundeliegende korrekte Wortfolge $W = w_1, w_2, ...w_n$ (Abb. 2.2, (a)) zu finden. Zu Beginn wird das in Zeit- und Wertebereich kontinuierliche Signal diskretisiert (Abb. 2.2, (c)). Die Abtastung in der Zeit wird zu äquidistanten Zeitpunkten im Abstand ΔT vorgenommen, die Abtastrate ergibt sich aus dem Quotienten $1/\Delta T$. Um den potentiellen Informationsverlust zu kompensieren, muss die Abtastrate in Abhängigkeit der gewünschten maximalen Frequenz gewählt werden: Laut Shannons Abtasttheorem ([63]) muss die Abtastfrequenz f_{sample} mindestens doppelt so groß sein wie die maximale Frequenz im Signal f_{max}, die noch verarbeitet werden soll. Da die relevanten Sprachfrequenzen bis etwa 8 kHz reichen ([46], S. 65f), muss die Abtastfrequenz bei mindestens 16 kHz liegen. Zur Verarbeitung ist in fast allen Systemen, die innerhalb dieser Arbeit benutzt wurden, demnach eine Abtastrate von 16 kHz gewählt worden, anderfalls ist die Abtastrate explizit aufgeführt.

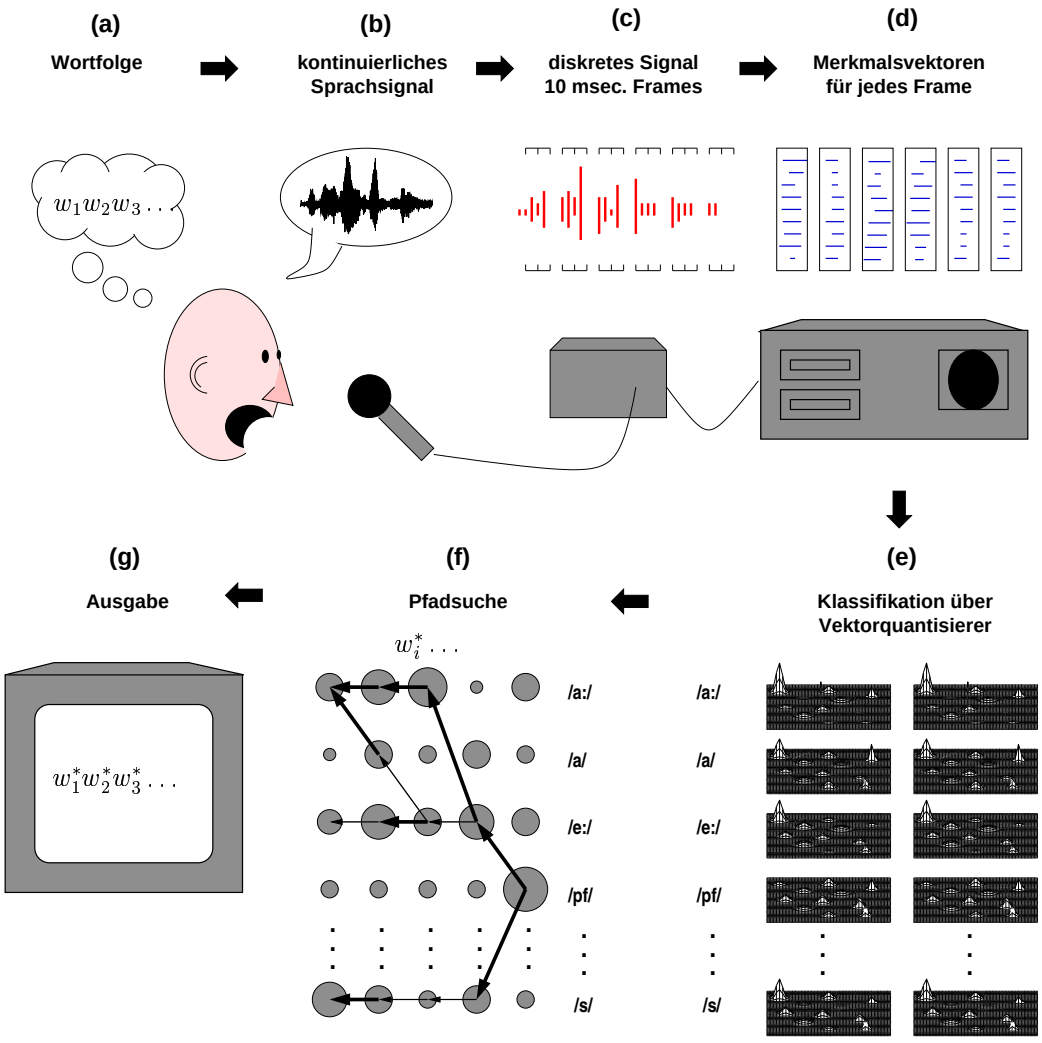


Abbildung 2.2: Ablaufschema eines Spracherkennungssystem. Ziel ist das Transkript (g) der ursprünglichen Wortfolge (a) aus dem gegebenen (kontinuierlichen) Sprachsignal (b) zu rekonstruieren. Dazu wird das Signal diskretisiert (c) und in Zeitscheiben (Frames) unterteilt, auf denen Merkmalsvektoren (d) berechnet werden. Diese werden über einen Vektorquantisierer auf Phonemmerkmale (e) abgebildet, auf denen dann die wahrscheinlichste Wortfolge ermittelt wird (f). Weitere Erläuterungen im Text.

Für die Quantisierung der Abtastwerte, die Überführung der kontinuierlichen Zahlwerte an den diskreten Zeitpunkten in einen diskreten und endlichen Wertebereich, reichen bei Sprache 11 Bit [47] aus. Da im Computer nicht mehr auf jedes

Bit geschaut werden muss, rundet man auf 16 Bit auf. Das entspricht 2^{16} Quantisierungstufen, also 65536 diskreten Zahlwerten. Im Gegensatz zur Abtastung nach Shannon ist die Quantisierung verlustbehaftet, kann aber bei der gewählten Anzahl der Quantisierungsstufen vernachlässigt werden: Das menschliche Ohr kann die Quantisierungssprünge nicht auflösen und hört eine perfekte Tonwiedergabe. Die gewählten 16 Bit entsprechen im Übrigen auch der Auflösung der Compact-Disk (CD), die als optimales Tonspeichermedium für allgemeine Anwendungen genutzt wird[8].

Zur Reduktion der Daten und Betonung der für die Erkennung essentiellen Anteile wird das Signal nach der Digitalisierung auf Merkmalsvektoren pro Zeitscheiben (Frames) reduziert (Abb. 2.2, (d)). Gängige Größen für die Frames sind 16 Millisekunden bei einer Abtastrate von 16 kHz, wobei benachbarte Frames überlappen und effektiv nur 10 Millisekunden neue Information bieten. Diese Größe umfasst 256 Abtastwerte und reicht aus, um das Signal quasi-stationär zu interpretieren, eine wichtige Voraussetzung der späteren statistischen Verarbeitung. Merkmalsberechnungen, die die schnelle Fouriertransformation (FFT, Fast Fourier Transformation) benutzen, können nur Dateneinheiten im Umfang von Zweierpotenzen bearbeiten. Jede kleinere Einheit müsste erst künstlich auf eine entsprechende Größe erweitert werden, ohne dass dabei zusätzliche Informationen abfallen würden. Daher wurden Abtastung und Framegröße entsprechend gewählt.

Auf diesen sich überlappenden Portionen des Sprachsignals werden repräsentative Merkmale berechnet. Dabei werden als Merkmale in gängigen Systemen fast ausschließlich Cepstral Koeffizienten (MFCC, mel-frequency weighted cepstral coefficients) oder Koeffizienten der linearen Vorhersage (LPC, linear predictive coding) benutzt.

Die **MFCCs** beruhen auf dem gemeinsamen Auftreten bestimmter Frequenzen bei bestimmten Lauten. Sie werden gewonnen, indem ein Signalausschnitt über eine Fouriertransformation vom Zeitraum in den Frequenzraum transformiert wird. Da die Fouriertransformation nur auf periodischen Signalen arbeitet, wird das Signalfragment kontinuierlich mit sich selbst fortgesetzt. Um dabei potentiell auftretende störende Sprünge an den Fragmentgrenzen 0 und $M-1$ zu vermeiden, wird das Signalstück mit einem zum Rand abflachenden Filter geglättet, beispielsweise mit der *Hammingfunktion* $a_\nu = 0.54 - 0.46\cos(2\pi\nu/(M-1)), \nu = 0, \ldots, M-1$. Zur einfacheren Verarbeitung wird die über die Transformation entstehende komplexe Zahldarstellung durch Betragsbildung in das Leistungspektrum bzw. durch zusätzliche Logarithmierung in das Leistungsdichtespektrum überführt. Da auch das menschliche Gehör die Phaseninformation nicht nutzt, wird ihr Verlust bei

[8] Mittlerweile werden allerdings auch Standards für Audio DVDs (Digital Versatile Disks) entwickelt, die für Musikexperten noch höhere Auflösungen vorsehen und damit (theoretisch) auch eine höhere Dynamik zur Verfügung stellen.

der Betragsbildung billigend in Kauf genommen. Die Frequenzachse wird nach *Mel* ([68]) 'gehörrichtig' verzerrt, das heißt analog der Verarbeitung im menschlichen Ohr werden unterschiedliche Anzahlen von Frequenzen mittels Dreiecks- oder Trapezfiltern zusammengefasst.

Durch die Rücktransformation des Leistungsdichtespektrums wird eine Art Frequenzanalyse des Spektrums im Pseudozeitbereich durchgeführt[9] und das Cepstrum[10] gewonnen. Die Cepstralkoeffizienten haben den Vorteil, dass sie praktisch unkorreliert vorliegen, also keine redundante Information beinhalten und weiterhin nach ihrer Wichtigkeit geordnet sind.

Den **LPC**-Merkmalen liegt die Idee zugrunde, dass ein Abtastwert f_n über eine Linearkombination einer bestimmten Anzahl m vorangegangener Abtastwerte $f_n = -\sum_{\mu=1}^{m} a_\mu f_{n-\mu}$ vorhergesagt werden kann. Die Koeffizienten a_μ modellieren das Bildungsprinzip der Abtastwerte und können als Modell des menschlichen Stimmapparates interpretiert werden: Eine Quelle (Lunge) erzeugt in unterschiedlichen Stärken (Luftdruck) entweder eine Impulsfolge (schwingende Stimmbänder bei leicht geöffneter Glottis) oder Rauschen (offene Glottis), das durch eine Reihe von Filterantworten (Vokaltrakt) moduliert wird. Die Berechnung der Koeffizienten erfolgt durch Minimierung des Vorhersagefehlers. Dadurch, dass mehr Abtastwerte in einem Frame sind als für die Geschichte der Berechnung benötigt werden, kann beispielsweise über das Gauss-Jordan-Verfahren oder auch die Levinsonrekursion eine Lösung bestimmt werden.

Zusätzlich zu diesen akustischen Merkmalen wird auch die Signalenergie herangezogen. Da sie in Abhängigkeit der Aufnahmebedingungen stark variieren kann, wird sie bzw. das zugrunde liegende Signal normiert.

Beide Arten von Merkmalen spiegeln nur die stationären Eigenschaften des Sprachsignals wider. Da wesentliche Bestandteile der Sprachlaute Änderungen im Frequenzverhalten sowie der Amplituden umfassen, müssen zusätzlich dynamische Eigenschaften berücksichtigt werden. Daher werden zur Ergänzung Ableitungen der Merkmale herangezogen. Eine typische Merkmalsvektorgröße ergibt sich somit zu 39 Merkmalen: Signalenergie + 12 Cepstralkoeffizienten sowie deren 1. und 2. Ableitung. Allerdings kann durch geschickte Auswahl der Merkmale die Vektorgröße reduziert werden (s. Kapitel 6).

Diese Merkmalsvektoren $Y = \vec{y}_1, \vec{y}_2, \ldots, \vec{y}_T$ werden durch einen Vektorquantisierer in (unspezifische) Klassen (Abb. 2.2, (e)) eingeteilt, über die dann die wahrscheinlichste Wortfolge $W^* = w_1^*, w_2^*, \ldots$ gesucht wird ((Abb. 2.2, (f)). Um W^* zu erhalten, wird die a posteriori Wahrscheinlichkeit $P(W|Y)$ (s. Gleichung 2.1) maximiert. Durch Einsatz der *Bayes*-Formel kann die Gleichung in einen

[9] Durch die Logarithmierung ist es nicht tatsächlich das Spektrum.

[10] engl. spectrum mit gespiegelter erster Silbe. Filter auf dem Cepstrum werden analog als *Lifter* bezeichnet.

Quotienten überführt werden, dessen Divisor die a-priori-Wahrscheinlichkeit der Merkmalsvektoren $P(Y)$ ist und nicht von W abhängt, über das maximiert wird.

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|Y) \tag{2.1}$$

$$\overset{Bayes}{=} \underset{W}{\operatorname{argmax}}\, \frac{P(W)P(Y|W)}{P(Y)}$$

$$\overset{P(Y)const.}{=} \underset{W}{\operatorname{argmax}}\, P(W)P(Y|W) \tag{2.2}$$

Entspricht W^* der geäußerten Wortfolge, liegt eine perfekte Erkennung vor. Bestimmte Abweichungen können allerdings ebenfalls toleriert werden, wenn das intendierte Dialogziel erreicht wird. Ansonsten ist ein Erkennungsfehler aufgetreten, den es zu minimieren gilt. Die Gleichung 2.2 lässt sich maximieren, wenn beide Produktkomponenten $P(W)$ und $P(Y|W)$ gemeinsam maximiert werden. $P(W)$ entspricht der a-priori-Wahrscheinlichkeit $P(Y|W)$ der Wortfolge W und kann über Sprachmodelle (s. Abschnitt 5.3.2) bestimmt werden. Die Wahrscheinlichkeit, die Vektorfolge Y zu beobachten, wenn die Wortfolge W gesprochen wurde, kann über Hidden-Markov-Modelle (HMMs) effektiv berechnet werden. Theorie und Einsatz der HMM werden im folgenden Abschnitt betrachtet.

2.3 Hidden-Markov-Modelle (HMM)

Da über den Aufbau und den Einsatz von Hidden-Markov-Modellen (HMM) eine Vielzahl Bücher und Artikel geschrieben wurden, beschäftigt sich dieser Abschnitt nicht mit der Realisierung im Detail, sondern führt auf einer abstrakteren Ebene in die Probleme und ihre Lösungen beim Einsatz der HMMs ein. Die weiteren Ausführungen orientieren sich primär an [61], weitere Informationen bieten [50] und vor allem [25].

Der Grundgedanke im Einsatz von HMMs für die Spracherkennung ist die statistische Modellierung der Sprachproduktion. Ein HMM in dieser Anwendung ist somit ein generierender, endlicher Automat. Dieser Automat ist definiert über seine N Zustandsknoten $S = \{1, 2, \ldots, N\}$. Welche Zustände zu bestimmten Zeitpunkten $t = 1, \ldots, T$ angenommen werden, kann über eine Folge von Zufallsvariablen $\underline{s} = s_1, s_2, \ldots, s_T$ beschrieben werden. Der Übergang von einem Zustand in den nächsten hat eine gewisse Wahrscheinlichkeit und hängt, nach der Markovbedingung, von der endlichen Zahl der vorher eingenommenen Zustände ab:

$$P(s_{t+1} = j(t+1)|s_t = j(t), s_{t-1} = j(t-1), \ldots, s_1 = j(1)) \tag{2.3}$$

Benutzt man HMMs erster Ordnung, reduziert sich die Anzahl der für den neuen Zustand berücksichtigten Vorgänger auf lediglich einen, die Zustandsübergangswahrscheinlichkeiten (Gleichung 2.3) vereinfachen sich zu $P(s_{t+1} = j(t+1)|s_t = j(t))$. Nimmt man an, dass der Prozess stationär, also nicht von der Zeit abhänging ist, kann als Kurzschreibweise aller Übergangswahrscheinlichkeiten eine $N \times N$ Matrix A gewählt werden:

$$A = [a_{ij}] = P(s_{t+1} = j|s_t = i), \quad 1 \leq i, j \leq N \tag{2.4}$$

Um die Traversierung der Zustände starten zu können, wird die Initialwahrscheinlichkeit jedes Zustands i gesetzt als $\underline{\pi} = [\pi_i] = P(s_1 = i)$. Für alle a_{ij} und natürlich auch die π_i gilt die Stochastizitätsbedingung, wonach keine negativen Werte angenommen werden und die Summe über alle Übergangswahrscheinlichkeiten eines Knotens 1 ergibt: $a_{ij} \geq 0, 1 \leq i, j \leq N$ und $\sum_j a_{ij} = 1$, sowie $\pi_i \geq 0$ für $1 \leq i \leq N$ und $\sum_i \pi_i = 1$. Damit ist die Traversierung des Automaten als stochastischer Prozess definiert.

Es existiert noch ein weiterer Prozess dieser Art, der eine Beobachtungsfolge $\underline{o}$ generiert. Jedesmal, wenn ein Zustand erreicht wird, wird ein $o_t \in O, 1 \leq t \leq T$ emittiert, das aus einem endlichen Vorrat von K Symbolen $O = \{O_1, O_2, \ldots, O_K\}$ für diskrete HMMs mit der vom Zustand abhängigen Wahrscheinlichkeit $P(o_t = O_{l(t)}|s_t = i)$ gewählt wurde. Für kontinuierliche HMMs (s. u.) entsprechen die beobachteten Werte Vektoren aus dem m-dimensionalen Merkmalsraum $\mathbb{R}^m$.

Auch dieser zweite stochastische Prozess lässt sich in einer Matrix kompakt repräsentieren. Für diskrete HMMS lautet die Matrix B der Emissionswahrscheinlichkeiten:

$$B = [b_i(o_t = O_l)] = [b_{il}] = P(o_t = O_l|s_t = i) \tag{2.5}$$

Wobei auch für B die Stochastizitätsbedingung $b_{il} \geq 0$ für $1 \leq i \leq N$ und $1 \leq l \leq K$ sowie $\sum_l b_{il} = 1$ gilt.

Ein HMM λ lässt sich vollständig als Tripel aus dem Vektor der Anfangswahrscheinlichkeiten $\underline{\pi}$ sowie den beiden Wahrscheinlichkeitsmatrixen A und B beschreiben. Da potentiell jeder Zustand jedes Symbol emittieren kann, jedoch lediglich die Symbole beobachtet werden können, ist nicht bekannt, in welchem Zustand sich ein HMM befindet. Daher werden diese Modelle als *Hidden*-Markov-Modelle bezeichnet.

Nach der Definition der HMMs stellen sich im weiteren vier Probleme im Umgang mit ihnen:

- 1. Wie wählt man die Anzahl Zustände und die Art ihrer Verbindungen für ein HMM aus?
- 2. Wie wahrscheinlich wird eine gegebene Beobachtungsfolge $\underline{o}$ von einem gegebenen HMM λ generiert?

- 3. Wie lautet die optimale Zustandsfolge $\underline{s}*$, anhand derer eine gegebene Beobachtungsfolge von einem gegebenen HMM am wahrscheinlichsten generiert wird?

- 4. Wie werden die Parameter eines gegebenen (initialen) HMMs für eine gegebene Beobachtungsfolge $\underline{o}$ angepasst, so dass $\underline{o}$ mit maximaler Wahrscheinlichkeit generiert wird?

Von den genannten Problemen lässt sich das erste nur heuristisch lösen, die anderen dagegen analytisch. Im weiteren werden diese Fragestellung separat behandelt.

2.3.1 Effiziente Berechnung einer Beobachtungsfolge

Um für ein gegebenes HMM $\lambda = (\underline{\pi}, \underline{A}, \underline{B})$ die Produktionswahrscheinlichkeit für eine bestimmte Oberservationsfolge $\underline{o} = o_1, o_2, \ldots, o_T$ zu berechnen, kann in einem direkten Ansatz für alle möglichen Pfade $\underline{s} = s_1 s_2 \ldots s_T$ der Länge T die jeweilige Pfadwahrscheinlichkeit $P(\underline{s}|\lambda) = \prod_{t=1}^{T} a_{s_{t-1}s_t}$, mit $a_{s_0 s_1} := \pi_{s_1}$ berechnet werden. Ebenso kann für jeden Pfad die Produktionswahrscheinlichkeit von $\underline{o}$ mit $P(\underline{o}|\underline{s}, \lambda) = \prod_{t=1}^{T} b_i(o_t)$ bestimmt werden. Die gesamte Produktionswahrscheinlichkeit des HMMs ergibt sich durch Aufsummierung der Wahrscheinlichkeiten für alle möglichen Pfade, also $P(\underline{o}|\lambda) = \sum_{\underline{s}} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(o_t)$. So verlockend einfach dieser Ansatz scheint, ist er in der Praxis nicht einsetzbar: Bei einem ergodischen, also komplett verzweigten, HMM kann zu jedem Zeitpunkt t jeder der N Zustände eingenommen werden, demnach existieren N^T Pfade der Länge T. Somit werden $(N^T - 1)$ Additionen und $(N^T - 1)(T - 1)$ Multiplikationen zur Berechnung eines HMMs benötigt, was einem unakzeptablen Rechenaufwand von $O(N^T)$ entspricht.

Berücksichtigt man allerdings, dass lediglich HMMs *erster* Ordnung benutzt werden, hängt für alle möglichen Pfade der Nachfolgezustand aller Knoten ausschließlich von ihrem direkten Vorgänger ab. Durch eine Rekombination der einzelnen Segmente erreicht man, dass es für jedes der T Pfadelemente lediglich N Vorgänger gibt. Ein großer Teil der oben durchgeführten Berechnungen ist also redundant. Realisiert wird dieses Verfahren im sogenannten *Forward-Backward Algorithmus*, dessen Namen von der geteilten Vorgehensweise rührt, die Beobachtungsfolge von vorne und von hinten zu iterieren.

Der *Forward* Teil benutzt die sogenannte Vorwärtsvariable $\alpha_t(i)$, die die Wahrscheinlichkeit repräsentiert, die partielle Observationsfolge $o_1, o_2, \ldots, o_t$ zu produzieren und zum Zeitpunkt t im Zustand i zu sein:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, s_t = i|\lambda) \tag{2.6}$$

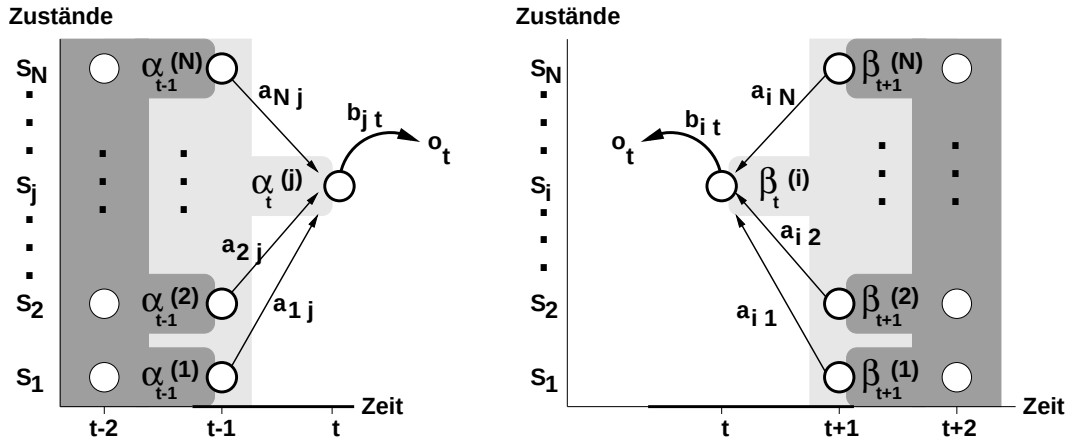


Abbildung 2.3: Rekursive Berechnung der Vorwärtsvariablen $\alpha_t(j)$ (links) und der Rückwärtsvariablen $\beta_t(i)$ (rechts). Die dunkelgraue Fläche stellt die Pfadstücke dar, die in den Variablen des vorherigen Schrittes repräsentiert sind. Die aktuellen Werte ergeben sich durch die Formeln 2.7 und 2.9.

Die Berechnung der α_t geschieht im Vorwärtsalgorithmus rekursiv:

$$\begin{array}{lll} 1) & \alpha_1(j) = \pi_j b_j(o_1) & 1 \leq j \leq N \\ 2) & \alpha_t(j) = (\sum_i \alpha_{t-1}(i) a_{ij}) b_j(o_t) & 1 \leq j \leq N, 1 < t \leq T \\ 3) & P(\underline{o}|\lambda) = \sum_j \alpha_T(j) & \end{array} \tag{2.7}$$

Analog erfolgt die Definition des Rückwärtsalgorithmus. Für die Berechnung von $P(\underline{o}|\lambda)$ reicht lediglich einer der beiden Algorithmen, da aber beide für die Berechnungen in Abschnitt 2.3.3 nötig sind, werden sie hier gemeinsam eingeführt.

Die Rückwärtsvariable $\beta_t(i)$ beschreibt die Wahrscheinlichkeit, die partielle Observationsfolge $o_{t+1}, o_{t+2} \ldots, o_T$ zu produzieren, falls man zum Zeitpunkt t in Zustand i ist:

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots, o_T | s_t = i, \lambda) \tag{2.8}$$

$$\begin{array}{lll} 1) & \beta_T(i) = 1 & 1 \leq i \leq N \\ 2) & \beta_t(i) = (\sum_j a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)) & 1 \leq i \leq N, T > t \geq 1 \\ 3) & P(\underline{o}|\lambda) = \sum_i \pi_i b_i(o_1) \beta_1(i) & \end{array} \tag{2.9}$$

Wie die Grafik 2.3 visualisiert, benötigt der Algorithmus zur Berechnung für jedes Symbol der beobachteten Folge die Betrachtung von allen N Vorgängern für jeden der N Knoten (falls ein ergodisches HMM vorliegt). Somit wird das Problem der Berechnung der Produktionswahrscheinlichkeit auf einen akzeptablen Aufwand von $O(N^2T)$ reduziert.

2.3.2 Wahrscheinlichste Zustandsfolge

Das zweite Problem in der Behandlung der HMMs, die Bestimmung, welche Zustandsfolge tatsächlich eine Beobachtungsfolge generiert hat, ist aufgrund der Tatsache, dass potentiell jeder Zustand jedes Symbol emittieren kann, in dieser Form nicht lösbar. Allerdings existiert mit dem *Viterbi- Verfahren* [66] die Lösung für das weniger strikte Problem, welche Zustandsfolge mit der höchsten Wahrscheinlichkeit eine Beobachtungsfolge generiert hat. Gesucht wird also

$$\begin{aligned} P(\underline{s}^*|\underline{o},\lambda) &= \max_{\underline{s}} P(\underline{s}|\underline{o},\lambda) \\ &= \max_{\underline{s}} \frac{P(\underline{s},\underline{o}|\lambda)}{P(\underline{o}|\lambda)} \\ &= \max_{\underline{s}} P(\underline{s},\underline{o}|\lambda) \end{aligned}$$

Die Lösung wird ähnlich wie beim *Forward-Backward* bestimmt. Da sich die Wahrscheinlichkeiten entlang der möglichen Pfade aufmultiplizieren und nur der beste Pfad bestimmt werden soll, reicht es aus, statt der Aufsummierung der partiellen Wahrscheinlichkeiten aller direkten Vorgänger lediglich deren wahrscheinlichsten Vertreter, das Maximum, zu behandeln. Dazu wird die Hilfsvariable $\delta_t(i)$ zur Repräsentation des bis zum Zeitpunkt t besten Ergebnisses für den Knoten i definiert:

$$\delta_t(j) = \max_{s_1 \ldots s_{t-1}} \{P(o_1, \ldots, o_t, s_1, s_2, \ldots, s_{t-1}, s_t = j|\lambda)\} \tag{2.10}$$

Es genügt jedoch nicht, den Wert des besten Pfades nach Abarbeitung der Beobachtungsfolge $\underline{o}$ berechnen zu können, zusätzlich muss nach Lokalisierung des Pfades eine Rückverfolgung über die tatsächlich gewählten Alternativen bis zum Anfang stattfinden. Dazu wird für alle Knoten j und für jeden Zeitpunkt t der gewählte Vorgängerknoten in der Variablen $\psi_t(j)$ abgelegt.

Der Viterbialgorithmus ist dann wie folgt definiert:

1) $$\delta_1(j) = \pi_j b_j(o_1) \qquad 1 \le j \le N \tag{2.11}$$
$$\psi_1(j) = 0$$

2) $$\delta_t(j) = \max_i\{\delta_{t-1}(i)a_{ij}\}b_j(o_t) \tag{2.12}$$
$$\psi_t(j) = \operatorname*{argmax}_i\{\delta_{t-1}(i)a_{ij}\}$$

3) $$P(s^*, \underline{o}|\lambda) = \max_j \delta_T(j) \tag{2.13}$$
$$s_T^* = \operatorname*{argmax}_j\{\delta_T(j)\}$$

4) $$s_t^* = \psi_{t+1}(s_{t+1}^*) \qquad T-1 \le t \le 1 \tag{2.14}$$

Nach der Initialisierung 2.11 wird für die gesamte Beobachtungsfolge die Berechnung 2.12 durchgeführt. Erst dann existiert das Ergebnis 2.13 für den wahr-

scheinlichsten Pfad. Der tatsächlich beste Pfad wird schließlich durch Rückverfolgung der Variablen ψ in 2.14 bestimmt. Dies bedeutet, dass eine Lösung erst vorliegen kann, wenn die komplette Beobachtungsfolge abgearbeitet wurde. Da für Interaktionen mit Spracherkennungssystemen eine schnelle Reaktion auf Benutzereingaben gewünscht wird, kann diese Vorgehensweise bei langen Äußerungen zu ungewollt hohen Verzögerungen führen. Dieses Problem kann durch ein inkrementell arbeitendes Verfahren gelöst werden, das in Abschnitt 3.3.2 näher betrachtet wird.

2.3.3 Bestimmung der Parameter

Nachdem in den vorigen beiden Abschnitten geklärt wurde, wie effektiv mit HMMs gerechnet werden kann, bleibt ein wesentliches Problem: Wie werden die Parameter des HMMs bestimmt? Da es keine Möglichkeit gibt, ein HMM direkt zu errechnen, wird für eine gegebene Beobachtungsfolge (der Trainingsstichprobe) ausgehend von einen existierenden HMM λ ein neues, besseres System $\hat{\lambda} = (\hat{\underline{\pi}}, \hat{A}, \hat{B})$ bestimmt, mit $P(\underline{o}|\lambda) \leq P(\underline{o}|\hat{\lambda})$. Dabei wird ähnlich dem *boots trappen* beim Compilerbau das Ursystem manuell konstruiert. Mit jedem Iterationsschritt kann das HMM System bei der Modellierung der Trainingstichprobe besser werden bzw. stagnieren. Kann keine signifikante (s. Abschnitt 2.4) Verbesserung mehr erreicht werden, wird das Training beendet. Das nach seinen Entwicklern als *Baum-Welch*-Training bekannte Verfahren berechnet zunächst $P(\underline{o}|\lambda)$ und passt dann in Abhängigkeit der dabei eingetretenen Übergänge und Emissionen die Parameter an.

Um eine einfachere Beschreibung der Berechnungsvorschriften zu realisieren, werden zwei Hilfsvariablen eingeführt: $\xi_t(i,j)$ und $\gamma_t(i)$.

$\xi_t(i,j)$ ist die a-posteriori-Wahrscheinlichkeit, zum Zeitpunkt t den Zustand i in Richtung j zu passieren bei Beobachtung von $\underline{o}$. Die Hilfsvariable ξ kann mit der Vorwärts- und Rückwärtsvariablen ausgedrückt werden:

$$\begin{aligned} \xi_t(i,j) &= P(s_t = i, s_{t+1} = j \,|\underline{o}, \lambda) \\ &= \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \end{aligned} \tag{2.15}$$

$\gamma_t(i)$ ist die Wahrscheinlichkeit, bei Beobachtung von $\underline{o}$ zum Zeitpunkt t in Zustand i zu sein. Die Variable kann als Summe über alle möglichen Übergänge von i in einen Nachfolgeknoten zum Zeitpunkt t beschrieben werden:

$$\begin{aligned} \gamma_t(i) &= P(s_t = i|\underline{o}, \lambda) \\ &= \sum_j \xi_t(i,j) \\ &= \sum_j \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \\ &\overset{(2.9)}{=} \alpha_t(i)\beta_t(i) \end{aligned}$$

Die neuen Initialwahrscheinlichkeiten $\underline{\hat{\pi}}$ werden mittels γ_1 berechnet und entsprechen der Wahrscheinlichkeit zum Startzeitpunkt 1 in Zustand i zu sein, gewichtet mit der Wahrscheinlichkeit, überhaupt die Beobachtungsfolge zu produzieren.

$$\begin{aligned}
\hat{\pi}_i &= P(s_1 = i|\underline{o}, \lambda) \\
&= \gamma_1(i) \\
&= \sum_j \xi_1(i,j) \\
&\overset{(2.15)}{=} \frac{\alpha_1(i)\beta_1(i)}{P(\underline{o}|\lambda)} \\
&\overset{(2.7.3)}{=} \frac{\alpha_1(i)\beta_1(i)}{\sum_k \alpha_T(k)}
\end{aligned}$$

Die neuen Übergangswerte $\hat{a}_{ij}$ ergeben sich durch den Quotienten aus der Wahrscheinlichkeit, den jeweiligen Übergang ij gemacht zu haben, und der Wahrscheinlichkeit überhaupt in i zu sein:

$$\begin{aligned}
\hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)}
\end{aligned}$$

Die optimierten Emissionswahrscheinlichkeiten $\hat{b}_{jk}$ lauten schließlich:

$$\begin{aligned}
\hat{b}_{jk} &= \frac{\sum_{t=1, o_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \\
&= \frac{\sum_{t=1, o_t=k}^{T} \alpha_t(j)\beta_t(i)}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)}
\end{aligned}$$

Zu beachten ist, dass sich sämtliche neuen Parameterberechnungen als Kombination der Vorwärts- und Rückwärtsvariablen ausdrücken und somit mit wenig mehr Aufwand als der in den Formeln 2.9 und 2.7 aufgeführte *Forward-Backward Algorithmus* realisieren lassen.

2.3.4 Modellierung der Verteilungsdichten

In der bisherigen Betrachtung der HMMs wurde stillschweigend vorausgesetzt, dass lediglich diskrete Symbole o_k aus einem endlichen Alphabet $\underline{O}$ von den HMMs emittiert werden (sogenannte diskrete HMMs). Da in der Sprachverarbeitung ein einzelner Wert zu wenig ist, um Sprache in einem Zeitabschnitt zu

charakterisieren, müsste mittels eines Vektorquantisierers (s. Abschnitt 2.3.5) eine Abbildung aus dem Merkmalsraum auf diskrete Symbole stattfinden. Da dieses Verfahren durch die damit verbundenen Quantisierungsfehler sehr früh eine Verfälschung der Eingabedaten für das System bedeutet, wird im Allgemeinen eine andere Vorgehensweise gewählt. Statt diskreter HMMs werden kontinuierliche HMMs (*continous density HMM*, CDHMM) verwendet. Anstelle der Matrix $B = [b_{ik}], b_{ik} = P(o_t = O_k | s_t = i)$ (s. auch Abbildung 2.5, C1) werden die Emissionen für eine Folge von oberservierten Vektoren $\underline{x} \in \mathbb{R}^m$ über Emissionsdichten definiert (Abb. 2.5, C2):

$$B = [b_j(\underline{x})], b_j(\underline{x}) = p(o_t = \underline{x} | s_t = j) \tag{2.16}$$

Da der zentrale Grenzwertsatz der Statistik aussagt, dass viele natürlich vorkommende Zufallsprozesse bei einer ausreichenden Zahl von Beobachtungen näherungsweise normalverteilt sind, werden die Emissionen der HMMs über Normalverteilungen modelliert. Um mehrere Häufungsgebiete annähern zu können, wird eine Linearkombination von Verteilungen benutzt:

$$b_j(\underline{x}) = \sum_{l=1}^{L_j} c_{jl} N(\underline{x} | \underline{\mu}_{jl}, \underline{K}_{jl}) = \sum_{l=1}^{L_j} c_{jl} g_{jl}(\underline{x}) \tag{2.17}$$

Die $g_{jk}(\underline{x})$ werden als Mischungskomponenten bezeichnet.

Somit bildet ein CDHMM einen 3-stufigen stochastischen Prozess: neben den Zustandsübergangswahrscheinlichkeiten wird eine Normalverteilung gemäß dem Gewicht c_{jl} gewählt und schließlich ein Ausgabevektor $\underline{x}_t$ generiert.

Um das *Baum-Welch*-Training für die CDHMM benutzen zu können, wird die Hilfsvariable $\zeta_t(j, l)$ definiert. Sie entspricht der Wahrscheinlichkeit, in Zustand j zu sein und die $l-$te der L_j Mischungsverteilungkomponenten für die Emission zu wählen. Um $\zeta_t(j, l)$ zu berechnen, müssen alle Zustandsübergänge von i nach j zum Zeitpunkt t berücksichtigt werden, eingeschränkt auf die l-te Normalverteilung:

$$\begin{aligned} \zeta_t(j,l) &= p(s_t = j, l | \underline{o}, \lambda) \\ &= \frac{p(s_t = j, l, \underline{o} | \lambda)}{p(\underline{o} | \lambda)} \\ &= \frac{1}{p(\underline{o} | \lambda)} \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} c_{jl} g_{jl}(o_t) \beta_t(j) \end{aligned}$$

Mit ζ lassen sich dann die neuen Gewichte der Verteilungen $\hat{c}_{jl}$ sowie die Parameter der Dichten, die Mittelwertvektoren $\hat{\mu}_{jl}$ und Kovarianzmatritzen $\underline{\hat{K}}_{jk}$, neu berechnen:

$$\hat{c}_{jl} = \frac{1}{\sum_t \gamma_t(j)} \sum_{t=1}^{T} \zeta_t(j, l) \tag{2.18}$$

$$\hat{\mu}_{jl} = \frac{1}{\sum_t \zeta_t(j,l)} \sum_{t=1}^{T} \zeta_t(j,l)\underline{x}_t \tag{2.19}$$

$$\hat{\underline{K}}_{jl} = \frac{1}{\sum_t \zeta_t(j,l)} \sum_{t} \zeta_t(j,l)(\underline{x}_t - \hat{\mu}_{jl})(\underline{x}_t - \hat{\mu}_{jl})^T \tag{2.20}$$

Mit diesen Modifikationen kann das *Baum-Welch*-Training auch für CDHMMs durchgeführt werden. Der Vorteil gegenüber den diskreten HMMs liegt in der besseren Modellierung des Sprachprozesses, der Nachteil in der weit höheren Zahl der zu bestimmenden Parameter. Pro Zustand j werden L_j Mischverteilungen mit jeweils dem Gewicht c_{jl}, dem m-dimensionalen Mittelwertvektor μ_{jl} und der $m \times m$-dimensionalen Kovarianzmatrix benötigt. Um dieses Problem einzuschränken, können anstelle der vollen Kovarianzmatritzen lediglich deren Diagonalen benutzt werden, was allerdings die Erkennungsleistung verschlechtern [26] und damit den Vorteil zunichte machen kann.

Die Vorteile beider Verfahren, die hohe Approximationsfähigkeit der CDHMMs einerseits und die handliche Parameterzahl der diskreten HMM andererseits, werden in den *semikontinuierlichen* HMMs (SCHMM) vereint. Wie bei den kontinuierlichen HMMs werden Normalverteilungen zur Modellierung der Emissionswahrscheinlichkeiten benutzt. Anstatt jedoch jedem Zustand i seine eigenen Verteilungen zu geben, teilen sich alle Zustände dieselben Dichten aus einem Pool und benutzen lediglich eigene Gewichte zur Verrechnung. Damit wird die Gleichung 2.17 vereinfacht zu:

$$b_j(\underline{x}) = \sum_{l=1}^{L_j} c_{jl} N(\underline{x}|\underline{\mu}_l, \underline{K}_l) = \sum_{l=1}^{L_j} c_{jl} g_l(\underline{x}) \tag{2.21}$$

Die Bestimmung der Mittelwertvektoren μ_l aus Gleichung 2.19 und Kovarianzmatritzen $\hat{\underline{K}}_{jk}$ aus Gleichung 2.20 werden zu:

$$\hat{\mu}_l = \frac{1}{\sum_t \sum_j \zeta_t(j,l)} \sum_{t=1}^{T} \sum_{j=1}^{N} \zeta_t(j,l)\underline{x}_t \tag{2.22}$$

$$\hat{\underline{K}_l} = \frac{1}{\sum_t \sum_j \zeta_t(j,l)} \sum_{t} \sum_{j=1}^{N} \zeta_t(j,l)(\underline{x}_t - \hat{\mu}_{jl})(\underline{x}_t - \hat{\mu}_{jl})^T \tag{2.23}$$

Mit dieser Gleichung wird der letzte Baustein für das Training von SCHMMs über eine Stichprobe definiert.

2.3.5 Vektorquantisierung

Sowohl bei der Modellierung von reellwertigen Merkmalen durch diskrete HMMs als auch bei der Auswertung der Verteilungen bei SCHMMs wird ein Vektorquantisierer (VQ) eingesetzt. Aufgabe des VQs ist die Zuordnung eines gegebenen

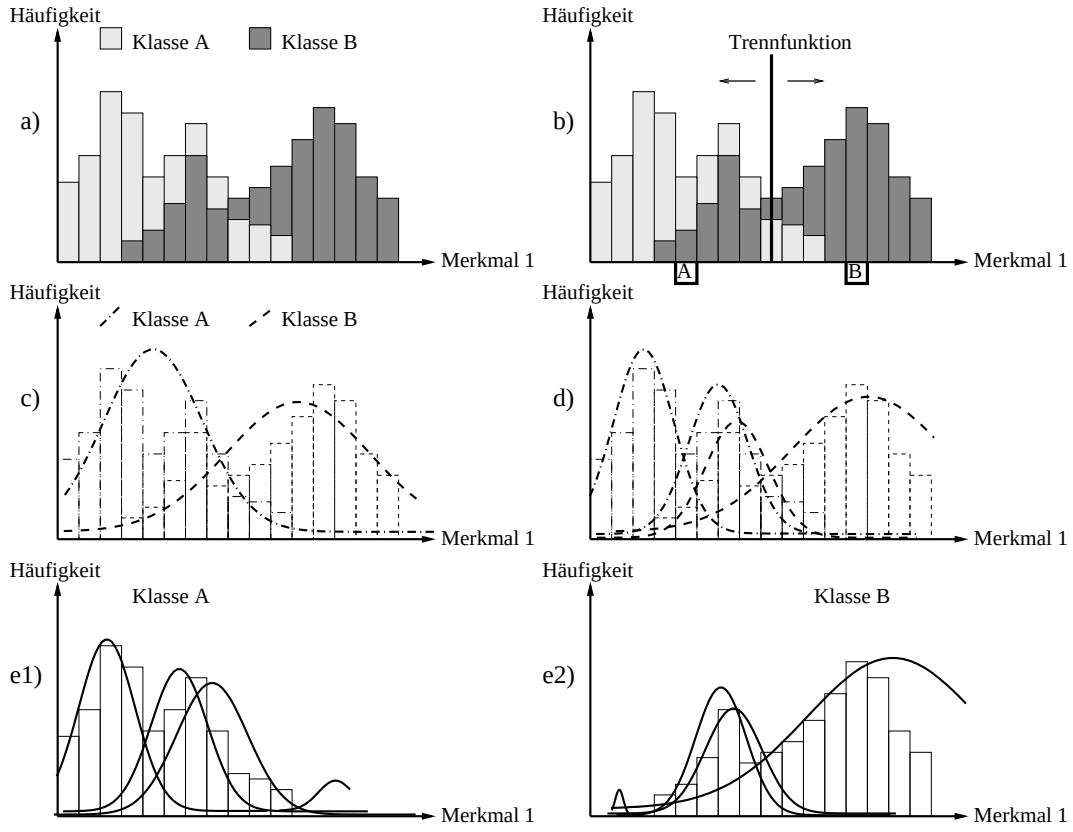


Abbildung 2.4: Partitionierung eines eindimensionalen Merkmalsraumes (nach [34]). a) Merkmalsverteilung zweier Klassen. b) Partitionierung mittels Trennfunktion. c) Klassenmodellierung über unimodale Dichten. d) Mischverteilungsansatz mit bimodalen Dichten e) Semikontinuierliche Variante, der gleiche Satz von Dichten mit unterschiedlichen Gewichten modellieren Klasse A (e1) und B (e2).

Vektors zu einem von R Häufungsgebieten (s. Abb. 2.4, a). Die zentralen Punkte (=Kodewörter) (s. Grafik 2.4, b) A und B) der Häufungsgebiete stehen dabei als Repräsentanten für ihren Cluster. Wird der VQ überwacht trainiert, entsprechen die Cluster Klassen.

Die Zuordnung der Eingabevektoren geschieht anhand eines Abstandsmaßes, beispielsweise des euklidischen Abstands $d(\underline{x}_1, \underline{x}_2) = \sqrt{(\underline{x}_1 - \underline{x}_2)^T(\underline{x}_1 - \underline{x}_2)}$. Um einen VQ zu konstruieren, wird iterativ eine Trainingsstichprobe ausgehend von einem initialen Kodebuch (=Gesamtheit der Kodewörter) in die gegebene Anzahl Klassen aufgeteilt und anschließend die neuen Kodewörter als Zentroiden der Klassen berechnet. Das Verfahren wird so lange durchgeführt, bis ein Optimalitätskriterium wie minimaler Gesamtintraklassenabstand oder maximaler

Gesamtinterklassenabstand erreicht wird. Eine ausführliche Betrachtung verschiedener Algorithmen zur Konstruktion von VQs findet sich bei [18]. Anstelle der Kodewörter kann eine Klasse auch durch eine hochdimensionale Verteilung $p(\underline{x})$ repräsentiert werden. In diesem Fall besteht die Möglichkeit, bei der Quantisierung statt einer harten Einklassenentscheidung auch eine gewichtete Zuordnung zu mehreren Klassen zu realisieren. Für die Dichten werden häufig Normalverteilungen benutzt (s. Abb. 2.4 c) und d)). Ein solcher 'weicher' VQ wird zur Auswertung der Emissionswahrscheinlichkeiten bei den SCHMMs benutzt (s. Abb. 2.4 e)). Anders als beim VQ-Einsatz bei den diskreten HMMs kann die Optimierung des Quantisierers abwechselnd mit dem *Baum-Welch*-Training durchgeführt werden und ist somit nicht anfällig für massive Quantisierungsfehler.

2.3.6 Wortmodellierung

Die vorherigen allgemeinen Betrachtungen der HMMs sind in dieser Form für diverse Mustererkennungsanwendungen anwendbar. In diesem Abschnitt werden die speziellen Einschränkungen erläutert, die für den Einsatz in Spracherkennungssystemen gemacht werden.

Es gibt verschiedene Möglichkeiten, HMMs aufzubauen und die Zustände zu verbinden, im Extremfall ist jeder Zustand mit jedem weiteren verknüpft (s. Abb. 2.5, B). Da in der Spracherkennung HMMs zur Modellierung von Wortuntereinheiten, beispielsweise Phonemen (s. Abschnitt 5.3.1), benutzt werden, deren Äußerung in verschiedene aufeinanderfolgende Phasen zerlegt werden kann, wählt man lineare HMMs, die zyklenfrei in einer Links-Rechts-Topologie angeordnet werden.

Somit reduziert sich die $N \times N$-große Übergangsmatrix A auf die obere Dreiecksmatrix der Größe $\frac{N^2+N}{2}$. Die Merkmale werden über zeitgetaktete Frames gerechnet, deren Abfolge nur kleinere Zustandssprünge zur Sprechgeschwindigkeitskompensation gestattet. Daher werden die Modelle noch weiter beschnitten, indem lediglich Selbstübergänge (a_{ii}) und Übergänge zum nächsten (a_{ii+1}) und übernächsten Zustand (a_{ii+2}) erlaubt werden (s. Abb. 2.5, C). Da jedes Muster einen Anfang und ein Ende besitzt[11] werden auch die HMMs mit einem dedizierten Start- und Endeknoten versehen. Dadurch reduziert sich die Zahl der Übergänge auf $3N - 2$ Kanten und die Definition der Anfangswahrscheinlichkeiten $\underline{\pi}$ auf einen Vektor, der lediglich an der Position des Anfangsknoten eine 1 enthält, ansonsten mit 0 gefüllt ist.

[11] Bei sehr langen Äußerungen können in bestimmten zeitlichen Abständen Abschnittsenden forciert werden (s. Abschnitt 3.3.2).

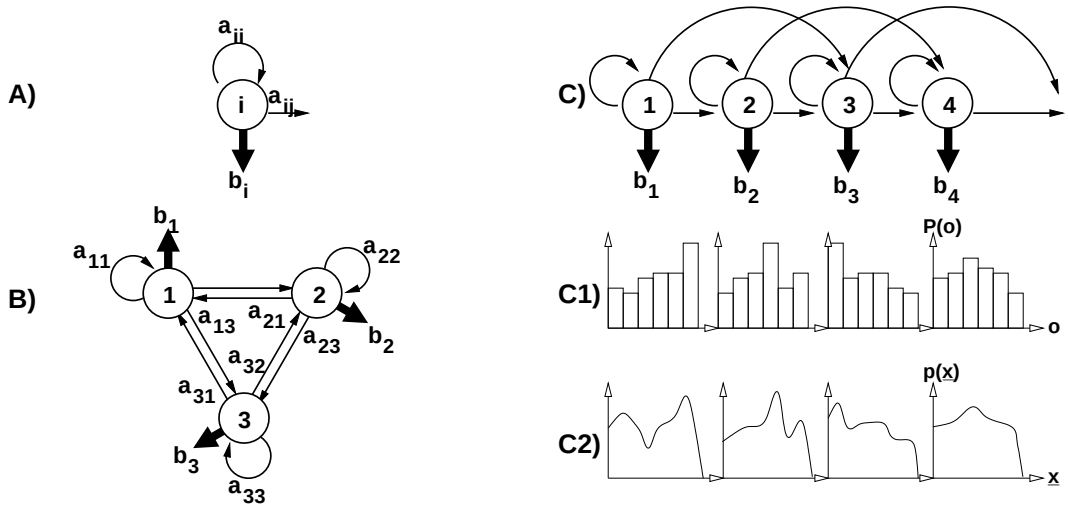


Abbildung 2.5: Verschiedene HMMs, A) isolierter Zustand i, B) ergodisches Modell, C) Bakis Modell mit C1) diskreten C2) kontinuierlichen Ausgabeverteilungen. Aus Gründen der Übersichtlichkeit ist C) nicht vollständig beschriftet.

2.3.7 Zuordnung des Sprachsignals und der HMMs

Wie erwähnt, werden die Zustände der HMMs im Takt der Sprachsignalframes traversiert. Da die Dauer der Frames in der Größenordnung von 10 Millisekunden liegt, wird zur Modellierung eines kompletten Wortes von 0.5 Sekunden Dauer (z.B. *Navigation*) ein HMM mit etwa 20 Zuständen[12] benötigt. Solche *Ganzwortmodelle* haben eine hervorragende Modellierungseigenschaft für ihr jeweiliges Wort, benötigen zu ihrem Training aber auch genügend Vorkommen dieses Wortes. Da für praktisch jeden Korpus Wörter existieren, die nicht häufig genug im Trainingsmaterial zu finden sind, muss ein besseres Verhältnis zwischen Daten und repräsentierten Einheiten geschaffen werden. Durch Einführung von *Wortuntereinheiten* (engl. subwordunits, SWU) werden Wörter in kleinere Einheiten zerlegt, die jeweils durch eigene HMMs dargestellt werden und somit mehr Trainingsmaterial erhalten können. Als kleinste sinnunterscheidende Einheit bilden die *Phoneme*, die Atome einer Sprache (s. Kapitel 5.3.1 für die Definition und Anhang A.1 für die Umschrift), die minimalen SWUs. Im Deutschen existieren nur etwa 50 Phoneme, so dass lediglich diese Anzahl an Modellen, die phonemabhängig 1 bis 10 Zustände besitzen, trainiert werden müssen.

Leider können die in der Sprache auftretenden Koartikulationseffekte durch diese Modellierungsart nicht erfasst werden, da sie mehr Kontext benötigen. Dieser

[12] Durch die Selbstübergänge können Zustände jeweils mehr als 1 Frame konsumieren.

wird bei den *Triphonen* durch Berücksichtigung des linken und rechten Nachbarn eines zentralen Phonems gegeben. Das Wort `Auto` kann aus folgenden vier Triphonen zusammengesetzt werden: `#/a/U a/U/t U/t/o t/o/#`, mit `#` als künstlichem Delimiter für Wortgrenzen. Der guten Modellierungseigenschaft und hohen Rekombinierbarkeit der Triphone steht allerdings ihre große Zahl gegenüber: bei 50 Phonemen existieren etwa $50^3 = 125000$ Triphone[13]. Wie bei den Ganzwortmodellen ist die Chance, sämtliche Triphone im Training in ausreichender Zahl zu sehen, gering. Daher ist eine Zusammenfassung verschiedener Triphone zu größeren Einheiten üblich. Phonologisch motiviert werden verschiedene Phoneme zu Lautklassen zusammengefasst, etwa *Plosive* wie `p`, `t` und `k` oder *Affrikaten* wie `pf` und `ts` (s. Anhang A.1 für eine mögliche Einteilung). Sind einzelne Triphone nicht häufig genug beobachtet worden, teilen sie sich die Parameter mit ihren generalisierten Versionen, etwa `a/U/t` mit allen Triphonen, die als rechten Kontext einen Plosivlaut haben. Ein verwandter Ansatz kann auch datengetrieben durchgeführt werden. Dabei bestimmt die Ähnlichkeit verschiedener Triphon-HMMs und die ihnen zugrunde liegende Datenmenge, ob sie zusammengefasst werden können. Lee gibt in [38], S. 104 ff. eine ausführliche Beschreibung von bisher untersuchten Ähnlichkeitsmaßen. Wie auch immer die Zusammenfassung durchgeführt wird, mit ihr können Triphone auch bei etwas zu geringer Datenlage traininert und eingesetzt werden.

2.3.8 Darstellung von Wahrscheinlichkeiten

Wie bereits in den vorherigen Abschnitten deutlich wurde, bedingt der Umgang mit HMMs Multiplikationen von langen Folgen von Wahrscheinlichkeiten. Da dabei immer kleinere Zahlwerte entstehen, bergen Implemetierungen auf Computern die Gefahr, früher oder später die Rechengenauigkeit der Maschine zu unterlaufen und somit keine korrekten Ergebnisse mehr zu produzieren. Um dieses Problem zu lösen, gibt es zwei Möglichkeiten: entweder wird vor jedem Rechenschritt eine Skalierung durchgeführt oder anstelle der Wahrscheinlichkeiten werden ihre Logarithmen verwendet.

Skalierung der Wahrscheinlichkeiten

Da die Wahrscheinlichkeiten mit fortschreitendem t exponentiell gegen Null laufen, bietet es sich an, die Skalierung ebenfalls von t bzw. den Zeitschritten abhängig zu machen. In [51] wird zur Berechnung der Vorwärtsvariablen der Skalierungsterm c_t eingeführt mit

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{2.24}$$

[13] Da nicht alle möglichen Kontexte existieren, ist dies nur ein Näherungswert.

und die skalierte Variante den Vorwärtsvariablen $\tilde{\alpha}_t(i) = c_t \alpha_t(i)$ definiert. Analoges gilt für die Rückwärtsvariable $\tilde{\beta}_t(j)$. Die so gewählte Skalierung wirkt zu jedem Zeitpunkt gegen das Verschwinden der Terme und erlaubt so die maschinelle Verarbeitung. Da die Skalierung vor jedem Rechenschritt ausgeführt werden muss, entsteht allerdings ein nicht zu vernachlässigender zusätzlicher Rechenaufwand.

Logarithmierung der Wahrscheinlichkeiten

Beim Übergang auf die logarithmische Darstellung müssen alle Formeln, die Wahrscheinlichkeiten benutzen, umgestellt werden. Sämtliche Multiplikation werden zu Additionen, sämtliche Additionen erfordern zuvor eine (aufwändige) Delogarithmierung der Wahrscheinlichkeiten.

Da im Viterbi-Alogrithmus keinerlei Additionen vorkommen, kann die Bestimmung der $\delta_t(j)$ in 2.12 einfach angepasst werden:

$$2a) \quad \delta_t(j) = \max_i\{\delta_{t-1}(i) + a_{ij}\} + b_j(o_t) \tag{2.25}$$

Bei der Berechnung der Vorwärts- und Rückwärtsvariablen in den Gleichungen 2.7 und 2.9 werden dagegen Summen benutzt. Um die Addition von zwei logarithmisch repräsentierten Wahrscheinlichkeiten $\log_b(p_1)$ und $\log_b(p_2)$ durchführen zu können, müssen zuerst beide Werte delogarithmiert, aufaddiert und schließlich wieder logarithmiert werden. Da logarithmische Operationen zeitaufwändig sind und innerhalb der Berechnung häufig benutzt werden, gibt es mit der *Kingsbury-Rayner*-Formel eine effektive Lösung:

$$\log_b(p_1 + p_2) = \log_b(p_1) + \log_b(1 + b^{log_b(p_2) - \log_b(p_1)}) \tag{2.26}$$

Da sowohl $\log_b(p_1)$ und $\log_b(p_2)$ bekannt sind, muss lediglich der Teilterm $\log_b(1 + b^{(\cdots)})$ ausgewertet werden. In gängigen C-Compilern ist eine solche Funktion bereits über *Lookup table* effektiv implementiert, damit können auch große Summen von logarithmierten Werten schnell berechnet werden.

In [61] findet eine ausführliche Betrachtung der Darstellungsgenauigkeit bei der Logarithmierung von Wahrscheinlichkeiten statt: Danach bedeutet eine 16 Bit Repräsentation bei einem maximalen Fehler von 1 Promille die Darstellung von etwa 57 Dezimalstellen. Dies ist deutlich mehr als mit der üblichen Zahldarstellung erreicht werden kann.

2.4 Bewertung eines Erkenners

Bei der Entwicklung eines Spracherkennungssystems liegt das Hauptaugenmerk natürlich auf der Qualität der Erkennungsergebnisse. Für andere Aspekte zur

Bewertung des Systems gibt es intuitive Maßzahlen. Die Größe des Moduls etwa kann in Kilobyte Arbeitsspeicherbelegung des Programmes und seiner Daten erfolgen, die Verarbeitungsgeschwindigkeit in Zeitverzug zwischen Äußerung und Ergebnis ins Verhältnis gesetzt zur Leistungsfähigkeit der Hardwareplattform.

Die Bestimmung der Güte der Erkennung ist dagegen komplexer. Ihre einfachste Form, ‚Äußerung erkannt/nicht erkannt', ist im Normalfall zu grob. Eine feinere Granularität erreicht man durch Bestimmung und Auszählung von verschiedenen Fehlerarten anhand einer Stichprobe. Um möglichst objektive Ergebnisse zu bekommen, kann die Teststichprobe sorgfältig ausgewählt werden: Sie muss vom Trainingsmaterial disjunkt sein, da sich der Erkenner ansonsten im Training auf das Testmaterial spezialisieren kann und zwar gute Ergebnisse darauf produziert, diese Leistung aber nicht notwendigerweise im späteren Einsatz erbringt. Für einen sprecherabhängigen Erkenner sollten es unterschiedliche Äußerungen in Test und Training sein, für sprecherunabhängige unterschiedliche Sprecher, für umgebungsvariante Systeme verschiedene Arten von Geräuschkulissen. Es empfiehlt sich, sämtliche Systemparameter auf den späteren Einsatz anzupassen, etwa

der Typ des Mikrofons evtl. sogar verschiedener Mikrofone, falls das spätere System nicht an ein bestimmtes gebunden sein soll,

die Art der Sprache, wird sie wie im Diktat eher gelesen oder ist sie spontan mit Phänomenen wie Hesitationen, ungrammatikalischen Äußerungen, Wortabbrüchen?

die Beschaffenheit der akustischen Umgebung, gibt es kontinuierliche und impulshafte Störgeräusche, wie ist der Signalrauschabstand?

die Definition des Lexikons, ist das Vokabular klein/groß oder gar erweiterbar, müssen unbekannte Wörter behandelt werden?

Zusätzlich muss eine Mindestgröße der Teststichprobe vorausgesetzt werden, um signifikante Änderungen messen zu können.

Die Standardmaße für die Erkennungsrate werden auf Wortebene bestimmt. Potentielle Fehler (und ihre Abkürzungen) sind somit bei Vergleich zwischen Erkennungsergebnis und Referenztext *Vertauschungen (S, Substitution)*, *Einfügungen (I, Insertion)* und *Löschungen (D, Deletion)* von Wörtern. Das Zuordnen des Ergebnisses und der Referenz wird mit dem Levenshteinabstand bewertet und über einen Ansatz des dynamischen Programmierens erreicht (s. Abschnitt 5.3.2). Über diese Fehler kann die Wortakkuratheit(WA) definiert werden:

$$WA = 100(1 - \frac{\#S + \#D + \#I}{\#W}) \tag{2.27}$$

#S,#D,#I sind die Anzahl der einzelnen Fehler, #W die Anzahl der Wörter im Referenztext. Die höchste Wortakkuratheit von 100% wird nur bei Übereinstimmung von gesprochenen und erkannten Wörtern erreicht. Da im Prinzip beliebig viele Einfügungen generiert werden könnten, auch mehr als Wörter in der Referenz sind, kann die Wortakkuratheit im schlechtesten Fall negativ werden.

Die früher häufiger benutzte prozentuale Angabe der richtigen Wörter (*Word Correct, WC*) verzichtet auf die Verrechnung der Einfügungen, sie entspricht Gleichung 2.27 ohne den Term der Einfügungen $+\#I$. Sie hätte im Fall einer Ausgabe, in der für jedes gesprochene Wort einfach das gesamte Lexikon erkannt wurde, einen Wert von 100%. Im Gegensatz zur WA existiert für die WC ein unterer Grenzwert. Er liegt bei 0%, d.h. kein richtiges Wort wurde erkannt. Aber wie im obigen Fall erläutert, ist die WC nicht sehr aussagekräftig.

Ein einfaches Beispiel aus der Domäne des fahrzeugbasierten Erkenners illustriert verschiedene Probleme der Fehlerbestimmung (C=Korrekt):

Referenztext	**Erkannter Text**	**Fehler**
Computer	Computer	C
	4	I
Telefonnummer	Telefon	S
	Nummer	I
5	5	C
5	5	C
5		D
7	6	S
6	6	C
0	0	C
0	0	C
wählen	wählen	C
	4	I

Beim Finden der korrespondierenden Wörter gibt es in der Regel mehrere Alternativen, so kann das Paar *Telefonnummer/Telefon* als Substitution und *Nummer* als Insertion gewertet werden, aber genauso kann auch *Telefon* die Insertion und *Nummer/Telefonnummer* die Substitution sein. Analog kann jede einzelne *5* des 5er Blocks weggelassen worden sein. Aber unabhängig davon, welche der Zuordnungen gewählt wird, die Anzahl und Art der Fehler bleibt identisch[14]: sieben Wörter sind richtig erkannt worden bei sechs Fehlern: drei Einfügungen, zwei

[14] Wählt man für die Bestimmung des Levenshteinabstandes unterschiedliche Gewichte für Substitution, Deletion und Insertion, stimmt diese Aussage nicht mehr. Beispielsweise kann bei einer hohen Gewichtung von Substitutionen diese günstiger durch eine Löschung und eine Einfügung erreicht werden.

Vertauschungen und eine Löschung. Die Beispielreferenz besteht aus 10 Wörtern, somit ergibt sich die WA zu $100(1 - \frac{2+1+3}{10} = 40\%$[15]. Die WC wirkt dagegen viel positiver: $100(1 - \frac{2+1}{10}) = 70\%$.

Das Beispiel illustriert ein weiteres interessantes Problem der Spracherkennung: Da kontinuierliche Sprache außer zur Betonung, Satztrennung oder bei Wortfindungsproblemen nur wenige Pausen enthält, können Komposita durchaus in ihren Komponenten erkannt werden. Je nachdem, wie die nachgeschaltete Verstehenskomponente damit umgehen kann, ergeben sich zusätzliche Erkennungsfehler, die schon beim Design des Erkennungslexikons vermieden bzw. durch Nachschalten einer morphologischen Komponente (s. [3] und [4] für Aufbau und Realisierung) kompensiert werden können.

Wird als Erkennungsziel nicht die wahrscheinlichste Wortkette, sondern ein Worthypothesengraph (WHG) angestrebt, verkompliziert sich die Güteangabe des Spracherkenners. Ein WHG ist gerichtet und enthält in seinen Knoten entweder mehrere Zeitpunkte oder einen Zeitpunkt mit kurzen Intervallen und an seinen Kanten Worthypothesen, gewöhnlich mit einer Bewertung versehen. Benachbarte Kanten spiegeln alternative Wörter wider, die verschiedenen Pfade durch den Graphen somit potentielle Erkennungsergebnisse. Die Aufgabe der Auswahl, wie eine Äußerung tatsächlich lautete, wird mit dem Graph zu einer nachgeschalteten Verstehenskomponente übergeben. Evaluiert man lediglich den Erkenner, erreicht er eine perfekte Erkennungsleistung, wenn im WHG ein einziges Mal die korrekte Äußerung zu finden ist. Um die Ergebnisse nicht ähnlich wie bei der WC zu verschleiern, muss die WHG-Dichte angegeben werden, eine Zahl, die die durchschnittliche Anzahl an alternativen Kanten im Graphen widerspiegelt. Dennoch sind Vergleiche zwischen WHG-Systemen mit unterschiedlicher Dichte schwierig. Ist ein WA von 85% bei einer Dichte von 4.5 besser als 80% bei einer Dichte von 3? Die Antwort darauf kann nur durch die Evaluation des gesamten Systems, also mit Berücksichtigung der nachgeschalteten Analysekomponente, in Hinsicht auf WA und Verarbeitungsgeschwindigkeit gegeben werden.

Neben der Güte auf Wortebene kann man die Erkennungsleistung auch auf Äußerungsebene messen, indem das Verhältnis von 100% korrekt erkannter Sätze zur Gesamtzahl der Sätze berechnet wird. Dieses *Sentence right, SR*, genannte Maß ist allerdings nur für Diktieranwendungen interessant. Bei anderen Anwendungen wie Auskunftsystemen ist ein wichtigerer Faktor, ob die Anfrage eines Kunden korrekt bearbeitet werden konnte, notfalls auch nach Rückfragen durch eine Dialogkomponente. Eine hundertprozentige Erkennung ist nicht nötig, wenn der Benutzer im Gesamtsystem, in dem der Spracherkenner nur eine Komponente ist, sein intendiertes Ziel erreicht. Daher sind einige EntwicklerInnen dazu übergegangen, nicht nur ihre Spracherkenner, sondern die Performanz ihrer gesamten Systeme zu messen, wie beispielsweise beim automatischen Auskunftski-

[15] Anmerkung: ein sehr schlechtes Ergebnis, das in Realität glücklicherweise übertroffen wurde.

osk *MASK*, der bis zu 99% aller Anfragen korrekt bearbeiten konnte, obwohl die Spracherkennung eine Fehlerrate von 15% aufwies ([36]).

Signifikanzbetrachtung

Die Berechnung der Wortakkuratheit hängt stark von der gewählten Teststichprobe ab. Geht man davon aus, dass die Teststichprobe und das Trainingsmaterial ursprünglich aus demselben Korpus kommen, sind keine Überlegungen in Richtung Lexikonauswahl nötig, man teilt einfach den ursprünglichen Korpus in Test- und Trainingsmaterial. Damit wird die Frage aufgeworfen, wie groß eine Stichprobe sein muss, um eine aussagekräftige Bewertung des Erkenners zu liefern. Und, damit zusammenhängend, wie stark muss sich die WA zweier Systeme unterscheiden, um sagen zu können, dass eines tatsächlich besser ist als das andere?

Da Teststichproben immer nur eine Teilmenge der potentiell möglichen Erkennereingaben sind, kann die angegebene WA nur eine Schätzung der tatsächlichen Leistungsfähigkeit des Systems sein. Wie gut diese Schätzung ist, lässt sich über Konfidenzintervalle angeben (s. [34], S. 163 ff): Durch Wiederholung des Tests mit unterschiedlichen Stichproben gewinnt man ein Intervall von Ergebnissen, das eine untere und obere Grenze G_u und G_o besitzt und in dem mit der Wahrscheinlichkeit γ die tatsächliche WA liegt. γ ist das Signifikanzniveau und wird i.d.R. auf 95% oder 99% festgesetzt. Das g_γ bezeichnet dann die Grenzen, in denen sich die Zufallssvariable $\#C$ der korrekt erkannten Wörter in Bezug gesetzt zu der ermittelten Wordakkuratheit WA und der Größe der Teststichprobe $\#W$ bewegt:

$$g_\gamma \in R : P(-g_\gamma \leq \frac{\#C - \#W \cdot WA}{\sqrt{\#W \cdot WA \cdot (1 - WA)}} \leq g_\gamma) \leq \gamma \tag{2.28}$$

Der untere bzw. obere Wert des Grenzintervalls ergibt sich nach einigen Umformungen (s. [34] für Details) zu

$$G_{u,o}(WA) = WA + / - g_\gamma \sqrt{\frac{WA(1 - WA)}{\#W}} \tag{2.29}$$

Somit kann bei der Evaluation eines Systems ein Signifikanzniveau berechnet werden, das objektiv Auskunft darüber gibt, ob ein Vergleichssystem besser, schlechter oder ähnlich gut ist.

3 Probleme der Spracherkennung

Nachdem im vorigen Kapitel die grundlegenden Verfahren zum Aufbau und Training eines Spracherkennungssystems erläutert wurden, liegt der Focus dieses Kapitels auf zwei interessanten Problemen, die beim Einsatz eines Systems auftreten:

1. Wie kann die Qualität der Spracherkennung bei neuen, nicht im Training vorkommenden Sprechern gewährleistet werden? Die frühen Spracherkenner waren eingeschränkt in der Zahl ihrer Benutzer (in der Regel einer) und der Größe des Vokabulars. Neue Benutzer wurden anfangs durch direkte Manipulation der Erkenner eingestellt, später durch langwierige *Enrollments*, dem Vorlesen definierter Texte zur automatischen Extraktion sprechspezifischer Parameter, die dann zur Adaption der Erkenner genutzt wurden.

2. Auf welche Art und Weise wird in dem System mit unbekannten Wörtern umgegangen? In den meisten Systemen werden unbekannte Wörter nicht einmal ignoriert, sondern auf bekannte abgebildet.

Beides sind Vorgehensweisen, die in modernen Systemen nicht mehr benutzt werden. Im Folgenden wird aufgezeigt, wie dies realisiert werden kann.

3.1 Sprecheradaption

Ein wichtiger Aspekt im Einsatz von Spracherkennungssystemen ist bisher lediglich angerissen worden: die Sprecheradaption. Zur Vereinfachung der Erkennungsleistung wird versucht, *Freiheitsgrade* im Signal auszuschalten. Neben der Lautstärke, dem Signal-Rauschabstand und den Umgebungsgeräuschen (die alle in Kapitel 4 behandelt werden) ist ein variabler Faktor der Sprecher selber. Damit die Sprache einer Person erkannt werden kann, obwohl das System bisher keine Daten von ihr erhalten hat, wurden Merkmale gewählt, die möglichst sprecherunabhängige Eigenschaften der Sprache repräsentieren. Allerdings können diese nicht alle Effekte ausreichend kompensieren, die sprecherübergreifend, aber auch innerhalb eines Sprechers, variieren. Daher arbeitet ein Erkenner für unterschiedliche Sprecher mit unterschiedlicher Qualität, genauso wie für denselben Sprecher bei verschiedenen Wiederholungen der gleichen Äußerung.

Zur Kompensation dieser Effekte gibt es unterschiedliche Problemlösungsansätze, die teilweise aber nicht nur speziell die sprachspezifischen Variabilitäten,

sondern auch andere Arten von Signalabweichungen behandeln. Die Verfahren setzen auf verschiedenen Ebenen an: Direkt beim Signal, bei den Merkmalen oder auf Ebene der HMMs. Zu den gängigen Verfahren gehören:

- die Vokaltraktlängennormierung auf Signalebene
- die *Maximum Likelihood Linear Regression* auf Ebene der Codebuchklassen
- die Dauermodellierung beim Training der HMMs

3.1.1 Vokaltraktlängennormierung

Die Vokaltraktlängennormierung (VTLN) versucht die durch die unterschiedlich geformten Vokaltrakte der Sprecher (s. Abschnitt 2.2, LPC) variienden Frequenzspektren in der Sprache zu normieren. Dazu wird das Frequenzspektrum eines Signals linear verzerrt, um es auf das einen Normsprechers abzubilden. Der Verzerrungsfaktor α kann aus einem Sprachfragment des zu normalisierenden Sprechers auf verschiedene Arten gewonnen werden:

- Die erste Möglichkeit ist relativ simpel: Über das Iterieren verschiedener Faktoren aus einer Menge von Werten[1] und Auswahl desjenigen, der die Produktionswahrscheinlichkeit des Äusserungsfragments für das gegebene HMM-System maximiert [49].
- Etwas komplizierter, aber ohne den Nachteil der mehrfachen Erkennung, ist die Methode, durch Verrechnen der in der Merkmalsextraktion gefundenen Formantenfrequenzen [11] zu normieren.

Die Einrechnung des so gewonnenen Verzerrungsfaktors wird durch Multiplikation der Filterbankmittelpunkte erreicht. Bei Faktoren > 1 können Frequenzen auftreten, die aus dem gewünschten Wertebereich austreten, in diesem Fall wird die Energie des höchsten noch darstellbaren Frequenzbandes aus den Nachbarn interpoliert.

Das Verfahren der VTLN wird in der Regel sowohl im Training als auch in der Erkennung eingesetzt, kann potentiell aber auch bei einem in Hinsicht auf die Vokaltraktlänge ausgewogenen Trainingskorpus nur in der Arbeitsphase des Systems verwendet werden.

3.1.2 Maximum likelihood linear regression

Die *Maximum likelihood linear regression* (MLLR) realisiert eine Codebuchadaption, die bei relativ geringem Rechenaufwand mit kleinen Trainingsstichproben

[1] Beispielsweise 12 äquidistante Werte aus dem Intervall $[0.8 \ldots 1.2]$

im laufenden System durchgeführt werden kann. Dabei werden die Mischverteilungen

$$b_j(\underline{x}) = \sum_{l=1}^{L_j} c_{jl} N(\underline{x}|\underline{\mu}_{jl}, \underline{K}_{jl}) \tag{3.1}$$

(s. Gleichung 2.17) der Emissionswahrscheinlichkeiten so optimiert, dass die Produktionswahrscheinlichkeit der Stichprobe erhöht wird. Dies entspricht exakt der Vorgehensweise des *Baum-Welch*-Algorithmus (s. Abschnitt 2.3.3). Im Gegensatz zu ihm wird die MLLR allerdings nicht mehrfach iteriert, sondern in nur einem Schritt durchgeführt [39].

Da sich bei unterschiedlichen Sprechern die Lage der Phoneme im Merkmalsraum, aber nicht deren Verteilung verändert [49], beschränkt sich das Verfahren auf die Manipulation der Mittelwertvektoren, die über eine lineare Transformation stattfindet: $\mu'_{jl} = A\mu_{jl} + b$. Daher müssen lediglich wenige Parameter neu geschätzt werden, wodurch die (kleine) Stichprobe besser ausgenutzt werden kann. Im Gegensatz zu anderen Verfahren bedient sich die MLLR der Zusammenfassung von Codebuchklassen in Regressionsklassen, die über ein Ähnlichkeitsmaß (etwa Abstand im Merkmalsraum) definiert werden. Dadurch können auch Klassen adaptiert werden, für die keine Beispiele im Trainingsmaterial enthalten sind. Alle Mitglieder derselben Regressionsklasse benutzen dieselbe Transformation.

3.1.3 Dauermodellierung

Ein interessanter Aspekt sowohl in der sprecherabhängigen als auch -unabhängigen Erkennung ist die Adaption an die Sprechgeschwindigkeit. Menschen sprechen nicht mit einer einheitlichen Geschwindigkeit. Selbst bei ein und derselben Person kann es zu Variationen der Geschwindigkeit, sogar innerhalb einer Äußerung, kommen, wenn der Benutzer beispielsweise bei der Wortfindung zögert oder Worte durch langsame Artikulation betonen möchte. Bei der Realisierung von Wortmodellen über HMMs wird die Mindestdauer eines Wortes über die minimale Zustandsfolge des Phonemkonkatenats und seinem Durchlaufen im Frametakt vorgegeben. Dagegen wird die Maximaldauer durch mehrfaches Selbstreferenzieren der einzelnen Zustände modelliert. Damit wird das Verweilen in einem Zustand mit der Zeit zwar unwahrscheinlicher, lässt sich aber nicht exakt angeben[2]. Da die Selbstübergangswahrscheinlichkeit fix ist, kann lediglich eine hyperbolisch mit der Zeit abfallende Verweildauer dargestellt werden, aber nicht Verweildauern innerhalb eines Intervalls (s. Abb. 3.1, oben). Dazu wären geringe Wahrscheinlichkeiten außerhalb der Intervallgrenzen und höhere innerhalb erforderlich (s. Abb. 3.1, unten).

[2] Außer in den Trivialfällen $a_{ii} = 1$ und $a_{ii} = 0$, wenn nur Selbstübergänge bzw. keine erlaubt sind.

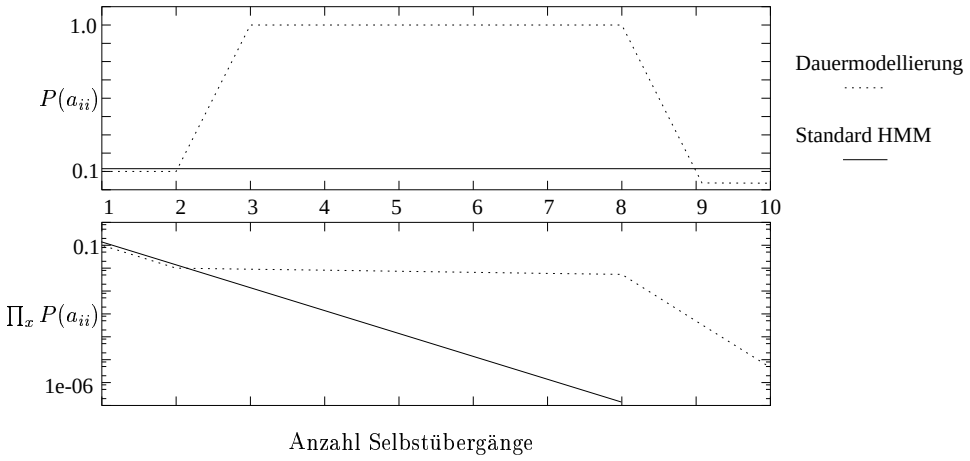


Abbildung 3.1: Wahrscheinlichkeit der Selbstübergänge in Abhängigkeit der bisherigen Selbstreferenzen eines HMM-Zustandes

Um diese Problematik zu umgehen, kann eine explizite Dauermodellierung eingesetzt werden. Die Übergangswahrscheinlichkeiten $a_{ij} = P(s_{t+1} = j | s_t = s_{t-1} = \ldots = s_{t-k+1} = i), \quad 1 \leq i, j \leq N$ aus Abschnitt 2.3 werden durch um den Zähler der Selbstübergänge k erweiterte Wahrscheinlichkeiten ersetzt:

$$A = [a_{ijk}] = P(s_{t+1} = j | s_t = i, i = j), \quad 1 \leq i, j \leq N, 1 \leq k \leq c(a_{ij}) \tag{3.2}$$

Dabei ist $c(a_{ij})$ die tatsächlich beobachtete maximale Sequenz der Übergänge i, j in Folge. Der Wert kann nur für Selbstübergange grösser als 1 werden. Die k können auf phonetisch annotiertem Material ausgezählt und die zugehörigen Wahrscheinlichkeiten entsprechend gesetzt werden.

3.2 Behandlung unbekannter Wörter

Einfache Spracherkennungssysteme benutzen im Allgemeinen ein festes Lexikon und haben daher einen beschränkten Wortschatz. Wörter, die nicht im Lexikon enthalten sind, werden als 'neue Wörter', unbekannte Wörter oder auch kurz OOV (engl. **O***ut* **O***f* **V***ocabulary*) bezeichnet. Wird ein Erkenner mit unbekannten Wörtern konfrontiert, werden sie im schlimmsten Fall auf Wörter mit ähnlichen akustischen Merkmalen (und/oder einer hohen Bewertung des Sprachmodells) abgebildet und produzieren Erkennungsfehler. Da dies nicht notwendigerweise durch jeweils eine einzelne Substitution geschieht, sondern auch von Einfügungen begleitet werden kann, erzeugt jedes unbekannte Wort im Durchschnitt mehr als

einen Fehler. Für den *Verbmobil* [3]-Korpus hat Fetter die Zahl auf 1.2 - 1.6 Fehler pro unbekanntem Wort ermittelt [14] und drei Jahre später noch nach oben auf 2 korrigiert [13]. Hetherington berichtet für den englischen *ATIS* -Korpus[4] mit 1.5 Fehlern eine ähnliche Rate [24].

Wird keinerlei Vermeidung dieser Fehler durch den Erkenner durchgeführt, ergibt sich ein ernsthaftes Problem bei der Benutzerinteraktion. Der Benutzer kennt nicht notwendigerweise den Gesamtwortschatz eines Erkenners, kann also, vor allem bei einem ungeführten Dialog und Spontansprache, OOVs äußern. Die dann auftretenden Fehler können von ihm nicht von *normalen*, z.B. akustischen, Erkennungsfehlern unterschieden werden. Also wird ein kooperativer Benutzer versuchen, seine Aussage zu wiederholen, evtl. mit anderer Betonung/erhöhter Lautstärke etc. Das unbekannte Wort kann aber nicht richtig erkannt werden, egal wie man es ausspricht. Somit steigt die Benutzerfrustration und sinkt die Benutzerakzeptanz.

Unbekannte Wörter zeichnen sich durch folgende Eigenschaften aus: Wenn der Erkenner gängigerweise mit Wortuntereinheiten arbeitet und ein ausreichend großes Vokabular besitzt, werden hinreichend viele verschiedene Wortuntereinheiten erkannt. Damit bestehen OOVs größtenteils aus bekannten Komponenten, haben also auf Wortuntereinheitenebene hohe Ähnlichkeit mit bekannten Wörtern. Desweiteren kommen OOVs potentiell überall in einer Äußerung vor und haben, da kleinere Funktionswörter in der Regel bereits im Korpus abgedeckt sind, eine etwas größere Durchschnittslänge.

Unbekannte Wörter können in verschiedene Klassen eingeordnet werden:

Flexionen Durch Deklination von Nomen und Konjugation von Verben ist die deutsche Sprache reich an Flexionen. Die Flexionen bieten von phonotaktischem Standpunkt aus betrachtet häufig nur subtile Unterschiede zwischen einzelnen Wörtern, etwa *brauch-e* und *brauch-en*

Derivate Anhängen bestimmter Prä- und Suffixe an freie Morpheme ('sauber','Sauberkeit'), wobei häufig ein Übergang von einer Wortart in eine andere stattfindet, bezeichnet man als Derivate, wobei nach [8] in neueren Untersuchungen die explizite Unterscheidung zwischen Derivaten und Flexionen als überflüssig angesehen wird.

Komposita Im Deutschen können nahezu beliebig lange Komposita zusammengesetzt werden (Beispiel *Verbmobil*: *Diaabendweintrinkrevisionstreffen*), deren einzelne Komponenten durchaus im Korpus existieren können.

[3] *Verbmobil* ist ein deutsches Großprojekt das sich mit automatischen, sprachgesteuerten Übersetzungssystemen aus der Domäne 'Terminabsprache' beschäftigt. Weitere Details in Abschnitt 6.2

[4] *ATIS* - Air Travel Information Service, ein Flugplanauskunftskorpus

Wortabbrüche Noch während ihrer Aussprache werden unpassende oder falsch formulierte Wörter abgebrochen und eventuell noch von Korrektursilben verbessert, was zu in der Sprache nicht existierenden Wörtern führt.

Eigennamen Über die Benutzung von Adresslisten und Telefonbüchern können beliebige Namen in das Erkennungslexikon kommen. Die Anzahl ist praktisch unbeschränkt. Dadurch, dass Namen auch aus anderen Sprachen mit einem vom Deutschen unterschiedlichen Phoneminventar übernommen werden, ist diese Klasse schwer zu handhaben.

Hesitationen Fülllaute wie *'öhm', 'äh', 'uhm'* aber auch komplette Wörter und Floskeln *'irgendwie', 'im Prinzip', 'gell'*

neue Wörter Wörter, die nicht im Lexikon enthalten sind.

Die Häufigkeitsverteilung dieser verschiedenen Kategorien unbekannter Wörter ist nach Untersuchungen auf dem englischen *WSJ*[5]-Korpus mit einem 45%-Anteil dominiert von den Flexionen, gefolgt von 27% für die Eigennamen, 6% für die Komposita. 22% subsummieren schließlich die verbliebenen Klassen [64]. Da die englische Sprache im Gegensatz zur deutschen kaum Komposita besitzt, ist der Anteil der Komposita an den unbekannten Wörtern im Deutschen höher. Bei Untersuchungen des *Verbmobil* -Korpus wurde der Bestandteil der Komposita im Lexikon auf 40% und im Korpus (in dem die Wörter des Lexikons unterschiedlich häufig vorkommen können) auf 11% geschätzt[13].

In Abhängigkeit der gewählten Anwendung können verschiedene OOV-Klassen ausgeschlossen werden. Beispielsweise werden in der Ansteuerung eines Navigationssystems mit gezieltem Dialog (Benutzer:*'Navigationssystem'*. System: `'In welche Stadt möchten Sie fahren?'` B:*'Hamburg'* ...) keinerlei Flexionen und Derivate vorkommen. Können aufgrund weniger eingeschränkter Eingaben Flexionen vorkommen, die aber innerhalb des Systems nicht aufgelöst werden, besteht die Möglichkeit, das Erkennungssystem auf Stammformen basieren zu lassen. Nach Geutner können die verhältnismäßig kurzen Wortenden ignoriert werden [19]. Des Weiteren sind Hesitationen und Wortabbrüche spontansprachliche Phänomene, die nur in entsprechenden Korpora vorkommen, dort aber bei kleinem Grundvokabular fast ausschließlich die OOVs bilden. Schließlich können Komposita bedingt verarbeitet werden, wenn ihre Komponenten bereits Bestandteil des Erkennungslexikons sind. Die OOVs werden dann dekompositioniert und ihre Einzelteile erkannt, was durch eine nachgeschaltete morphologische Analyse vorgenommen und an eine Verstehenseinheit gemeldet werden kann (s. [4] für eine detailierte Beschreibung).

[5] *WSJ - Wall Street Journal*, Korpus aus gelesenen Börsennachrichten

Es gibt beim Umgang mit unbekannten Wörtern verschiedene Aufgaben mit steigendem Schwierigkeitsgrad, die in den nachfolgenden Abschnitten behandelt werden:

Vermeidung von OOVs: Werden beim Design eines Systems ausreichende Restriktionen an die zu erkennenden Äußerungen gestellt, kann das Auftreten von OOVs verhindert werden.

OOVs finden und überspringen: Kann ein Bereich im Sprachsignal als unbekanntes Wort klassifiziert werden, lässt sich das Problem durch Ignorierung des entsprechenden Abschnitts lösen.

OOVs finden und wiederfinden: Wenn zwei unabhängig voneinander detektierte unbekannte Wörter als identisch identifiziert werden können, bietet sich die Möglichkeit, das Erkennungslexikon dynamisch zu erweitern.

OOVs transkribieren: Das Wiedererkennen kann auch über die Erweiterung des Erkennungsmodelle durch Übernahme der Transkription des OOV-Wortes geschehen.

OOVs interpretieren: Eine sinnvolle Verarbeitung von OOVs setzt voraus, dass das System sie in geeigneter Weise verarbeitet. Dazu muss ein OOV 'verstanden' werden.

3.2.1 Vermeiden von unbekannten Wörtern

Die einfachste Problemlösung ist das Vermeiden von unbekannten Wörtern: Treten sie nicht auf, brauchen keine rechenzeitintensiven Verarbeitungsstrategien eingesetzt werden. Das Aufgabengebiet des Erkenners wird so exakt wie möglich definiert und sein Vokabular daraufhin so stark erweitert, dass alle Wörter bekannt sind. Dies setzt voraus, dass der Benutzer eines solchen Systems das Vokabular genau kennt und diszipliniert genug ist, nur genau dieses zu äußern, bzw. dass in der Entwicklung des Systems eine exakte Vorhersage von Benutzeräußerungen gemacht werden kann. Nach Auswertung der ersten Testläufe des telefonischen Zugauskunftsystems von Philips fanden die Techniker in den Mitschnitten eine Reihe von fingierten Pizzabestellungen. Daraufhin wurde das Erkennerlexikon mit den entsprechenden Wörtern erweitert und die Dialogkomponente modifiziert, so dass sie die Anrufer an eine Pizzeria verweisen konnte.

Die Lexikonerweiterung ist dabei nicht auf tatsächliche Wörter beschränkt, sondern kann durch Hinzufügen von gängigen Hesitationen in Form der 'Unwörter' *äh*, *hm*, *uhm* auch gegen diese spontansprachlichen Phänomene bedingt robust gemacht werden.

Vom Benutzer kann und soll aber nicht verlangt werden, die komplette Liste des Vokabulars zu kennen und nur sie anzuwenden. Die Vorgehensweise der Ausschließung unbekannter Wörter ist utopisch außer für sehr kleine Korpora, etwa die Zahlen von *null* bis *neun*. Dennoch kann man die Zahl der zu erwartetenden OOVs minimieren und damit das Problem wenigstens einschränken.

3.2.2 Detektion unbekannter Wörter

Die Detektion von unbekannten Wörtern im Sprachsignal ist die Voraussetzung aller folgender Verfahren im Umgang mit den OOVs. Es gibt verschiedene Ansätze zur Detektion, die hier unter folgenden Oberbegriffen zusammengefasst werden:

- Lexikonerweiterung
- Garbagemodelle
- Schwellwertverfahren
- Konkurrierende Erkenner

Allerdings ist bei den meisten Verfahren eine scharf abgegrenzte Einteilung nicht möglich. Garbagemodelle können beispielsweise so komplex sein, dass sie der Leistung (und dem Rechenaufwand) eines eigenständigen Erkenners entsprechen. Genauso können bei der Verrechnung von Modellen der Lexikonerweiterungen zu Modellen des Grundlexikons auch Schwellwertverfahren eingesetzt werden.

Lexikonerweiterung

Die Erweiterung des Lexikons durch eine Anzahl unterschiedlicher Wörter ist eine ungerichtete Variante der unter Abschnitt 3.2.1 *Vermeidung unbekannter Wörter* beschriebenen Vorgehensweise. Anstatt wie dort das Lexikon um zu erkennende, sinnvolle Wörter zu vergrößern, wird es einfach durch eine bestimmte Anzahl von Wörtern erweitert, die in ihren Längen und ihrer phonetischen Zusammensetzung ausgewogen sind. Diese Wörter sind Bestandteil des Trainingskorpus und können daher wie der Rest des Lexikons trainiert werden. In der Erkennungsphase werden sie allerdings alle mit demselben OOV-Symbol identifiziert. Dieser Ansatz geht davon aus, dass die erweiterten Modelle bei Vorkommen von unbekannten Wörtern besser passen als die Modelle für die bekannten. Fetter berichtet allerdings, dass Experimente einer bis zu 75-prozentigen Lexikonerweiterung auf dem Verbmobilkorpus keine wesentliche Verbesserung der OOV-Detektion mit sich brachten [14].

Garbagemodelle

Die im vorigen Abschnitt erwähnten *natürlichen* Modelle für unbekannte Wörter können auch *künstlich* konstruiert werden. Die ersten expliziten Modelle wurden 1990 von Asadi vorgestellt [5]. Es waren einfache lineare Phonemkonkatenate (die

sich als unzureichend erwiesen), Cluster mit beliebigen Phonemsequenzen einer bestimmten Länge (wobei sich die Mindestlänge von 3 Phonemen als am stabilsten erwies), sowie ein komplexes Diphonmodell (das Wörter so gut modellierte, dass es auch bei bekannten Wörtern anschlug).

Eine Weiterentwicklung wurde 1995 von Sakamoto untersucht. Die besten Resultate wurden dabei von einem Konsonanten-Phonemcluster erreicht, das in etwa der Silbenstruktur des verwendeten Japanisch entsprach [56].

Unrestringierte Modelle neigen dazu, sehr rechenaufwändig zu sein, da viele Alternativen parallel betrachtet werden müssen. Jusek hat daher erfolgreich Experimente mit einer einschränkenden Phonotaktik auf einem Garbagemodell durchgeführt. Die Phonotaktik wurde mit Linguisten entwickelt und spiegelt den Aufbau deutscher Silben wider [31]. Das Verfahren wurde in Form von Übergangsnetzen auf phonetischer Ebene umgesetzt, die die komplexen Regeln für den Aufbau von allen Arten von Silben im Deutschen widerspiegeln. Obwohl dieses Verfahren für Wörter aus dem alltäglichen Sprachgebrauch gut funktioniert, ist es nicht restriktiv genug, um die Modelle für einen echtzeitfähigen Einsatz zu restringieren. Ein zusätzliches Problem ergibt sich beim Auftreten von fremdsprachlichen Lehnwörtern und Eigennamen, die durch die Phonotaktik nicht abgebildet werden.

Schwellwertverfahren

Schwellwertverfahren beruhen auf der Berechnung von Maßzahlen für Wörter und der schwellwertbasierten Einteilung in bekannte und unbekannte Wörter. Im Gegensatz zu den anderen Verfahren wird hier keine explizite, sondern eine implizierte Modellierung benutzt.

Eine einfache Möglichkeit, Maßzahlen zu erhalten, ist die Verwendung der Worthypothesenbewertungen. Während des Erkennungsprozesses in der Viterbisuche werden für alle Hypothesen Bewertungen in Form von Wahrscheinlichkeiten berechnet. Diese können lediglich akustisch basiert sein oder durch Verrechnung mit einem Sprachmodell oder einer Grammatik erweitert werden. Relativ höhere Bewertungen zeichnen wahrscheinlichere Wörter aus, aber in Abhängigkeit der Aufnahmebedingungen des Signals sowie durch implementierungsbedingte Vereinfachungen können die absoluten Werte in einem gewissen Maß variieren. Aufgrund dieser Varianz sind die absoluten *Scores* zur Bewertung der Erkennungsgüte unbrauchbar. Statt dessen haben sich relative Differenzen bewährt: Hat die wahrscheinlichste Hypothese eine große Differenz zur nächstwahrscheinlichen, liegt eine sichere Erkennung vor [32].

Eine andere Möglichkeit ist die Detektion über die Anzahl der aktiven Wörter. Wenn zu einem Zeitpunkt sehr viele Hypothesen aktiv sind, bedeutet dies ebenfalls, dass ihre Bewertungen sehr ähnlich sind, unabhängig von ihrem absoluten Wert. In diesem Fall scheint kein Modell richtig zu passen, und ein potentiell un-

bekanntes Wort liegt vor [35]. Somit wird hier, anders als bei expliziten Garbagemodellen oder Lexikonerweiterungen, jedes Wortmodell zur Detektion eingesetzt. Der Vorteil besteht darin, dass die Detektion nicht über Modelle geschieht, die sich von den Wortmodellen in ihrem Aufbau oder ihrem Training unterscheiden und daher nicht direkt vergleichbar sind.

Neben der akustischen Wahrscheinlichkeit und dem kombinierten *Score* kann auch analog die Bewertung des Sprachmodells allein herangezogen werden. Eine andere Art, das Sprachmodell zu berücksichtigen, ist die Beobachtung, wie häufig ein 'Backing Off'[6] auftrat [57].

Konkurrierende Erkenner

Als letztes Verfahren sei noch der parallele Einsatz unterschiedlicher Erkenner erwähnt. Werden mehrere Erkenner, beispielsweise ein Phonemerkenner und ein Standarderkenner mit Lexikon, eingesetzt, können deren Bewertungen verglichen werden. Übersteigt die Differenz einen zuvor festgelegten Wert, wird ein unbekanntes Wort hypothetisiert [23].

Aus Gründen des Rechenaufwands besteht die Möglichkeit, den durch eine Grammatik gesteuerten Phonemerkenner nur bei bestimmten Wortklassen zuzuschalten. Itou aktivierte den Phonemerkenner beispielsweise nur für Nomen, der Wortklasse, zu denen auch Eigennamen gezählt werden. Eine weitere Beschränkung des Erkenners kann durch Einsatz eines restringierenden Sprachmodells für Phonemfolgen erreicht werden [27].

Von allen Verfahren berichten die jeweiligen Autoren Erfolge, die sich allerdings aufgrund der uneinheitlichen zugrundeliegenden Korpora, Lexikongrößen, experimentellen Bedingungen, gewählten Gütemaße etc. nicht direkt miteinander vergleichen lassen. Erschwert wird dies zusätzlich durch einen wesentlichen Aspekt, der ebenfalls selten Berücksichtigigung findet: den Rechenaufwand. Es ist leicht nachzuvollziehen, dass ein parallel geschaltetes Modell, das eine beliebige Folge von Phonemen erkennen kann, einen großen Suchraum aufspannt und enorm viel Verarbeitungszeit benötigt. Andererseits ist beim Einsatz von wenigen zusätzlichen Modellen für Wörter aus einer Lexikonerweiterung nur ein geringer zusätzlicher Rechenaufwand nötig, vor allem, da nicht mehrere Erkenner parallel laufen müssen.

[6] Bei zu geringen Vorkommen bestimmter Wortkombinationen wird deren Verteilung durch Neuschätzung der Verteilungen aller beobachteter Wortkombinationenen simuliert. Das Verfahren dient zur Verhinderung von unmöglichen Wortpaaren, s. Abschnitt 5.3.2 für eine detailierte Beschreibung.

Bewertung der Detektionsleistung

Um die Leistung eines Erkenners im Umgang mit unbekannten Wörtern auszudrücken, werden Maßzahlen eingesetzt. Während einige Verfahren mit Zahlpaaren wie *False-Alarm*-Rate und *Detection- Correct*-Rate (s.u.) arbeiten, hat Jusek die Angabe auf eine prägnante Zahl reduziert, die *Detektionsakkuratheit*.

Es gibt bei der Detektion von unbekannten Wörtern zwei mögliche Fehler: Zum einen werden bekannte Wörter fälschlicherweise als unbekannte eingestuft, zum anderen werden unbekannte Wörter nicht gefunden.

Die Angabe, wie häufig fehlerhafte Detektionen von unbekannten Wörter stattfinden, die *False-Alarm-(FA)*-Rate, wird aus dem Quotienten der fälschlicherweise als unbekannt detektierten Wörter $\#U_{false}$ und der Gesamtzahl der bekannten Wörter $\#K_{tot}$ multipliziert mit 100 berechnet:

$$FA := 100 \cdot \frac{\#U_{false}}{\#K_{tot}} \tag{3.3}$$

Je höher der Wert ist, desto häufiger hat das System versagt. Die andere Fehlerquelle wird über das *Detection-Correct*-Maß bestimmt. Sie bezeichnet das Verhältnis der korrekt eingestuften unbekannten Wörter $\#U_{correct}$ zu der Gesamtzahl der unbekannten Wörter $\#U_{tot}$:

$$DC := 100 \cdot \frac{\#U_{correct}}{\#U_{tot}} \tag{3.4}$$

Höhere Werte zeichnen hier also bessere Systeme aus. Um die Leistung von Systemen vergleichen zu können, müssen jeweils beide Werte ermittelt und miteinander verrechnet werden. Eine elegante Möglichkeit wurde von Jusek in [30] eingeführt: die Detektionsakkuratheit. Sie ist in Analogie zur Wortakkuratheit (s. Formel 2.27), die ihrerseits zur Beurteilung der Worterkennungsleistung eines Systems dient, definiert:

$$DA := 100 \cdot \frac{\#U_{correct} - \#U_{false}}{\#U_{tot}} \tag{3.5}$$

Sie wird aus der Differenz von korrekt erkannten und falsch hypothetisierten OOV, gesetzt ins prozentuale Verhältnis zur Gesamtzahl der unbekannten Wörter in der Stichprobe, berechnet. Wie bei der Wortakkuratheit ist der höchste erreichbare Wert 100%, wohingegen der schlechteste Wert durch hohe fälschliche Detektion negativ werden kann.

3.2.3 Wiedererkennen

Die Detektion und anschließende Ignorierung von unbekannten Wörtern ist die einfachste sinnvolle Behandlung von OOVs. Die nächstschwierige Behandlung ist das Einbinden der detektierten Wörter in den Erkennungsprozess. Somit wird

dem Benutzer die Möglichkeit gegeben, selber im laufenden Betrieb ein Erkennungssystem zu erweitern, etwa durch Zuordnung von Namen, die dem System bis dahin unbekannten waren, zu Telefonnummern und anschließender Anrufmöglichkeit durch Nennung des Namens.

Eine simple Möglichkeit, die auch von diversen Telefonen und Handys unterstützt wird, entspricht der früher üblichen sprecherabhängigen Spracherkennung: Aus den Sprachsignalen eines mehrfach gesprochenen Wortes werden durchschnittliche Merkmalsmasken berechnet und abgespeichert. Die Erkennung eines Signals erfolgt dann, indem die gleiche Art Merkmale[7] berechnet werden und ein Mustervergleich mit allen abgespeicherten Masken durchgeführt wird. Um zeitliche Variationen zu kompensieren, wird der Vergleich über *dynamic time warping* (s. Abschnitt 5.3.2) durchgeführt. Der Vorteil dieses Verfahrens ist die einfache Realisierbarkeit. Im Gegensatz dazu steht allerdings der Speicheraufwand zum Ablegen aller Schablonen sowie die schlechte Einbindbarkeit in einen bestehenden Erkenner: Wie sollen diese völlig andersgearteten Bewertungen mit denen der Erkennermodelle verglichen werden? In speziellen Anwendungen kann zwar dafür gesorgt werden, dass unbekannte Wörter nur in definierten Kontexten vorkommen, etwa Floskeln wie '*Telefon* `(NAME)` *anrufen*', so dass dort auf die Schablonenerkennung umgeschaltet werden könnte. Praktischer wäre jedoch die Integration durch Erweiterung der Modelle des Erkenners. Damit wäre eine Umschaltung im Erkennungsprozeß nicht nötig und zusätzlich auch noch die sprecher*un*abhängige Erkennung des neuen Wortes möglich.

Für die Modellerweiterung ist es notwendig, eine auf denselben Einheiten wie die Standardmodelle basierende Transkription zu besitzen, in der Regel Phoneme. Die benutzerfreundlichste Gewinnung der Transkription ist der Einsatz sogenannter *phonetischer Schreibmaschinen*. Diese speziellen Erkenner berechnen auf Äußerungen Phonemsequenzen, die allerdings durch die verhältnismäßig kleinen Einheiten relativ ungenau sind. Gängige Systeme erreichen eine Phonemakkuratheit (Definition analog der Wortakkuratheit in Gleichung 2.27) von lediglich 51%, die auch nur durch Einsatz von Phontrigrammen erreicht werden [27].

Eine bessere Transkription kann erzielt werden, wenn der Benutzer das zu lernende Wort mehrfach äußert. Dies setzt allerdings voraus, dass das System schon erkannt hat, dass das Lexikon erweitert werden soll und daher mehrfach die Aufnahme vom Benutzer einfordert und bearbeitet. Dann können, analog dem obigen Verfahren, alle Versionen transkribiert und schließlich die zwischengespeicherten Äußerungen durch alle so gewonnenen Modelle berechnet werden. Dasjenige Modell, das die höchste Durchschnittswahrscheinlichkeit produziert, wird in das System integriert. Eine andere vorgeschlagene Methode ist die Bildung einer

[7] Diese Merkmale sind nicht notwendigerweise identisch mit denen für die sprecherunabhängige Erkennung, da sie nicht auf Eigenschaften zurückgreifen müssen, die sprecherübergreifend einheitlich sind.

Durchschnittsäußerung, die anschließend auf die existierenden Wortuntereinheiten abgebildet wird [21]. Im Rahmen dieser Arbeit wurden diverse Experimente durchgeführt, die auf diesen Methoden basieren und für einen isolierten Bereich des Sprachsignals eine automatische Transkription mit hoher Wiedererkennungswahrscheinlichkeit generieren konnten (s. Abschnitt 6.7).

Eine für den Benutzer umständlichere und für das System kompliziertere Vorgehensweise ist die textuelle Eingabe des zu erkennenden Wortes und anschließende automatische Umsetzung in eine Phonemsequenz. Die automatische Übersetzung der Orthographie entspricht diversen Zeichenersetzungen, Silbenzerlegungen, Generierung verschiedener Modellhypothesen auf diesen Silben und der Integration der Silbenkonkatenate in den Erkenner ([30], S. 91ff). Da die Korrespondenz zwischen gesprochenem und geschriebenem Deutsch recht hoch ist, wurden auf diesem Wege gute Ergebnisse erzielt. Je nach Anwendung können als unbekannte Wörter allerdings auch Eigennamen auftreten, die durchaus aus Sprachen kommen, bei denen diese Korrespondenz nicht mehr so hoch ist, etwa dem Französischen. In diesem Fall funktioniert das Verfahren nicht mehr. Die textuelle Eingabe geschah im Experiment über eine Tastatur, kann aber auch durch einen Buchstabiererkenner (s. Abschnitt 5.4) erfolgen.

3.2.4 Verstehen

Das *Verstehen* unbekannter Wörter ist die schwierigste Disziplin im Umgang mit ihnen. Wie kann das System mit einem detektierten, unbekannten Wort umgehen? Es kann zum Beispiel der Name zu einer Telefonnummer, die Bezeichnung eines Ortes, ein Versprecher oder auch ein Fluch sein (der ein Indiz für zuvor aufgetretene Erkennungsfehler sein kann).

Ein OOV soll nicht nur detektiert und eventuell im Erkenner integriert, sondern darüber hinaus sinnvoll verarbeitet werden können. Falls unbekannte Wörter nur in bestimmten Kontexten auftreten und etwa dialoggesteuert vom System erfragt werden können, ist in der Regel die Wortklasse wie beispielsweise *Eigenname* bei benutzerdefinierten Objektbezeichnungen vorher definiert. Somit kann das Wort auch in Sprachmodellen mit Kategorien (wie eben Eigenname) sowie Parsern berücksichtigt und auf Verstehensebene korrekt verarbeitet werden.

Dies liegt allerdings außer Reichweite dessen, was innerhalb dieser Arbeit behandelt werden soll. Weiterführende Literatur zum Thema bieten beispielsweise die Artikel von Roy, in denen über zusätzliche optische Eingabekanäle Wörtern aus der Spracherkennung eine Bedeutung in Form von Attributwerten ('rot', 'rund') zugewiesen wird [55] bzw. angeregt durch Untersuchungen vom Spracherwerb bei Kindern ähnliche Schemata auch in Maschinen umgesetzt werden [54].

3.3 Spezielle Probleme der Spracherkennung im Fahrzeug

Nachdem zuvor auf allgemeine Probleme in der Spracherkennung eingegangen wurde, sollen zwei wesentliche Probleme, die beim Einsatz eines Spracherkenners im Fahrzeug auftreten, betrachtet werden:

Die Geräuschumgebung

In der Labor- und Büroumgebung treten an sich kaum Hintergrundgeräusche auf. Diese können durch Nahbesprechungsmikrofone komplett kompensiert werden, so dass das Erkennungssystem ein reines Nutzsignal erhält. Im Gegensatz dazu treten in der Fahrzeugumgebung eine Vielzahl von Störgeräuschen auf: Motorenlärm bei unterschiedlichen Geschwindigkeiten und Drehzahlen, Windrauschen, Scheibenwischer, Blinker, Unterhaltungsgeräte etc. Dazu kommen neben dem Lärm vom umgebenenden Verkehr und Personen im Fahrzeug auch noch das hardwarebedingte Rauschen durch die Aufnahme und den Transport des Signals sowie dessen Analog-/Digital-Wandlung durch die Soundkarte. Damit die Erkennung möglichst gut arbeitet, müssen verschiedene Strategien zur *Signalverbesserung* benutzt werden. Beim Einsatz im Fahrzeug wird ein Großteil dieser Arbeit von speziellen Mikrofonen bzw. Mikrofonarrays, also zwei oder mehr miteinander verschalteten Mikrofonen, übernommen.

Die Hardwarerestriktionen

Die im Fahrzeug einzusetzende Hardware muss möglichst robust sein, sie muss Temperaturschwankungen von $-20°$ bis $+60°$ Celsius, hohe Luftfeuchtigkeit und Erschütterungen verkraften können. Um schnell verfügbar zu sein, darf das System nicht erst langwierig initialisiert werden, sondern sollte mit Umdrehung des Zündschlüssels verfügbar sein. Analoges gilt für die Deaktivierung des Systems beim Ausschalten des Fahrzeuges. Da gängige Computer diesen Anforderungen nicht genügen, muss auf kleinere, langsamere, aber robustere Rechner zurückgegriffen werden. Damit auch auf diesen Plattformen die Spracherkennung zufriedenstellend läuft, müssen zeit- und speicheraufwändige Rechenoperationen vermieden oder optimiert werden und so eine *Kompensation der Hardwarerestriktionen* durchgeführt werden.

Im weiteren werden Lösungsstrategien für die beiden Probleme vorgestellt.

3.3.1 Signalverbesserung

Im Allgemeinen wird versucht, ein Mustererkennungssystem möglichst robust zu realisieren und vor allem auch für ein gegebenes Gütekriterium zu optimieren. Um

dies nicht durch verrauschte Daten unnötig zu erschweren, ist es wünschenswert, dass dem Erkenner, sei er zum Verarbeiten von Bildern, Texten oder natürlich auch Sprache, einheitliche Daten präsentiert werden.

Die Datenwerte sollten innerhalb eines festgelegten Intervalls bzw. bei Wertevektoren in einem definierten Ausschnitt des Merkmalsraumes liegen und eine gewisse Qualität aufweisen. Die Definition der Qualität sollte über die Evaluation eines kompletten Mustererkennersystems (s. [47], S. 20 ff) geschehen, aus Gründen des Aufwands wird sie in der Regel aber lediglich subjektiv bewertet oder über simple Gütemaße wie beispielsweise dem Signal-/Rauschabstand (s.u.) angenähert. Obwohl die Optimierung des Erkenners nicht notwendigerweise von einer subjektiven Verbesserung der Eingabesignale abhängt, wird sie dennoch zur Bewertung herangezogen.

Um uniforme Daten zu gewährleisten, werden in einem Vorverarbeitungsschritt direkt bei oder nach der Signalaufnahme über Filterungen und Transformationen potentielle Störungen minimiert oder, wenn möglich, komplett entfernt. Die Fahrzeugumgebung ist, akustisch gesehen, ein sehr komplexer Einsatzort für einen Spracherkenner. In ihr gibt es nicht nur einfache Störquellen wie die Unterhaltungsgeräte, die direkt abgegriffen, isoliert betrachtet und aus dem gemischt vorliegenden Gesamtsignal getilgt werden können, sondern auch unbekannte Störgeräusche. Diese liegen nur mit dem Sprachsignal vermischt vor, wie etwa die Fahrgeräusche bei unterschiedlichen Geschwindigkeiten, Äußerungen von Beifahrern, Außengeräusche etc. Bedingt können auch sie durch an entsprechenden Positionen im Fahrzeugraum angebrachten, zusätzlichen Mikrofonen abgegriffen und wie die bekannten Störgeräusche kompensiert werden. Alle bisher aufgeführten Arten von Lärm stören das Sprachsignal additiv und können daher durch die unten aufgeführten Verarbeitungsschritte elimiert werden. Zusätzlich dazu gibt es noch multiplikative Störungen in Form des Halls in der geschlossenen Fahrzeugkabine. Der Hall verstärkt diverse Frequenzen, während andere gedämpft werden, und verändert sich mit dem Volumen und dem Inhalt der Fahrzeugkabine, beispielsweise durch Passagiere oder geöffnete Fenster.

Signal-Rauschabstand

Um die Aufnahmequalität eines Signals in Werten ausdrücken zu können, bedient man sich des sogenannten Signal-Rauschabstands (*SNR, engl. Signal Noise Ratio*). Er gibt das Energieverhältnis von Nutz- und Störsignal in Dezibel (dB) an. Die Einheit Dezibel drückt den Schalldruckpegel aus und berechnet sich aus dem zwanzigfachen dekadischen Logarithmus des Quotienten aus einem (in Pascal) gemessenen Schalldrucks P_m und einem Referenzschalldruck P_r, also *Schalldruckpegel* [dB]$= 20 \log P_m/P_r$. Nimmt man die Hörschwelle als Referenzschalldruck, ergeben sich als typische Werte für verschiedene Signale (nach [9], S.208 ff):

130 dB	Heavy Metal Konzert
100 dB	Blaskapelle oder Sirenen
80 dB	Turbinengeräusche in einer Flugzeugkabine
70 dB	normale Sprache
60 dB	Geräuschpegel in einer Büroumgebung
40 dB	leise Sprache
20 dB	Restlärm in einem schallarmen Labor
0 dB	Hörschwelle

Der Signalrauschabstand ergibt sich aus der Differenz der dB-Werte von zwei Signalen, die in Bezug zum gleichen Referenzwert gesetzt wurden oder (wie sich leicht zeigen lässt) indem man in obige Gleichung den Schalldruck des Störsignals als Referenzdruck P_r einsetzt und den des Nutzsignal als gemessenen Wert P_m. Der SNR für Sprachaufnahmen in einem schallarmen Raum ist demnach 70 db - 20 dB = 50 dB. Je lauter das Hintergrundgeräusch wird, umso kleiner wird der SNR, bis im Extremfall das Nutzsignal komplett übertönt wird und der SNR negative Werte annimmt. So liegt der SNR für laute Umgebungen wie etwa Sprachaufnahmen in einem *Opel Corsa* bei 120km/h bei lediglich 3.4 dB und in einem Rennwagen bei lediglich 160km/h bei -4 dB. Um ein Sprachsignal zu verbessern, gilt es entweder den Geräuschanteil abzudämpfen oder den Anteil des Sprachsignals zu verstärken. Sprachsignale haben einen Frequenzbereich von etwa 200 bis 5000 Hz. Die Grundfrequenz von Männern liegt dabei zwischen 80-200 Hz, von Frauen zwischen 150-300 Hz und von Kindern zwischen 200-500 Hz [9]. Während bei Vokalen hauptsächlich niederfrequente Anteile auftreten, besitzen Konsonanten wie die Frikative `/s/` und `/f/` hochfrequente Anteile. Eine einfache Art der Geräuschreduktion lässt sich mit einem Bandpassfilter realisieren, der das Signal entsprechend obiger Restriktion beschneidet. Zusätzlich können über die sogenannte Präemphase niederfrequente Störungen, wie z.B. Motorenlärm, reduziert werden, indem eine gewichtete Differenz zwischen jedem Abtastwert und seinem direkten Vorgänger als neuer Abtastwert eingesetzt wird [51]. Da hochfrequente Anteile im Signal, zu denen auch die Sprache gehört, stärker oszillieren als niederfrequente, folgt daraus eine Anhebung des Signalspektrums im oberen Bereich bei gleichzeitiger Dämpfung im unteren (s. Anhang A.3). Kompliziertere Filterverfahren wurden in dieser Arbeit nicht eingesetzt, da jedes Experiment mit einem Filter ein komplettes Neutraining des Erkenners nach sich zieht. Um die zusätzlichen Freiheitsgrade in der Zahl der einzustellenden Parameter nicht zu vergrößern, wurde darauf verzichtet.

Sprachdetektion

Egal wie man versucht, die Hintergrundgeräusche zu behandeln, ein damit verbundenes Problem ist die Sprachdetektion (*VAD* , engl. **V**oice **A**ctivity **D**etection). Eine robuste *VAD* trennt Signalbereiche mit Sprache von solchen

ohne Sprache und sorgt dafür, dass der Erkenner nicht unnötig aktiv ist und somit weder Rechenzeit verbraucht noch ungewollt Geräte durch angebliche Steuerbefehle manipuliert. Praktisch alle momentan existierenden Spracherkenner im Fahrzeug benutzen keine automatische Sprachdetektion, sondern überlassen die Entscheidung dem Fahrer (s. a. Kapitel 4): Mittels eines Knopfes oder Hebels wird die Spracherkennung aktiviert. In dieser Arbeit wurde darauf verzichtet und eine zweistufige Absicherung gegen ungewollte Aktivierung der Erkennung getroffen. Nach einer *VAD* wird auf dem potentiellen Sprachsignal eine Detektion durchgeführt. Nur wenn ein langes und damit robustes Schlüsselwort gefunden wurde, werden Kommandosequenzen akzeptiert. Simple Verfahren zur *VAD* beruhen auf der Signalenergie. Wenn eine gewisse Energieschwelle, die durchaus dynamisch angepasst werden kann, überschritten wird, liegt Sprache vor. Weiterentwickelte Verfahren berücksichtigen bestimmte Spektralenergien, also nur die Frequenzanteile, die primär in Sprache auftreten [7]. In dieser Arbeit wurde ein datengetriebener Ansatz mit einem Polynomklassifikator eingesetzt. Als Merkmale dienen die Filterbankausgaben, also pro Frame 32 Merkmale. Der Klassifikator unterscheidet 2 Klassen, *keine Sprache* und *Sprache*. Das System wird mit einen gleitenden Mittelwert eingesetzt und kann somit kurze Fehlklassifikationen in einem längeren, einheitlich klassifizierten Abschnitt unterdrücken. Da die *Sprache*-Klasse möglichst robust erkannt werden soll, sind Fehlklassifikationen zu *keine Sprache* kritisch. Daher wird die *Sprache*-Klasse mit einem gewissen Nachlauf erkannt und somit der anderen Klasse in kritischen Bereichen des Signals vorgezogen. Die nachfolgende Schlüsselworterkennung kompensiert voreilig als Sprache detektierte Signalabschnitte.

3.3.2 Kompensation der Hardware-Restriktionen

Die Anforderungen an die Rechnerhardware im Auto erlauben nicht den Einsatz herkömmlicher portabler Computer, sondern benötigen robuste DSP-Boards, die allerdings von der Speichergröße und der Verarbeitungsgeschwindigkeit restringiert sind. Da das Erkennungssystem noch nicht auf diese Plattform portiert wurde, werden im folgenden einige theoretische Überlegungen und Verfahren vorgestellt, die die Dimensionen des Erkenner reduzieren können.

Verkleinerung der Suchräume

Bei der Pfadverfolgung (s. Abschnitt 2.3.7) in der HMM-Berechnung werden in der Regel nicht alle Alternativen betrachtet, sondern nur die, deren Bewertung innerhalb eines relativen Abstands zur momentan bestbewerteten stehen. Diese Vorgehensweise wird als *Beamsearch* (=Strahlsuche) bezeichnet. Je nachdem, wie schmal der Beam gewählt wird, überleben zu den entsprechenden Zeitpunkten weniger Alternativen, die damit auch keinen weiteren Arbeits- und Speicheraufwand

für das Restsignal benötigen. Zu kleine Beams beschneiden die Suche allerdings so stark, dass lediglich ein einziges Pfadstück dominiert und alle anderen gelöscht werden. Da der richtige, also der am besten bewertete, Pfad nicht notwendigerweise von Anfang an auch die höchste Bewertung erhält, sorgen zu kleine Beams für eine zwar schnelle Erkennung, aber suboptimale Ergebnisse [43].

Statt die Suchräume nachträglich zu beschneiden, kann stattdessen bzw. zusätzlich versucht werden, sie erst gar nicht groß werden zu lassen. Die Größe des Suchraumes ist unter anderem auch mit der Dauer einer Äußerung gekoppelt: je länger sie andauert, umso mehr mögliche Kombination von Phonemen und Phonemfolgen existieren und umso höher wird der Suchaufwand. Andererseits ist es aber auch klar, dass der Anfang einer langen Äußerung nicht notwendigerweise Einfluss (wenigstens nicht auf einer Ebene unterhalb der Semantik) auf deren Ende hat. Demnach bietet es sich an, die Verarbeitung von Sprachsignalen mit einem geringen Zeitversatz schritthaltend durchzuführen. In [16] wurde für das *Verbmobil* -Korpus und das *CityMobil* -Korpus ein Versatz von etwa 750 Millisekunden für optimal festgestellt. Kürzere Abstände ließen die Erkennungsergebnisse signifikant einbrechen (relative 200% für 250 msec), höhere brachten dagegen keinerlei Verbesserung. Dieses Verfahren eignet sich für den Einsatz im Spracherkennungssystem dieser Arbeit allerdings (noch) nicht: durch das Vokabular und die verarbeitbaren Kommandos sind die meisten Äußerungen nicht lang genug, als dass sich der Einsatz des Verfahrens bemerkbar machen würde. Jedoch wird er interessant werden, sobald die Funktionalität des Erkenners erweitert wird und längere Sätze, etwa für das Diktieren von E-Mails, auftreten können.

Reduktion der Parameter

Die Reduktion des Speicherbedarfs geht in dem Erkenner meistens auch mit einem Geschwindigkeitszuwachs einher. Beispielsweise können die internen Zahlrepräsentationen auf kleineren Einheiten mit geringerer, aber ausreichender Genauigkeit umgestellt werden, so dass neben der Minimierung der Speicherrepäsentationen des Codebuchs und der Zustände auch die benötigten Berechnungen schneller werden.

In eine ähnliche Kategorie fällt die Reduktion der Merkmale. Wenn weniger Merkmale gleich gute Ergebnisse liefern, werden weniger Dichten für ihre Darstellung benötigt und die Kovarianzmatrizen entsprechend verkleinert. Ein positiver Nebeneffekt davon ist, dass weniger Parameter beim Training des Erkenners auch weniger Daten benötigen, mit dem gleichen Material also besser trainiert bzw. mit weniger Trainingsdaten ein eben so gutes Ergebnis wie beim ursprünglichen, großen Trainingsset erreicht werden kann. In [47] (S. 136 ff) werden verschiedene Verfahren zur Merkmalsauswahl vorgestellt, eines ist stellvertretend evaluiert und in Abschnitt 6.6 beschrieben.

4 Spracherkennung im Fahrzeug

Seit den Anfängen des Automobils, als dessen Aufgabe der simple Transport von Mensch und Material war, sind die Fahrzeuge mit immer mehr zusätzlichen Geräten ausgestattet worden, um das Reisen einfacher und angenehmer zu gestalten. Mittlerweile sind Navigationssysteme auf dem Vormarsch, während Radios mit CD-Wechsler sowie Klimaanlagen bereits häufig zur Standardaustattung gehören. Der nächste Schritt wird die Verbindung des Fahrzeugs mit dem Internet sein, womit unzählige Informations- und Unterhaltungsdienste zur Verfügung gestellt werden. Diese Weiterentwicklung ist bereits in einigen wenigen Prototypen realisiert worden, beispielsweise dem 'Internet-on-Wheels' Konzeptfahrzeug von *DaimlerChrysler*.

Beim Design eines Spracherkenners für den Einsatz im Fahrzeug stellt sich neben dem Aspekt der umgebungsbedingten Einschränkungen grundsätzlich die Frage, welche Art von Anwendungen gesteuert werden soll, genauer: Welche speziellen Anforderungen muss die Sprache erfüllen können. In diesem Kapitel werden verschiedene Interaktionsmodalitäten diskutiert und bestehende Systeme vorgestellt.

4.1 Anzusteuernde Geräte

Zunächst soll betrachtet werden, welche potentiell per Sprache ansteuerbaren Geräte in einem Fahrzeug zu finden sind. Ihre bisherige Handhabung und die dabei auftretenden Schwierigkeiten dienen als Hintergrund für die Frage, wie zukünftig sprachliche Bedienung realisiert werden könnte.

Dabei ist es der wesentliche Aspekt der Spracherkennung, dem Fahrer die Primäraufgabe des Fahrens zu erleichtern und nicht durch unnötig hohe Anforderung an die Aufmerksamkeit abzulenken. Weiterhin dürfen keine sicherheitsrelevanten Funktionen verbal angesteuert werden, da niemals eine hundertprozentige Erkennung garantiert werden kann. Alle Fahrfunktionen wie Steuerung, Geschwindigkeitsregulierung und Lichtaktivierung sind von vornherein ausgeklammert. Ebenso sind verschiedene Funktionen nicht im Fahrbetrieb möglich, etwa die Manipulation der Sitze. Um zu verhindern, dass sich der Benutzer um fälschlicherweise angesteuerte Geräte kümmern muss, sollen die Geräte eindeutig identifiziert werden. Ein gangbares Verfahren ist, die Anweisungen durch möglichst sicher erkennbare Floskeln einzuleiten. Diese Systemansprachen sollten natürlich

nicht Bestandteil von allgemeinen Gesprächen sein und umso robuster arbeiten, je wichtiger die Funktion ist und je häufiger sie benutzt wird. Nachdem die Spracherkennung für ein Gerät aktiviert wurde, arbeitet sie jeweils nur so lange, bis eine Aufgabe, etwa das Einstellen eines neuen Senders im Radio, erfolgreich durchgeführt werden konnte. Anschließend deaktiviert sie sich wieder. So kann gewährleistet werden, dass die Gefahr einer unbeabsichtigten Manipulation von Geräten bei Unterhaltungen im Fahrzeugraum minimal bleibt. Eine häufig praktizierte Methode ist der Einsatz eines Aktivierungsknopfes am Lenkrad. Soll die Erkennung arbeiten, wird während einer kompletten Äußerung der Knopf gedrückt[1]. Da die Einführung der Spracherkennung im Fahrzeug eine Benutzerentlastung sein soll, wurde dieser Ansatz in der vorliegenden Arbeit nicht verfolgt. Stattdessen wird die Erkennung direkt durch Sprache[2] ausgelöst und ist anschließend sensibilisiert für das Auftreten spezieller Aktivierungswörter. Bleiben diese aus, verharrt das System, auch wenn potentiell interpretierbare Kommandosequenzen folgen.

Im Folgenden wird betrachtet, welche Geräte im Auto genutzt werden und wie sie bisher bedient wurden. Daran schließen sich die Fragen an, welche Anforderungen an einen Spracherkenner und dessen Vokabular gestellt werden und wie die Interaktion mit einem Sprachinterface aussieht.

4.1.1 Navigationssysteme

Die Aufgabe von Navigationssystemen ist das Ermitteln einer geeigneten Route von einem gegebenen Startpunkt zu einem gewählten Zielpunkt, die beide in der Form eines Ort-Straßenpaares gegeben werden, und die anschließende Führung des Fahrers auf der berechneten Strecke. Bessere Systeme sind dabei in der Lage, spezielle Benutzerwünsche, wie bestimmte Zwischenstationen oder Streckenarten (Landstraße/Autobahnen), zu berücksichtigen. Sie lassen sich im laufenden Betrieb an vom Fahrer irrtümlich gewählte Alternativwege anpassen und können die Route neu berechnen.

Gängige Navigationssysteme, wie beispielsweise die Systeme *TravelPilotDX-N* von *Blaupunkt/Bosch* oder *CarMultimediaSystem MS 5000* von *Carin* haben zur Darstellung von Fahranweisungen und Systeminformationen ein kleines Display (s. Abbildung 4.1) sowie häufig ein Sprachsynthesesystem oder eine Auswahl zuvor aufgenommener Textschablonen, um während der Fahrt Anweisungen wie `'In 200 Metern links abbiegen'` geben zu können. Desweiteren kann die Navigation neben einem GPS-Gerät[3] vor allem durch Berücksichtigung der seit dem Start

[1] Daher wird diese Form der Aktivierung auch *PTT - push to talk* (drücke zum Sprechen) genannt.

[2] *VAD* - voice activity detection (Detektion von Sprachaktivität) s. Abschnitt 3.3.1 für weitere Informationen

[3] *GPS* - Global Positioning System. Über Satelliten und Dreieckspeilungen kann die räumliche

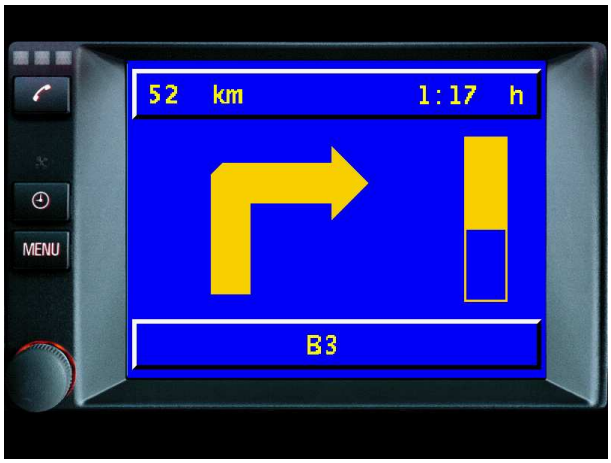


Abbildung 4.1: Display eines Navigationssystems, zu sehen sind textuell die gefahrene Strecke und Zeit, die symbolische Anweisung nach rechts abzubiegen sowie die relative Distanz bis zu diesen Ereignis.

der Fahrt erfolgten einzelnen Radumdrehungen einen Abgleich der tatsächlich gefahrenen Strecke mit der berechneten durchführen.

Die direkte Informationseingabe des Fahrers erfolgt im allgemeinen über eine einfache Tastatur mit wenigen Knöpfen zur buchstabenweisen Eingabe der Start- und Zieldaten (wobei der Startpunkt in der Regel noch bekannt ist und meistens vorgeschlagen wird). Über wenige Tasten lässt sich ein Alphabet durchblättern und das aktuelle Zeichen auswählen. Da *a priori* alle Orts- und Straßennamen bekannt sind, kann nach Angabe einer ausreichenden Zahl von Buchstaben die Anzahl der möglichen Orte so stark eingeschränkt werden, dass die noch verfügbaren Alternativen zur direkten Auswahl im Display präsentiert werden können.

Soll ein Navigationssystem mit der oben aufgeführten Funktionalität über einen Spracherkenner angesteuert werden, gehört zum Erkennerlexikon neben einigen wenigen Schlüsselwörtern wie *Abbrechen*, *Ja*, *Nächster* (s. Anhang A.2 für eine vollständige Liste) auch das komplette Namensverzeichnis der Städte und Straßen, zu denen Routen berechnet werden können. Von vorn herein sind zwar alle Namen bekannt, aber die extrem große Zahl der Städte[4] und der Straßennamen[5],

Position eines GPS-Empfängers auf wenige Zentimeter ermittelt werden, vorausgesetzt eine genaue Karte steht zur Verfügung.

[4]Nach dem Bundesamt für Statistik in Wiesbaden gibt es 12896 Städte und Gemeinden mit unterschiedlichen Namen in Deutschland, ein gängiges Naviagtionssystem berücksichtigt auch kleinere Orte und besitzt etwa 65.000 Einträge

[5]Laut Pressemitteilung der Deutschen Post AG vom 18.07.01 existieren in Deutschland 1.182.517 Straßen, die insgesamt 396.345 verschiedene Namen haben.

ihre hohe phonetische Ähnlichkeit sowie die häufig schwer zu bestimmende Aussprache machen die Erkennung zu einem äußerst komplexen Problem.

Verhältnismäßig simpel dagegen ist die Interaktion mit dem Benutzer: Nach sprachlicher Aktivierung der Navigationskomponente kann in einem Dialog der Startort vorgeschlagen und notfalls korrigiert sowie das Ziel erfragt werden. Der Erkenner liefert aufgrund der vokabularbedingten hohen Fehlerrate jeweils nicht nur das beste Resultat, sondern bietet eine Liste der besten Erkennungsergebnisse auf dem Display oder über die Synthese zur Auswahl. Sollte das richtige Resultat nicht angeboten werden, kann der Benutzer die Ganzworterkennung umgehen und stattdessen einen Buchstabiererkenner benutzen, der ähnlich der manuellen Eingabe bei hinreichender Einschränkung die verbliebenen Resultate anzeigt. Die Erkennung einzelner Buchstaben ist aufgrund der kurzen Einheiten und ihrer Ähnlichkeit (*b*, *d*, *e*, *g* ...) fehlerbehaftet. Um diesem Phänomen entgegenzuwirken, wird durch Berücksichtigung potentieller Verwechslungspartner und Abgleich mit dem Lexikon das Verfahren robuster gestaltet (s. Abschnitt 5.4).

4.1.2 Telefon

Die Funktionsfähigkeit eines modernen Handys übersteigt die eines herkömlichen standortgebundenen Telefons: neben der Hauptaufgabe der Gesprächsübermittlung besitzt es noch die Möglichkeit,

- SMS-Nachrichten[6] zu übermitteln,
- Telefonbücher von derzeit etwa um die 100 Einträgen zu verwalten,
- über Mailboxen einen Anrufbeantworterservice anzubieten und
- Internetdienste abzufragen (WAP Handys).

Des Weiteren bieten diverse Handys auch noch ein Arsenal an Spielen an sowie die Möglichkeit, komprimierte Musikdateien (MP3s) abzuspielen. Zusätzlich gibt es noch integrierte Systeme wie beispielweise den *Nokia Communicator*, die neben der Handyfunktionalität noch die eines PDA[7] beinhalten. Aufgrund der Komplexität dieser zusätzlichen Funktionen und ihrer geringen Relevanz für die angestrebte Domäne werden diese Erweiterungen hier nicht weiter betrachtet.

[6] *SMS - short messaging service*: Kurznachrichten für andere Handys, die mittlerweile bis zu 1000 Zeichen lang sind und in der Regel über die Tastatur des Handys eingegeben werden. Seit Anfang 2001 können bei einigen Anbietern SMS auch ins Festnetz geschickt werden, wo sie über eine Sprachsynthese vorgelesen werden.

[7] *PDA - personal digital assistant*, portable Minicomputer, die primär als elekotronisches Address- und Notizbuch benutzt werden, aber auch Applikationen aus der Büroumgebung anbieten.

Die primäre Eingabe bei herkömmlichen Handys geschieht über etwa 20 Tasten, die neben den zehn Ziffern noch diverse Funktionstasten zum Blättern in Dateien und Menus, zum Bestätigen oder Abbrechen von Eingaben und natürlich zum 'Abheben' und 'Auflegen' des Telefons besitzen. Die Zahlentasten werden menukontextabhängig auch zur Eingabe von Buchstaben benutzt, dabei sind die Tasten mehrfach belegt und müssen gegebenenfalls wiederholt betätig werden, um alle Zeichen auswählen zu können. Zum Wählen einer Nummer kann diese direkt eingegeben oder aus dem privaten Telefonbuch ausgewählt werden. Der Benutzer erhält Informationen über das Display, das in der Regel 4 Zeilen mit 10 Zeichen und zusätzlichen Statussymbolen über Batterieladezustand und Empfangsstärke besitzt. Weiterhin werden verschiedene Handyzustände per Pieper akustisch oder per Vibrationsalarm taktil angezeigt, etwa der Eingang einer Nachricht oder zu niedrige Batteriespannung.

Neben Display und Tastatur verfügen Handys noch zusätzlich über eine Schnittstelle (Seriell und/oder Infrarot) zum Anschluss an alphanumerische Tastaturen zur schnelleren Eingabe von Namen und Nachrichten, an Computer, um diese mit dem Internet zu verbinden, an andere Handys, um direkt Informationen austauschen zu können sowie an Freisprecheinrichtungen für freihändiges Telefonieren im Auto. Der Einsatz des Handys im Fahrzeug benötigt noch die Freisprecheinrichtung selbst, die aus einer Anschlusshalterung für das Telefon besteht und entweder einen Lautsprecher und ein Mikrofon besitzt oder mit den im Fahrzeug integrierten Komponenten verbunden wird. Neuerdings können Handys auch mit dem *Bluetooth*[8]- Verfahren anschlussfrei betrieben werden. Damit reicht die bloße Anwesenheit des Handys im Fahrzeug aus, sei es in der Jackentasche des Fahrers oder im Kofferraum[9].

Das Vokabular zur primären Interaktion mit dem Handy ist überschaubar. Neben den Zahlen müssen lediglich ein paar Kommandos wie *abnehmen*, *wählen* und *auflegen* erkannt werden. Schwieriger gestaltet sich die Ansteuerung des Telefonbuchs, da die Namen nicht *a priori* bekannt sind und vom Benutzer ergänzt und verändert werden können, was das Lexikon offen und unbeschränkt macht. Ein ähnliches Problem ist die Versendung von SMS-Nachrichten, die beliebige Wörter und diverse Symbole enthalten dürfen.

Auch die Benutzerinteraktion an sich ist komplizierter als bei den anderen Geräten. Da sowohl die Ansteuerung als auch die Benutzung des Handys über das Medium Sprache geschieht, entsteht das Problem, die Steuerbefehle vom Rest eines Gesprächs zu unterscheiden. Die Spracherkennung kann nicht einfach deaktiviert werden, sobald ein Gespräch zustande gekommen ist und aktiviert werden, wenn der Gesprächsteilnehmer aufhängt, da dies den Benutzer zwingt, bei Bedarf

[8] *Bluetooth* ist ein Standard, dessen Etablierung Ende 1998 von *Ericsson* initiiert wurde. Er soll die kabelfreie Verbindung verschiedenster Geräte über integrierte Funkchips ermöglichen.

[9] Laut Ankündigung der französischen Firma *Parrot* auf der CeBit 2001.

ein Telefonat manuell zu beenden. Daher ist es bei der Handyansteuerung essentiell, dass eine Einleitungsfloskel für die Erkennung sich möglichst grundlegend von anderen Gesprächsinhalten unterscheidet. Je länger und ungewöhnlicher diese Phrase gewählt wird, umso sicherer ist der Benutzer vor irrtümlicher Aktivierung anderer Geräte und vorzeitiger Beendigung eines Telefonats.

Die Erweiterung des privaten Telefonbuches erfordert eine Wiedererkennung bisher nicht bekannter Namen. Anders als die Städte- und Straßennamen im Navigationssystem sind die Eigennamen der Telefonpartner nur dann bekannt, wenn sie zuvor eingegeben wurden. Diese Eingabe kann über einen Buchstabenerkenner geschehen. Eleganter ist das Abspeichern einer Merkmalsschablone, die nach Mehrfachnennung des Namens berechnet werden kann. Obwohl dies einen zuätzlichen Aufwand zur Erweiterung des Spracherkenners darstellt, lohnt es sich, da die Anzahl der Einträge im Vergleich zu denen des Navigationssystems gering ist und die Namen häufiger aufgerufen werden. Wurde das Telefonbuch teilweise oder komplett manuell eingegeben, gibt es die Möglichkeit, das Erkennungssystem auf Basis der Orthographie zu erweitern (s. Abschnitt 5.3), allerdings mit der Einschränkung, dass der Benutzer mit Abkürzungen und Namenszusätzen die Einträge nicht unaussprechbar macht.

Das Vokabular für SMS-Nachrichten ist ebenfalls offen, aber bei weitem nicht so überschaubar wie das auf 100 Namen beschränkte Telefonbuch. Daneben werden aufgrund der Längenbeschränkung auf 160 Zeichen SMS-Meldungen häufig mit Abkürzungen und einzelnen Zeichen versehen, wie etwa die zwei Buchstaben *cu* (für engl. *see you*) als Abschiedsgruß oder sogenannte *Emoticons/Smileys*, die aus verschiedenen Zeichen gebildet werden, wie Semikolon/schließende Klammer `;)` um ein augenzwinkerndes Grinsen zu repräsentieren. Daher bietet es sich an, neben dem Erkenner für Steuerkommandos wie *SMS versenden* einfach einen um die Sonderzeichen erweiterten Buchstabenerkenner zu benutzen.

4.1.3 Radio/CD/MP3/Tape Player

Das Unterhaltungs- und Informationsmedium im heutigen Fahrzeug ist das Radio. Neben Musik und Reportagen überträgt es vor allem auch Verkehrshinweise. Die Sender des Radios werden entweder automatisch gesucht, aus vorgewählten Speicherplätzen gewählt oder manuell eingestellt. Dabei kann in der Regel durch Auswahl des *TA*[10]-Modus forciert werden, dass nur Sender mit Verkehrsmeldungen angewählt werden. Neben ihrem Programm senden die Stationen auch nicht hörbare Information[11] wie den Sendernamen, die Musikrichtung und das momentane Programm. Zusätzlich wird noch ein akustisches Aktivierungssignal für Verkehrsmeldungen übertragen, um stand-by geschaltete Radios zu aktivieren.

[10] *TA - Traffic Announcement*, Verkehrsdurchsagen.
[11] Dieser Service wird *RDS - Radio Data System* genannt

Neben dem Radio gibt es im Fahrzeug an Unterhaltungsmedien noch Cassettenabspielgeräte sowie CD-Wechsler und mittlerweile auch MP3-Player[12]. Diese Geräte spielen zuvor aufgenommene Signale ab, die auf Cassetten, CDs oder komprimiert auf diversen Speichermedien vorliegen. Cassetten werden in der Regel sequentiell abgespielt, außer Vor- und Zurückspulen kann die Seite der Cassette gewählt oder das Tape ausgeworfen werden. Moderne Geräte erlauben auch das automatische Spulen zum nächsten Stück, wobei Stille auf dem Band als Separator genutzt wird. CD-Wechsler haben Zugriff auf ca. 10 CDs, die in der Wechseleinheit im Kofferraum liegen. Neben direkter Anwahl der CDs und ihrer Stücke bieten die meisten Geräte auch eine zufällige Musikabfolge an. MP3-Player zeichnen sich dadurch aus, dass den Musikstücken noch kleine Informationseinheiten mit Titel und Interpreten zugeordnet sind, die während des Abspielens angezeigt werden können. Je nachdem, welches Speichermedium unterstützt wird, reicht die Anzahl der Musikstücke von 10 bei Speicherkarten über 100 bei Daten-CDs bis zu mehreren 1000 bei Festplatten. Die letzten beiden unterstützen eine Verzeichnishierarchie geringer Tiefe, um die Musikstücke zu Sinneinheiten zusammenzufassen und entsprechen in der Funktionalität dann dem CD-Wechsler.

Allen Geräten ist gemeinsam, dass sie über Lautsprecher im Fahrzeug Audiodaten abspielen und ihre Lautstärke per Tasten oder Potentiometer einstellbar ist. Sind die Geräte eine Einheit, teilen sie sich ein Display, in dem zumindest numerisch die gewählten Frequenzen bzw. CD-/Musiknummern angezeigt werden. Bei besseren Systemen wird alternativ die über RDS empfangene Senderkennung ausgegeben. Radios werden über Drehregler und diverse Tasten zum Eintragen und Abrufen von Stationen und Aktivierung spezieller Modi wie *stand-by*, automatische Senderverfolgung und Ignorierung von Nicht-Verkehrsfunksender angesteuert. Cassettenabspieler besitzen Tasten für die oben genannten Funkionen. CD-Wechsler und MP3-Player besitzen Tasten zur Auswahl der nächsten Scheibe bzw. des nächsten Musikstückes. Wegen der großen Anzahl der Stücke auf den Geräten wird meistens zusätzlich eine numerische Tastatur zur direkten Anwahl der CDs bzw. der Verzeichnisse unterstützt.

Die sprachliche Ansteuerung erfordert neben Floskeln zur Aktivierung der einzelnen Komponenten wie *Radio einschalten* auch die relative und absolute Regelung der Lautstärke (*Radio lauter*, *Radio halbe Lautstärke*). Die Frequenzen der Stationen und deren Namen müssen erkannt werden. Die UKW[13] Frequenzen in *MHz* reichen von *87* bis *108* und können eine Dezimalstelle besitzen. Während ältere Sender fast ausschließlich leicht handhabbare Buchstaben-

[12] *MP3* steht für *Moving Pictures Expert Group Ebene 3 Audiokomprimierung* und beschreibt ein Komprimierungsverfahren für Audiodaten, die bei subjektiv empfundener CD-Qualität nur etwa 1/10 Speicher benötigt. Eine Minute Musik benötigt nur etwa 1MB.

[13] Langwelle, Mittelwelle und Kurzwelle erfordern bis auf den Frequenzbereich prinzipiell dieselbe Ansteuerung wie UKW, sind im Fahrzeug aber aus technischen Gründen nicht so gut zu empfangen und daher hier nicht weiter aufgeführt.

/Zahlen-Kombinationen (*WDR 2*, *SWR 1*, ...) oder Länderkennung mit Nummern (*Bayern 3*, *Hessen 1*, ...) besitzen, ist mit Einführung der Lokalsender die Anzahl der Namen massiv gestiegen. Etabliert hat sich die Kombination aus *Radio* und Kreisname (*Radio Bielefeld*, *Radio Lippe*, ...). Für die einfache Ansteuerung der anderen Musikkomponenten reichen Kommandosequenzen wie *nächstes Stück*, *Cassette umdrehen* bzw. *nächste CD/nächstes Verzeichnis*, *zufällige Reihenfolge* aus. Für direkte Zugriffe werden zusätzlich noch Zahlen benötigt.

Obwohl bei den MP3-Playern Namen und Interpreten textuell vorliegen und somit ähnlich den Städtenamen des Navigationssystems durchaus für eine Erkennung herangezogen werden könnten, ist das Problem der fremdsprachlichen Namen störender als bei den im Verhältnis dazu harmlosen Städte- und Straßennamen. Als Beispiel für das Problem der Graphem-Phonem-Umsetzung seien hier nur die amerikanische Band *Huey Lewis and the News*, die Italienerin *Gianna Nannini* oder der Franzose *Gilbert Becaud* genannt.

4.1.4 Internet und Fernsehen

Bislang lediglich in einigen Prototypen realisiert, werden Fernsehen und Internet in Zukunft vermehrt in Fahrzeugen zu finden sein. Ein Indiz dafür sind die Allianzen, die zwischen Fahrzeugherstellern und Internetanbietern geschlossen wurden: *General Motors* mit *America OnLine*, *Ford* mit *Yahoo* und *DaimlerChrysler* mit *T-Online* [12]. Während das Fernsehen den Benutzer durch Präsentation von empfangenen Programmen oder abgespielten Aufnahmen in einer passiven Konsumentenrolle lässt, ist das Internet auf Interaktion ausgelegt. Es zeigt multimediale Inhalte in Form von Webseiten, erlaubt Korrespondenz über E-Mail und Austausch von Daten, im Fahrzeug speziell zur Routenplanung und zur Telediagnose.

Im Fahrzeug ist mindestens ein Bildschirm, im besten Fall für jeden Insassen ein eigenes Display eingebaut. Für das Fernsehen können relativ oder absolut verschiedene Programme gewählt und die Lautstärke reguliert werden. Steht ein DVD-Spieler[14] zur Verfügung, müssen noch zusätzlich Cursor-Tasten für die Film-Menus benutzt werden können. Die benutzergesteuerte Internetanbindung erfordert einen Mechanismus zur Verfolgung von Links auf Webseiten, entweder über Cursor und Entertaste oder über ein raumsparendes Mousedevice, d.h. einen *Trackball* oder ein *Touchpad*. Weiterhin müssen zur direkten Angabe von Webseiten, zur Eingabe in Masken und zur Erstellung von E-Mails komplette Tastaturen verfügbar sein.

Das Lexikon zur verbalen Ansteuerung besteht neben den bereits bei den anderen Geräten erwähnten Aktivierungswörtern beim Fernsehen oder bei der DVD

[14] *DVD - Digital Versatile Disk*, digitale, vielseitige Scheibe. Ein Speichermedium mit hoher Kapazität, auf dem komplette Filme abgelegt werden können.

aus Zahlen zur Sender- bzw. Kapitelwahl. Die komplette Unterstützung der verbalen Internetansteuerung erfordert nicht nur ein offenes Vokabular, sondern durch die Vielsprachigkeit des *Webs* eines für jede Sprache, daher sollte auf diese Art der Ansteuerung verzichtet werden. Handhabbarer wiederum ist das Navigieren eines Cursors, der über die Links und Eingabefelder verschoben wird. In diesem Fall genügen die Kommandos *rechts/links*, *hoch/runter*, *nächster/voriger* und *auswählen*. Für kleinere Eingaben kann wiederum der Buchstabenerkenner hinzugezogen werden.

Mit der Fernseh-/Internet-Anbindung des Fahrzeugs werden mehrere Probleme aufgeworfen: Die bildschirmbasierte Verarbeitung von Informationen während der Fahrt kann bei einfachen Symbolen wie den Pictogrammen eines Navigationssystems (s. Abb. 4.1) für den Fahrer noch realisierbar sein. Die Informationsflut von Webseiten und Filmen lenkt ihn dagegen zu stark ab. Sie sind daher den Mitfahrern vorbehalten. Die wiederum haben ihre Hände frei und können ihre Aufmerksamkeit ganz der Bedienung der Geräte widmen. Demnach ist die sprachliche Ansteuerung in diesem Fall nicht unbedingt notwendig. Lediglich die Realisierung der Interaktion mit den anderen Unterhaltungskomponenten und damit die Abdeckung des für die einfache Interaktion benötigten Vokabulars bietet an, die Sprache auch für Fernsehen und Internet zu nutzen. Wird hingegen wenigstens statt der Bildschirmausgabe eine Syntheseeinheit benutzt (was für Filme natürlich keinen Sinn ergibt), können E-Mails problemlos mitgeteilt werden. Webseiten hingegen sind als Hypertexte nicht sequentiell angelegt und müssten, wie etwa für WAP Handys,[15] aufbereitet sein, um vorgelesen zu werden.

Wird die Sprachsteuerung auch den Mitfahrern gestattet, müssen für sie zusätzliche Mikrofone bereitstehen. Neben der Problematik divergierender Anweisungen der Insassen ergibt sich erschwerend, dass Kinder als Sprecher auftreten können. Kinder haben sehr kurze Vokaltrakte und damit ein von den Trainingsdaten stark abweichendes Frequenzspektrum was zu einem massiven Einbruch in der Erkennungsleistung führt. Da der Fahrer des Fahrzeuges vom Gesetz her wenigstens 18 Jahre[16] sein muss, ist die Einschränkung des Trainingsmaterials auf Erwachsene legitim.

4.1.5 Andere Geräte

Neben den oben Genannten befinden sich im Fahrzeug noch diverse einfach anzusteuernde Geräte wie beispielsweise die Klimaanlage, Sitzeinstellung, Fensterheber oder die Außenspiegel. Die Bedienung dieser Geräte geschieht über einfache

[15] *WAP - Wireless Access Protocol.* Die damit ausgestatteten Handys können auf ihrem Minimaldisplay speziell zugeschnittene Internetseiten anzeigen.

[16] Europaweit ist 18 das Mindestalter (Ausnahme England, dort ab 17 Jahren), um Autos fahren zu dürfen. In Amerika kann zwar schon mit 16 der Führerschein gemacht werden, aber auch in diesem Alter ist der Vokaltrakt nahezu ausgewachsen.

Taster (Fensterheber), Drehregler (Klimaanlage) oder kleine Joysticks (Außenspiegel). Im Falle einer Sprachsteuerung ist das Vokabular beschränkt auf Anweisungen, das jeweilige Gerät zu aktivieren und anschließend einzustellen. Dazu dienen bei der Klimaanlage relative oder absolute Temperaturangaben *Klimaanlage wärmer* bzw. *Klimaanlage 22 Grad* und für die anderen Systeme Kommandos wie *vor/zurück*, *hoch/runter* und *kippen/aufrichten*. Im extremen Fall kann die nähere Spezifikation, welches Element eines Gerätes eingestellt werden soll, zu einer unhandlichen Bezeichnungsflut sorgen. Moderne Sitze können nicht nur geneigt und verschoben werden, sondern haben die Möglichkeit, bestimmte Bereiche der Lehne zu versteifen, die Sitzfläche in der Höhe zu variieren etc. Um diese verschiedenen Funktionen verbal anzusteuern, müssen einfache und eindeutige Bezeichnungen gewählt werden, um zu verhindern, dass dem Benutzer Äußerungen wie *'Sitz vertikale Lodorse vor'* aufgenötigt werden.

Ausserdem gibt es einfach einstellbare Geräte, die per Sprache nicht adäquat manipuliert werden können: Wer jemals in einem Auto ohne vom Fahrersitz aus verstellbaren rechten Spiegel den Beifahrer angeleitet hat, diesen einzustellen, wird nicht ohne ikonische Handgesten ausgekommen sein. Da multimodale Eingaben nicht Inhalt dieser Arbeit sind, werden diese Anwendungen dieser Art daher ausgeklammert.

Bei den Sitzeinstellungen ist zusätzlich zu beachten, dass sie aufgrund der Gefährdung der Primäraufgabe nicht während der Fahrt geschehen dürfen, die Dialogkomponente muss also Kontakt mit einem Sensor für Fahrzeugbewegungen haben.

4.1.6 Zusammenfassung

Zusammenfassend betrachtet können für die verbale Ansteuerung verschiedener Geräte im Fahrzeug folgende Vorüberlegungen getroffen werden (s. Tabelle 4.1):

Für die Ansteuerung des Navigationssystems wird ein immens großes Vokabular benötigt. Es enthält zwar keine unbekannten Wörter, dafür besteht die Möglichkeit, dass Benutzer ihnen unbekannte Namen nicht korrekt aussprechen.

Wenn lediglich die Grundfunktionalität des Handys genutzt wird, ist das Vokabular klein, die Erkennung wird aber dadurch erschwert, dass Äußerungen im Gespräch von Steuerkommandos unterschieden werden müssen. Soll das integrierte Telefonbuch per Namenswahl angesteuert werden, erweitert sich das Vokabular um etwa 100 Wörter, die allerdings vorab unbekannt sind. Weitere unbekannte Wörter werden durch das verbale Aufsetzen von SMS-Nachrichten in das System gebracht, die zusätzlich noch einen Vorrat Sonderzeichen beinhalten können.

Für die Ansteuerung des Radios gibt es neben Kommandos für die üblichen Funktionen noch die Namen der Radiosender. Bei den Komponenten, die auf

Anwendung	Vokabular-größe	Unbekannte Wörter	erweiterbar
Navigationssystem	500.000	-	-
Handy (Primärfunktion)	20	-	-
(Telefonbuch)	offen (100)	+	+
(SMS)	offen	+	-
Radio	1000	-	-
CD/MP3/Tape	je 20	-	-
mit Namenwahl	offen (10.000)	-	-
Internet (Navigation)	20	-	-
eMail (lesen)	20	-	-
eMail (schreiben)	offen	+	-
Klimaanlage etc.	je 20	-	-

Tabelle 4.1: Übersicht der Anforderungen an das Vokabular zur Sprachansteuerung für verschiedene Anwendungen. Die in Klammern gesetzten Zahlen entsprechen der Größenordnung an sinnvoll verarbeitbaren Wörtern. In der Spalte 'erweiterbar' ist markiert, ob ein Benutzer eigene Wörter integrieren kann.

Datenträger zugreifen, ist das Vokabular, ähnlich dem des Handys, verhältnismäßig klein für die Standardansteuerung, jedoch beliebig groß für die Namenswahl. Aber im Gegensatz zum kleinen Telefonbuch kann die Anzahl der Lieder/Interpreten/Alben für MP3 Spieler sehr groß werden.

Die weiteren Funktionen schließlich benötigen jeweils nicht mehr als etwa 20 Kommandos, können aber nicht intuitive und damit wenig benutzerfreundliche Wörter haben.

4.2 Hardwareansteuerung

Im Zusammenhang der Fragestellung dieser Arbeit soll eine Spracherkennung das Verarbeiten von geeigneten Sprachsignalen zu Anweisungssequenzen leisten, die von einem nachgeschalteten Parser analysiert und ausgeführt werden sollen. Im Folgenden soll ein für diese Arbeit zwar nebensächlicher, für die reale Umsetzung jedoch wesentlicher Aspekt betrachtet werden: Wie werden die vielen verschiedenen Geräte tatsächlich physikalisch angesteuert?

Für das Handy wurde bereits das *Bluetooth*-Konzept erwähnt, das Verbindungen kabelfrei über kleine Funkchips mit Reichweiten zwischen 10 und 100 Metern zur Verfügung stellt. Die Verbindung geschieht im weltweit nicht lizensierten, also überall zugänglichen, 2.4-Gigahertz-Band. Neben einer ähnlich den Netzwerkkar-

ten eindeutigen Identifizierungsnummer jedes dieser Chips wird die Sicherheit durch Verwendung eines Authentifizierungsschlüssels und einer Datenchiffrierung von bis zu 128 Bit ermöglicht. Bis zu 8 Geräte können in einem sogenannten *Pico-Netz* miteinander kommunizieren, wobei das erste Gerät im Netz als eine Art Server agiert, alle Folgenden als Clients. Da die Geräte nach jedem Sende-/Empfangsvorgang ihre Frequenz ändern, was bis zu 1600 Mal pro Sekunde geschehen kann, ist die Verbindung robust gegen Störungen von außen. Die Übertragungsgeschwindigkeit liegt bei synchroner Verbindung bei 64kBit/s (entspricht also ISDN-Qualität) und bei asynchroner Verbindung zwischen 57 und 720 kBit/s. Beides reicht aus, um Sprache in guter Qualität zu übertragen. Obwohl das Verfahren bisher erst in wenigen Geräten praktisch realisiert wurde, spricht die hohe Zahl an Mitgliedsfirmen[17] von Entwicklern und Vertreibern entsprechender Produkte in der *Bluetooth Special Interest Group* (kurz *SIG*) für eine Etablierung als zukünftiger Standard.

Bluetooth bietet sich als Kommunikationsmedium im Fahrzeug an: Die kabelfreie Lösung erlaubt ein einfaches Nachrüsten von Fahrzeugen und die Möglichkeit, Geräte wie Handys oder PDAs durch simple Anwesenheit im System einzubinden, was das Verfahren sehr benutzerfreundlich macht (vgl. [2]). Neben der kabellosen Verbindung gibt es Bestrebungen, schon beim Zusammenbau des Fahrzeugs Multimediabusse in physikalischer Form zu integrieren. Die anzusteuernden Geräte kommunizieren über Lichtleiterkabel, die, laut IEEE[18]-Standard 1394[19], mit bis zu 1.2 GigaBit/s eine wesentliche höhere Übertragungsrate als das oben erwähnte funkbasierte Verfahren haben. Somit können auch Musikdaten, die pro Kanal in CD-Qualität etwa 0.7 MBit/s benötigen, zwischen den einzelnen Instanzen der Bordstereoanlage ausgetauscht werden. Alle Geräte benötigen spezielle Adapter, um den Bus nutzen zu können. Diese können laut Hersteller (z.B. *Oasis SiliconSystems AG, Karlsruhe*) allerdings kostengünstig realisiert werden. Um die Anzahl zur Verwaltung benötigter Komponenten gering zu halten, wird das System in einer Ringtopologie realisiert. Dabei soll sowohl synchroner als auch asynchroner Datentransfer unterstützt werden. Die nachträgliche Aufrüstung zum Bussystem dürfte sehr kostenintensiv sein, der prophylaktische Einbau in der Fahrzeugherstellung dagegen günstig. Die Glasfaserlösung macht weiterhin die bisher in Fahrzeugen benutzten Kupferkabel überflüssig. Ihre Funktion, sei es der Transport von Steuerdaten für Bremsunterstützungssysteme oder Musikdaten für die Lautsprecher, wird vom neuen System übernommen. Zusätzlich ist das Sy-

[17] Anfang 2001 waren es 1400 Partner, darunter praktisch alle großen Firmen der Unterhaltungs- und Handyindustrie.

[18] *IEEE - Institute of Electrical and Electronics Engineers*, ist ein internationaler Dachverband von Wissenschaftlern aus technischen Bereichen, der unter anderem Standards definiert.

[19] *Standard 1394* ist auch unter dem von Apple geprägten Namen *FireWire* und von dem Sony etablierten Namen *i.Link* bekannt.

stem wegen der lichtleiterbedingten galvanischen Trennung durch elektrische und magnetische Strahlung sowohl weniger störend als auch weniger störbar.

Da bisher *Bluetooth* noch nicht sehr verbreitet ist, wird momentan (Stand Mitte 2002) der zuletzt beschriebene Standard von der *AMI-C*[20] prferiert. Was sich zukünftig durchsetzen wird, bleibt abzuwarten.

Die Ansteuerung der Geräte mit aktueller Technologie ist möglich und stellt keine besonderen Anforderungen an die Spracherkennung oder die anzusteuernden Geräte.

4.3 Bestehende Systeme

Häufig wird mit der Spracherkennung im Fahrzeug der Begriff *Telematics* in einem Atemzug genannt. *Telematics* ist ein englisch adaptiertes, ursprünglich französisches Kunstwort aus *Télécommunication* und *Informatique* (Telekommunikation und Informatik) und beschrieb ursprünglich lediglich die Verschmelzung der beiden Disziplinen. Der Begriff wird heute fast ausschließlich im Bereich der drahtlosen Verbindung von Computern im Fahrzeug mit einer menschlichen oder maschinellen Gegenstelle zum beidseitigen Datenaustausch benutzt. Zu den Aufgaben werden beispielsweise die Überwachung der Fahrzeugfunktionen mit automatischer Benachrichtigung von Pannen- und Rettungsdiensten, das Anbieten von Navigationshilfen und Verkehrsmeldungen sowie das Bereitstellen von Informations- und Unterhaltungsprogrammen gezählt. Seit 1995 existieren verschiedene *Telematics*-Anbieter auf dem Markt, die anfangs die Kunden durch menschliche Dienstleister in Callcentern betreuen ließen, später aber für einige automatisierbare Anfragen zusätzlich auf Spracherkennungssoftware setzten.

Da die Spracherkennung im Fahrzeug nicht nur ein aktueller Forschungsgegenstand ist, sondern vor allem als Interfacetechnik für die Ansteuerung diverser Geräte im Auto einen neuen, großen Markt erschließt, halten sich die Entwicklungsabteilungen der entsprechenden Firmen mit Veröffentlichungen sehr zurück. Der Großteil der Publikationen wird von universitären Forschungseinrichtungen herausgegeben und beschränkt sich dabei auf einzelne Komponenten sowie Korpora. Die Industrie hingegen tritt erst an die Öffentlichkeit, wenn ein Prototyp verfügbar ist, um damit für ihre technische Überlegenheit zu werben. Dabei werden interne Informationen nie bekannt gegeben, um einen Wissensvorsprung vor der Konkurrenz zu halten. Daher basiert die folgende Aufstellung von bestehenden Systemen primär auf Eigenwerbungen der entsprechenden Hersteller sowie einigen Testberichten.

[20] *AMI-C - Automotive Multimedia Interface Collaboration.* Zusammenschluss diverser Fahrzeughersteller zur Definition und Etablierung von Standards für die multimediale Verbindung von Komponenten im Fahrzeug.

Exklusiv für den *Jaguar S-Type* wurde Ende 1999 von der *Visteon Corporation*, Michigan, ein Spracherkennungssystem entwickelt. Das System hat in seinem 80 Wörter großen Vokabular Kommandos zur Ansteuerung eines Audiosystems, der Klimaanlage, eines Navigationssystems (bei der gegebenen Lexikongröße ohne verbale Adresseneingabe), eines Handys und der Türverriegelung. Für die Aktivierung des Erkenners muss ein Knopf gedrückt werden (PTT). Das gleiche System soll Anfang 2002 in ein weiteres Fahrzeug der Luxusklasse integriert werden, dem *Infiniti Q45*.

Mitte 2001 hat *Visteon* bekanntgegeben, dass der neue *Jaguar X-Type* mit der weiterentwickelten, 140 Wörter großen Variante des hauseigenen Spracherkenners versehen wird. Das System arbeitet ohne PTT-Aktivierung und kann zusätzlich zu den oben genannten Geräten auch einen Fernseher(!) ansteuern. Auf Wunsch gibt es das Gerät in britischem und amerikanischem Englisch, Deutsch, Französisch und Italienisch.

Seit 1996 bietet die Firma *ATX*, San Antonio, einen Telematics-Service an, bei dem diverse Informationen via Handy von einem menschlichen Operator abgefragt werden können. Dieser Dienst kostet eine jährliche Pauschale, die sich pro Kunde auf etwa $200 pro Jahr beläuft. Seit Ende 2000 wird dieser Service durch Einsatz von *IBM Direct Talk* und *ViaVoice* über eine Spracherkennung realisiert, bei der die Erkennungssoftware allerdings beim Provider und nicht im Fahrzeug läuft. Wie bei anderen automatischen Diensten bleibt für den Notfall weiterhin ein menschlicher Operator verfügbar.

Einen ähnlichen Dienst bietet *OnStar*, ein Tochterunternehmen des Fahrzeugherstellers *General Motors*, an. Auch hier wurde die ursprünglich menschliche Dienstleistung durch eine Spracherkennung teilweise ersetzt. Die Software dazu stammt von *Nuance* und *General Magic*. Benutzer können seit Ende 1999 verbal ihre E-Mail abfragen und sich über eine TTS-Einheit vorlesen lassen sowie ein Telefon über ein Dutzend Kommandos und Namenswahl bedienen. Die Namen werden dazu als gesprochene Schablonen auf einem Server hinterlegt. Das System wird nach eigenen Angaben in über einer Millionen Fahrzeugen betrieben.

Anfang 1999 wurde von *Buick* das Konzeptfahrzeug *Cielo* mit dem Spracherkennungssystem *Quiet Servant*, ebenfalls von *Visteon* entwickelt, vorgestellt. Verbal lassen sich Fenster und Dach öffnen, die Verriegelung der Türen manipulieren, Radio und Klimaanlage einstellen sowie das Navigationssystem ansteuern. Über das Vokabular des Navigationssystems werden keinerlei Informationen angegeben, daher wird es sich vermutlich um ein simples System, etwa ein verbales Pendant zur haptischen Variante handeln. Anfang 2001 wurde das weiter entwickelte System auch in das nächste Konzeptfahrzeug von *Buick*, den *Bengal*, integriert. Die Erkennung versteht 6 Sprachen, besitzt ein Lexikon von 118 Kommandos und bedient praktisch jede Funktion im Fahrzeug, so dass auf Schalter und Knöpfe bis auf einen auf verschiedene Funktionen umschaltbaren Joystick verzichtet wur-

de. Zu den bisher angesteuerten Geräten kommen Scheinwerfer, Scheibenwischer, Sitzeinstellung und Tempomat hinzu.

Im Jahr 2000 stellte DaimlerChrysler den Prototyp Truck *MAXXCab Dodge* vor, der ein Spracherkennungssstem benutzt, um ein Telefon, die Fahrzeugdiagnostik, einen Telematicsdienst und sogar das Internet anzusteuern. Wie die Realisierung tatsächlich aussieht und was die verarbeitbare Vokabulargröße ist, wird allerdings verschwiegen.

Alle bis Mitte 2002 veröffentlichten Spracherkennungssysteme für den automobilen Einsatz sind entweder nur in Prototypen realisiert worden oder basierend auf schon verfügbaren 'stationären' Spracherkennern aufgebaut, die serverseitig laufen und vom Fahrzeug aus angerufen werden müssen. Im Gegensatz dazu steht das im Rahmen der Arbeit entwickelte System, das auf einem integrierten Baustein lauffähig und somit für den Einsatz in (beliebigen) Fahrzeugen geeignet ist.

5 Ein System zur fahrzeugbasierten Spracherkennung

Im Rahmen dieser Arbeit wurde ein komplettes sprachgesteuertes Interface für den Einsatz im Fahrzeug realisiert, in das mehrere unterschiedlich aufgebaute Spracherkenner zusammen mit einer Analysekomponente integriert wurden.

Die einzelnen Erkenner sind der phonembasierte *Verbundworterkenner*, der alle Steuerkommandos und die Interaktion mit den einfacheren Komponenten realisiert, der graphembasierte *Städteerkenner* zur Bearbeitung von Eingaben für das Navigationssystem und schließlich der widerum phonembasierte, aber unspezifischere *Buchstabenerkenner* als Rückfalllösung zur Eingabe von in einem Lexikon enthaltenen Wörtern[1]. Das komplette System mit dem Namen *Cars* (**C**ar-based **a**utomatic **r**ecognition of **s**peech) wird in seiner Funktion im folgenden Abschnitt dargestellt. Anschließend folgt eine detaillierte Beschreibung der einzelnen integrierten Erkenner, mit Schwerpunkt auf den speziellen Eigenschaften des Städteerkenners.

5.1 Die Funktionalität

Als Repräsentanten für die verschiedenen anzusteuernden Geräte und Funktionen wurden das Handy, das Radio, das Navigationssystem sowie die Benutzerauthentisierung über einen Zahlcode gewählt. Diese Auswahl repräsentiert alle diejenigen Geräteklassen aus dem vorangegangenen Kapitel, für die eine Sprachansteuerung sinnvoll ist. Für Benutzerfeedback unterstützt die Parser- und Steuerungskomponente die Verwendung eines Displays und einer Sprachausgabe, die entweder über eine Syntheseeinheit oder mittels gesprochener Textschablonen geschieht.

Da das System nicht nur wie ein einfacher Spracherkenner ein Transkript der gesprochenen Äußerungen liefern, sondern auch auf die Eingaben reagieren soll, wurden die Erkenner unter einer Parserkomponente mit integrierter Funktionalität zusammengefasst. Das Parsen eines Kommandos geschieht über die Traversierung eines endlichen Automaten, der als Graph in den folgenden Abbildungen dargestellt werden kann (der komplette Automat ist zusätzlich in textueller Form

[1] Wie in Abschnitt 5.4 festgestellt wird, ist die Erkennung einzelner Buchstaben zu ungenau, um sie zur Eingabe von unbekannten Wörtern zu benutzen. Die Abbildung einer unsicheren Buchstabensequenz auf bekannte Wörter hingegen lässt sich realisieren

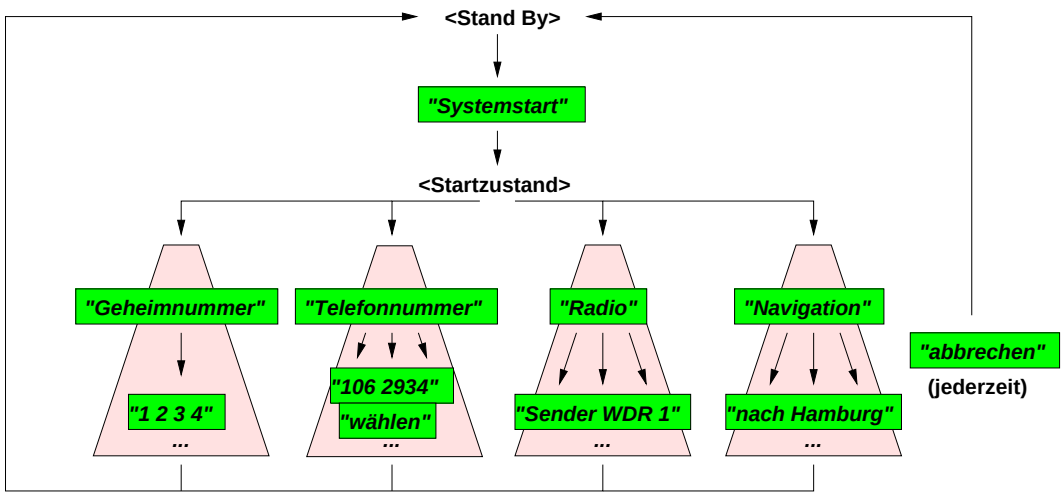


Abbildung 5.1: Übersichtsschema der Äußerungsverarbeitung

im Anhang A.4 zu finden). Dieser Automat wird über eine Liste von Quadrupeln (s, w, f, n) deklariert, die jeweils aus einem Zustandsnamen s, einem zu verarbeitenden Wort bzw. Wortgruppe w, einer Funktion f und einem Nachfolgezustandsnamen n bestehen. Wenn der Automat in einem bestimmten Zustand eines der akzeptierbaren Wörter bzw. Wortfolgen erhält, wird die entsprechende Funktion ausgeführt und anschließend der passende Nachfolgezustand eingenommen. Der Automat besitzt einen ausgezeichneten Startzustand namens '<Stand by>' (s. Abb.5.1), der beim Einschalten des Systems, nach der erfolgreichen vollständigen Traversierung eines Astes, nach Äußerung eines Abbruchkommandos oder beim Auftreten einer bestimmten Anzahl ungültiger Wörter eingenommen wird.

In jedem Knoten wird nur eine bestimmte Untermenge der möglichen Wörter aus dem Vokabular verarbeitet, bei der Eingabe von Geheimnummern beispielsweise die Menge der Ziffern von 0 bis 9. Jedes nicht zu den erlaubten Eingaben gehörende Wort wird als solches erkannt und gezählt. Traten mehrere in Folge auf (in Experimenten hat sich die Zahl Drei bewährt), wird die Eingabe als nicht interpretierbar angesehen und das System wieder in seinen Startzustand zurückversetzt. Somit bilden alle momentan nicht aktiven Wörter ein dynamisches *Garbagemodell* (s. Abschnitt 3.2.2), das umso besser funktioniert, je größer das Erkennerlexikon und damit auch die Anzahl der zur Modellierung der ungewünschten Eingaben heranziehbaren Wörter ist. Um beispielsweise das Sprachinterface zu aktivieren, d.h. den Automat von '<Stand by>' in den '<Startzustand>' zu schalten, wird als einzige Eingabe das Wort *'Systemstart'* akzeptiert. Somit stellt das gesamte Lexikon bis auf dieses Wort die Garbagemodelle, was zu einer sehr robusten Erkennung führt und damit verhindert, dass das Interface irrtümlich aktiviert wird. Im nächsten Schritt werden die anzusteuernden Komponenten

Benutzeranweisungen...	**Systemausgaben**
...Geheimnummer	
Systemstart Geheimnummer eins zwei drei vier	`Geheimnummer richtig`
...Radio	
Systemstart Radio Sender Bayern drei	Sprache: `Suche Sender` Display: `Suche Sender Bayern 3`
...Telefon	
Systemstart Telefonnummer drei fünf fünf (pause)	`3 5 5`
zehn sechs (pause)	`1 0 6`
löschen zwo null sechs wählen	`Ich wähle 3 5 5 2 0 6`
...Navigation	
Systemstart Navigation ich möchte nach Viernheim	`Sie möchten nach Weinheim?`
Nein ich möchte nach ich buchstabiere V I E R N H E	`Sie möchten nach Viernheim?`

Tabelle 5.1: Beispieläußerungen mit den entsprechenden Antworten des Basissystems für alle Verarbeitungszweige der Grafik 5.1.

ausgewählt, d.h. die Liste der erlaubten Wörter umfasst nun die vier Einträge *'Geheimnummer', 'Telefonnummer', 'Radio'* und *'Navigation'*. Obwohl die Zahl der zu erkennenden Wörter damit viermal höher ist, überwiegt die Anzahl der ungenutzten Wörter hier um ein Vielfaches. Somit ist auch diese Erkennung, genauso wie die in allen weiteren Verarbeitungsschritten, sehr robust.

Um die Geräte zu verwalten, werden verschiedene Informationen in internen Variablen vorgehalten, auf die die auszuführenden Funktionen lesend und schreibend zugreifen können. In diesen Variablen wird beispielsweise eine in einzelnen Ziffern gesprochene Telefonnummer zusammengesetzt oder der aktuelle Status des Radios (Aktivierungszustand, Sender, Lautstärke) abgelegt. Zur Erstellung des Automaten und Integration in das Gesamtsystem sind verschiedene Programme im Rahmen der Arbeit entwickelt worden, die eine einfache Erweiterung der gesamten Anwendung auf andere Geräte und andere Kommandos erlauben.

In der Tabelle 5.1 ist für jede der vier Anwendungen beispielhaft jeweils ein Benutzer-System-Dialog aufgeführt. Von oben nach unten entsprechen sie der Traversierung der dargestellten Teilpfade aus den nachfolgenden Grafiken 5.2 bis 5.4. Die Codeabfrage, links in Abbildung 5.2, ist das einfachste Anwendungsbeispiel, da lediglich ein linearer Pfad verfolgt wird. Dies stellt weder an die Konzentration des Benutzers noch an die Verarbeitungsstrategie des Systems große Anforderungen. Der Sinn der Codeabfrage ist die Erkennung des Benutzers, um einerseits den Zugriff auf weitere Systemfunktionen zu ermöglichen, andererseits aber auch um zuvor abgespeicherte, benutzerspezifische Einstellungen im Fahrzeug (Sitz-/Lenkradpositionierung, gespeicherte Sendereinstellungen, Anruferlisten etc,) und in der Spracherkennung (Adaptionsparameter) zu aktivieren. Die Einstellung des Radios (Abb. 5.3) erfordert aufgrund der Namens- und Frequenz-

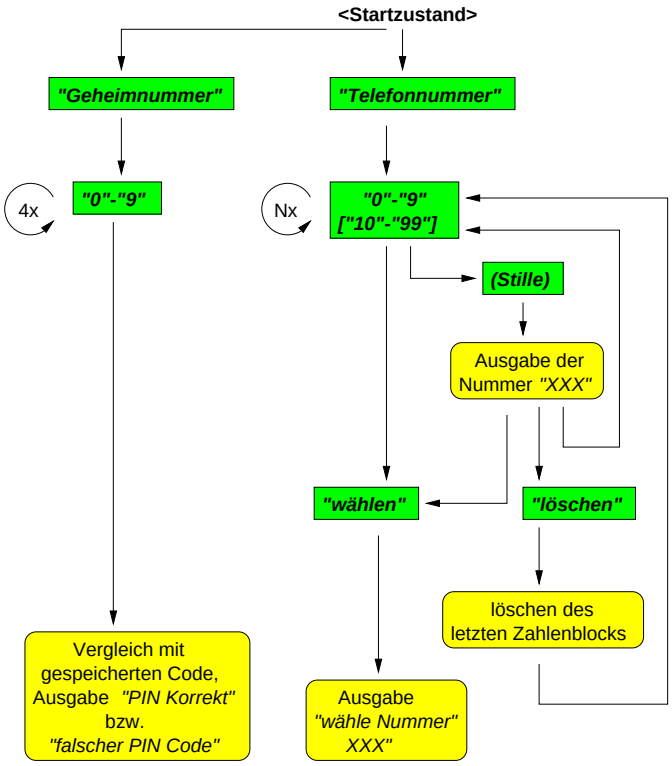


Abbildung 5.2: Ablaufschema zur Ansteuerung der Authentisierung und des Telefons. Eckige Kästen enthalten die jeweils gültigen Eingaben, abgerundete die Verarbeitungsschritte. Außer dem Startzustand sind alle Knoten durch das Kommando markiert, das zu ihnen führt.

Wahlmöglichkeit, sowie der potentiellen Buchstabierung verfügbarer Sender[2] eine kompliziertere Verarbeitungsstrategie, die aber immer noch weitgehend linear ist. Dagegen weist das nächste Beispiel, die Ansteuerung des Telefons (in Abbildung 5.2 rechts) durch die Möglichkeit, Ziffern in Blöcken zu nennen und auch wieder zu löschen, diverse Zyklen auf. Um den Benutzer während der Aufnahme der kompletten Telefonnummer eine Kontrollmöglichkeit zu geben, wird beim Auftreten von Sprechpausen der letzte Zahlenblock wiederholt und kann dann im Bedarfsfall gelöscht und verbessert werden. Die Bedienung des Navigationssystems (Abb. 5.4) schließlich ist im komplexesten Graphen dargestellt. Neben dem Basiserkenner werden im Bedarfsfall noch zusätzlich der Buchstaben- und der Städteerkenner eingesetzt. Im Beispiel aus der Tabelle 5.1 wird bei Nennung eines Städtenamens der Städteerkenner aktiviert, liefert aber das falsche Ergeb-

[2] Alle Sendernamen sind nicht unbedingt a priori bekannt, können aber via RDS beim Betreten des Sendebereichs vom Radio übernommen und potentiell in das Senderlexikon der Erkenners integriert werden.

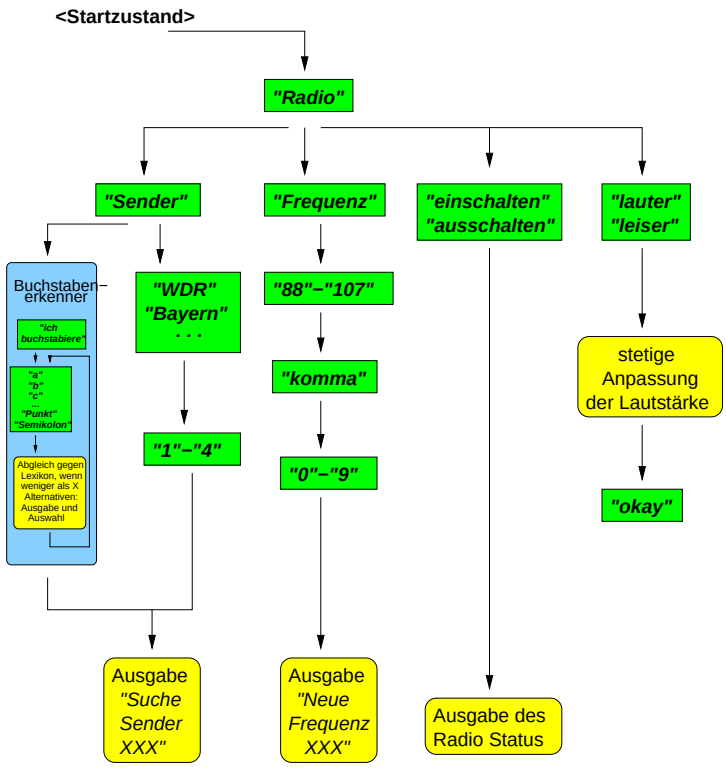


Abbildung 5.3: Ablaufschema zur Ansteuerung des Radios. Der Buchstabierer-kenner ist in Grafik 5.4 vergrößert dargestellt

nis[3]. Ein erneuter Versuch, diesmal mit dem Buchstabenerkenner, ist erfolgreich. Da bei dieser Erkennung ein Lexikonabgleich stattfindet, bei dem nach Erreichen einer gewissen Präfixlänge keine Alternativen mehr übrigbleiben, wird dazu nicht die komplette Buchstabierfolge benötigt.

Für eine ansprechendere Visualisierung wurde eine grafische Oberfläche entwickelt, die die Ausgabe des Erkenners, die Rückgabe des Systems (auch als Sprachschablone) und den Zustand des Traversierungsautomaten anzeigt sowie die angesteuerten Geräte simuliert. Diese Visualisierung ist primär für Demonstrationszwecke und nicht für den tatsächlichen Einsatz im Fahrzeug konzipiert worden, da sie zu komplex ist und ablenkend wirken würde. Im Fahrzeug wird stattdessen lediglich akustisches Feedback zur Verfügung gestellt, das auf die nötigen Ausgaben beschränkt bleibt. Die Abbildung 5.5 zeigt einen Screenshot der Oberfläche im Betrieb. Die geäußerten Anweisungen (s. unteren Fensterbereich in der Abbildung) folgen im Wesentlichen dem Beispiel aus der Tabelle 5.1.

[3]Das Beispiel ist aus Gründen der Anschaulichkeit vereinfacht. In Wirklichkeit liefert der Städteerkenner nicht nur ein einziges sondern bis zu fünf Ergebnisse zur verbalen Auswahl durch den Benutzer.

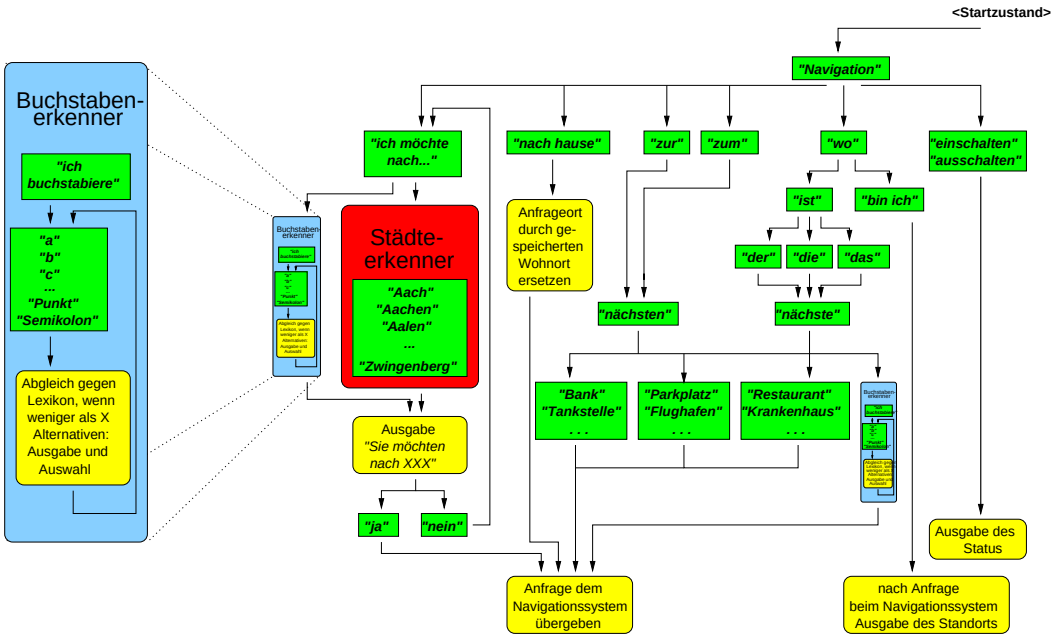


Abbildung 5.4: Ablaufschema zur Ansteuerung des Navigationssystems. Links ist die Vergrößerung des Buchstabiererkenners, der in mehreren Teilpfaden (s.auch Grafik 5.3) vorkommt.

Weitere Details zu den einzelnen Verarbeitungspfaden der Traversierungsgraphen befinden sich in den drei folgenden Erkennerbeschreibungen.

5.2 Der Verbundworterkenner

Der Verbundworterkenner versteht die wesentlichen Kommandos, die zur Bedienung einzelner Komponenten im Fahrzeug sowie zur Kontrolle des Erkenners selber benötigt werden. Nur über ihn kann die Verarbeitung gestartet (s. Abbildung 5.1 *Systemstart*) und gegebenenfalls Knoten erreicht werden, die die anderen Erkenner aktivieren. Durch ihn erfolgt die primäre Auswahl der anzusteuernden Komponenten (s. Abbildung 5.1 *Navigation*, *Geheimnummer*, ...) sowie der Abbruch der aktuellen Traversierung durch das entsprechende Kommando.

Der Erkenner des Systems ist auf den *SLACC*-Korpus[4] trainiert worden, der im Rahmen der Arbeit unter realen Bedingungen, das heißt während der Fahrt in verschiedenen Fahrzeugen, von 22 Sprechern über zwei unterschiedliche Mikrofone aufgenommen wurde (s. Abschnitt 6.2 für Details).

Der Basiserkenner [15] benutzt als Merkmale, wie die anderen hier aufgeführ-

[4] *SLACC* - (**S**poken **LA**nguage **C**ar **C**ontrol) Fahrzeugsteuerung mit Sprache.

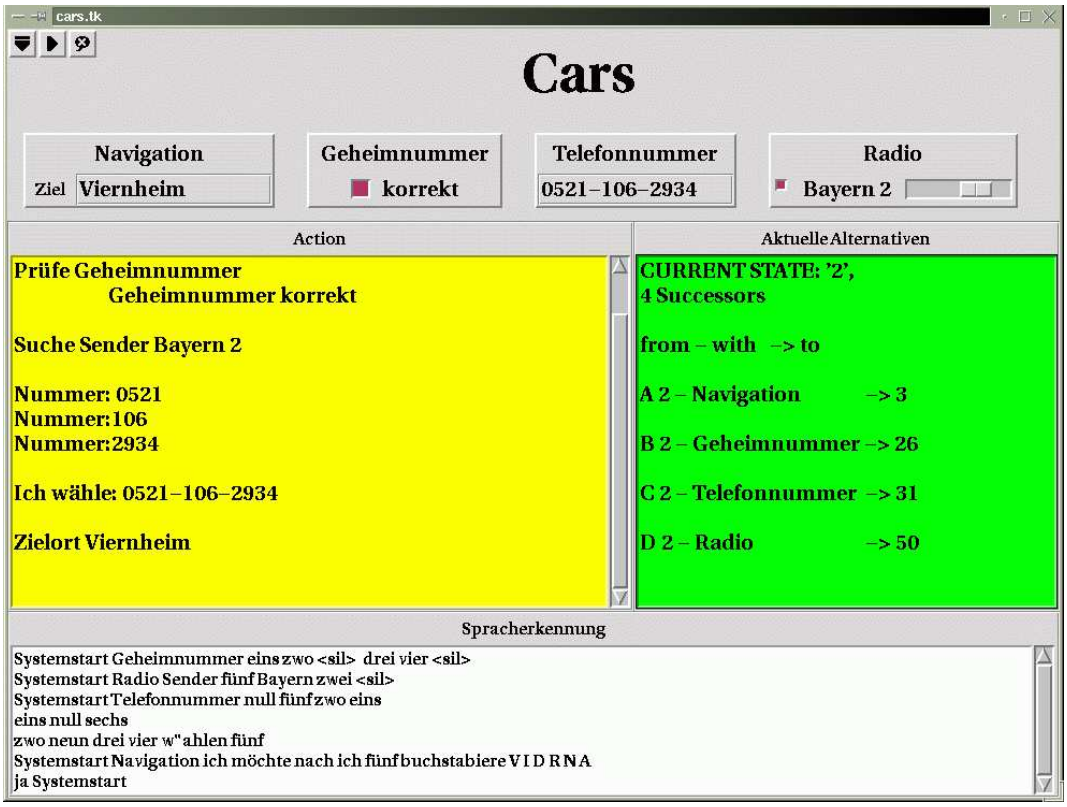


Abbildung 5.5: Visualisierung des Basissystems in der Laptop Variante. Im oberen Viertel befinden sich die simulierten anzusteuernden Geräte, darunter im linken Fenster die Systemantworten, im rechten die aktuell möglichen Teilpfade, unten die Ergebnisse der Erkennungskomponente.

ten Erkenner, 39komponentige Merkmalsvektoren, die aus den ersten 12 Cepstralkomponenten und der skalierten Signalenergie bestehen, sowie deren erste und zweite Ableitung. Das System basiert auf semikontinuierlichen HMMs, die auf 1009 Gaussverteilungen zurückgreifen. Bis auf einige Ausnahmen werden die Worte über Triphone als Wortuntereinheiten modelliert. Kam eine ausreichende Zahl von Trainingsbeispielen vor, sind Ganzwortmodelle gewählt worden. Die Implementierung folgt im Wesentlichen dem in Abschnitt 2.2 beschriebenen Aufbau. Da der Erkenner zusammen mit der Verstehenskomponente eingesetzt wird und das Konzept der dynamischen Garbagemodelle unterstützt werden soll, wurde keine Grammatik und kein Sprachmodell für die Steuerung des Erkennungsprozesses eingesetzt. Als Trainingsmaterial dienten etwa 13 Stunden Aufnahmen aus dem *SLACC*-Korpus. Das Lexikon umfasst etwa 650 Wörter, wobei die Frequenz der Wörter im Trainingskorpus ihre Wichtigkeit widerspiegelt: Die höchste Priorität

haben die Steuerkommandos, gefolgt von den Zahlen zur Ansteuerung des Telefons und Auswahl aus Listen.

5.3 Städteerkenner

Das Modul *Städteerkenner* ist ein eigenständiger Spracherkenner, dessen Lexikon zwar bekannt, aber sehr groß ist. Da mit steigender Vokabulargröße die Erkennung instabiler wird und das Grundsystem möglichst stabil funktionieren muss, wurde die Städteerkennung nicht direkt in das Basissystem integriert, sondern als ein Modul realisiert [59].

Die Aufgabe des Städteerkenners ist die Verarbeitung von Adressen zur Ansteuerung des Navigationssystems. Damit der erhebliche Erkennungsaufwand des großen Vokabulars eingeschränkt wird, wurde das System auf Einzelworterkennung reduziert. Eine Äußerung (also ein in Pausen eingebettetes Stück Sprachsignal) für diesen Erkenner entspricht demnach exakt einem Städtenamen. Zur vollständigen Eingabe eines Ortes in das Navigationssystem gehört neben der Stadt auch noch eine Straße. Es gibt in Deutschland etwa 400.000 verschiedene Straßennamen, was die Erkennungsaufgabe theoretisch um eine Größenordnung komplexer macht. Da in jeder Stadt aber nur eine Teilmenge der möglichen Straßennamen vorkommt, wird nach Erkennung der Stadt das Erkennerlexikon im aufwändigsten Fall[5] in etwa die gleiche Größenordnung haben, in der Regel jedoch viel kleiner sein. Daher ist die Straßenerkennung in dieser Arbeit als redundantes Beispiel ausgeklammert worden.

Die Integration des Städteerkenners in das Gesamtsystem erfolgt über die Aktivierung des Navigationssystems durch eine Floskel (*'Navigation, ich möchte nach ...'*, s. Abb.5.4 Mitte). Falls der Benutzer es wünscht, kann die Eingabe allerdings auch über den Buchstabenerkenner erfolgen, der im nächsten Abschnitt behandelt wird.

A priori ist das Erkennungslexikon für die Aufgabe bekannt, es entspricht allen im Navigationssystem enthaltenen Namen. Da aufgrund der Lexikongröße die Erkennung nicht perfekt arbeitet, wird beim Generieren des Ergebnisses nicht nur ein Name, sondern eine Liste von n Namen zur Verfügung gestellt, aus der der Benutzer mit simplen Blätteranweisungen oder durch Nummernnennung auswählen kann. Zusätzlich zu den Städtenamen gibt es noch eine Anzahl von dynamischen Ortsbezeichnungen, wie *'nach hause'*, die vom Benutzer vorab definiert werden müssen oder auch *'zur nächsten Bank/Tankstelle/Sehenswürdigkeit'* etc., die entweder vom Navigationssystem oder einem Online-Informationsdienst auf bestimmte Adressen abgebildet werden müssen.

Bei der Erstellung des Städteerkenners musste ein sehr großes Vokabular betrachtet werden. Ein allgemeines Problem in der Spracherkennung beim Umgang

[5] Mit 13.000 Straßennamen ist dies Berlin.

mit großen Wortschätzen ist, dass von allen Wörtern Transkriptionen verfügbar sein müssen sowie eine ausreichende Anzahl Sprachaufnahmen existieren muss, um Ganzwortmodelle trainieren zu können. Im allgemeinen bedeutet dies für den Ersteinsatz eines neuen Vokabulars einen erheblichen personellen und zeitlichen Aufwand. Für den Städteerkenner wurde daher eine andere Vorgehensweise untersucht: Anstatt auf Phonemmodellen und den damit verbundenen phonetischen Transkriptionen aller Namen wurde das System auf Graphemmodellen aufgebaut. Ein ähnlicher Ansatz wurde schon 1993 in [62] als potentiell einsetzbar vorgeschlagen, aber später nicht mehr verfolgt. Die graphemische Transkription ist identisch mit dem Namen des Lexikoneintrags. Abgesehen von drei sehr simplen, regelgetriebenen Umsetzungen, der Abbildung von *sch*, *ch* und *ck* auf jeweils ein einzelnes Sonderzeichen, basiert das System lediglich auf diesen textuellen Repräsentationen. Somit ist diese Art der Erkennung potentiell auch für andere Anwendungen geeignet, in denen eine phonetische Transkription nicht vorliegt, wohl aber die Orthographie des Wortes, wie beispielsweise beim Telefonbuch eines Handys.

Der Städteerkenner unterstützt nur die Erkennung der Städtenamen, nicht die der variablen Ortsbezeichnungen. Diese wurden bei der Erstellung des SLACC-Korpus und dem darauf basierenden Training des Grundsystems schon vorgesehen. Damit existieren für diese Wörter sicher erkennbare Phonemmodelle. Um diesen Vorteil nicht durch die Integration in den Städteerkenner zu stören, werden die variablen Ortsbezeichnungen noch auf Ebene des Basissystems erkannt.

Der nachfolgende Abschnitt definiert zunächst ein paar linguistische Begriffe und erläutert dann den Aufbau verschiedener graphembasierter Verfahren zur Erkennung der Städtenamen.

5.3.1 Phonem - Graphem oder Man schreibt wie man spricht

Alle gesprochenen Sprachen setzen sich aus Lauten zusammen, den Phonemen. Die gängige Definition der Phoneme und damit des Phoneminventars einer Sprache basiert auf Minimalbildung von Wortpaaren:

Die **Phoneme** einer Sprache sind die kleinsten Lauteinheiten, die zwei verschiedene Wörter voneinander unterscheiden können [8], z.B. `/I/`[6] und `/a/` in *Binde* und *Bande*.

Es gibt auf der Welt etwa 140 verschiedene Phoneme [41], von denen jede Sprache lediglich ein paar Dutzend nutzt. Für das Deutsche sind es etwa 50 Phoneme, aber je nach Grad der Auflösung und der Feinheit des Gehörs kann diese Zahl variieren. Innerhalb dieser Arbeit wurde ein Satz von 46 Phonemen benutzt, die in der *SAMPA*[7]-Notation aufgeführt werden (s. Anhang A.1).

[6] Um Phoneme zu markieren werden sie durch Schrägstriche "/" geklammert, Grapheme werden im weiteren mit Hochkomma "'" markiert.

[7] SAMPA = **S**peech **A**ssessment **M**ethods **P**honetic **A**lphabet, *SAMPA*(s. [20], S. 684ff.)

Von den Phonemen kann es unterschiedliche Ausprägungen, sogenannte **Allophone** geben. Einerseits können diese Unterschiede unabhängig vom phonetischen Kontext auftreten, beispielsweise in dialektischer Variation wie dem Zungen-r und dem Zäpfchen-r, andererseits gibt es sie auch kontextabhängig, etwa bei Koartikulation nach vorderen und hinteren Lauten (wie die Umsetzung von 'i' und 'a'). Im Beispiel *Licht (in SAMPA* `/lICt/`*)* und *lacht (in SAMPA* `/laxt/`*)* wird das 'ch' bedingt durch den davor liegenden Vokal ebenfalls vorne bzw. hinten artikuliert. Die tatsächliche Ausprägung eines Allophones bezeichnet man schließlich als **Phon**. Da die Zahl der Phone der Gesamtheit der von allen Menschen bisher geäußerten Lauten entspricht und damit eine ständig wachsende und sehr flüchtige Menge von Artikulationen bezeichnet, wird im weiteren der Begriff 'Phonem' synonym benutzt.

Während das Phonem als Lauteinheit (die akustisch wahrgenommen wird) Segmente eines Sprachsignals repräsentiert, entspricht das **Graphem** den Atomen eines Textes, respektive eines Zeichens (das optisch wahrgenommen wird).

Im Gegensatz zum Phonem stellt das Graphem eine simplere Einheit dar. Es gibt sie nicht in verschiedenen Ausprägungen, sozusagen Allographeme, sondern nur als einzelne Zeichen des Alphabets. Das Grapheminventar des Deutschen ist somit 30 Zeichen groß, 26 Buchstaben, 3 Umlaute und die Sz-Ligatur[8].

Weiterhin ist ein Text in der abstrakten Graphemdarstellung unveränderlich, wogegen eine phonetische Darstellung einer Äußerung stark vom Sprecher beeinflusst wird. Die Grapheme liegen tatsächlich als abstrakte Einheiten wie beispielsweise ihrem ASCII-Code vor und nicht als Realisierungen (wie etwa die Phone) in Form von verschiedenen handschriftlichen Dokumenten.

Für den folgenden Erkenner ist die grundlegende Frage, ob es eine (eindeutige) Abbildung der geäußerten Phoneme auf die zu erkennenden Grapheme gibt. Der Erkenner selbst basiert auf Graphemen als Wortuntereinheiten, während er die Lauteinheiten der Signale als Trainingsmaterial benutzt bzw. in der Arbeitsphase erkennt. Damit ist die Problematik umgekehrt zu der eines Sprachsynthesesystems, in dem aus geschriebenen Text eine phonetische Transkription und daraus ein Sprachsignal erzeugt wird.

Von der Entstehungsgeschichte der Alphabetschrift her sollte man hoffen, dass eine eindeutige Abbildung existiert. Die ursprüngliche Motivation der Verschriftlichung war eine materielle Fixierung von Sprache, also eine Zuordnung von Lauten zu schreibbaren Zeichen (Buchstaben). Basierend auf dem lateinischen Alpha-

hat gegenüber anderen Notationen wie IPA (**I**nternationales **P**honetisches **A**lphabet) den Vorteil, dass es mit dem Standard-Ascii-Zeichensatz auskommt.

[8] Anmerkung: andere Zeichen der Schriftsprache, vor allem der Interpunktion, werden im Weiteren nicht betrachtet, obwohl diese zur Sinntrennung und Typbestimmung von Sätzen nicht vernachlässigbar ist und natürlich zur Markierung von Abkürzungen wie etwa *St. Peter Ording* oder *Parkstr.* benutzt wird. Letzteres Problem wurde in den Experimenten durch Ausformulierung umgangen.

bet entwickelten die europäischen Sprachen in Abhängigkeit ihrer eigenen Phonologien verschiedene Alphabete, die mehr oder weniger gut die Sprache abbilden konnten. Dadurch und durch historische Veränderungen resultieren verschiedene orthographische Inkonsistenzen [8]. Obwohl es erst kürzlich in der Rechtschreibreform durch die Eindeutschung von Fremdwörtern ('Delphin' wurde zu 'Delfin'), durch Konservierung von Morphemen[9], um Wörter leichter ableiten zu können ('Gräueltaten'), und durch eine einfachere Regelung der Kennzeichnung langer Vokale wieder den Versuch gab, das graphemische System dem phonetisch-phonologischen zu nähern, schreibt man leider nicht immer, wie man spricht:

Das einfache Beispiel der Wörter *'Stil'*, *'Stiel'* und *'stiehl'*, die alle als `/Sti:l/` gesprochen werden, zeigt dies. Im Übrigen ist die andere Richtung, dank der Aussprachevarianten, genauso mehrdeutig: `/bU6k/`, `/bUak/`, und `/bu6X/` wird (fast) jeder als *'Burg'* verstehen.

Das Zuordnungsproblem existiert in alle Richtungen:

1. es gibt Grapheme, die für verschiedene Laute stehen (z.B. 's', in 'stiehlst' steht für die Laute `/S/` und `/s/`) bzw. durch Assimilationseffekte und Auslautverhärtung wie andere Grapheme gesprochen werden ('n' in 'fünf' wird als `/m/` gesprochen, 'b', 'd' und 'g' an Silbengrenzen als `/p/`, `/t/` und `/k/`).

2. Verschiedene Graphemsequenzen können den gleichen Laut ausdrücken (die oben aufgeführten `/i:/` Varianten).

3. Einzelne Grapheme können für mehrere Laute stehen ('z' für `/ts/` und 'x' für `/ks/`).

4. Unterschiedliche Graphemfolgen werden auf nur einen Laut abgebildet ('sch' auf `/S/`, 'ch' auf `/C/` oder `/x/`, 'ck' auf `/k/`)

5. bestimmte Graphemsequenzen führen zu Aussprachevеränderungen von benachbarten Zeichen (Vokalverkürzungen nach Doppelkonsonanten, Veränderung des 'ch' nach `/i/`, `/e/` und `/a/`, `/o/`, `/u/`,)

Die Frage der Eindeutigkeit der Abbildung muss also verneint werden. Dennoch zeigen gängige Text-to-Speech-Systeme, dass es für den Übergang von Graphemen zu Phonemen dank eines fundierten linguistischen Regelwerks mittlerweile praktikable Lösungen gibt, wobei diese Verfahren Ausnahmen von den Regeln, etwa Leihwörter aus anderen Sprachen, nur mit einem Lexikon, das wieder ein phonetisches Transkript der gängigsten Fremdwörter enthält, behandeln können.

[9] Ein Morphem ist die kleinste, bedeutungstragende Einheit einer Sprache. Das Wort 'zerstört' setzt sich aus den Morphemen 'zer' 'stör' und 't' zusammen. Morpheme sind nicht Gegenstand dieser Arbeit, für eine ausführliche Betrachtung s. [45].

Sicherlich kann man ähnliche Regeln für die benötigte Rücktransformation angeben, aber dazu gehört ein erschöpfendes linguistisches Wissen und wiederum eine spezielle Ausnahmefallbehandlung.

Der hier untersuchte Ansatz geht einen ganz anderen Weg. Ausgehend von einem ausreichend großen Korpus wurde eine streng datengetriebene Vorgehensweise gewählt, ohne explizites Fachwissen hinzuzuziehen. Es wurde nahezu eine 1:1-Umsetzung von Phonemen auf Grapheme gewählt. Die kleinen Einschränkungen der letzten Bemerkungen begründen sich darin, dass es eine minimale Konversion gab. Drei Graphemketten sind auf ein gesondertes Zeichen abgebildet worden: *'sch', 'ch'* und *'ck'*, da sie im wesentlichen als lediglich ein Laut realisiert werden. Die Umsetzung des 'ck' gestaltet sich unkritisch, da in (deutschen) Wörtern die 'c' Endung niemals auftritt und die Zeichenkette somit auch nicht in einer Aussprachevariante an Silbengrenzen vorkommt. Das 'ch' ist aus oben aufgeführten Gründen etwas kritisch, da es in den zwei Versionen *vorne* und *hinten* artikuliert existiert. Das 'sch' ist noch problematischer, da es sowohl zusammenhängend gesprochen, wie in 'Fisch' (`/fiS/`) als auch zerlegt, wie beispielsweise in Diminutiva von auf 's' endenden Wörtern vorkommt, etwa in 'Mäuschen' `/mOIsCn/`. Da dieser Fall aber im gewählten Korpus sehr selten vorkam, wurde dieses Problem ignoriert.

Die durchgeführten Experimente teilen sich in zwei unterschiedliche Ansätze auf: ein *Graphem**-Erkenner mit unterschiedlich erstellten Sprachmodellen verschiedener Längen (s. Grafik 5.6 A) und ein baumbasierter Erkenner mit trigraphemischen Wortuntereinheiten (s. Grafik 5.6 B).

5.3.2 *Graphem**-Erkennung

Im ersten Ansatz wurde ein auf Monographemen[10] basierender Erkenner entwickelt, dessen Ausgabe beliebig lange Graphemketten[11] sind. Da die Erkennung aufgrund der kleinen Wortuntereinheiten nicht sehr verlässlich ist, wird sie durch den Einsatz eines Sprachmodells unterstützt. Die so erhaltene Ergebniskette muss gegen ein bestehendes Lexikon abgeglichen werden und das ähnlichste bzw. die $n-$ähnlichsten Wörter des Lexikons bilden die Ausgabe des Moduls. Im Weiteren behandelt dieser Abschnitt zunächst die Theorie von statistischen Sprachmodellen, gefolgt von verschiedenen Variationen der Abstandsmessungen zwischen der Graphemkette und jedem Wort eines gegebenen Lexikons.

[10] Die für die Erkennung benutzten Wortuntereinheiten waren alleinstehende Grapheme.

[11] In Anlehnung an die Darstellung regulärer Ausdrücke werden die Ketten als *Graphem** bezeichnet.

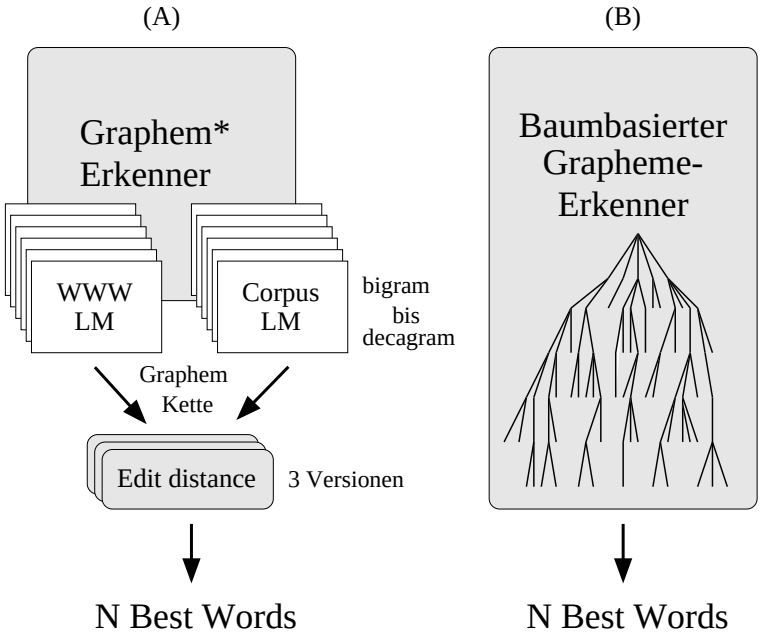


Abbildung 5.6: Überblick der Experimente zur Graphemerkennung

Sprachmodelle für die *Graphem**-Erkennung

Um die Erkennung eines allgemeinen Spracherkenners zu verbessern, kann Weltwissen über Sprachmodelle (engl. Language Model, LM) in das System integriert werden. Zu den unterschiedlichen Arten von Sprachmodellen gehören die binär entscheidenden formalen Grammatiken (Symbolfolge gehört oder gehört nicht zum Sprachumfang der Grammatik), die Wahrscheinlichkeiten angebenden stochastischen Grammatiken (Symbolfolge gehört mit einer bestimmten Wahrscheinlicheit, die auch 0 sein kann, zu dieser Grammatik) und die potentiell alle Symbolfolgen zulassenden statistischen Sprachmodelle (mit unterschiedlichen Wahrscheinlichkeiten).

Eine formale Grammatik wurde im baumbasierten Alternativexperiment (s. S.84 für Details) benutzt, sie eignet sich jedoch wegen ihrer Restriktivität nicht für eine beliebige Graphemfolgenerkennung. Von den beiden Wahrscheinlichkeiten angebenden Verfahren ist die stochastische Grammatik aufgrund ihrer arbeitsaufwändigen Realisierung ungeeignet, da das gesamte Verfahren der Graphemerkennung unüberwacht/datengetrieben durchgeführt werden soll. Im weiteren geht es an dieser Stelle daher nur noch um statistische Sprachmodelle.

Einfach ausgedrückt sind Sprachmodelle Wahrscheinlichkeitsfaktoren von Sequenzen definierter Länge. Im Allgemeinen werden sie auf Wortebene eingesetzt, beschreiben also je nach gewählter Tiefe Wahrscheinlichkeiten von Wortpaaren (Bigramm), Worttriplets (Trigramm) etc. Innerhalb dieses Abschnitts wird die

Erkennung auf Zeichenebene betrachtet, daher entsprechen die Sprachmodelle Wahrscheinlichkeiten von *Graphem*sequenzen.

Während ein Spracherkenner i.A. kompliziert zu gewinnende annotierte Sprachsignale für das Training benötigt, geschieht die Berechnung eines Sprachmodells auf leichter zu beschaffendem reinem Textmaterial, wobei aber auch hier gilt, dass das resultierende System umso besser wird, je näher die Daten an der späteren Einsatzdomäne liegen.

Die Wahrscheinlichkeit einer Zeichenfolge[12] W der Länge k entspricht der Verbundwahrscheinlichkeit aller Einzelkomponenten

$$\begin{aligned} P(W) &= P(w_1 \dots w_k) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_k|w_1w_2 \dots w_{k-1}) \\ &= \prod_{i=1}^{k} P(w_i|w_1 \dots w_{i-1}) \end{aligned} \tag{5.1}$$

Gleichung 5.1 ist aufgrund der beliebigen Größe der Geschichte einerseits in der Praxis nicht handhabbar, andererseits ist in den wenigsten Fällen ein am Anfang einer längeren Äußerung aufgetretenes Zeichen von Belang für deren Ende. Daher beschneidet man die Geschichte auf $n-1$ Ereignisse und erhält somit $n-$Gramme:

$$P(W) \approx \prod_{i=1}^{k} P(w_i|\underbrace{\underbrace{w_{i-n+1} \dots w_{i-1}}_{n-1\,\text{Werte}}}_{n\,\text{Werte}}) \tag{5.2}$$

Die Gewinnung der Sprachmodelle geschieht durch simples Auszählen auf einem Trainingstext. Da aber vor allem für große n viele Ereignisse nicht gesehen werden[13] und ihnen, anders als bei den stochastischen Grammatiken, eine gewisse Grundwahrscheinlichkeit gegeben werden muss, wird ein zusätzliches Verfahren benötigt, um mit den ungesehenen Ereignissen umzugehen.

Der Schätzwert $\hat{P}$ für eine $n-$ elementige Zeichenfolge $w_1 \dots w_n$ ist zunchst die relative Häufigkeit (Anzahl Vorkommen der Kette $= c(w_1 \dots w_n)$) im Trainingstext mit N Ketten.

$$\hat{P}(w_1 \dots w_n) \approx \frac{c(w_1 \dots w_n)}{N}$$

[12] Im Folgenden werden aus Konsistenzgründen zu gängigen Definition von Sprachmodellen, die über Wörter geschehen, auch für Zeichen die Symbole w_i benutzt.

[13] Die Anzahl verschiedener Ereignisse hängt natürlich nicht nur von der Größe der Geschichte, sondern im Wesentlichen von der Größe des Vokabulars ab. Bei der Graphemerkennung betrgt die Vokabulargröße lediglich 33 Zeichen, somit ist die Anzahl der möglichen Folgen für ein Pentagramm lediglich $33^5 \approx 40.000.000$ groß. Bei 50 Zeichen wären es schon $50^5 \approx 312.500.000$ Folgen.

Analog kann die bedingte Wahrscheinlichkeit durch

$$\begin{aligned} \hat{P}(w_n|w_1 \ldots w_{n-1}) &= \frac{\hat{P}(w_1 \ldots w_n)}{\hat{P}(w_1 \ldots w_{n-1})} \\ &\approx \frac{c(w_1 \ldots w_n)}{c(w_1 \ldots w_{n-1})} \end{aligned} \tag{5.3}$$

berechnet werden.

Um unerwünschte Null-Wahrscheinlichkeiten für unbeobachtete Wortfolgen zu umgehen, wird zuerst etwas Wahrscheinlichkeitsmasse von den gesehenen Ereignissen abgezogen, d.h. eine Neuschätzung der gesehenen Wahrscheinlichkeiten durchgeführt, und anschließend die Masse auf die ungesehenen Ereignisse verteilt.

Zur Gewinnung der Wahrscheinlichkeitsmasse und zu ihrer anschließenden Verteilung gibt es viele verschiedene Verfahren. Beispielhaft wird hier für beide Schritten jeweils eines vorgestellt. Weitere Informationen finden sich in [28].

Neuschätzung der gesehenen Ereignisse

Ein in der Praxis bewährtes Schätzverfahren ist *shift* β. Von jedem gesehenen $n-$Gramm wird ein konstanter Faktor β mit $(0 < \beta \leq 1)$, subtrahiert (engl. *absolute discounting*, im Gegensatz zu anderen Verfahren, in denen proportionale Faktoren abgezogen werden, *linear discounting* genannt). Die neugeschätzte, reduzierte Häufigkeitsverteilung f^* für ein Zeichen z mit der $n-1$ elementigen Geschichtssequenz $\underline{y}$ ergibt sich somit zu:

$$f^*(z|\underline{y}) = \frac{c(\underline{y}z) - \beta}{c(\underline{y})} \qquad \forall c(\underline{y}z) > 0 \tag{5.4}$$

die Neuschätzung ist demnach höchstens gleich, in der Regel aber geringer als die alte Verteilung:

$$f^*(z|\underline{y}) \leq \frac{c(\underline{y}z)}{c(\underline{y})} = f(z|\underline{y}) \tag{5.5}$$

Die somit gewonnene Wahrscheinlichkeitsmasse λ berechnet sich aus der Anzahl der verschiedenen beobachteten $n-$Gramme mit der Geschichte $\underline{y}$, die als $d(\underline{y}, \_)$ bezeichnet werden:

$$\lambda(\underline{y}) = \frac{d(\underline{y}, \_)\beta}{c(\underline{y})} \tag{5.6}$$

Umverteilung der Wahrscheinlichkeitsmasse

Die im vorigen Schritt gewonnene Wahrscheinlichkeitsmasse muss nun auf die ungesehenen Ereignisse verteilt werden. Auch dafür gibt es verschiedene Algorithmen. Im Folgenden soll als Beispiel das *backing-off-(engl. zurückweichen)*-Verfahren erläutert werden.

Die neue a-posteriori-Wahrscheinlichkeit für ein Zeichen z mit der Geschichte $\underline{y}$ ergibt sich aus der reduzierten Häufigkeitsverteilung, falls die Sequenz $\underline{yz}$ existiert (also $c(\underline{yz}) > 0$ gilt). Ansonsten wird auf die Wahrscheinlichkeit von z mit der um ein Element verminderten Geschichte zurückgegriffen, multipliziert mit der zu verteilenden Wahrscheinlichkeitsmasse der gesamten Geschichte. Das Verfahren wird im schlechtesten Fall immer wieder iteriert, bis die Geschichte auf das Unigramm zurückfällt.

$$P(z|\underline{y}) = \begin{cases} f^*(z|\underline{y}) & \text{falls} \quad c(\underline{yz}) > 0 \\ K_y \lambda(\underline{y}) P(z|y_2 \dots y_{n-1}) & \text{falls} \quad c(\underline{yz}) = 0 \end{cases} \tag{5.7}$$

mit K_y als Normierungsfaktor, um die Bedingung $\sum_z P(z|\underline{y}) = 1$ zu erfüllen. Aus $\sum_z f^*(z|\underline{y}) = 1 - \lambda(\underline{y})$ folgt für K_y

$$K_y = \frac{1 - \sum_{z:c(\underline{yz})>0} f^*(z|\underline{y})}{\lambda(\underline{y}) \sum_{z:c(\underline{yz})=0} P(z)} \tag{5.8}$$

Während die Wahrscheinlichkeitsmasse vorab gewonnen wird, wird die Umverteilung auf ungesehene Ereignisse erst im laufenden Betrieb des Erkenners gemacht. Allein dadurch ist es möglich, ohne gigantische[14] Lookup-Tabellen auszukommen.

Qualität der Sprachmodelle

Die Güte eines Sprachmodells wird über seine Restriktivität gemessen: je besser es ist, umso weniger Alternativen werden favorisiert. Um eine qualitative Aussage treffen zu können, wird das Sprachmodell an einer (repräsentativen) Teststichprobe evaluiert. Das so gewonnene Maß ist die Perplexität. Zu ihrer Berechnung bedient man sich eines Wertes aus der Informationstheorie: der Entropie H einer Sprache L:

$$H(L) \quad = \quad \sum_{w_1 \dots w_m} \frac{1}{m} P(w_1 \dots w_m) \log_2 \left(\frac{1}{P(w_1 \dots w_m)} \right)$$

[14] Bei einem Dekagramm über 33 Grapheme gibt es 33^{10} Ereignisse, die jeweils eine Wahrscheinlichkeit haben. Wird diese in 4 Byte abgelegt, nehmen alle Wahrscheinlichkeiten 5.9 Petabyte ein.

$$= -\sum_{w_1 \ldots w_m} \frac{1}{m} P(W) \log_2 P(W) \tag{5.9}$$

Die Entropie gibt die für die Vorhersage des nächsten Wortes benötigte Anzahl Bits an. Die Entropie eines Sprachmodells wird nur auf der N-elementigen Stichprobe $W = w_1 \ldots w_N$ geschätzt, ihr Schätzwert ergibt sich aus:

$$\hat{H}_{LM} = -\frac{log_2(P_{LM}(W))}{N} \tag{5.10}$$

dabei ist $P_{LM}(W)$ die durch das Sprachmodell LM berechnete Wahrscheinlichkeit der Sequenz.

Aus dem Schätzwert 5.10 kann nun die Perplexität PP des Sprachmodells berechnet werden:

$$PP_{LM} = 2^{\hat{H}_{LM}} = \frac{1}{\sqrt[m]{P_{LM}(W)}} \tag{5.11}$$

Je kleiner die Perplexität des Sprachmodells ist, umso stärker kann es einschränkend auf den Erkennungsprozess wirken. Dabei hängt diese Zahl natürlich auch von der Größe des zugrunde liegenden Vokabulars ab. Große Perplexitäten können nur von entsprechend großen Wortschätzen erzeugt werden. Die Erkennung von beliebigen Ziffernfolgen hat eine Perplexität von 10 (wobei bei beliebigen Ziffernfolgen ein Sprachmodell irrelevant wäre). Die Perplexität des *CityMobil* -Bigramms beträgt bei einer Lexikongröße von 1081 Wörtern lediglich 24 (s. Abschnitt 6.2), die des RM-Korpus bei 1000 Wörtern 60 und die des *Verbmobil* -Bigramms für über 5000 Wörter 118. Das zeigt an, dass im RM-Korpus das folgende Wort wesentlich schlechter vorhergesagt werden kann, als im *CityMobil* -Korpus.

Korpora für die Sprachmodelle

Wie bereits erwähnt, ist für statistische Sprachmodelle ein Trainingskorpus obligatorisch. Wie gut ein Modell später im Einsatz ist, hängt von der Menge und vor allem der Qualität des Materials ab. Je besser es zur späteren Anwendung passt, umso restriktiver kann es die Erkennung unterstützen. Allerdings kann man vermuten, dass für die *Graphem**-Erkennung ein beliebiger (aber ausreichend großer) vorhandener Korpus gewählt werden kann, wenn er nur die möglichen Graphemfolgen repräsentativ widerspiegelt. Im Kontrast dazu wurde ein spezieller Korpus für einen 10.000-Städtenamen-Erkenner konstruiert. Da die Städtenamen bekannt waren, bildeten sie die Grundlage für das Material. Die Anzahl der Namen ist für ein Training zu gering, daher wurde das Material vervielfältigt. Eine simple Multiplizierung bietet leider keine zusätzliche Information, deshalb wurden die

Städtenamen entsprechend ihrer 'Wichtigkeit' wiederholt. Pauschal wurde angenommen, je mehr Einwohner eine Stadt hat, umso wahrscheinlicher wird sie als Ziel der Erkennung gewünscht. Unglücklicherweise zeigte es sich, dass man nicht ohne Schwierigkeiten an aufbereitete Daten der Einwohnerzahlen aller deutschen Städte, ja nicht einmal an eine Liste der Städte selber kommt. Abhilfe schaffte das World Wide Web und eine Postleitzahlenliste. Aus der Liste wurden 10.000 Namen extrahiert und anschließend automatisch mittels der Suchmaschine *AltaVista* im WWW auf deutschen Seiten gesucht. Die zurückgegebenen 'Hits', also die Anzahl der gefundenen Seiten, die den gesuchten Begriff namentlich enthielten, waren das rohe Maß der Wichtigkeit der Stadt. Die Liste musste noch nachbearbeitet werden, um verschiedene Namen, die nicht nur Städte repräsentieren, wie etwa 'Eiche' (der Baum) oder 'Spielberg' (der Regisseur), zu dämpfen. Da 'Berlin' die Hauptstadt ist und eine entsprechende Größe hat, wurde ihre 'Hit'-Zahl als Obermaß eingesetzt und alle anderen Städte daraufhin beschnitten.

Somit wurden Sprachmodelle auf 2 Korpora erstellt, die jeweils dank des eingeschränkten Vokabulars von nur 33 Graphemen vom Bigramm bis zum Decagramm berechnet werden konnten.

Lexikonabgleich

Da die Ausgabe des *Graphem**-Erkenners lediglich eine Zeichenkette ist, die gewünschten Resultate aber Wörter aus einem Lexikon sind, muss ein zusätzlicher Verarbeitungsschritt durchgeführt werden. Die n Wörter, die der Buchstabenkette am ähnlichsten sind, werden als Ausgabe weitergegeben. Zum Messen der Ähnlichkeit einer Buchstabenfolge mit einem Wort des Lexikons wird ein Ansatz des dynamischen Programmierens, die sogenannte *edit distance* bzw. *Levenshtein-Abstand* (s. [40]), eingesetzt. Das Abstandsmaß ist dabei die minimale Anzahl von Einfügungen, Löschungen und Vertauschungen die gemacht werden müssen, um eine Textkette r der Länge n_r in eine Kette t der Länge n_t zu überführen. In der Regel sind alle drei Transformationen gleich gewichtet ($= 1$), werden aber im hier verfolgten Ansatz später automatisch angepasst werden. Die Berechnung wird iterativ durchgeführt (Algorithmus s. Anhang A.5), indem eine $(n_r + 1) \times (n_t + 1)$ Abstandsmatrix zuerst in der ersten Zeile mit den Zahlen von 0 bis n_r und in der ersten Spalte mit Zahlen von 0 bis n_t initialisiert wird. Der Rest der Matrix ergibt sich durch systematischen Vergleich jedes Zeichens der einen Kette gegen jedes Zeichen der anderen Kette unter Aufsummierung der entsprechenden Gewichte und der besten Transformationsbewertung des vorigen Schrittes. Die Distanz ist dann an der Position $(n_r + 1), (n_t + 1)$ abzulesen. Grafik 5.7 zeigt am Beispiel der Texte `manhain` und `mannheim` eine resultierende Matrix, in der der Abstand der beiden Texte 3 ist (s. rechte untere Ecke der Matrix).

Dieses Vorgehen birgt ein Problem in sich: dadurch, dass sehr viele Wörter mit der erkannten Kette abgeglichen werden, gibt es auch entsprechend viele

Test \ Referenz		m	a	n	n	h	e	i	m
	0	1	2	3	4	5	6	7	8
m	1	0	1	2	3	4	5	6	7
a	2	1	0	1	2	3	4	5	6
n	3	2	1	0	1	2	3	4	5
h	4	3	2	1	1	1	2	3	4
a	5	4	3	2	2	2	2	3	4
i	6	5	4	3	3	3	3	2	3
n	7	6	5	4	3	4	4	3	3

Abbildung 5.7: Levenshteinabstand der vom Graphemerkenner ausgegebenen Textkette `manhain` und dem Lexikoneintrag `mannheim`. Einfügungen entstehen beim Übergang in Y-Richtung, Löschungen beim Übergang in X-Richtung.

gleichbewertete Hypothesen. Im Beispiel waren dies für `manhain` neben der genannten Stadt *Mannheim* auch noch *Manching*, *Marnheim*, *Monheim*, *Mandern* und *Molzhain*[15], um nur einige zu nennen. Um eine bessere Unterscheidung treffen zu können, muss ein weniger diskretes Maß eingesetzt werden.

Ausgehend davon, dass sich manche Verwechslungen nicht so negativ auf die Erkennung auswirken wie andere – ein 'k' gegen ein 'g' zu vertauschen ist nicht so fatal wie ein 'a' gegen ein 'z' – wurden neue Gewichte für die Abstandsmessung gesucht. Da die Verwechslung jedes Zeichens gegen jedes Zeichen betrachtet wird, ergibt sich eine 33×33 Zeichen große Verwechslungsmatrix, wobei auf der Diagonalen die richtigen Paarungen stehen. Erweitert man diese Matrix noch durch ein zusätzliches leeres Symbol ϵ, kann man in ihr auch noch die Gewichte für Einfügungen, als Verwechslung von ϵ mit einem Graphem, und Löschungen, als Verwechslung von einem Graphem mit ϵ, repräsentieren. Somit erhält man eine 34×34 große Gewichtsmatrix G, in der nicht nur die Gewichte für alle Fehlerarten, sondern auch noch zusätzlich die für richtige Paarungen enthalten sind.

Da das ganze Verfahren zur Graphemerkennung eine möglichst einfache, nicht manuelle Strategie verfolgt, bietet es sich an, auch hier einen datengetriebenen Ansatz zu benutzen. Um diese Matrix zu berechnen, wurde ein Testkorpus im Erkenner evaluiert und die Ergebnisse mit einer über der Standard-Levenshtein-initialisierten Matrix mit allen Gewichten $G(i, j) = 1$ für $i \neq j$ mit dem Lexikon abgeglichen. Innerhalb der Levenshteinmatrix kann nicht nur der Abstand zweier Zeichenketten abgelesen werden, sondern durch Pfadrückverfolgung von $L(n_r, n_s)$ nach $L(1, 1)$ über die bis zu drei jeweiligen Vorgänger $L(x-1, y)$,$L(x, y-1)$ und

[15] s. Abschnitt 6.4.1.

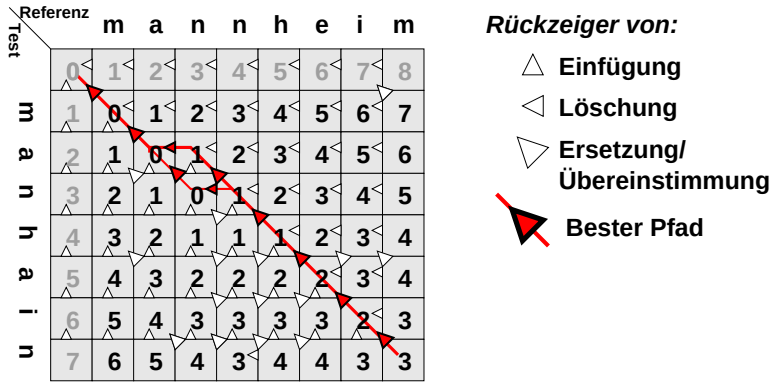


Abbildung 5.8: Pfadrückverfolgung in der Matrix

$L(x-1, y-1)$ für jeden Eintrag $L(x, y)$ auch sämtliche *Alignments*, die diesen Abstand haben (s. Grafik 5.8). Dabei steigt die Anzahl der potentiellen Pfade mit der Länge der zu vergleichenden Texte.

Im Beispiel sind, bis auf die zwei möglichen Paarungen des `'n'` aus `manhain` zu einem `'n'` des Doppel-`'n'` aus `mannheim`, die Zuordnungen eindeutig und ergeben lediglich zwei beste Pfade.

Nach Auswertung aller Erkennungsergebnisse können die tatsächlich aufgetretenen Graphempaare auf allen Pfaden gezählt und mit der Anzahl aller Pfade normiert werden. Die resultierenden Zahlen von 1 subtrahiert ergeben die neue Gewichtsmatrix. Um zusätzlich die Übereinstimmung von Graphemen zu bevorzugen, wurde in weiteren Experimenten die Diagonale auf 0 bzw. auf negative Werte gesetzt. Nachdem dieses Verfahren mehrfach iteriert wurde, stellten sich stabile Gewichtsmatrizen ein. Diese Matrizen wurden dann für die tatsächliche Erkennung eingesetzt. Im Beispiel führte dies dazu, dass sowohl *Manheim* als auch *Mannheim* unter den besten drei Ergebnissen waren.

5.3.3 Baumbasierter Städteerkenner

Der zweite Ansatz zur graphembasierten Erkennung entspricht in der Vorgehensweise dem bereits beschriebenen Standarderkenner. Allerdings sind aus oben genannten Gründen statt Triphonen Trigrapheme eingesetzt worden. Des Weiteren wurde statt einer kontinuierlichen Spracherkennung das System auf Einzelworterkennung reduziert, um den Umgang mit dem großen Lexikon zu vereinfachen. Zusätzlich wurde das System um eine $n-$best Ausgabe erweitert, um dem Benutzer potentielle Alternativen zum präferierten Ergebnis aufzeigen zu können. Das Lexikon des Erkenners wurde automatisch in einen Präfixbaum überführt, der zur Laufzeit im Arbeitsspeicher verfügbar sein muss. Durch den Übergang

auf Einzelworterkennung entfallen die sonst üblichen Baumkopien, wodurch der Speicheraufwand relativ gering bleibt.

5.4 Buchstabenerkenner

Der Buchstabenerkenner ist als Notfallsystem für Namen konzipiert: sollte der Benutzer einen existierenden Namen aus dem Telefonbuch, dem Städte- und Straßenverzeichnis des Navigationssystems oder der Senderliste nicht als ganzes Wort eingeben können, wird die Möglichkeit gegeben, den Namen zu buchstabieren. Da das zu äußernde Wort im System textuell vorliegt bzw. von den angeschlossenen Geräten geholt wird, kann während des Buchstabiervorgangs ständig ein Lexikonabgleich stattfinden. Gibt es bei diesem Abgleich lediglich noch eine einzige Lösung, wird die Eingabe vom System unterbrochen und dem Benutzer vorgeschlagen (Städteerkennung und Namenswahl) bzw. sofort zur Verarbeitung weitergereicht (Sendernamenanwahl). Diese Vorgehensweise verkürzt die vom Benutzer einzugebende Zeichenfolge. Allerdings können identische Einträge, wie beispielsweise mehrfache Vorkommen des gleichen Namens in einem Handytelefonbuch ohne zusätzliche Informationen nicht sinnvoll verarbeitet werden.

Potentiell ergibt sich beim Buchstabieren das Problem, dass Buchstaben akustisch sehr kurze Einheiten darstellen, die phonetisch teilweise hochähnlich sind. Um die damit verbundene hohe Verwechslungswahrscheinlichkeit zu umgehen, bieten sich längere Wörter als Platzhalter für die Buchstaben an. Diese Problematik existiert nicht nur seit der Entwicklung von Spracherkennungssystemen, sondern schon seitdem Mensch-Mensch Kommunikation über Telephone oder Funkverbindungen durchgeführt wird. Das Problem wurde damals durch Einführen von längeren, untereinander schwer verwechselbaren Buchstabenplatzhaltern in Form der Funkbuchstabieralphabete gelöst. Fast jede Sprache hat ein eigenes dieser Alphabete, für Internationale Kommunikation wird in der Regel das englische gewählt. Allen gemein ist, dass Buchstaben durch Wörter, die mit ihnen beginnen, ersetzt werden. Der Anfang des internationalen Buchstabieralphabetes lautet zum Beispiel *Alpha, Bravo, Charlie, Delta, etc.*, der des deutschen *Anton, Berta, Cäsar, Dora, usw.*. Diese langen Wörter können, auch von einem Spracherkenner, sehr sicher erkannt werden, haben für den ungeübten Benutzer aber einen entscheidenden Nachteil: sie müssen gelernt werden. Da die Motivation zum Einsatz der Spracherkennung als Interface unter anderem die intuitive Benutzbarkeit ist, kann ein vorher trainiertes Buchstabieralphabet also leider nicht vorausgesetzt werden.

Eine denkbare andere Lösung wäre es, dem Benutzer zu gestatten, selber Buchstabierwörter zu erfinden. Gäbe es die Möglichkeit, ein automatisches Transkript für lediglich das erste Zeichen eines unbekannten Wortes anzugeben, wäre dies

die Methode der Wahl. Leider sind die Erkenner dafür noch nicht leistungsfähig genug.

Um das Dilemma der schlechten Erkennungsleistung aufgrund der kurzen, hochverwechselbaren Buchstabenwörter zu umgehen, muss das System von der Ausgabe eines wahrscheinlichsten Ergebnisses auf einen Hypothesengraph erweitert werden. Aber anders als beim Städteerkenner des vorigen Abschnitts können durch zuvor durchgeführte Experimente die Verwechslungspaare a priori geschätzt und in einem dem Buchstabenerkenner nachgeschalteten Prozeß realisiert werden. Diese Vorgehensweise kann beim Städteerkenner zwar zusätzlich eingesetzt werden, aber aufgrund der größeren und damit nicht so leicht verwechselbaren Wörter nicht alleinstehend. Anstelle der Rückverfolgung von Traversierungsbäumen im Erkenner wird lediglich dessen Ausgabe durch alle Buchstaben mit ihren wahrscheinlichsten Verwechslungspartnern versehen (die auf den SLACC-Korpus ermittelte Verwechslungsmatrix befindet sich im Anhang A.6). Durch einen Lexikonabgleich, der die jeweiligen Alternativbuchstaben für jedes Zeichen ebenfalls in Betracht zieht, kann die tatsächlich intendierte Lösung gefunden werden. Der dabei auftretende Arbeitsaufwand entspricht bei durchschnittlich zwei zusätzlichen Alternativen pro Zeichen und einer mittleren Länge von 7 Buchstaben pro Name 3^7, also etwa 2000 Variationen pro erkannter Sequenz. Bei einem Abgleich gegen eine 13.000 Namen umfassende Liste entspricht dies im schlimmsten Fall mehr als 28 Millionen Vergleichen, vorausgesetzt, dass eine komplette Suche durchgeführt wird. Durch experimentell ermittelte Verwechslungspaarungen der einzelnen Zeichen können die zu erkennenden Namen automatisch in Sequenzen von Zeichengruppen übersetzt werden, in denen alle Buchstaben auftreten, mit denen ein Zeichen verwechselt werden kann – für den Buchstaben *'B'* sind dies beispielsweise *'BDW'*. Für jedes Zeichen existiert genau eine Gruppe. Die zu erkennenden Namen können dann effektiv in einer Präfixbaumhierarchie aufgelistet werden, wodurch sich der Suchaufwand enorm reduziert. Im schlechtesten Fall gibt es für den ersten erkannten Buchstaben *'B'* insgesamt $1347 + 532 + 979 = 2858$ mögliche Namensanfänge (alle Städte die mit *'B'*, *'D'* oder *'W'* beginnen), so dass 2858 Alternativen übrig bleiben. Unter ihnen befinden sich beispielsweise *Baden Baden* und *Darmstadt*. Mit dem nächsten Zeichen (im schlechtesten Fall die Gruppe *'AHK'* mit $666 + 1 + 0 = 667$ Vorkommen) reduziert sich die Anzahl der möglichen Lösungen widerum bis schließlich nur eine übrig bleibt, in der Regel bevor das letzte Zeichen geäußert wurde. Die Einträge *Baden Baden* und *Darmstadt* unterscheiden sich beispielsweise schon beim dritten Zeichen, da *'D'* und *'R'* nicht gemeinsam in einer Gruppe auftreten.

Sollte am Ende einer Buchstabiersequenz mehr als eine Alternative vorhanden sein, wie es beispielsweise bei den Orten *Sehlem, Nordrhein-Westfalen* und *Sehlen, Rheinlandpfalz* und *Niedersachsen* auftritt, und nur noch wenige Alternativen verbleiben, kann dem Benutzer über ein Display visuell oder beim Einsatz einer Syntheseeinheit akustisch eine Auswahlliste, die mit den zusätzlichen Infor-

mationen versehen wurde, präsentiert werden. Steht lediglich die Ausgabe über Textschablonen zur Verfügung, kann das System die vermutete Lösung buchstabieren. Aus Gründen der besseren Verständlichkeit wird dies über ein Buchstabieralphabet, das Benutzer zwar nicht unbedingt auswendig sprechen, aber dennoch verstehen können, realisiert. Sollte der Benutzer Korrekturen durch Unterbrechungen anbringen, kann das System durch das mittlerweile bekannte Präfix und dem Ausschluß des abgelehnten Zeichens an der aktuellen Vergleichsstelle die Anzahl der verbleibenden Alternativzeichen reduzieren und weiter nennen. Für den Fall, dass identische Alternativen, wie etwa die zwei Ortschaften *Sehlen*, übrigbleiben, müssen neben dem Namen zusätzliche Informationen, wie etwa das Bundesland oder andere geographische Spezifikationen, zur Verfügung gestellt und vom Klärungsdialog berücksichtigt werden:

```
Möchten Sie nach
1.) Frankfurt am Main
2.) Frankfurt an der Oder
```

Am Ende der Prozedur liegt ein vom Benutzer über eine Buchstabiersequenz eingegebener Name aus einer Liste von dem System bekannten Wörtern vor. Potentiell könnte dieser Erkenner auch zum Erweitern des Handytelefonbuchs sowie dem Versenden von SMS-Nachrichten und E-Mails benutzt werden. Praktisch ist dafür aber die Erkennungsleistung nicht ausreichend, so dass der Benutzer zu häufig in einen Klärungsdialog geführt werden müsste und dadurch abgelenkt würde.

Da die Verwechslungen zwischen den Buchstaben als statistischer Prozess analog Kapitel 2.3 aufgefasst werden kann und die entsprechenden Routinen schon für die Erkenner erstellt worden sind, wurde der Lexikonabgleich der unsicher erkannten Buchstabensequenzen über diskrete HMMs realisiert. Die Emissionswahrscheinlichkeiten der diskreten Buchstaben modellieren dabei die Verwechslungsmatrix.

Der Buchstabenerkenner entspricht in seiner Realisierung mit einer Einschränkung exakt dem Basissystem: das Vokabular wurde auf die Buchstabiersymbole reduziert. In seiner Integration in das gesamte Interface werden dadurch keine dynamischen Garbagemodelle zur Verfügung gestellt. Aufgrund der sehr kurzen Dauer der einzelnen Buchstaben wären diese nur potentielle Fehlerquellen. Um dennoch die benötigte Robustheit zu leisten, wird die Ausgabe dieses Teilerkenners nicht direkt verwendet, sondern mit der Verwechslungmatrix erweitert und dem Lexikonabgleich zugeführt. Ist lediglich ein einziges Wort aus dem Lexikon als Alternative übrig, wird es zur nachfolgenden Verarbeitung weitergereicht. Ist ein Wort vollständig buchstabiert, aber aufgrund der Verwechslungsgruppen nicht eindeutig, wird zuvor noch ein kurzer Klärungsdialog über den Basiserkenner geführt, bevor das dann gewählte Wort weitergereicht wird.

Die tatsächliche Realisierung des Lexikonabgleichs geschieht über diskrete HMMs. Für jedes Zeichen wird ein Zustand etabliert, der mit der zuvor beobachteten Wahrscheinlichkeit nicht nur das entsprechende Zeichensymbol, sondern auch seine Verwechslungspartner emittieren kann. Alle HMMs sind parallel geschaltet und werden über einen Pseudostart- und Pseudoendeknoten miteinander verbunden. Das Lexikon für den Abgleich liegt als Präfixbaum vor, bei dem lineare Pfade zu den Endeknoten beschnitten werden, so dass eine eindeutige Lösung schon ausgegeben werden kann, wenn zwar das gesamte Wort noch nicht buchstabiert wurde, aber es keinerlei Alternativen mehr gibt. Solange noch Alternativen existieren, deren Zahl die der sinnvoll ausgebbaren Namen (etwa fünf) überschreitet, werden weitere Buchstabiersymbole verlangt. Der schlechteste Fall, ein Lexikon mit identischen Einträgen, kann nicht auftreten, da die ausgewählten Anwendungen, mit Navigationszielen, Senderliste und Telefonbuch nicht redundant vorliegen. Es gibt immer zusätzliche Informationen, um Einträge eindeutig zu bestimmen.

6 Ergebnisse

In diesem Kapitel werden zunächst die Entwicklungsumgebung für die im Rahmen der Arbeit erstellten Erkenner sowie die zum Training und Testen verwendeten Sprachkorpora präsentiert. Anschließend werden die Ergebnisse der Erkenner und Algorithmen vorgestellt und diskutiert.

6.1 Die Entwicklungsumgebung ESMERALDA

Das Akronym *ESMERALDA* steht für **E***nvironment for* **S***tatistical* **M***odel* **E***stimation and* **R***ecognition on* **A***rbitrary* **L***inear* **D***ata* **A***rrays*[1]. *ESMERALDA* [15] ist eine Entwicklungsumgebung mit einer Vielzahl von Progammen zur Erstellung, zum Trainieren, zur Manipulation und Auswertung von HMMs. Es wird seit 1990 an der Technischen Fakultät der Universität Bielefeld unter der Leitung von Dr. Gernot A. Fink entwickelt und kann, dank flexibler Struktur des Systemaufbaus, neben der Sprachverarbeitung auch in anderen Bereichen der Mustererkennung eingesetzt werden, wie etwa der Erkennung von handgeschriebenen Texten und der Analyse textuell vorliegender genetischer Sequenzen. Die weitere Beschreibung des Programmpaketes beschränkt sich ausschließlich auf die für die Spracherkennung benötigten Teile.

Der Ablauf der Spracherkennung vom Signal bis zur Ausgabe der Transkription mit dem *ESMERALDA* -System ist schematisch in Abbildung 6.1 dargestellt. Die Verarbeitungsschritte stehen in den weißen Kästchen, die von ihnen benutzten Daten in den hellgrauen Zylindern, bzw. die verwendeten heuristischen Grundlagen in der Wolke darüber. Am oberen Rand sind in dunkelgrau die Verfahren zur Erstellung der Daten aufgeführt. Zur Extraktion der Merkmale aus dem Sprachsignal werden auf psychoakustischem Wissen basierende, heuristische Methoden benutzt, die Merkmale sind die in Abschnitt 2.2 erläuterten *Mel Frequency cepstral coefficients*. Die verschiedenen Merkmale aus einem Signalabschnitt werden zu Vektoren zusammengefasst, die mittels eines auf Mischverteilungen basierenden Vektorquantisierers auf Wahrscheinlichkeitsverteilungen abgebildet werden. Aus diesen wird in der anschließenden Pfadsuche über HMMs, auf Wunsch mit Unterstützung durch Sprachmodelle und/oder Grammatiken die wahrscheinlich-

[1] Umgebung zur Schätzung statistischer Modelle und Erkennung von beliebigen linearen Datenfeldern

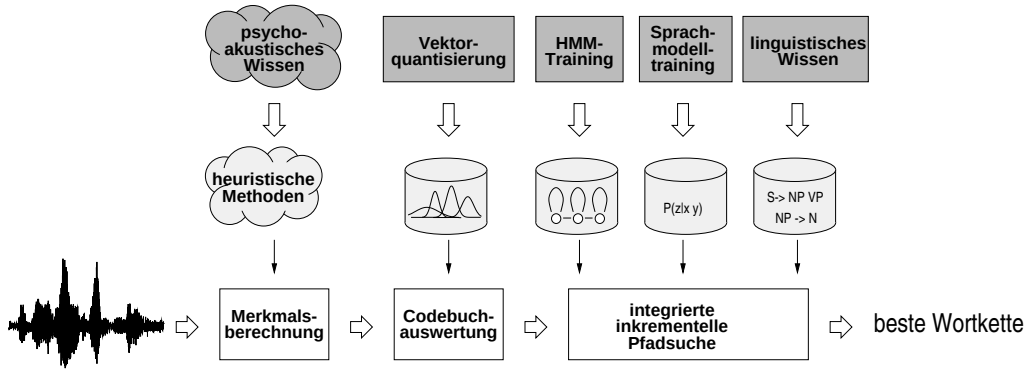


Abbildung 6.1: Schematischer Aufbau eines ESMERALDA Spracherkenners (nach [15]). Weitere Erläuterungen im Text.

ste Wortkette berechnet. Zu jedem der in den Kästen erwähnten Verarbeitungsschritten bietet *ESMERALDA* Programme an, die einerseits durch eine Vielzahl einstellbarer Parameter flexibel einsetzbar sind, andererseits aber auch im Bedarfsfall durch Codeerweiterung an spezielle Aufgaben angepasst werden können. Weiterführende Informationen zur Entwicklungsumgebung finden sich in [15].

6.2 Sprachdaten

Zur Generierung und zum Testen der erstellten Spracherkenner und Verfahren wurden mehrere Sprachkorpora benutzt. An Sprachkorpora werden, je nach geplanter Anwendung, unterschiedliche Anforderungen gestellt. Im allgemeinen gilt aber, dass jedes Korpus in Hinsicht auf Wortwahl, Aufnahmeausrüstung, Hintergrundgeräusche, Sprechart etc. möglichst der späteren Anwendung entsprechen sollte. Das Korpus muss unter Berücksichtigung aller zu trainierenden Modelle ausreichend groß und phonetisch[2] ausgewogen sein. Um verschiedene Systeme miteinander vergleichen zu können, muss weiterhin genug Material, das nicht im Training eingesetzt wurde, vorliegen. Die Menge hängt von der gewünschten Signifikanz der Qualitätsmessung ab (s. Abschnitt 2.4). Da die Erstellung und Bearbeitung eines Korpus sehr arbeitsintensiv ist, sind die meisten Korpora entweder kaum bezahlbar oder gar nicht erst der Öffentlichkeit zugänglich. Aus diesem Grund wurden für einige der später aufgeführten Experimente anwendungsfremde Korpora eingesetzt, die zum jeweiligen Zeitpunkt zur Verfügung standen und sich in der untersuchten Kategorie nicht vom intendierten Korpus unterschieden. Für den innerhalb der Arbeit entwickelten Erkenner wurde schließlich ein eige-

[2] oder graphemisch, je nach Wahl der Wortuntereinheiten.

nes Korpus Namens *SLACC* erstellt. Im weiteren folgt eine Beschreibung der verwendeten Korpora.

SLACC

Das *SLACC* Korpus[3][58] wurde speziell für die im Rahmen dieser Arbeit entwickelten Erkennungssysteme erstellt. Es beinhaltet Phrasen zur Steuerung des Radios, des Navigationssystems, der Klimaanlage und des Telefons. Zusätzlich sind verschiedene Städtenamen, Buchstabiersequenzen und kleine Anweisungen für die Ansteuerung eines Taschenrechners und eines Kalenders enthalten. Damit ausreichend Zahlen im Korpus auftreten, wurde jede Äußerung mit einer dreistelligen Zahl nummeriert, die mitgesprochen wurde. Die Aufnahmen wurden von Probanden während der Fahrt gelesen (aus Versicherungsgründen natürlich nur auf dem Beifahrersitz) und simultan von zwei Mikrofonen aufgezeichnet, einem Nahbesprechungsmikrofon und einem vom Verband deutscher Automobilhersteller (VDA) zugelassenen Mikrofon, das unter anderen für Freisprecheinrichtungen im Fahrzeug genutzt wird. Jede Versuchsperson bekam ein eigenes Set von 300 Äußerungen, die zweimal gesprochen wurden, wobei die gewählte Fahrstrecke bei beiden Lesungen unterschiedliche Fahrgeschwindigkeiten und damit auch unterschiedliche Innenraumgeräusche und Sprechverhalten (Lombardeffekt) zur Folge hatten. Insgesamt wurde von 16 Sprechern und 6 Sprecherinnen pro Mikrofon 11.5 Stunden Material aufgenommen, das in der Rohfassung mit 48kHz[4] vorlag und anschließend auf 16kHz neu abgetastet wurde. Für die Aufnahmen wurden 5 Fahrzeuge, vom Kleinwagen bis zur Oberklasse benutzt. Damit standen unterschiedliche Innenraumakustiken sowie Motorgeräusche zur Verfügung. Weitere Variabilität wurde durch unterschiedliche Witterungsbedingungen wie Regen und Schnee in das Korpus gebracht. Die Lexikongröße beträgt 718 Wörter, wobei viele im Korpus vorkommende Zahlen durch Zerlegung in ihre Einzelbestandteile (*227* etwa in *zwei hundert sieben und zwanzig*) dargestellt werden können und sich damit das Lexikon mit 370 Wörtern auf fast die Hälfte reduzieren lässt. Eine detailierte Beschreibung der einzelnen Fahrzeugtypen, Aufnahmebedingungen und Probanden findet sich in [58]. Für weitere Arbeiten ist mittlerweile ein noch nicht öffentlicher Nachfolgekorpus basierend auf *SLACC* erstellt worden. Das Korpus besitzt ein größeres Lexikon und wurde von mehr Probanden in einer kleineren Anzahl Fahrzeugen als der Vorgänger gesprochen. Von diesem Korpus wurden einige Äußerungen extrahiert, die lediglich Wörter aus dem *SLACC* Lexikon enthalten und für ein Experiment zur Merkmalsreduktion eingesetzt.

[3] *SLACC* - **S***poken* **La***nguage* **C***ar* **C***ontrol* Fahrzeugsteuerung durch Sprache

[4] Das Nahbessprechungsmikrofon hat einen Frequenzgang bis 20kHz, die Abtastung musste also wenigsten 40kHz betragen. Da das spätere System mit 16kHz arbeiten sollte, wurde das nächsthöhere Vielfache von 16kHz, also 48kHz, benutzt.

Verbmobil

Das *Verbmobil* Korpus [33] stammt aus dem gleichnamigen BMBF[5]-Projekt, das in 2 Phasen von 1992 bis 1997 und 1997 bis 2000 lief. Ziel des Projektes war die Entwicklung eines portablen Übersetzungscomputers, der für Dialogpartner mit den Eingabesprachen Deutsch und Japanisch eine Kommunikation mit Englisch als Vermittlungssprache ermöglichen soll, ausgehend davon, dass alle Teilnehmer passive Englischsprecher sind. Die gewählte Domäne für die erste Phase war *Terminabsprache* und für die zweite Phase *Reiseplanungen*. Innerhalb dieser Arbeit wurde lediglich das Korpus aus der Domäne Terminabsprache benutzt. Es umfasst spontansprachliche Dialoge von jeweils zwei Probanden, denen lediglich die Aufgabe gestellt wurde, einen passenden Termin anhand von ausgegebenen Terminplanern zu finden. Da keinerlei Restriktion für die Art der Termine noch den zu führenden Dialog gestellt wurde, ergab sich ein mit über 6000 Wörtern (96er Evaluationsset) recht umfangreiches Lexikon, das teilweise eher ungewöhnliche Begriffe wie etwa *Diaabendweintrinkrevisionstreffen* oder *Safaribüchse* enthält. Die Aufnahmen fanden mit hochwertigen Tischmikrofonen in ruhiger Büroumgebung statt und umfassen 33 Stunden deutschsprachiges Material von 654 Sprechern und Sprecherinnen aus ganz Deutschland.

Erba

Als weiteres Korpus wurde Material aus dem *ERBA* -Projekt[6] [53] benutzt. Das Korpus besteht ausschließlich aus gelesenem Material, das Erstanfragen von Zugverbindungen beinhaltet. Da die zu lesenden Turns durch eine vorgegebene Grammatik generiert wurden, ist ihr Aufbau systematisch und das abgeschlossene Vokabular mit knapp 950 Wörtern handlich. Das Lexikon besteht zur Hälfte aus Städtenamen und etwa einem Drittel Uhrzeiten und Datumsangaben, inklusive Feiertagen. Das Material wurde von 110 Sprechern gelesen und umfasst 15 Stunden Büroaufnahmen. Durch den hohen Anteil an Städtenamen diente dieses Korpus zur Evaluation der Städteerkenner.

CityMobil

Der *CityMobil* Korpus wurde in Bielefeld im Rahmen des NRW-Verbundprojektes *Virtuelle Wissensfabrik, Mensch-Maschine-Interaktion* erstellt. Das Ziel des Verbundes war die Untersuchung multimodaler Eingaben über Sprache, Geste und Blickrichtung in einem virtuellen Konstruktionsszenario. Das Vokabular bestand dementsprechend aus verschiedenen Bauelementen (Räder, Leisten, Schrauben

[5] **B**undes**m**inisteriums für **B**ildung und **F**orschung
[6] **Er**langer **Ba**hnanfragen

...), unterschiedlichen Attributen (Farben, Formen, Relationen, Längen), Anweisungen zur Generierung und Manipulation von Objekten, Steuerkommandos (speichern, löschen, abbrechen), sowie Ziffern für Stück- und Maßzahlen, insgesamt 1081 Wörter. Ähnlich dem *ERBA* -Korpus wurde das Material per Grammatik erzeugt und von 93 Probanden beiderlei Geschlechts in Büroumgebung gelesen. Die Korpusgröße beträgt etwa 13 Stunden. Dieses Korpus wurde primär für Untersuchungen zur automatischen Transkription unbekannter Wörter verwendet.

Weitere Korpora

Für die Erstuntersuchungen in der Fahrzeugdomäne wurde ein kleiner, nicht öffentlicher Korpus von 7 Sprechern und 6 Sprecherinnen aufgenommen, die diverse kurze Befehlssequenzen und Ziffernfolgen unter drei unterschiedlichen Bedingungen sprachen: im Büro, in einem Fahrzeug im Stadtverkehr und im Fahrzeug während einer Autobahnfahrt bei hoher Geschwindigkeit. Das Vokabular setzte sich aus Zahlen und 12 Wörtern zusammen, die aus dem Lexikon des bereits beschriebenen und damals erst geplanten *SLACC* -Korpus unter Berücksichtigung der phonetischen Ausgewogenheit gewählt wurden. Mit einer Gesamtdauer von knapp über 2 Stunden ist das im Weiteren *Initialisierung* genannte Korpus das kleinste der hier vorgestellten und erlaubte damit eine Vielzahl an Experimenten mit unterschiedlichen Konfigurationen innerhalb kurzer Zeit.

Neben den bereits erwähnten Korpora, die sowohl zum Training als auch zum Testen (Trainingsset und Testset waren jeweils disjunkt) genutzt wurden, existiert noch ein reines Testset mit über 18 Stunden Sprachaufnahmen von zwei Sprechern und einer Sprecherin, die jeweils 10.000 Städte in bis zu neun Wiederholungen aufgenommen haben. Das Set ist der Öffentlichkeit leider nicht zugänglich, da es für einen aus dieser Arbeit entwickelten Spracherkenner von einer Firma der Fahrzeugindustrie zur Verfügung gestellt wurde. Alle Aufnahmen entstanden durch automatische Mitschnitte des Erkenners während mehrtägiger Testläufe. Die Probanden beobachteten beim Vorlesen der Städte die Ausgabe des Erkenners und wiederholten sie bei Fehlern. Da die Aufnahmen erst nach der Sprachdetektion geschrieben wurden, kam es in einigen Fällen durch zu leise Sprache zu Verstümmelungen der Äußerungen, was später in den Experimenten zur Baumerweiterung noch spezielle Beachtung findet. Im weiteren wird dieses Testset als *10k-Städte* referenziert.

Außerdem wurden, ebenfalls für die Städte- bzw. Straßenerkennung, noch verschiedene, reine Textkorpora erstellt, die zur Erweiterung des graphembasierten Erkenners dienten. Die Städteliste umfasst etwas über 50.000 Einträge und wird daher fortan als *50k-Liste* bezeichnet. Von 950 Städten aus dieser Liste wurden komplette Straßenlisten erstellt, die zwischen 20 und 13.000 Namen lang sind. Insgesamt liegen für Testzwecke somit 339.000 Straßen vor, davon 168.500 ver-

Korpus	Lexikon-größe	Anzahl Sprecher	Korpus-größe [h]	Domäne	Umgebung/ Sprachstil
Initialisierung	52	13	2	Fahrzeugsteuerung	Büro und Fahrzeug gelesen
SLACC	718	22	2 x 11.4	Fahrzeugsteuerung	Fahrzeug, gelesen
Verbmobil	6266	654	33	Terminabsprache	Büro, spontan
ERBA	942	110	15	Bahnauskunft	Büro, gelesen
CityMobil	1081	93	13	3D Konstruktion	Büro, gelesen
10k-Städte	9875	3	18.2	Städtenamen	Büro, gelesen

Tabelle 6.1: Übersicht der verwendeten Korpora.

schiedene Straßennamen. Die tatsächliche Anzahl der in Deutschland vorkommenden Straßen und Straßennamen ist lediglich dreimal so hoch, für die hier durchgeführten Experimente ist das kleinere Set daher ausreichend.

6.3 Auswahl der Systemparameter

Für die Erstellung des Basissystems wurden verschiedene Entwürfe der HMMs betrachtet. Da alle weiteren Untersuchungen und Fortentwicklungen auf dem Basissystem aufsetzen, wurde die Entscheidung früh mit Experimenten auf dem der späteren Einsatzumgebung nahe stehenden Initialisierungskorpus getroffen.

Während das Trainieren und Auswerten der HMMs algorithmisch durchgeführt werden kann, ist das Grunddesign der Modelle ein heuristischer Prozess. Welche Arten von Verknüpfungen sollen zwischen den Zuständen gewählt werden, wie werden die Emissionswahrscheinlichkeiten realisiert und wie können die Parameter möglichst effektiv verwendet werden? Als beste Konfiguration etablierten sich folgende Parameter: Die HMMs wurden als Links-Rechts-Modelle mit einem Selbstübergang und einem Nachfolger realisiert. Die Emissionswahrscheinlichkeiten sind als Gauss-Mischverteilungen modelliert worden. Zur Erstellung der Codebücher wurden verschiedene Experimente mit der Anzahl der Klassen und der Art der Kovarianzmatrix, voll gegenüber diagonal, gemacht. Dabei zeigte sich, dass der Erkenner für das kleine Vokabular mit dem diagonalen Codebuch aufgrund des geringeren Parameterbedarfs 512 Klassen etablieren konnte. Mit der vollen Kovarianzmatrix konnten nur 256 Klassen berechnet werden, die auf dem Initialen-Set ein schlechteres Ergebnis lieferten.

6.4 Graphemerkennung

Gängige Spracherkennungssysteme basieren auf phonetisch motivierten Wortuntereinheiten. Da die zu ihrer Erstellung benötigte phonetische Transkription eines Lexikons sehr arbeitsaufwändig wäre, wurden für die großen Lexika der Städte- und Straßennamen verschiedene graphembasierte Erkenner, die auf Basis von Buchstaben als Wortuntereinheiten arbeiten, untersucht. Das Ziel der Erkennung ist die Ausgabe einer Liste von Ergebnissen, aus denen der Benutzer in einem Dialog dann das korrekte Resultat auswählt bzw. bei einem Fehlschlag entweder einen neuen Versuch starten oder den intendierten Namen buchstabieren kann (s. Abschnitt 5.4). Die Experimente für die Graphemerkennung wurden in zwei unterschiedlichen Ansätzen durchgeführt. Der erste Satz von Experimenten hatte als Ziel, einen durch ein Sprachmodell gesteuerten Graphemkettenerkenner zu realisieren, dessen Ausgabe mit einem bestehenden Lexikon abgeglichen wird und aufgrund eines Ähnlichkeitsmaßes die n-besten Ergebnisse produziert (s. Abschnitt 5.3.2). Der zweite Ansatz ging von einer präfixbaum-gesteuerten Suche aus und generierte 'direkt' die gewünschte n-best Liste (s. Abschnitt 5.3.3). Als Grapheme dienten die 26 Buchstaben des Alphabets, die drei Umlaute `'ä'` , `'ö'` und `'ü'` , die Sz-Ligatur `'ß'` und die zu jeweils einem Symbol zusammengefassten Zeichensequenzen `'sch'`, `'ch'` und `'ck'`. Zusätzlich konnten beide Ansätze noch durch eine der Erkennung nachgeschaltete Erweiterung der Ausgabelisten verbessert werden. Als Testset wurden 978 Städtenamen von 10 Sprechern aus dem *ERBA* -Korpus gewählt.

6.4.1 *Graphem**-Erkennung

Wie bereits erwähnt, zerfällt die *Graphem**-Erkennung in zwei Teilaufgaben: die Erkennung einer Graphemkette unter Zuhilfenahme eines Sprachmodells und den Abgleich der Kette gegen Wörter aus einem Lexikon. Der *Graphem**-Erkenner basiert lediglich auf Monographemen, die auf Material aus dem *Verbmobil* -Korpus trainiert wurden. Bevor der Lexikonabgleich erläutert wird, werden zunächst die verwendeten Sprachmodelle näher betrachtet.

Sprachmodelle

In Abschnitt 5.3.2 wurde die Konstruktion eines Sprachmodells aus einem gegebenen Textkorpus und dessen Bewertung aufgrund der Perplexität beschrieben. Je kleiner die Perplexität ist, um so restriktiver arbeitet das Sprachmodell und um so besser kann die Spracherkennung unterstützt werden. Um das geeignete Korpus zu finden, wurden verschiedene Sprachmodelle berechnet und die im Einsatz

	Verbmobil LM		WWW LM	
n-gram	GA	(PPX)	GA	(PPX)
2	38.77	(21.59)	43.93	(13.52)
3	39.28	(31.86)	52.07	(9.02)
4	40.73	(43.08)	60.02	(6.25)
5	41.40	(47.97)	66.96	(4.82)
6	41.13	(48.14)	70.90	(4.22)
7	41.32	(47.77)	72.07	(3.99)
8	41.25	(47.51)	**72.89**	(3.93)
9	41.24	(47.44)	72.55	(3.93)
10	41.24	(47.42)	72.67	(3.93)
none	24.24			

Tabelle 6.2: Graphemakkuratheit (GA) und Sprachmodellperplexität (PPX) bei n-gram Sprachmodellen von unterschiedlichen Korpora auf dem *ERBA* Testset mit 978 Städten.

mit dem Erkenner erreichten Graphemkettenakkuratheiten[7] gemessen. Da mit lediglich 33 Graphemen gearbeitet wurde, konnten mit dem vorhandenen Textmaterial Sprachmodelle vom Bigramm, das lediglich das direkte Vorgängergraphem in Betracht zieht, bis zum Dekagramm, das eine Geschichte von 9 Graphemen berücksichtigt, berechnet werden.

Das erste betrachtete Korpus für das Sprachmodell basiert auf demselben Material, mit dem auch die akustischen Modelle trainiert wurden, *Verbmobil* . Obwohl das Korpus für die Akustik mit seiner großen Zahl an Äußerungen (mehr als 300.000 Wörter) eine Menge unterschiedlicher Graphemkombinationen zur Verfügung stellt, wurde noch ein besser auf die Anwendung zugeschnittenes Korpus erstellt, das hier *WWW*-Korpus genannt wird. Es beinhaltete lediglich die 10.000 Städte des Testsets, mit einer Frequenz entsprechend ihrer Wichtigkeit. Die Messung der Wichtigkeit wurde durch eine Anfrage der einzelnen Städtenamen an *World Wide Web*-Suchmaschinen und Zählung der jeweils gefundenen deutschsprachigen Seiten realisiert. Die so erhaltenen Maßzahlen wurden anschließend beschnitten auf die Zählung der angenommenen wichtigsten Stadt, der Hauptstadt Berlin. Diese etwas umständliche Vorgehensweise wurde gewählt, da es nicht möglich war, auf andere Art an die gewünschten Daten[8] zu kommen.

Die auf diesen Korpora erzielten Ergebnisse sind in Tabelle 6.2 zusammengefasst. Die Graphemerkennung ohne jegliches Sprachmodell ist mit einer Gra-

[7] Die Graphemakkuratheit über einem Wort ist analog der Wortakkuratheit über einem Satz definiert.

[8] Wenigstens nicht ohne eine vierstellige Summe zu bezahlen.

phemakkuratheit (GA) von lediglich 24.24% deutlich schlechter als jedes andere System mit Sprachmodell. Aufgrund der hohen Verwechselbarkeit und der kleinen Einheiten ist eine nicht restringierte Erkennung sehr unzuverlässig.

Vergleicht man die unterschiedlichen Korpora miteinander, ist festzustellen, dass bei steigender Länge der Sprachmodelle die Perplexität des zum Training der HMMs verwendeteten Korpus von 21.59 zunächst bis zum Hexagramm zunimmt und danach nur leicht abfällt. Im Gegensatz dazu sinkt die Perplexität des speziell für die Anwendung erstellten Korpus (WWW) von 13.52 bis 3.93, ab dem Octagramm ist sie konstant. Das Korpus-Sprachmodell passt aufgrund der hohen Perplexitäten offensichtlich nicht zur Graphemfolge des Zielvokabulars, während das WWW-Sprachmodell sehr gute Modellierungseigenschaften hat, was sich auch in der deutlich höheren GA des WWW Korpus zeigt. Der leichte Anstieg der Erkennungsleistung für das Korpus Sprachmodell trotz steigender Perplexität lässt sich auf die hohe Anzahl nicht gesehener Graphemkombinationen und den damit verbundenen Rückfall des Sprachmodells auf die jeweiligen Modelle niedrigerer Ordnung zurückführen. Die leichten Unterschiede unterliegen der statistischen Variabilität und befinden sich unterhalb der Signifikanzschwelle von etwa 3.6. Zusammenfassend zeigt sich, dass das lediglich auf dem zu erkennenden Vokabular basierende Sprachmodell ab einer Tiefe von 7 Graphemen am effektivsten arbeitet. Da die zu erkennenden Wörter a priori bekannt sind, ist es legitim, sie zur Erstellung des Sprachmodells heranzuziehen.

Lexikonabgleich

Die mit Sprachmodellunterstützung gewonnene Graphemkette entspricht noch nicht der endgültigen Ausgabe des Systems, und die Graphemakkuratheit darf nicht mit der Wortakkuratheit verwechselt werden. Um die Ergebnisliste zu erhalten, muss die Graphemkette mit den Namen aus dem Lexikon verglichen und eine Liste der n ähnlichsten Einträge ausgegeben werden. Wie bereits in Abschnitt 5.3.2 beschrieben, wurde die Messung der Ähnlichkeit über den Levenshtein-Abstand realisiert. Bei Benutzung der selben Gewichte für alle Arten von Einfügungen, Löschungen und Ersetzungen ist der Abstand immer eine ganze Zahl, die sich zwischen 0 (Testwort und Vergleichswort sind identisch) und dem Maximum der beiden Wortlängen (Beide Wörter haben kein gemeinsames Zeichen[9]) bewegt. Damit ergibt sich das Problem, dass vor allem bei Abgleich gegen große Lexika sehr viele gleich gut bewertete Resultate generiert werden. Die Tabelle 6.3 zeigt in der mittleren Spalte einen Auszug der Ergebnisse mit einem Abstand von 2-3 zu der Graphemkette *manhain*. Die Beschneidung der Ergebnisliste auf eine Länge von n ist somit problematisch. Da aber nicht alle Arten

[9] Dies gilt nur, wenn alle Gewichte auf 1 gesetzt sind. Werden für die Ersetzung, Löschung und Verwechslung unterschiedliche Werte benutzt, stimmt die Aussage nur noch in der Größenordnung.

Stadt-name	Abstand zu *manhain*	
	Levenshtein	Gewichtsmatrix
Manheim	2	1.61
Manching	3	1.64
Mannheim	3	1.66
Manhagen	2	1.79
Marnheim	3	1.96
Maxsain	2	2.03
Monheim	3	2.23
Mandern	3	2.46
Molzhain	3	3.02
Mahlis	3	3.38
Marzahn	3	3.55
...	...	...

Tabelle 6.3: Abstände zwischen der aus der Äußerung *Mannheim* erkannten Graphemfolge *manhain* und verschiedenen Lexikoneinträgen für Standardlevenshtein und Levenshtein mit Gewichtsmatrix.

von Verwechslungen, Löschungen und Einfügungen gleich kritisch zu bewerten sind – eine Verwechslung 't' und 'd' ist, vor allem an Silbengrenzen, praktisch vernachlässigbar, ein Auslassung von Vokalen hingegen nicht – wurde mit dem unter Abschnitt 5.3.2 beschriebenen Verfahren eine Neuberechnung der Gewichte durchgeführt. Die dritte Spalte der Tabelle 6.3 gibt die so erreichten Abstandsmaße wieder. Die jetzt feinere Bewertung erlaubt eine klare Sortierung der Ausgabe sowie eindeutige Beschneidung auf eine $n-$best-Liste.

6.4.2 Baumbasierte Erkennung

Im Gegensatz zu der oben untersuchten, relativ aufwändigen *Graphem**-Erkennung wurde im zweiten Ansatz ein Standarderkenner auf Trigraphembasis realisiert, der zur Steuerung der Suche das Lexikon in Form eines Präfixbaum benutzt. Obwohl diese Vorgehensweise simpler erscheint, hat sie im Gegensatz zum anderen Verfahren entscheidende Nachteile: der gesamte Präfixbaum in Form von jeweils um ein Trigraphem wachsenden Präfixen muss im Speicher gehalten werden, bei einer vollständigen deutschen Städteliste von etwa 65.000 Einträgen entspricht dies mehr als 20 MB – zuviel, um in dieser Form auf einen DSP gebracht zu werden. Des Weiteren kann das Verfahren nicht einfach aufgeteilt und somit notfalls, wie es beim *Graphem**-Erkenner möglich ist, auf mehrere Einzelprozesse, die auf unterschiedlichen Prozessoren laufen können, aufgeteilt werden. Schließlich lässt sich das Lexikon für den Baumerkenner nicht so einfach manipu-

n best	Erkennungsrate Grapheme		Erkennungsrate Baum
	Levenshtein	Levenshtein, negative diag.	*Verbmobil*
1	49.69	53.16	61.45
5	53.06	61.55	78.62
10	53.98	64.95	79.65

Tabelle 6.4: Erkennungsraten in Prozent für n-best Ausgaben des *Graphem**- und des Baumerkenners für *Verbmobil* -Material.

lieren: Während die andere Variante einfach Wörter aus dem Lexikon entfernen und einfügen kann (solange diese grob den Restriktionen des Sprachmodells folgen), benötigt der Präfixbaum aufwändige Anpassungen. Diese Einschränkung gilt natürlich nur für den Fall, dass das Lexikon des Navigationssystems variabel ist.

6.4.3 Vergleich beider Systeme

Für den Vergleich beider Erkenner ist die in Hinsicht auf Graphemakkuratheit beste Version des *Graphem**-Systems gewählt worden, die Variante mit dem *WWW*-Korpus-basierten 8-Gramm-Sprachmodell. Wie bereits zuvor aufgeführt, erzeugte der Lexikonabgleich mit dem gleichgewichteten Levenshteinabstand zu viele identisch bewertete Resultate, daher wurden lediglich die zwei modifizierten Abstandsmaße benutzt.

Obwohl die Graphemakkuratheit fast 73% betrug, erreichte das System lediglich eine Wortakkuratheit (WA) von 49.69% auf dem besten Resultat (s. Tabelle 6.4). Aufgrund seines Umfangs gab es im Lexikon immer noch einige Namen, die besser als die korrekten passten, auch wenn fast dreiviertel der Grapheme richtig erkannt wurden. Daher beträgt bei der Ausgabe der besten 5 bzw. 10 Ergebnisse die WA lediglich 53.06% bzw. 53.98%. Diese Resultate konnten signifikant verbessert werden, indem die Matrixvariante mit der negativen Diagonalen benutzt wurde, absolut stieg die WA für das beste Ergebnis um 3.5%, für die besten 5 um 8.5% und für die 10 besten sogar um 11%.

Vergleicht man diese Zahlen mit denen des baumbasierten Erkenners, sind sie allerdings deutlich niedriger. Ein wesentlicher Grund dafür ist der Einsatz von Trigraphemen, die robuster als die Monographeme im *Graphem**-Ansatz sind. Ausserdem unterstützt die Baumverarbeitung den Erkennungsprozesses restriktiver als die Sprachmodelle und damit effektiver. Der baumbasierte Erkenner erreichte 61.45% beim besten Ergebnis und fast 80% bei den 10 besten.

Um diese Ergebnisse zu verifizieren, wurde das Experiment mit einem anderen Trainingskorpus wiederholt. Von den in der Evaluation nicht verwendeten Sprechern des *ERBA* -Korpus war ausreichend Material vorhanden, um ein Neu-

n best	Erkennungsrate Grapheme mapped	mapped, negative diag	Erkennungsrate Baum *ERBA*
1	48.87	49.07	82.71
5	57.97	59.10	89.57
10	60.73	62.57	89.67

Tabelle 6.5: Erkennungsraten für n-best Ausgaben des Graphem- und des Baumerkenners für *ERBA* -Material.

training der Erkenner zu realisieren. *ERBA* und *Verbmobil* unterscheiden sich im Vokabular, den Aufnahmebedingungen und in der Sprechart. *ERBA* besteht aus gelesenem Material, *Verbmobil* aus spontansprachlichem. Diese Unverträglichkeit der Korpora spiegelt sich in den deutlich besseren Ergebnis der baumgesteuerten Erkennung wider: Das System hat für das beste Ergebnis mit 82.71% eine mehr als 21% absolut höhere WA als das auf *Verbmobil* basierende System. Auch die Bestenlisten der Länge 5 und 10 sind deutlich besser, wenn auch aufgrund der Höhe des Wertes mit 10% nicht mehr so stark. Dagegen zeigt das *Graphem**-System nahezu ähnliches Verhalten für beide Trainingskorpora, mit einem etwas besseren Ergebnis auf dem *Verbmobil* Korpus. Die Ursache dafür liegt vermutlich an dem vom Umfang her größeren Trainingsmaterial, das durch besser angepasste Monographeme den Mismatch des Trainings- und Testmaterials kompensieren konnte.

6.4.4 Baumerweiterung

Nachdem sich in den vorherigen Experimenten gezeigt hat, dass die baumbasierte Erkennung die besten Resultate liefert, wurden von ihr ausgehend weitere Systemverbesserungen untersucht. Da bis zu einem Einsatz des Systems einerseits zusätzliche Arbeit in die Reduktion der Baumrepräsentation investiert werden kann und andererseits größere, günstige Speicherbausteine verfügbar sein werden, wurde der erhöhte Speicheraufwand des Verfahrens in Kauf genommen.

Die am einfachsten realisierbare Optimierung ist die Erweiterung der Baumstruktur: durch zusätzliche Transkriptionen, die manuell und automatisch erfolgen können, werden von den einzelnen Wörtern mehrere Versionen im Erkenner aufgenommen, die jeweils aber auf die selbe Grundform abgebildet werden. Wie bereits mehrfach erwähnt wurde, basiert der Städteerkenner auf Graphemen, also Buchstaben. Daher kann die Expansion auch von nicht phonetisch ausgebildeten Benutzern in Form von Neutranskription/Lexikoneintrag-Paaren durchgeführt werden. Die Erweiterung wird automatisch durch Zerlegung der Neutranskripte in bekannte Trigraphemsequenzen realisiert. Tauchen dabei unbekannte Trigrapheme auf, wird das Neutranskript verworfen, da sich die Reduktion auf Mono-

grapheme als zu fehlerträchtig erwiesen hat. Allerdings ist das Trainingsmaterial bewusst variantenreich gewählt worden, damit dieser Fall möglichst nicht auftritt.

Obwohl die manuelle Erweiterung recht einfach machbar ist, kann sie aus Zeitgründen verständlicherweise nicht für alle Einträge[10] eingesetzt werden. Sie wurde bisher lediglich verwendet, um bestimmte regionale Aussprachevarianten, z.B. `stuagatt` für Stuttgart, sowie unregulär gesprochene Namen, etwa Oeynhausen (`öhnhausen`) oder unsicher erkannte Namen wie Calw (`kalf`) zu integrieren. Die kritischen Namen wurden auf dem *10k-Städte-* Korpus bestimmt: wurde von allen Aufnahmen eines Namens wenigstens eine pro Sprecher korrekt erkannt, galt der Name als nicht kritisch. Um die Erkennungsrate der kritischen Fälle zu verbessern, ohne neue Fehler zu produzieren, wurden in vier automatischen Baumerweiterungen jeweils die nicht mehr erkannten gegen die neu erkannten Städte aufgerechnet und die nächste Modifikationsiteration daraufhin abgestimmt. Unter den nicht erkannten Namen wurde versucht, Gemeinsamkeiten in der Schreibweise von ihnen bzw. den mit ihnen verwechselten Städten zu finden, um diese für eine zusätzliche Regel zu benutzen. Die so entstandenen Regelwerke wurden schließlich eingesetzt, um auch die Bäume für die Erkennung der *50k-Liste*-zu erweitern. Die Evaluation dieser Erkenner wurde ebenfalls mit dem *10k-Städte*-Korpus sowie mit nicht aufgezeichneten Tests von mehreren Probanden durchgeführt. Bis zur Fertigstellung der Arbeit stand noch kein Korpus zur Verfügung, der das gesamte Material abdeckte[11]. Die folgenden Evaluationen im Bereich Baumerweiterung sowie Verwechslungsmatrix sind aufgrund der Beschaffenheit des Testmaterials anwendungsnäher durchgeführt worden. Da im intendierten Gesamtsystem der Benutzer bei Erkennungsfehlern durch Wiederholung der Äußerung einen erneuten Erkennungsversuch starten kann, gilt eine Stadt dann als erkannt, wenn sie innerhalb von drei Wiederholungen wenigsten einmal hypothetisiert wurde. Auf diese Art und Weise stören auch die durch die automatische Schneidung teilweise verstümmelten Äußerungen das Ergebnis nicht mehr, da in der Regel die Wiederholungen besser gesprochen wurden und somit einen korrekten Mitschnitt lieferten.

Die ersten Experimente zur Erweiterung der Bäume auf dem 10k-System gibt Tabelle 6.6 wieder. Das Basisline-System erreichte unter den oben genannten Evaluationsbedingungen über alle Sprecher eine Erkennungsleistung von 83.3%.

Die erste Baumvariante wurde durch zuätzliche Einführung von Transkriptionen ohne führende `h` und `f` generiert. Dadurch wurde es möglich, beispielsweise *Frankfurt* zu erkennen, auch wenn der Anlaut durch die Sprachdetektion unterdrückt wurde. Dies geschah verhältnismäßig häufig bei *Sprecher b*, wodurch seine Erkennungsrate signifikant um 0.4% verbessert werden konnte. Nicht ganz so

[10] In den Größenordnungen 60.000 für Städtenamen und 400.000 für Straßennamen

[11] Basierend auf den Erfahrungen beim Erstellen des *10k-Städte*-Korpus erfordert das dreimalige Sprechen und Nachbearbeiten des erweiterten Materials pro Person etwa 30 Tage.

Baum	Sprecherin	Sprecher a	Sprecher b	Durchschnitt
Standard	79.10	82.86	87.94	83.30
1	79.27	82.37	88.32	83.32
2	79.32	82.67	88.39	83.46
3	79.42	82.61	88.85	83.63
4	79.22	83.08	88.68	83.66

Tabelle 6.6: Erkennungsraten [%] auf dem *10k-Städte*-Korpus für unterschiedlich stark erweiterte Bäume des **10k**-Systems unter Berücksichtigung von jeweils 2 Wiederholungen.

Baum	Sprecherin	Sprecher a	Sprecher b	Durchschnitt
Standard	65.20	71.43	79.11	71.91
1	64.98	71.13	78.98	71.70
2	65.00	71.43	79.09	71.84
3	64.86	70.89	79.71	71.82
4	64.64	70.65	79.61	71.63

Tabelle 6.7: Erkennungsraten [%] auf dem *10k-Städte*-Korpus für unterschiedlich stark erweiterte Bäume des **50k**-Systems unter Berücksichtigung von jeweils 2 Wiederholungen.

häufig trat dieses Problem bei der *Sprecherin* auf, was ebenfalls zu einer Verbesserung führte, wenn auch kleiner und nicht signifkant. Die deutliche, gut zu detektierende Sprache von *Sprecher a* schließlich benötigte diesen Kunstgriff nicht, im Gegenteil, das Vorgehen führte zu einer signifikanten Verschlechterung, die aber im Gesamtergebnis von den anderen Sprechen mehr als kompensiert wurde.

Der zweite Baumvariation beinhaltete den vorigen Baum mit einer zusätzlichen Ergänzung durch alternative Umsetzung aller Vorkommenden `y` durch `i` und `ü`. Dies führte vor allem bei *Sprecher a* zu einer Verbesserung, wenngleich bei ihm der Baseline-Wert nicht erreicht wurde.

Bei der dritten Version des Baumes wurden nicht mehr so zaghafte Anpassungen gemacht. Sämtliche Doppelkonsonaten wurden zusätzlich noch auf die entsprechenden einzelnen abgebildet, `h` hinter `r` und `t` entfernt sowie `pf` auch als `f` eingesetzt. Bei *Sprecher a* führte dies zu einem minimalen Einbruch, die anderen wurde deutlich besser erkannt.

Die letzte Variation schließlich ergänzte noch Umschreibungen von `ar` zu `a` und Abbildung aller `ü` auf `i`, was vor allem die Erkennungsrate von *Sprecher a* verbesserte, endlich auch besser als der Ursprungswert.

Die gleichen Baumerweiterungen wurden sukzessiv auch für den 50k-Erkenner

eingesetzt; die Ergebnisse sind in der Tabelle 6.7 zusammengefasst. Dabei zeigte sich allerdings, dass lediglich für *Sprecher b* eine signifkante Verbesserung erreicht wurde, während die Erkennung bei den anderen beiden Sprechern signifikant schlechter wurde. Nach Überprüfung, wie häufig jeweils einzelne Namen hypothetisiert wurden, fiel auf, dass einige Städte mehrfach auftraten, diese Mehrfachnennungen aber von allen Versionen der Transkriptionen aus den Baumergänzungen stammten. Da die Ausgabe selber jeweils alle Namen nur einmal enthält, fällt diese Bevorzugung einzelner Städte nicht direkt nach außen auf. Intern behindert sie aber dennoch das Auftreten anderer Alternativen, was sich bei dem fünffach größeren Vokabular offensichtlich negativ bemerkbar macht.

Dennoch bietet sich die vorgeschlagene Vorgehensweise an, kleinere Vokabulare wie etwa die Straßenlisten, die pro Stadt nicht größer als 13.000 Einträge werden, besser zu erkennen.

Ein weiteres Problem für große Vokabulare ergibt sich aus dem erhöhten Speicheraufwand für den Baum. Bei Anwendungen aller Regeln, also in der vierten Baumalternative, wächst der Baum um etwa 20%. Das lässt sich für Lexika in der 10.000er Ordnung mit einem Zuwachs von etwa 1MB[12] noch tolerieren, ein Zuwachs von 5MB ohne deutliche Verbesserung der Erkennungsrate hingegen nicht.

6.4.5 Verwechslungsmatrix

Bei der zuvor diskutierten $n-$best-Erkennung gibt es neben den durch unvollständige Signale entstehenden Fehlern auch solche, die durch leichte Abweichungen in der Aussprache entstehen. Dieser Effekt kann teilweise in der Erkennung durch die Ausgabe der $n-$besten Ergebnisse kompensiert werden, allerdings zeigte sich, dass häufig wenige Hypothesen dominant waren und frühzeitig ähnlich klingende Alternativen aufgrund der Suchraumbeschneidung verdrängten. Daher wurden verschiedene Untersuchungen durchgeführt, um die Ausgabe des Erkenners nachträglich durch Verwechslungsalternativen zu erweitern. Zu jeder Stadt kann vorab eine Liste von potentiell ähnlich klingenden Namen automatisch und/oder manuell erstellt werden, die dann die Ergebnislisten erweitern können. Die vergrößerten Listen müssen anschließend umsortiert und auf n Namen beschnitten werden.

Finden der Verwechslungspartner

Um potentielle Verwechslungspartner zu finden, kann man die Buchstabenrepräsentation jedes Namens mit ähnlichen Regeln wie bei der zuvor beschriebenen Baumergänzung vereinfachen und anschließend alle Namen zusammenfassen, die dieselbe vereinfachte Schreibweise haben. Analog der Baumergänzung kann neben

[12] Diese Angabe ist bezogen auf einen nicht komprimierten, ASCII-repräsentierten Baum, die Binärversion mit Indizes hat etwa 25% der Ursprungsgröße.

der automatischen Erweiterung auch hier in den fertigen Listen manuell eingegriffen werden, um Verwechslungspaarungen zu ermöglichen, deren Schreibweise dies ansonsten nicht zulassen würde. Dies trifft im Besonderen auch auf Namen mit Zusätzen zu, beispielweise kann *Frankfurt* als Verwechslungspartner *Frankfurt am Main* und *Frankfurt an der Oder* bekommen, wobei letztere nicht im eigentlichen Erkennerlexikon enthalten sein müssen.

Der Grundstock der Regeln zur Vereinfachung der Schreibweisen wurde von der Baumergänzung genommen. Dies kollidiert nicht mit der baumgesteuerten Erkennung, da die Anwendung der Verwechslungsmatrix erst nach einem Erkennungslauf einsetzt, also nicht der oben geschilderten Problematik der konkurrierenden Hypothesen unterliegt. Weitere Regeln wurden durch Vergleich der phonetischen Transkripte von Wörtern aus dem *ERBA* -Korpus gemacht, die in Abschnitten ähnlich gesprochen, aber unterschiedlich geschrieben werden.

Mit den Regeln wurden unter anderem alle Doppelkonsonanten auf ihre einfachen Vertreter gesetzt, alle `c` die nicht von `k` oder `h` gefolgt wurden auf `k` abgebildet, alle `v` durch `f` ersetzt, sämtliche `h` nach `r` oder `t` entfernt, Auslautverhärtungseffekte durch rigoroses Austauschen aller `d`, `b` und `g` durch `t`, `p` bzw. `k` abgefangen etc. Die komplette Liste der Regeln befindet sich im Anhang A.7. Je großzügiger das Regelwerk gewählt wurde, umso mehr Verwechslungspaarungen konnten gefunden werden. Da im Extremfall mehr als n Namen in einer Verwechslungsgruppe auftreten können, die durch Beschränkung der Ausgabeliste auf n Einträge niemals ausgegeben werden könnten, wurde die Erweiterung der Regeln eingestellt, als die Gruppengrößen den kritischen Wert erreichten. Allerdings kann selbst mit dieser Beschränkung der Fall eintreten, dass die erweiterte Ausgabe mehr als n Einträge enthält, in diesem Fall wird sie einfach auf n Einträge beschnitten. Um dennoch möglichst gute Ergebnisse zu erhalten, wurden experimentell verschiedene Möglichkeiten der Listenerweiterung untersucht.

Erweitern der Ergebnisliste

Nachdem die Verwechslungsmatrix generiert wurde, stellt sich die Frage, wie die Ergebniserweiterungen am effektivsten in die Ausgabelisten integriert werden können, ohne dass die Limitierung auf n Namen überschritten wird. Basierend auf dem Korpus *10k-Städte* wurden drei verschiedene Erweiterungsarten untersucht. Allen ist gemeinsam, dass sie in ihrer Anwendung keinerlei Informationen über den Aufbau der Ausgabewörter benutzen, sondern lediglich Nachschlagen in abstrakten Listen von beispielsweise Indizes erfordern, demnach also schnell abgreifbar sind und nur wenig Speicher benötigen.

Im ersten Verfahren wurde die Ausgabeliste so sortiert, dass alle Namen direkt von ihren Verwechslungspartnern gefolgt werden. Das zweite Verfahren hängt an die Ausgabeliste jeweils die Alternativen als Block an. Das dritte ordnet die Alternativen ebenfalls nach den erkannten Namen an, fasst aber jeweils die Ver-

Baum	Verfahren 1	Verfahren 2	Verfahren 3	(ohne)
Standard	70.41	74.51	74.31	71.91
1	70.11	74.29	74.13	71.70
2	70.27	74.42	74.25	71.84
3	70.01	74.50	74.27	71.82
4	70.09	74.27	74.06	71.63

Tabelle 6.8: Erkennungsraten [%] auf dem *10k-Städte*-Korpus für unterschiedlich stark erweiterte Bäume des **50k**-Systems mit Verwechslungsmatrix und unterschiedlichen Einfügungsverfahren, unter Berücksichtigung von jeweils 2 Wiederholungen.

wechslungspartner in ihrer Reihenfolge zusammen. Schematisch stellen sich die Verfahren für die Namen a, b, c, d und e, die vom Erkenner geliefert werden, und ihre durch Indizes markierten Alternativen $a_0, a_1, b_0, b_1, b_2, c_0, c_1$ und e_0, die durch die Verwechslungsmatrix ergänzt werden, folgerdenmaßen dar:

Verfahren	Pos. 1	2	3	4	5	6	7	8	9	10	/	/	/
1	a	a_0	a_1	b	b_0	b_1	b_2	c	c_0	c_1	d	e	e_0
2	a	b	c	d	e	a_0	a_1	b_0	b_1	b_2	c_0	c_1	e_0
3	a	b	c	d	e	a_0	b_0	c_0	e_0	a_1	b_1	c_1	b_2

Die Evaluationen sind mit dem 50k-Erkenner auf dem *10k-Städte*-Korpus durchgeführt worden (s. Tabelle 6.8). Um etwaige Interaktionen durch die Baumerweiterungen zu berücksichtigen, wurden alle Experimente sowohl auf dem Standardbaum als auch auf seinen vier erweiterten Versionen überprüft. Beim Vergleich der unterschiedlichen Bäume zeigte sich, dass innerhalb der Verfahren Variationen existieren, die aber bei weitem nicht so extrem waren wie die zwischen den Verfahren. Nach obigen Überlegungen bietet sich die Baumerweiterung für das große Lexikon nicht an, dasselbe trifft beim zusätzlichen Einsatz der Ausgabelistenerweiterungen zu: die besten Ergebnisse werden auf dem handlichen Standardbaum erreicht. Das Experiment ohne Erweiterung (rechte Spalte) übertrifft mit 71.91% die durch Verfahren 1 modifizierte Version, die lediglich 70.41% erreichte. Offensichtlich hat das Einfließenlassen der Alternativen zwischen die Erkennungsergebnisse die an sich gute Erkennerausgabe gestört. Anders bei den beiden anderen Verfahren, an die Erkennerausgaben anhängen: Beide Ergebnisse sind mit 74.51% bzw. 74.31% deutlich besser als die Variante ohne Erweiterung, wobei aufgrund der etwas höheren Leistung Verfahren 2 die Methode der Wahl ist.

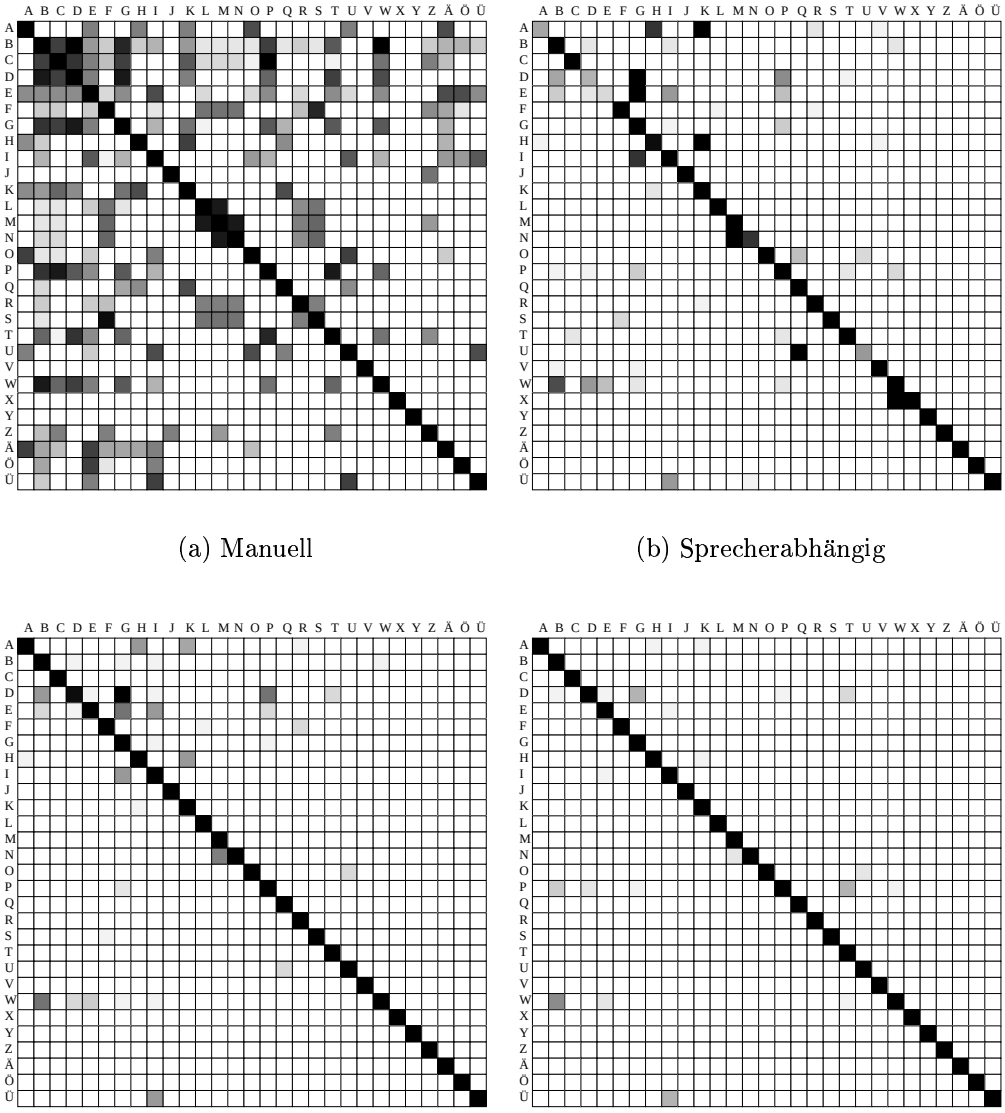
Um das Verfahren weiter zu verbessern, ist es möglich, die Alternativlisten für jeden Namen bei der automatischen Generierung zu sortieren, so dass die ähnlichsten Namen (phonetisch, ortografisch, Anzahl Buchstaben oder Vergleichbares) vorne in den Listen auftreten. Diese Möglichkeit wird in näherer Zukunft untersucht werden.

6.5 Buchstabenerkennung

Der Buchstabenerkenner (s. Abschnitt 5.4) erlaubt das Buchstabieren von bekannten Wörtern, um bei einer fehlgeschlagenen Standarderkennung doch noch verbal die gewünschte Eingabe durchzuführen. Da die einzelnen Buchstaben sehr kurz und phonetisch hochähnlich sind, ist die erreichte Buchstabenakkuratheit mit knapp 50% nicht sehr genau. Daher wird die Erkennung eines Zeichens durch potentielle Verwechslungsalternativen aufgestockt. Beim Abgleich des so generierten Buchstabenhypothesengraphen gegen das Lexikon bekannter Wörter werden alle Alternativen berücksichtigt.

Experimente mit verschiedenen Verwechslungsmatrizen auf Buchstabenebene wurden auf 500 buchstabierten Städtenamen aus dem *SLACC* Korpus durchgeführt. Die Verwechslungsmatrizen werden in Abbildung 6.2 dargestellt, dunkle Einträge entsprechen hohen Verwechslungswahrscheinlichkeiten zwischen den Zeichen. Die erste Matrix (Name 'Manuell') ist von einer ausgebildeten Linguistin aufgrund phonetischer Transkripte der Buchstaben manuell erstellt worden. Die dabei verwendeten Abstandsmaße werden im Anhang A.6 aufgeführt. Die anderen drei Matrizen sind automatisch erzeugt worden, indem erkannte Buchstabenfolgen auf Trainingsmaterial mit den Referenzzeichen verglichen und ausgezählt wurden. Für die 2. Matrix ('Sprecherabhängig') wurden die Resultate vom nicht im Test enthaltenen Material der einzelnen Sprecher für eine sprecherabhängige Matrix benutzt. Die beiden letzten Matrizen ('Sprachmodell 1' bzw. 'Sprachmodell 5') wurden sprecherunabhängig auf einer bigramm-unterstützten Erkennung mit Gewichtung von 1 bzw. 5 bei der Verrechnung mit der akustischen Bewertung generiert. Das Bigramm wurde auf phonetisch ausgewogenem Material aus dem *ERBA* -Korpus berechnet. Beim Vergleich der Abbildungen ist auffallend, dass nur die manuelle Variante symmetrisch an der Diagonalen ist, die automatischen dagegen nicht. Offensichtlich treten einige Vertauschungen nur in einer Richtung auf. Weiterhin ist eine starke Einfärbung der manuell erstellten Matrix zu erkennen, viele Verwechslungen treten mit nahezu gleicher Wahrscheinlichkeit auf. Von den vier Matrizen sind sich die Sprachmodell-1-Variante und das sprecherabhängige Modell am ähnlichsten, wobei Letzteres als Einziges keine durchgehend hochbewertete Diagonale hat. Für den Sprecher, auf dem die Abbildung basiert, gab es wenig erfolgreiche Erkennungen von *a, d, e* und *u*.

Die erreichten Ergebnisse sind nach dem Lexikonabgleich nur auf Wort- und



(a) Manuell

(b) Sprecherabhängig

(c) Sprachmodell 1

(d) Sprachmodell 5

Abbildung 6.2: Verwechslungsmatrizen für Buchstaben, basierend auf unterschiedlichem Material. Die Plots sind jeweils zeilenweise auf ihren Maximalwerten normiert, Schwarz=Maximum, Weiß=Minimum.

nicht mehr auf Zeichenebene angegeben, als Lexikon wurde das der *50k-Liste* verwendet. Das beste Ergebnis mit 89.11% (s. Tabelle 6.9) erreichte der Lexikonabgleich mit der sprecherabhängigen Verwechslungsmatrix, das schlechteste mit

Verwechslungs-matrix	Manuell	Sprecher-abhängig	Sprachmodell Gewicht 1	Sprachmodell Gewicht 5	Sprecherabhg. adaptiert
Erkennungsrate	75.25%	89.11%	88.45%	77.23%	96.5%

Tabelle 6.9: Erkennungsergebnisse für den Lexikonabgleich von Buchstabierfolgen mit unterschiedlichen Verwechslungsmatrizen.

75.25% die manuell[13] erstellte. Von den beiden auf einem Sprachmodell basierenden Versionen war mit 88.45% die kleinere Gewichtung der mit 77.23% deutlich schwächeren größeren Gewichtung überlegen. Wesentlich bei den Ergebnissen scheint sich die Korpusnähe auszuzeichnen: Das abstrakte, manuelle Verfahren schneidet am schlechtesten ab, gefolgt von dem durch die höhere Gewichtung dominanten Sprachmodell, danach dem schwächeren Sprachmodell und schließlich der sprecherabhängigen Variante, die auf Material aus demselben Korpus wie der Test basiert.

Das Resultat der sprecherabhängigen Version konnte anschließend noch durch dynamische Anpassungen auf 96.5% gesteigert werden. Die erste Version der Matrix wurde vor dem Einsatz erstellt und schließlich iterativ mit jedem Erkennungslauf verbessert. Da jeweils die erkannte Graphemfolge vorlag und die durch Auswahl und Bestätigung des Benutzers gewünschte Stadt und damit intendierte Buchstabenfolge bekannt war, konnten die entsprechenden Verwechslungswahrscheinlichkeiten adaptiert werden. Analog kann das Verfahren auch im Fahrzeug eingesetzt werden und, falls eine Fahreridentifizierung[14] benutzt wird, sich im Laufe der Zeit für den Benutzer optimieren.

6.6 Merkmalsreduktion

Bei den letztendlich eingesetzten System wurde ein 39-komponentiger Merkmalsvektor, bestehend aus der Energie und den ersten zwölf Cepstralkoeffizienten sowie deren ersten und zweiten Ableitungen eingesetzt. Obwohl sich gezeigt hat, dass diese Auswahl in Hinsicht auf Speicheraufwand, Verarbeitungsgeschwindigkeit und Qualität zufriedenstellend ist, wirft sich die Frage auf, wie eine Reduktion des Merkmalsvektors die entsprechenden Kategorien beeinflusst.

Durch Auslassen jeweils eines Merkmals und Vergleich der Wortakkuratheit der reduzierten Systeme auf dem *Verbmobil*-Korpus wurde die Wichtigkeit der einzelnen Merkmale bestimmt. Nachdem das wichtigste Merkmal gefunden war, wurde es aus dem Merkmalsvektor entfernt und im nächsten Schritt das dann beste Merkmal bestimmt. Dieses Verfahren wurde iteriert, bis eine komplette Eintei-

[13] Dieses Ergebnis sollte nicht pauschalisiert werden und die Arbeit der LinguistInnen abwerten.
[14] Die beispielsweise über personalisierte Schlüssel o.ä geschehen kann.

Rang	Merk-mal	Rang	Merk-mal	Rang	Merk-mal	Rang	Merk-mal	Rang	Merk-mal
1	1	9	4	17	5'	25	1"	33	5"
2	E'	10	4'	18	8	26	11'	34	12"
3	2	11	6	19	9'	27	11	35	10"
4	3	12	E"	20	7'	28	3"	36	9"
5	3'	13	7	21	12	29	10'	37	8"
6	5	14	9	22	2"	30	7 12'	38	11"
7	2'	15	6'	23	10	31	4"	39	6"
8	1'	16	E	24	8'	32	7"		

Tabelle 6.10: Merkmale sortiert nach Priorität, E=Energie, Zahlen=Cepstralkoeffizienten, '=1.Ableitung, "=2.Ableitung. Markiert sind die unterschiedlichen Vektorgrößen für die Experimente.

lung aller 39 Komponenten vorlag. Dabei war die Grundannahme, dass es keine Abhängigkeiten der Merkmale untereinander gibt, ansonsten hätten alle Kombinationsmöglichkeiten ausgewertet werden müssen, was eine nicht realisierbare Anzahl[15] von Testsystemen erfordert hätte. Bedingt durch diese Vorgehensweise sind die wichtigsten Merkmale eindeutig bestimmbar, die weniger wichtigen allerdings nicht mehr: Die Wortakkuratheit der einzelnen Testsysteme sinkt bedingt durch das Evaluieren von immer unwichtigeren Merkmalsuntermengen. Dadurch steigt aber auch das Niveau, das zum Erreichen von signifikanten Unterschieden benötigt wird, was zur Folge hat, dass ab 8 Merkmalen keine Unterschiede mehr messbar waren. Um die Rangfolgetabelle trotzdem zu vervollständigen, sind die absoluten Zahlen benutzt worden. Die sich ergebene Reihenfolge der Merkmale ist in Tabelle 6.10 aufgeführt.

Die Cepstralkoeffizienten treten etwa in ihrer Reihenfolge auf, die ersten sind, wie zu vermuten war, die wichtigsten. Interessant ist, dass vor den höheren Koeffizienten zuerst viele 1. Ableitungen der niederen zu finden sind. Weiterhin tragen die 2. Ableitungen, ausser die der Energie, am wenigsten zu einer guten Erkennung bei.

Basierend auf dieser Rangfolge sind neue Experimente aufgesetzt worden, die lediglich die besten 12, 16, 20 und 24 Merkmale aus der Tabelle benutzten. Während das entsprechende Basis-Experiment auf allen Merkmalen eine Wortakkuratheit von 65.13 erreichte, erreichten fast alle Systeme mit kleinerer Dimensionalität unter Berücksichtigung der Signifikanzschwelle von ± 2.3 die glei-

[15] Es gibt $\binom{39}{n}$ Möglichkeiten, n-dimensionale Merkmalsvektoren aus 39-komponentigen zusammenzustellen. Bei $n = 20$ sind das etwa $7 \cdot 10^{10}$.

Merkmale	WA (Klassen)	WA (Klassen)
39	65.13 (256)	- -
12	65.01 (256)	65.99 (512)
16	63.60 (256)	64.21 (512)
20	62.68 (256)	66.05 (392)
24	65.56 (256)	65.56 (378)

Tabelle 6.11: Wortakkuratheit der Experimente mit reduzierten Merkmalsvektoren. Weitere Informationen im Text.

che Qualität, lediglich der Erkenner mit den besten 20 Merkmalen lag knapp darunter.

Obwohl durch die verminderte Merkmalszahl weniger Parameter im System trainiert werden mußsen und somit potentiell mehr Klassen im Codebuch etabliert werden können, sind aus Gründen der Vergleichbarkeit alle Experimente mit einem 256-Klassen-Codebuch durchgeführt worden. Wird beim Erstellen der einzelnen Codebücher diese Schwelle erhöht, können mehr Klassen erstellt und damit eine feinere Auflösung erreicht werden, was zu einer Verbesserung der Ergebnisse führt. Allerdings waren, bis auf den Ausreißer bei 20 Merkmalen, keine im Vergleich zum Basissytem oder zur entsprechenden 256-Klassen-Variante signifikant besseren Ergebnisse erreicht worden. Die Resultate im Einzelnen sind in der Tabelle 6.11 aufgeführt.

Um die Reihenfolge der Merkmale auch auf dem Fahrzeugkorpus zu verifizieren, wurde das Experiment für den erweiterten *SLACC* -Korpus wiederholt. Der Erkenner benutzt ein diagonales Codebuch mit 512 Klassen und erreicht für den kompletten Merkmalssatz knapp unter 89% Wortakkuratheit bei einer Signifikanz von 1.9. Die Abbildung 6.3 stellt den Abfall der Wortakkuratheit bei abnehmender Merkmalszahl dar. Die Reihenfolge der Merkmale entspricht der auf dem *Verbmobil* -Korpus experimentell ermittelten. Erstaunlicherweise gibt es einen signifikanten Einbruch der Erkennungsleistung erst bei der Reduktion auf die 14 wichtigsten Merkmale. Ein wesentlicher Grund dafür liegt in der größeren Klassenzahl des Codebuchs und damit einfacheren Aufteilung des Merkmalsraumes.

6.7 Automatisches Transkribieren

Ein interessantes Problem bei der Behandlung von unbekannten Wörtern ist das *Lernen* neuer Wörter. Je nachdem, wie sie im System später wieder erkannt werden sollen, gibt es unterschiedliche Verarbeitungstrategien. Die einfachste ist dabei das Extrahieren von Merkmalsschablonen aus Beispielaufnahmen der entsprechenden Wörter und Mustervergleich über DTW (s. Abschnitt 5.3.2) von einer auf dem aktuellen Sprachsignal berechneten Merkmalsschablone und allen abgespei-

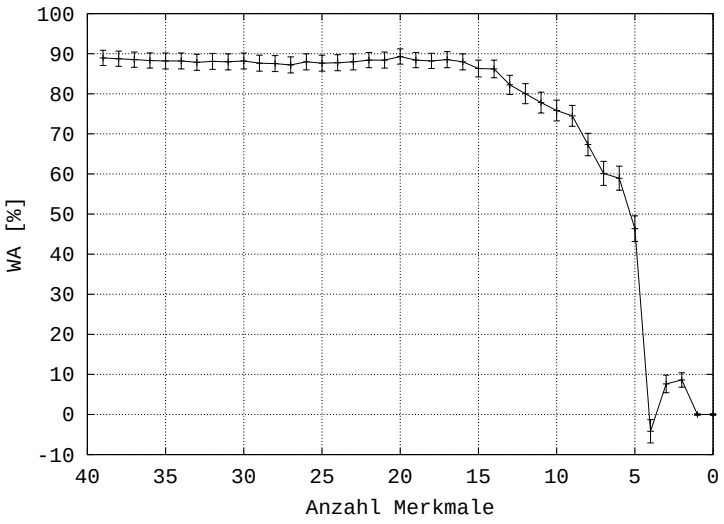


Abbildung 6.3: Verhalten der Wortakkuratheit bei abnehmender Merkmalszahl für den *SLACC* Korpus, vertikale Marker für die Signifikanz.

cherten. Damit einher geht allerdings das Problem der fehlenden Vergleichbarkeit der Resultate dieser Erkennungsform gegenüber denen der herkömmlichen Erkennung: Wenn beide Verfahren in Konkurrenz auf demselben Sprachabschnitt arbeiten, welches System liefert dann das richtige Ergebnis? Beide benutzen zwar interne Bewertungen, die aber auf unterschiedliche Arten gewonnen werden und damit nicht vergleichbar sind. Abgesehen davon sind die Merkmalsschablonen sehr abstrakt, sie lassen sich nicht mehr auf ein tatsächliches Wort abbilden und damit auch nicht mehr auf einem Display darstellen.

Eleganter und einfacher lassen sich neue Wörter in den Erkennerablauf integrieren, wenn ihre Transkription vorliegt. Theoretisch könnte man vom Benutzer eine Eingabe der Transkription, wenigstens der orthographischen, anfordern, praktisch wäre dies mit den eingeschränkten nonverbalen Eingabemedien aber nicht benutzerfreundlich. Die verbale Eingabe durch Buchstabieren des Namens ist ebenfalls nicht praktikabel, da sie aufgrund der ungenauen Erkennung lediglich zum Aussuchen von bekannten Wörtern aus einem Lexikon benutzt werden kann. Daher wurden Experimente mit einer automatischen Transkription von Wörtern durchgeführt. Als Korpus wurde *CityMobil* -Material gewählt. In den Untersuchungen wurden die häufig vorkommenden Wörter künstlich entfernt und im Erkenner mit einer simulierten Detektion[16] über ein automatisches *Alignment* mit jeweils einem eigenen Transkript versehen. Dieses Transkript wurde in Modellfolgen übersetzt

[16]Da die zu lernenden Wörter nur über einen definierten Dialog eingegeben werden, kann man von einer perfekten Erstdetektion ausgehen.

Trainings-beispiele	Anzahl Vorkommen	DA manuell	DA automatisch
1	745	58.26%	48.09%
2	607	76.11%	66.13%
3	482	80.50%	74.39%

Tabelle 6.12: Detektionsakkuratheit (DA) der simulierten unbekannten Wörter mit manueller und automatischer Transkription bei unterschiedlicher Anzahl von Testbeispielen.

und in den Erkenner integriert. Anschließend wurde die Qualität des erweiterten Systems in einer Evaluierung bestimmt. Um die Güte der Erkennung objektiv beurteilen zu können, wurden die Experimente zusätzlich mit einer manuellen, d.h. phonetisch korrekten, anstelle der automatischen Transkription wiederholt. Die Erkennungsleistung im 2. Ansatz entspricht dem Optimum des automatischen Verfahrens. Alle Tests fanden sprecherabhänging statt, d.h. aus dem gesamten Sprachmaterial eines jeden Sprechers wurde jeweils eine bestimmte Anzahl von Trainingsbeispielen extrahiert. Mit ihnen wurden dann die sprecherabhängigen Lexikonerweiterungen, einmal mit automatischem, einmal mit manuellem Transkript, erstellt. Anschließend erfolgte die Evaluation mit dem gesamten Restmaterial des Sprechers in Form einer Auszählung der korrekt erkannten, nicht erkannten und fälschlich hypothetisierten ehemals unbekannten Wörtern. Die Ergebnisse sind in Tabelle 6.12 zusammengefasst. Wie zu erwarten war, kann die Detektionsakkuratheit der manuellen, optimalen Transkription nicht von der automatischen erreicht werden, allerdings nähern sie sich mit steigender Anzahl von Trainingsbeispielen an, von relativen 17.62% bei einem Trainingsbeispiel, über 13.11% bei zwei bis zu 7.59% bei drei.

Die Gesamtleistung über alle simulierten unbekannten Wörter ist mit einer Detektionsakkuratheit von 74.39% nach Angabe von drei Trainingsäußerungen nicht sehr zuverlässig, bei Betrachtung der Leistung für einzelne Vertreter der unbekannten Wörter zeigt sich die Hauptursache des Problems: Kurze Wörter. Ein Vertreter dieser Klasse ist das Wort `Loch`. In den insgesamt 378 Testäußerungen wurden ein Drittel aller Vorkommen nicht gefunden (s. Tabelle 6.13) und stattdessen hauptsächlich als Pause, `doch` oder `noch` erkannt. In 17 Fällen ist es anstelle von `doch` und `noch` falsch hypothetisiert worden, was sich durch die hohe phonetische Ähnlichkeit und die schlecht artikulierten Äußerungen dieser beiden Wörter erklären läßt. Im Gegensatz dazu wurden lange Wörter wie `Computer` oder `Frontverkleidung` mit fast 82% bzw. 95.68% sehr gut detektiert und nicht ein einziges Mal falsch generiert. Beim Einsatz dieses Verfahrens zur Integration neuer Wörter im Lexikon muss offensichtlich eine Mindestlänge sowie eine pho-

Wort	Anzahl Test-äußerungen	richtig erkannt	nicht erkannt	falsch hypothetisiert	DA
Loch	378	264	114	17	65.34%
Computer	61	50	11	0	81.97%
Frontverkleidung	139	133	6	0	95.68%

Tabelle 6.13: Aufgeschlüsselte Auszählungen und DA für drei simulierte unbekannte Wörter bei automatischer Transkription.

netische Disjunktheit vom Restvokabular vorausgesetzt werden. Experimentell erwiesen sich Wortlängen ab drei Silben als stabil.

7 Zusammenfassung und Ausblick

Das Ziel der vorliegenden Arbeit war die Untersuchung von verschiedenen Aspekten der Spracherkennung unter Berücksichtigung der Anforderungen zur Bedienung aller nicht sicherheitsrelevanten Funktionen in einem Fahrzeug. Der Hauptaugenmerk lag dabei auf der Ansteuerung eines Navigationssystems. Das große Lexikon mit etwa 65.000 Städte- und 400.000 Straßennamen und die damit einhergehenden Probleme der fehlenden Transkriptionen für phonembasierte Erkenner und der teilweise hohen Ähnlichkeit in der Aussprache diverser Wörter machen diese Aufgabe wenig attraktiv. Bisher realisierte Spracherkenner für die Ansteuerung eines Navigationssystems berücksichtigen entweder nur die wichtigsten 1000 Städte, die transkribiert und traininert werden, oder ersetzen das alte haptische Interface durch eine verbale Variante mit nur wenigen Kommandos, die zur Simulierung der Tasten dienen. Mit dem Übergang auf einen graphembasierten Erkenner wurde in dieser Arbeit die arbeitsintensive Transkription überflüssig und damit eine Möglichkeit geschaffen, basierend auf einem vorhandenen Erkenner neue Domänen schnell zu erschließen sowie ad hoc textuell vorliegende Wörter in einen Erkenner zu integrieren. Diese können aus dem Telefonbuch eines Handys oder auch aus der Straßenliste eines Navigationssystems stammen, vorausgesetzt, dass es sich um aussprechbare Einträge handelt und keine uneindeutigen Abkürzungen, die sich nicht automatisch expandieren lassen.

Es wurden verschiedene Strategien zur graphembasierten Erkennung untersucht, wobei die präfixbaumgesteuerte Erkennung auf Basis von Trigraphemen (Graphemen mit einem linken und rechten Nachbarn als Kontext) der sprachmodellgesteuerten Graphemkettenerkennung überlegen war. Da die Erkennung für die großen Lexika, die zudem auch noch Homophone wie *Rot, Roth, Rodt* und *Rhodt* enthalten, nicht eindeutig funktionieren kann, wurde als Ausgabe eine Auswahlliste von Ergebnissen gewählt. Diese kann dem Benutzer zur Bestätigung angezeigt bzw. im Falle einer eingebundenen Sprachsyntheseeinheit auch vorgelesen werden. Um die Erkennungsrate weiter zu verbessern, wurden verschiedene Strategien zur Erweiterung der Präfixbäume einerseits und Generierung einer nachgeschalteten Verwechslungsmatrix auf Wortebene andererseits untersucht. Während sich die Baumerweiterungen bei Lexika um 10.000 Wörter als erfolgreich erwies, brachte sie keinerlei Verbesserung bei größeren Namenslisten. Dagegen konnte unabhängig von der Lexikongröße durch die nachträgliche Expandierung der Ausgabeliste auf n Einträge die Erkennungsrate signifikant verbessert werden.

Zusätzlich wurde des System mit einem Modul zur Erkennung von Buchstabiersequenzen ausgestattet, um bei kritischen Wörtern notfalls auf einzelne Buchstaben zurückgreifen zu können. Allerdings eignet sich dieses Modul nur für die Eingabe bekannter Wörter und nicht für das Buchstabieren beliebiger Sequenzen. Da die Buchstabenerkennung an sich aufgrund der zu kurzen, akustisch zu ähnlichen Buchstabenwörter sehr fehleranfällig ist, kann sie nur durch Abgleich mit einem Lexikon unter Berücksichtigung von potentiellen Verwechslungen einzelner Zeichen geschehen, arbeitet dann aber robust.

Um für das Baseline-System Trainings- und Testdaten aus der intendierten Domäne einsetzen zu können, wurde im Rahmen der Arbeit ein eigenes Sprachkorpus entworfen und aufgenommen. Das Material umfasst mehr als 11 Stunden Aufnahmen von 22 Probanden und wurde parallel mit einen hochwertigen Nahbesprechungsmikrofon und einem in der Automobilindustrie verwendeten Mikrofon aufgenommen. Damit ein breites Spektrum an Hintergrundgeräuschen im Korpus ist, wurden insgesamt 5 verschiedene Fahrzeuge benutzt und alle Daten zweimal bei unterschiedlichen Geschwindigkeiten aufgezeichnet.

Für Demonstrationszwecke ist der Erkenner um eine Verstehenskomponente erweitert worden, die über einen Traversierungsbaum gesteuert wird und die Ansteuerung beliebiger Komponenten durch Einbindung von C-Funktionen realisiert. Es wurden Vorschläge für das Design eines kompletten Systems in Hinblick auf Funktionalität und Sprachumfang gemacht und schließlich in einem Demonstrationssystem umgesetzt.

Die nächsten anstehenden Arbeiten umfassen die Reduktion des Speicherbedarfs und die Optimierung der Geschwindigkeit, um das komplette System auf die Zielhardware zu portieren. Die Zielhardware ist ein einfaches DSP-Board, das den Anforderungen für den Einbau im Fahrzeug entspricht: Es ist robust gegen Erschütterungen und Temperaturschwankungen, wartungsfrei und kostengünstig.

Im Weiteren ist geplant, einen Übergang auf andere Sprachen durchzuführen. Inwiefern sich die graphembasierte Erkennung dort bewährt, bleibt zu untersuchen. Die Vermutung liegt nahe, dass Sprachen, die eine eindeutige Korrespondenz zwischen Schrift und Aussprache haben, problemlos mit dieser Art Erkenner genutzt werden können. Demnach dürfte die Rangfolge in der Einfachheit der Umsetzung für eine Auswahl von Sprachen von Finnisch über Englisch (britisch), Englisch (amerikanisch), Italienisch, Spanisch bis zu Französisch als kompliziertester Fall gehen. Zusätzlich dazu wird das *SLACC* Korpus ebenfalls mit diesen Sprachen erweitert.

Das im Rahmen dieser Arbeit entwickelte Spracherkennungssystem wurde durch Evaluierung verschiedener Ansätze und deren Optimierung durch Heuristiken dem Ideal der sprachlichen Mensch-Maschine-Kommunikation näher gebracht, in welchem die eingebauten Geräte intuitiv bedient werden können und man dem Auto sagen kann, wohin die Reise geht.

Literaturverzeichnis

[1] *An Introduction to Text-to-Speech Synthesis.* Kluwer Academic Publishers, Dordrecht, 1997.

[2] *The Bluetooth Specification*, 2001.

[3] F. Althoff, G. Drexel, H. Lüngen, M. Pampel, and C. Schillo. Morphology and Speech Technology. In *Proceedings of ACL 1996*, pages 25–30, Santa Cruz, CA, 1996.

[4] F. Althoff, G. Drexel, H. Lüngen, M. Pampel, and C. Schillo. The Treatment of Compounds in a Morphological Component for Speech Recognition. In D. Gibbon, editor, *Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference*, pages 71–76, Berlin, 1996. Mouton de Gruyter.

[5] A. Asadi, R. Scgwartz, and J. Makhoul. Automatic detection of new words in a large vocabulary continuous speech recognition system. In *Proc. Int. Conf. on Acoustic, Speech, and Signal Processing*, pages 125–128, Albuquerque, 1990.

[6] H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The philips automatic train timetable information system. *Speech Communication*, 17:249–262, 1995.

[7] N. Boukas, P. Naylor, and T. Stathaki. Voice activity detection using source separation techniques, 1997.

[8] H. Bussmann. *Lexikon der Sprachwissenschaft.* Kroner, Stuttgart, 1995.

[9] J. Clark and C. Yallop. *an Introduction to Phonetics & Phonology.* Blackwell, Oxford, 1990.

[10] K. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. In *Journal of the Acoustic Society of America*, volume 24, pages 637–642, 1952.

[11] E. Eide and H. Gish. A parametric approach to vocal track length normalisation. In *International Conference on Acoustic, Speech and Signal Processing*, pages 346–348, Atlanta, 1996.

[12] C. Feifel. *Wechselwirkungen - Jahrbuch der Universität Stuttgart*, chapter Internet ins Fahrzeug. Universität Suttgart, 2000.

[13] P. Fetter. *Detecting and Transcription of OOV Words.* Report 231, 1998.

[14] P. Fetter, F. Class, U. Haiber, A. Kaltenmeier, U. Kilian, and P. Regel-Brietzmann. Detection of Unknown Words in Spontaneous Speech. In *Proc. European Conf. on Speech Technology*, pages 1637–1640, Madrid, 1995.

[15] G. A. Fink. Developing HMM-based recognizers with ESMERALDA. In V. Matoušek, P. Mautner, J. Ocelíková, and P. Sojka, editors, *Lecture Notes in Artificial Intelligence*, volume 1692, pages 229–234, Berlin Heidelberg, 1999. Springer.

[16] G. A. Fink, C. Schillo, F. Kummert, and G. Sagerer. Incremental speech recognition for multimodal interfaces. In *Proceedings 24th Annual Conference of the IEEE Industrial Electronics Society*, pages 2012–2017, Aachen, Sep. 1998.

[17] J. Forgie and C. Forgie. Results obtained from a vowel recognition program. In *Journal of the Acoustic Society of America*, volume 31, pages 1072–1081, 1959.

[18] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Communications and Information Theory. Kluwer Academic Publishers, Boston, 1992.

[19] P. Geutner. Using morphology towards better large-vocabulary speech recognition systems. In *International Conference on Acoustic, Speech and Signal Processing*, pages 445–448, Detroit, 1995.

[20] D. Gibbon et al. *Handbook of standards and resources for spoken language systems.* Walter de Gruyter & Co., 1997.

[21] R. Haeb-Umbach, P. Beyerlein, and E. Thelen. Automatic Transcription of Unknown words in a Speech Recognition System. In *Proc. Int. Conf. on Acoustic, Speech, and Signal Processing*, pages 840–843, Detroit, MI, 1995.

[22] E. Haszto, W. Longenbraker, and J. Scherer. Talking to the network. *AT&T Technology*, 9(1):22–27, 1994.

[23] S. Hayamizu, K. Itou, and K. Tanaka. Detection of Unknown Words in Large Vocabulary Speech Recognition. In *Proc. European Conf. on Speech Technology*, pages 2113–2116, Berlin, 1993.

[24] L. I. Hetherington. New Words: effect on recognition Performance and Incorporation Issues. In *Proc. European Conf. on Speech Technology*, pages 1645–1648, Madrid, 1995.

[25] X. Huang, Y. Ariki, and M. Jack. Hidden markov models for speech recognition. In *Information technology series*, number 7. Edinburgh University Press, 1990.

[26] X. Huang, W. Hon, M. Hwang, and K. Lee. A comperative study of discrete, semicontinuous, and continuous hidden markov models. In *Computer Speech & Language*, volume 7, pages 359–368. 1993.

[27] K. Itou, S. Hayamizu, and K. Tanaka. Detection of Unknown Words and Automatic Estimation of their Transcriptions in Continuous Speech Recognition. In *Proc. Int. Conf. on Spoken Language Processing*, pages 799–802, Banff, Canada, 1992.

[28] F. Jelinek, R. Mercer, and L. Bahl. *Handbook of Statistics*, chapter Continuous Speech Recognition, pages 549–573. Krishnaiah, P. and Kanal, R., 1982.

[29] B. Juang. The past, present, and future of speech processing. In *IEEE SIGNAL PROCESSING MAGAZINE*, volume 15, pages 24–48, MAY 1998.

[30] A. Jusek. *Detektion unbekannter Wörter in gesprochener Sprache.* PhD thesis, Technische Fakultät, Universität Bielefeld, 1997.

[31] A. Jusek, G. Fink, F. Kummert, and G. Sagerer. Automatically Generated Models for Unknown Words. In *Proceedings Sixth Australian International Conference on Speech Science and Technology, December 10-12, 96*, pages 301–306, Adelaide, South Australia, 1996.

[32] A. Kai and S. Nakagawa. Evaluation of Unkown Word Processing in a Spoken Work Recognition System. In *Proc. Int. Conf. on Spoken Language Processing*, pages 2151–2154, Yokohama, 1994.

[33] K. Kohler, G. Lex, M. Pätzold, M. Scheffers, A. Simpson, and W. Thon. Handbuch zur Datenaufnahme und Transliteration in TP 14 von VERBMOBIL – 3.0. Technical Report 11, Institut für Phonetik und digitale Sprachverarbeitung, Universität Kiel, 1994.

[34] T. Kuhn. *Die Erkennungsphase in einem Dialogsystem.* Dissertationen zur künstlichen intelligenz, band 80, 1994.

[35] R. Lacouture and Y. Normandin. Detection of Ambiguous Portions of Signal Corresponding to OOV Words or Misrecognized Portions of Input. In *Proc. Int. Conf. on Spoken Language Processing*, pages 2071–2074, Philadelphia, 1996.

[36] L. Lamel, S. Bennacef, J. Gauvain, H. Dartigues, and J. Temem. User evaluation of the MASK kiosk. In *Proc. Int. Conf. on Spoken Language Processing*, pages 2875–2878, 1998.

[37] M. E. Latoschik. *Multimodale Interaktion in virtueller Realität am Beispiel der virtuellen Konstruktion.* PhD thesis, Technische Fakultät, Universität Bielefeld, 2001.

[38] K. Lee. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Boston, MA, 1989.

[39] C. Legetter and P. Woodland. Maximum likelihood linear regression for speaker adaption of continuous density hidden markov models. In *Computer Speech & Language*, number 9, pages 171–185. 1995.

[40] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10:707–710, 1966.

[41] A. Linke and N. Nussbaum. *Studienbuch Linguistik.* Max Niemeyer Verlag, Tübingen, 1996.

[42] E. Lombard. Le signe de l'élévation de la voix. *Annales des maladies de l'oreilles, du larynx, du nez e t du pharynx*, 37:101–119, 1911.

[43] B. Lowerre. The Harpy Speech Recognition System. Technical report, Carnegie-Mellon University, 1976.

[44] J. A. Markowith. *Using Speech Recognition.* Prentice Hall, New Jersey, 1996.

[45] J. Mugdan. Was ist eigentlich ein morphem? *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 39:29–43, 1986.

[46] J. Neppert and M. Pétursson. *Elemente einer akustischen Phonetik.* Helmut Buske Verlag, Hamburg, 1986.

[47] H. Niemann. *Klassifikation von Mustern.* Springer Verlag, Berlin, 1983.

[48] H. Olson and H. Belar. From interface to content: Translingual access and delivery of on-line information, 1997.

[49] D. Pye and P. Woodland. Experiments in speaker normalisation and adaptation for large vocabulary speech recognition. In *International Conference on Acoustic, Speech and Signal Processing*, pages 1047–1059, München, 1997.

[50] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–285. 1989.

[51] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[52] T. Raman. *Auditory User Interfaces –Toward The Speaking Computer*. Kluwer Academic Publishers, Boston, 1997.

[53] S. Rieck. *Parametrisierung und Klassifikation gesprochener Sprache*. PhD thesis, Lehrstuhl für Informatik 5 (Mustererkennung), Universität Erlangen-Nürnberg, 1994.

[54] D. Roy. Learning from multimodal observations. In *Proceedings of the IEEE Int. Conf. Multimedia and Expo*, 2000.

[55] D. Roy and A. Pentland. Learning words from audio-visual input. In *Proc. Int. Conf. on Spoken Language Processing*, pages 1279–1282, 1998.

[56] H. Sakamoto and S. Matsunaga. Detection of Unknown Words Using Garbage Cluster Models for Continuous Speech Recognition. In *Proc. European Conf. on Speech Technology*, pages 2103–2106, Madrid, 1995.

[57] T. Schaaf and T. Kemp. Confidence measures for spontaneous speech recognition. In *Proc. Int. Conf. on Acoustic, Speech, and Signal Processing*, pages 875–878, München, 1997.

[58] C. Schillo. Das SLACC Korpus. Technical report, Faculty of Technology, Bielefeld University, 2001.

[59] C. Schillo, G. A. Fink, and F. Kummert. Grapheme based speech recognition for large vocabularies. In *International Conference on Spoken Language Processing*, volume 4, pages 584–587, Beijing, China, 2000.

[60] R. Schmidt, editor. *Grundriß der Sinnesphysiologie*. Springer Verlag, Berlin, 1985.

[61] E. Schukat-Talamazzini. *Automatische Spracherkennung*. Vieweg, Wiesbaden, 1995.

[62] E. G. Schukat-Talamazzini, H. Niemann, W. Eckert, T. Kuhn, and S. Rieck. Automatic speech recognition without phonemes. In *Proc. European Conf. on Speech Communication and Technology*, pages 129–132, Berlin, 1993.

[63] C. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, 1949.

[64] B. Suhm, M. Woszczyna, and A. Waibel. Detection and transcription of New Words. In *Proc. European Conf. on Speech Technology*, pages 2179–2182, Berlin, 1993.

[65] J. van Santen, R. Sproat, J. Olive, and J. Hirschberg. *Progress in Speech Synthesis.* Springer, New York, 1996.

[66] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transaction on Information Theory*, volume 13, pages 260–269, 1967.

[67] W. von Kempelen. *Mechanismus der menschlichen Sprache nebst Beschreibung einer sprechenden Maschine.* Friedrich Frommann Verlag, 1791. Faksimile-Neudruck 1970.

[68] E. Zwicker. *Psychoakustik.* Springer, Berlin, Heidelberg, New York, 1982.

A Anhang

A.1 SAMPA

Das deutsche SAMPA mit Beispielen und ihren Transkriptionen, sortiert nach Lautklassen (| ist das Silbentrennsymbol).

	Vokale			**Diphtonge**	
[a]	Land	lant	[aI]	Eis	aIs
[a:]	Bahnhof	ba:n|ho:f	[aU]	Auto	aU|to
[e]	Medizin	me|di|tsi:n	[OI]	deutsch	dOI|tS
[e:]	Gegend	ge:|g@nt		**Nasale**	
[E]	Bielefeld	bi:|l@|fElt	[m]	morgen	mOr|gn
[E:]	Flughäfen	flu:k|hE:|fn	[n]	nach	na:x
[i]	siebter	zip|t6	[N]	Anfang	an|faN
[i:]	sieben	zi:|bm		**Plosive**	
[I]	ich	IC	[p]	lebhaft	le:p|haft
[o]	Auto	aU|to	[t]	Tore	to:|r@
[o:]	Bahnhof	ba:n|ho:f	[k]	Zug	tsu:k
[O]	Forst	fO6st	[b]	Bauer	baU|6
[2]	Wörlitz	v26|lI|ts	[d]	danke	daN|k@
[2:]	deux	d2:	[g]	Tage	ta:g@
[9]	Wörter	v96|t6		**Frikative**	
[u]	Fußball	fus|bal	[f]	Bielefeld	bi:|l@|fElt
[u:]	Zug	tsu:k	[s]	Fußball	fus|bal
[U]	Burg	bU6k	[S]	deutsch	dOI|tS
[y]	Grüß_Gott	grys|gOt	[C]	ich	IC
[y:]	früh	fry:	[x]	ach	ax
[Y]	Würzburg	vYrts|bU6k	[v]	Wörter	v9r|t6
[@]	Analyse	a|na|ly:|z@	[z]	reisen	raI|zn
[6]	Bauer	baU|6	[j]	ja	ja:
	Affrikaten			**Liquide**	
[pf]	Pfalz	pfalts	[l]	einheitlich	aIn|haIt|lIC
[ts]	Zug	tsu:k	[r]	frisch	frIS

A.2 Vokabular SLACC

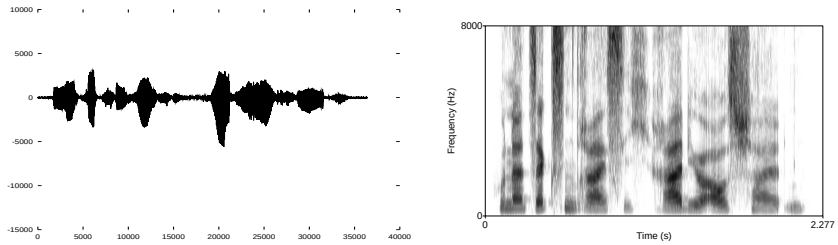
Das Vokabular des Spracherkenners zur Ansteuerung des Navigationssystems,der Klimaanlange, des Radios, des Handys und eines einfachen Taschenrechners.

Ä	6	Juni	Telefon	letzter
Ö	7	Juno	Telefonbuch	mache
Ü	8	Kalender	Telefonmenu	mal
A	9	Klammer_auf	Telefonnummer	merken
B	10	Klammer_zu	Tonwahl	minus
C	11	Klimaanlage	Uhr	mit
D	...	Komma	Verkehrsfunk	mittag
E	250	Krankenhaus	Wahlwiederholung	morgen
F	1ten	Leerzeichen	Waschanlage	multipliziert
G	2ten	März	Werkstatt	nächste
H	3ten	Mai	Zentrum	nächsten
I	...	Marktplatz	ab	nächster
J	31ten	Mittwoch	abbrechen	nach
K	Antenne	Montag	abend	nachmittag
L	Apotheke	Navigation	abrufen	nein
M	April	November	am	ok
N	August	Nummer	ausschalten	plus
O	Bahnhof	Oktober	bin	runter
P	Bank	Parkplatz	buchstabiere	spät
Q	Dezember	Polizei	das	speichern
R	Dienstag	Prozent	dem	stop
S	Display	Punkt	der	um
S-M-S	Donnerstag	Radio	die	versenden
S-W-R	Doppelpunkt	Raststätte	durch	von
T	Eins_live	Raute	eingeben	vorlesen
U	Februar	Restaurant	einschalten	vormittag
V	Feuerwehr	Samstag	es	wählen
W	Flughafen	Sender	geteilt	wärmer
W-D-R	Fragezeichen	September	hause	was
X	Freitag	Sonntag	heute	weiter
Y	Frequenz	Sparkasse	hoch	wie
Z	Geheimnummer	Spielplatz	ich	wieviel
0	Grad	Sportplatz	ist	wo
1	Handy	Statusreport	ja	zum
2	Hilfe	Sternchen	kälter	zur
3	Hotel	Systemstart	löschen	zurück
4	Januar	Tankstelle	lauter	
5	Juli	Taschenrechner	leiser	

A.3 Aufnahmen von SLACC

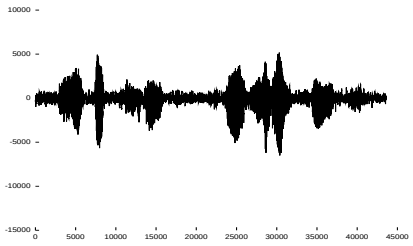
Die folgenden Abbildungen zeigen am Beispiel einer Äußerung die unterschiedlichen Signalqualitäten im *SLACC* Korpus. Die Aufnahme *"1 3 6 Radio ausschalten"* wurde von einer Sprecherin in einem fahrenden *Opel Corsa* gemacht. Von jedem Probanden wurde ein Set von 300 Phrasen aus der Domäne "Sprachliche Ansteuerung nicht sicherheitsrelevanter Funktionen" einmal bei langsamer und einmal bei hoher Geschwindigkeit vorgetragen. Die Aufnahmen geschahen parallel mit einem Nahbesprechungsmikrofon vom Typ *Sennheiser* und einem von der Automobilindustrie verwendeten, am Spiegel angebrachten *VDA* Mikrofon. Dadurch existieren von jedem Satz insgesamt vier Aufnahmen, wobei die Aufnahmen für eine Geschwindigkeit, die hier als *Stadtverkehr* bzw. *Landstraße* bezeichnet werden, jeweils gleichzeitig stattfanden. Die Signale sind in Abbildung A.2 auf der linken Seite dargestellt. Zum Vergleich ist dieselbe Äußerung noch einmal mit dem Nahbesprechungsmikrofon in ruhiger Laborumgebung aufgezeichnet worden, s. Abb. A.1. Beim Vergleich der Mikrofone ist deutlich die bessere Trennung von Sprachsignal und Hintergrundgeräusch beim *Sennheiser* Mikrofon zu erkennen. Desweiteren sieht man den höheren Energiegehalt (stärkere Amplitude) der *VDA* Aufnahmen im Allgemeinen, sowie den Anstieg des Gesamtsignals bei höheren Geschwindigkeiten bei beiden Mikrofonen. Interessant ist auch der Energiezuwachs beim Sprachsignal, ein Teilaspekt des *Lombard*-Effektes [42]: Um gegen den Hintergrundlärm anzukommen, sprachen die Personen lauter.

Durch die Vorverarbeitung mit der Präemphase, dargestellt auf der rechten Seite, findet eine Normierung der Signale statt. Beim Blick auf die Spektogramme der Signale (Abb.A.3) fällt auf, dass das Nahbesprechungsmikrofon auch die hohen Frequenzen noch gut auflösen kann, während das *VDA* Mikrofon hauptsächlich unterhalb von 6 kHz bleibt. Die Spektogramme nach der Präemphase zeigen dagegen für alle Signale eine deutlicheres Bild der hohen Frequenzen und eine vollständige Unterdrückung von stark niederfrequenten Schwingungen.

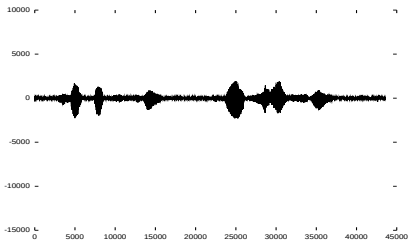


(a) Büroumgebung, Sennheiser (b) Spektogramm

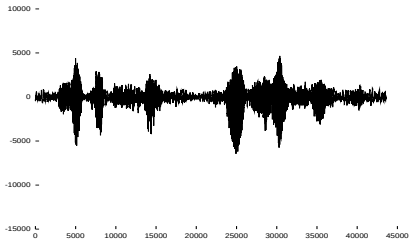
Abbildung A.1: Äußerung "*136 Radio ausschalten*" in Laborumgebung.



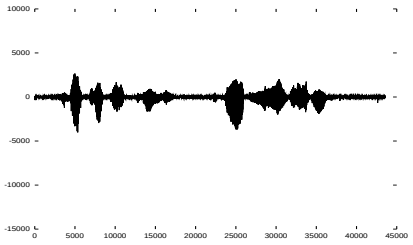
(a) Stadtverkehr, VDA



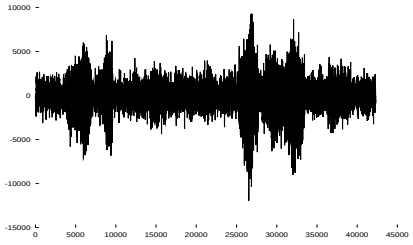
(b) nach Prämphase



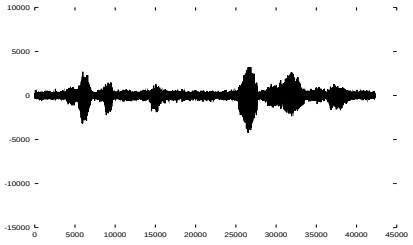
(c) Stadtverkehr, Sennheiser



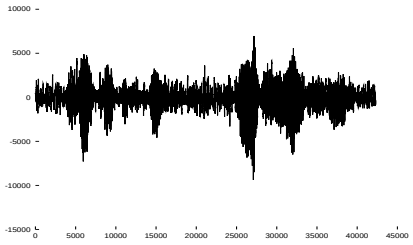
(d) nach Prämphase



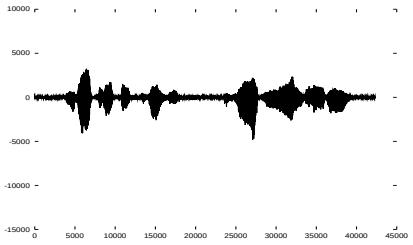
(e) Landstraße, VDA



(f) nach Prämphase

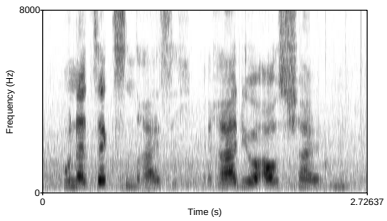


(g) Landstraße, Sennheiser

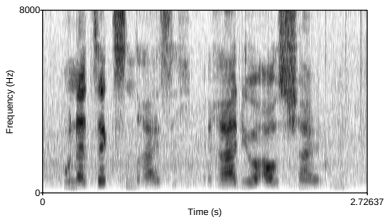


(h) nach Prämphase

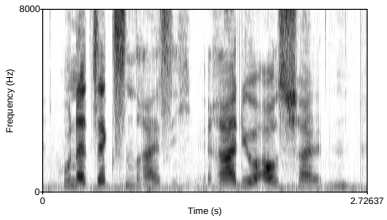
Abbildung A.2: Aufnahmen der Äußerung "*136 Radio ausschalten*"



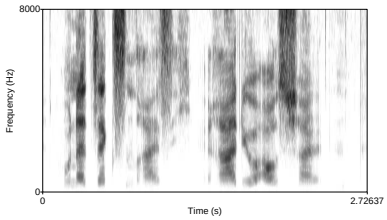
(a) Stadtverkehr, VDA



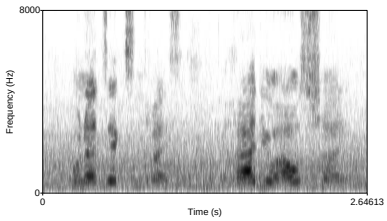
(b) nach Prämphase



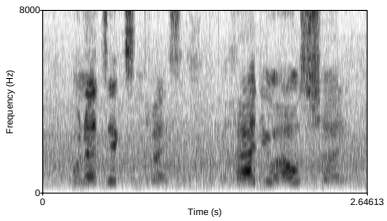
(c) Stadtverkehr, Sennheiser



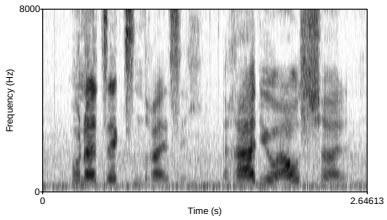
(d) nach Prämphase



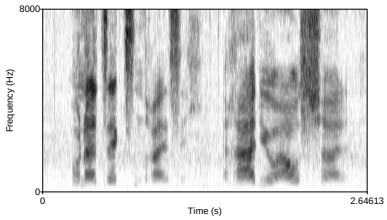
(e) Landstraße, VDA



(f) nach Prämphase



(g) Landstraße, Sennheiser



(h) nach Prämphase

Abbildung A.3: Spektogramme der Äußerung "*136 Radio ausschalten*"

A.4 Automat zum Sprachverstehen

Darstellung eines aus Gründen der Übersichtlichkeit vereinfachten[1] Automaten zur Verarbeitung von Äußerungen für den realsisierten Erkenner. Als Startzustand fungiert Knoten **SB**. Der Automat wird durch Ausgaben des Erkenners weitergeschaltet: kommt ein im aktuellen Zustand gültiges Wort, das auch über eine Wortklasse definiert sein kann, wird die entsprechende Funktion ausgeführt und der Nachfolgezustand eingenommen. Neben den explizit aufgeführten Wörter gibt es noch ein Metakommando (*abbrechen*) das jederzeit verstanden wird, die aktuelle Traversierung unterbricht und das System wieder in den Startzustand zurückversetzt.

Als Wortgruppen existieren:

`CLASS-ZAHL`	alle Zahlen
`CLASS-STATION`	alle Stationsnamen (WDR,Bayern,Hessen,HR,...)
`CLASS-STATIONI`	1,2,3,4 (Stationsindizes)
`CLASS-FREQUENZ`	Zahlen z mit $88 \leq z \leq 108$
`CLASS-NACHKOMMA`	Zahlen z mit $0 \leq z \leq 9$ oder $10 \leq z \leq 90$, $z \bmod 5 = 0$

Desweiteren gibt es noch die Wortklasse `STILLE`, deren Ereignis von von einem Äußerungsende bzw. Hesitationen ausgelöst wird.

Die Funktionen, die verschiedene Geräte (Handy, Radio, Sicherheitssperre) ansteuern, haben Zugriff auf interne Statusvariablen wie einen Zahlstring, der zum Akkumulieren von einzeln genannten Telefonnummern benötigt wird, oder aktuelle Statusinformation des Radios (Sender, Lautstärke, Aktiverungszustand).

Zu den Funktionen gehören:

no_op	keine Aktion
store	Anhängen der zuletzt genannten Zahl an den Zahlstring
pin_check	Vergleich des Zahlstrings mit dem Geheimcode und entsprechende Behandlung für Übereinstimmung und Differenz
show_number	Textuelle/Akustische Ausgabe des momentanen Zahlstrings, anhängen eines Blockseparators an den Zahlstring
kill_last_number	Löschen des letzten Zahlteilstrings bis zum Blockseparator
rad_on	Radio einschalten
rad_off	Radio ausschalten
rad_inc	Lautstärke kontinuierlich erhöhen
rad_dec	Lautstärke kontinuierlich verringern
rad_status	Radiostatus ausgeben (& Lautstärkemanipulation beenden)
add_num	Addieren der letzten Zahl zum Zahlstring
seek_stat	Sender mit Sendernamen suchen
seek_freq	Sender über Frequenz suchen

[1] Die Vereinfachung betrifft das Auslassen der Navigationsansteuerung und des Buchstabiererkenners.

Der Automat (s. Abbildung 5.1 für eine Baumdarstellung) ist dann durch folgende Quadrupel definiert:

Zustandsname	*Wörter/Wortklassen*	*Funktion*	*Nachfolgezustand*
SB	Systemstart	no_op	SZ
SZ	Geheimnummer	no_op	GN1
SZ	Telefonnummer	no_op	TS
SZ	Radio	no_op	RS
GN1	`CLASS-ZAHL`	store	GN2
GN2	`CLASS-ZAHL`	store	GN3
GN3	`CLASS-ZAHL`	store	GN4
GN4	`CLASS-ZAHL`	store	GN5
GN5	`STILLE`	pin_check	SB
TS	`CLASS-ZAHL`	store	TN
TN	`CLASS-ZAHL`	store	TN
TN	`STILLE`	show_number	TC
TN	wählen	dial	ENDB
TC	löschen	kill_last_number	TC
TC	`CLASS-ZAHL`	store	TN
TC	wählen	dial	ENDB
RS	einschalten	rad_on	SB
RS	ausschalten	rad_off	SB
RS	lauter	rad_inc	RSH
RS	leiser	rad_dec	RSH
RS	Sender	no_op	RSE
RS	Frequenz	no_op	RSF
RSH	okay	rad_status	SB
RSE	`CLASS-STATION`	store	RSEN
RSEN	`CLASS-STATIONI`	store	RSENC
RSENC	`STILLE`	seek_stat	SB
RSF	`CLASS-FREQUENZ`	store	RSFK
RSF	hundert	store	RSFL
RSF	einhundert	store	RSFL
RSFL	Komma	no_op	RSFN
RSFL	`CLASS-ZAHL`	add_num	RSFK
RSFK	Komma	no_op	RSFN
RSFN	`CLASS-NACHKOMMA`	store	RSFNC
RSFNC	`STILLE`	seek_freq	SB
END	`STILLE`	no_op	SB
ENDB	`STILLE`	no_op	SB

A.5 Levenshtein, Algorithmus

Algorithmus zur Berechnung des Levenshtein Abstandes

Initialisierung :

L(i,0) = i , i=0..n und L(0,j) = j , j=0..m

Iterierung :

Forall i=1..n

Forall j=1..m do

L(i,j) = min(L(i-1, j)+INS, L(i, j-1)+DEL, L(i-1, j-1) + d(r(i), t(j))*SUB)

mit

d(r(i), t(j)) = 0 , if r(i)==t(j)
1 , else

INS, DEL und SUB sind die Gewichte für Insertion, Deletion und Substitution (für Standardlevenshtein alle = 1).

A.6 Verwechslungsmatrix Buchstabenerkenner

Nach Evaluation des Buchstabenerkenners auf dem SLACC Korpus ergab sich für die 114 Buchstabensequenzen enthaltende Teststichprobe eine Wortakkuratheit von lediglich 54%. Ein Teil der Fehler fanden bei hohen Geschwindigkeiten (=niedriger Signalrauschabstand) statt und bestanden in Detektion von Buchstaben in Fahrgeräuschen. Die um 20 betroffene Äußerungen bereinigte Teststichprobe erreichte eine WA von 64%. Die verbleibenden Fehler bestanden größtenteils in Verwechslungen von phonetischen ähnlichen ausgesprochenen Buchstaben, etwa 'H' und 'K', 'B' und 'W' aber auch Zeichen aus den gleichen Lautgruppen, wie die Frikative 'S' und 'F'.

Die folgende Matrix zeigt alle aufgetretenen Verwechslungen. In den Zeilen stehen die erkannten Zeichen, in den Spalten die geäußerten (Leerzeichen = _). Da innerhalb der Teststichprobe kein 'Q' geäußert wurde, ist in der Matrix die entsprechende Zeile leer. Beim Einsatz dieser Matrix für das ermitteln von Verwechslungspartner für den Lexikonabgleich wurde die Zeile mit einer 50 für 'Q' modifiziert, da ansonsten für diesen Buchstaben keine Erkennung mehr möglich gewesen wäre. Ähnlich wurde für den Buchstaben 'X' verfahren.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ü
a	26		1					56		1	65		2	1				10	1	1		5							
b		74		11			4		10							2	1						9						
c			62								1							2		2			1						
d		17		14			45		2					1		22				4			2						1
e		34		19	26		148	1	59					1		43		4	1	2			8						
f						38					1	3						4									1		
g		1				1	74		4							16				1									1
h	5							56	1		58	1						3				5							
i					3		47	1	58			1	2					1											
j										2																			
k								4	1		33			1								1							
l		2				1		1	1			149	1										1				2		
m							1				1		90	3															
n								1			1	2	131	111	1	1	1		1										
o															55		16				11	2							
p		1		1			4									16				2			3						
q																													
r		1	1			1	1	7		2	13	2					1	189	1			3					1	1	
s			1			18					1								113	1									1
t			8				2					1		1		10		1		75									
u						1						1				1	72	1	1		30		1					1	
v		1					1															11							
w		5		3	2		1									1							6						
x																							1						
y																									3				
z			1																							21			
ä																											1		
ö																												11	
ü									8			1		2			1												18

Bestimmung einer Verwechslungmatrix von Buchstabiersymbolen basierend auf phonetischer Ähnlichkeit

Der Vergleich zweier Phonemsequenzen auf phonetische Ähnlichkeit wurde phonemweise durchgeführt, also nicht als Buchstabenzeichen *a, b, etc.* sondern als Transkripte `/a:/, /b/e:/, /t/s/e:/ ...`. Dabei wurden nur Konsonanten mit Konsonanten und Vokale mit Vokalen verglichen. Der Vergleich eines Vokals mit einem Konsonanten gibt automatisch einen Ähnlichkeitswert von 0. Die Gesamtähnlichkeit einer Sequenz wird über die einzelnen Phonemähnlichkeiten gemittelt. Die Abbildung der Phoneme aufeinander wird dabei mit hilfe des Levensthein-Abstandes bestimmt.

Beispiel: Berechnung der Gesamtähnlichkeit

$$\ddot{A}hnlichkeit(te:, pe:) = \frac{\ddot{A}hnlichkeit(t,p) + \ddot{A}hnlichkeit(e:, e:)}{2}$$

$$= \frac{0.88 + 1}{2} = 0.944$$

Berechnung der Einzelähnlichkeiten

$$\ddot{A}hnlichkeit(e:, e:) = 1$$

$$\ddot{A}hnlichkeit(t,p) = \frac{Artikulationsort(t,p) + Artikulationsart(t,p) + Stimmhaft(t,p)}{3}$$

$$= \frac{\frac{2}{3} + 1 + 1}{3} = 0.88$$

Vergleich zweier Konsonanten nach Ort, Art und Stimmhaftigkeit

$$Artikulationsort(t,p) = Artikulationsort(alveolar, bilabial) = \frac{2}{3}$$

$$Artikulationsart(t,p) = Artikkulationsart(Plosiv, Plosiv) = 1$$

$$Stimmhaft(t,p) = Stimmhaft(stimmlos, stimmlos) = 1$$

Konsonanten

Zum Vergleich zweier Konsonanten werden die drei Dimensionen Artikulationsart, -ort und Stimmhaftigkeit verwendet. Die Ähnlichkeit wird dabei folgendermaßen verrechnet:

$$\ddot{A}hnlichkeit(C1, C2) = \frac{Ort(C1, C2) + Art(C1, C2) + Stimmhaft(C1, C2)}{3}$$

Ähnlichkeit der Artikulationsorte (C1, C2):

	labial labiodental	dental alveolar postalveolar	palatal velar uvular	pharyngal glottal
labial..	1	$\frac{2}{3}$	$\frac{1}{3}$	0
dental..	$\frac{2}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$
palatal..	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{2}{3}$
pharyngal...	0	$\frac{1}{3}$	$\frac{2}{3}$	1

Ähnlichkeit der Artikulationsarten (C1, C2):

	Plosiv	Nasal	Frikativ	Approximant
Plosiv	1	0	0	0
Nasal	0	1	0	0
Frikativ	0	0	1	0
Approximant	0	0	0	1

Ähnlichkeit der Stimmhaftigkeit (C1, C2):

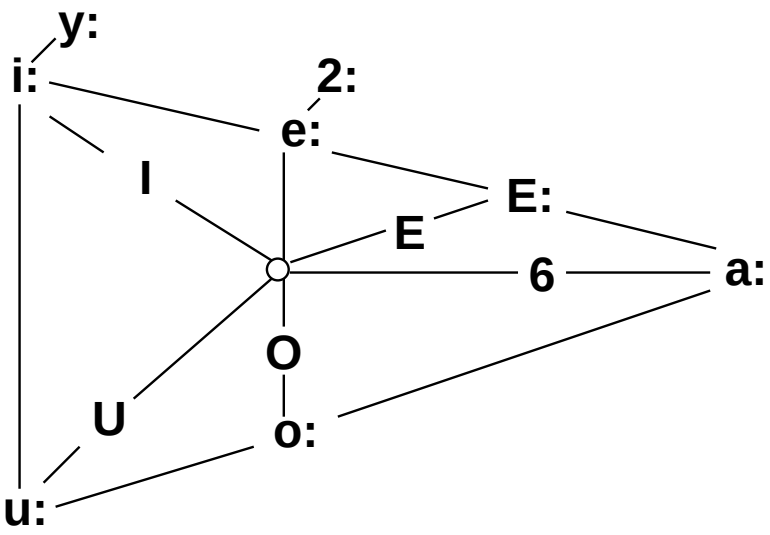
	stimmhaft	stimmlos
stimmhaft	1	0
stimmlos	0	1

Vokale

Die Ähnlichkeit von Vokalen wird über ihre Nähe im Vokaldreieck zueinander bestimmt. Pro Kante werden Kosten von 1 berechnet. Diese Kosten werden durch 4 geteilt und vom Maximalwert 1 abgezogen:

$$\ddot{A}hnlichkeit(V1, V2) = 1 - \frac{AnzahlKanten(V1, V2)}{4}$$

Beim Vergleich zweier gerundeter Vokale wie z.B. (u:, y:) wird die Kante zwischen i: und y: nicht mitgezählt. Das ist noch etwas unausgereift, da wegen der Gerundetheit eigentlich die Distanz (u:, i:) weiter sein müßte als (u:, y:).



Beispiele: Ähnlichkeit(V1, V2)

$$\ddot{A}hnlichkeit(a:, e:) = 1 - \frac{2}{4} = 0.5$$

$$\ddot{A}hnlichkeit(u:, y:) = 1 - \frac{1}{4} = 0.75$$

A.7 Umsetzungsregeln

Zur Erstellung von Verwechslungslisten für einzelne Städte wurden durch verschiedene Umsetzungen auf der orthographischen Darstellung vereinfachte Darstellungen der Namen generiert. Alle Namen mit jeweils identischer Reduktionsform wurden in einer eigenen Liste zusammengefaßt.

ä	→	e
ß	→	s
ö	→	oe
oo	→	o
ah	→	a, wenn kein Vokal auf ah folgt
eh	→	e, wenn kein Vokal auf eh folgt
y	→	i
ai	→	ei
ck	→	k
c	→	k, wenn kein h auf dem c folgt
v	→	f
mm	→	m
pp	→	p
nn	→	n
ll	→	l
ff	→	f
gg	→	g
rh	→	r
th	→	t
dt	→	t
d	→	t
b	→	p
g	→	k
Alle Konsonanten am Wortanfang	→	/

Beispiele für einige Verwechslungslisten und die jeweiligen reduzierten Formen:

Ahlendorf, Alendorf, Allendorf, Ballendorf	⇒	alentorf
Cobstädt, Lobstädt, Loopstedt, Obstädt	⇒	opstet
Altendeich, Altenteich	⇒	altenteich
Creussen, Greussen	⇒	kreussen
Scherneck, Schernegg	⇒	schernek
Heyerhöfen, Maierhöfen, Meyerhöfen	⇒	eieroefen
Krumbeck, Krummbek	⇒	krumpek