"Robuste Spracherkennung mittels der italienischen SPEECON-Datenbank"

Dokumentation

Wissenschaftliches Projektseminar
Master Informations- und Kommunikationstechnik

Frank Hackemesser (632775)

Betreuer: Prof. Dr.-Ing. Hans-Günter Hirsch

08.01.2009

Inhaltsverzeichnis

Zusammenfassung

Im Mittelpunkt dieses Projektseminars steht die Vorbereitung und Anwendung der italienischen SPEECON-Audiodatenbank für Spracherkennungsexperimente in ungestörten und gestörten akustischen Umgebungen. Zunächst wurden Änderungen an der Datenbankstruktur durchgeführt, um die Nutzbarkeit der Audiodaten zu verbessern. Unter Verwendung der HTK-Programmsammlung wurden Hidden-Markov-Modelle für alle isolierten Wörter aus ungestörten Audiodaten erstellt. Unter Verwendung dieser Modelle wurden mehrere Erkennungsexperimente mit Sprachaufnahmen aus unterschiedlichen Aufnahmeumgebungen durchgeführt und ausgewertet.

Das Dokument gliedert sich wie folgt: Das erste Kapitel enthält die zum Verständnis des Projekts notwendigen Grundlagen, die Zielsetzung des SPEECON Projekts sowie Aufbau und Inhalte der Datenbank. Außerdem werden die an der Datenbank durchgeführten Veränderungen aufgelistet und erläutert. Im zweiten Kapitel werden die Phasen des HMM-Trainingsprozesses dargestellt und alle in diesem Kontext entwickelten MATLAB-Programme dokumentiert. Das dritte Kapitel dient der Definition der durchgeführten Erkennungsexperimente sowie der Ergebnisaufbereitung und -analyse.

Kapitel 1

SPEECON Datenbank

Das folgende Kapitel enthält eine Zusammenfassung der auf dieses Projekt bezogenen Grundlagen. Nach einer kurzen Beschreibung des SPEECON-Projekts wird genauer auf Aufbau, Inhalte und Formatierung der italienischen SPEECON-Sprachdatenbank eingegangen. Im Anschluss werden die an der Datenbank durchgeführten Veränderungen sowie die Funktionsweise des hierfür verwendeten Programms dargestellt und erläutert.

1.1 Das SPEECON Projekt

Das mit dem Akronym SPEECON ("SPEEch driven interfaces for CONsumer applications") bezeichnete Projekt stellt eine Kooperation mehrerer internationaler Firmen dar, welche sich mit dem Ziel, allen Partnern eine effiziente Entwicklung und Produktion von interaktiven Sprachschnittstellen für Verbraucheranwendungen zu ermöglichen, zu einem Konsortium zusammengeschlossen haben. Als Grundlage dienen hierzu die im Verlaufe des Projektes geschaffenen Audiodatenbanken, welche 18 Sprachräume abdecken und deren Aufnahmen unter verschiedensten akustischen Rahmenbedingungen durchgeführt worden sind. Im Rahmen dieses Projektseminars wurde die italienische SPEECON-Sprachdatenbank verwendet, welche von der Firma Teleca erstellt wurde.

1.2 Inhalt und Formate

Die SPEECON Audiodatenbank enthält eine Mischung aus spontaner und gelesener Sprache, isolierten Wörtern sowie kontinuierlichen Äußerungen. Die individuellen Inhalte wurden so definiert, dass eine möglichst weit gefächerte Auswahl an Verbraucheranwendungen (Mobiltelefone, PDAs, Spielzeug, usw.) berücksichtigt werden kann. Gemäß Spezifikation wurden die Sprachdaten in drei inhaltliche Kategorien eingeordnet, welche als *core words*, *read speech* und *spontaneous speech* bezeichnet werden. Unter *core words* werden hierbei allgemein gebräuchliche Wörter und Phrasen sowie anwendungsspezifische Kommandowörter und -äußerungen eingeordnet, während die beiden Kategorien *read speech* und *spontaneous speech* hauptsächlich längere, zusammenhängende Sätze beinhalten. Ein Teil des Dateinamens jeder einzelnen Audiodatei besteht aus einem zweistelligen Code, welcher einen direkten Rückschluss auf den sprachlichen Inhalt ermöglicht. Die entsprechenden Codes können aus Tabelle 1.1 auf der nächsten Seite entnommen werden.

Zusätzlich zu den Sprachressourcen enthält die SPEECON-Datenbank mehrere Dateien, welche detaillierte Informationen über die Aufnahmebedingungen der Sessions und der Eigenschaften der individuellen Dateien enthalten. Relevante Dateien sind hierbei *RecCondition.tbl*, *test.dbml* und *train.dbml*. Die erstgenannte Datei enthält eine nach Session-Identifikationsnummer sortierte Liste der jeweils verwendeten Mikrofontypen und -anordnungen sowie Aufnahmeort und -konditionen. Die beiden *dbml*-Dateien stellen dagegen eine zweigeteilte Liste aller Sprachdateien mit ihren zugehörigen Informationen über Sprecher, Session, gesprochen Inhalt usw. dar. Die genaue zeilenweise Formatierung der Dateien werden in den Tabellen 1.2 und 1.3 auf Seite 4 dargestellt.

Während der Aufnahmen aufgetretene Fehler werden mit speziellen Fehlercodewörtern und -symbolen in der Label-Informationsspalte der *dbml*-Dateien gekennzeichnet. Tabelle 1.4 auf Seite 4 zeigt eine Liste der verwendeten Codewörter. Je nach Fehlertyp und Verwendungszweck muss die Relevanz des Fehlers unterschiedlich bewertet werden, da sich beispielsweise beim Training eines HMM-Modells ein unvollständiges Wort weitaus kritischer auf das Ergebnis auswirkt, als das Lippengeräusch eines Sprechers. Das Codewort *w_sil* ist hingegen kein definierter Standard, sondern zur Kennzeichnung von Stilleabschnitten innerhalb der Sprachdateien definiert.

Kategorie	Element	Inhalt
Calibration data		
"_"	01-06	noise recordings
N	01	The "silence word" recording
Free spontaneous items		
F	1-30	5 minutes of free spontaneous context items
Elicited spontaneous items		
E	D1 – D3	3 elicited dates
E	T1 – T2	2 elicited times
E	P1– P3	3 elicited proper names
E	C1 – C2	2 elicited city name
E	L1	1 elicited letter sequence
E	Q1 – Q2	2 elicited answers to questions
E	N1 – N3	3 elicited telephone numbers
E	O1	1 elicited language
Read speech		
S	01 – 30	30 phonetically rich sentences
W	01 – 05	5 phonetically rich words
Core words (read), 31 general, 208 application specific		
C	I1 – I4	4 isolated digits
C	B1	1 isolated digit sequence
C	C1 – C4	4 connected digit sequences
C	E1	1 telephone number
C	N1 – N3	3 natural numbers
C	M1	1 money amount
C	T1 – T2	2 time phrases
C	D1 – D3	3 dates
C	L1 – L3	3 letter sequences
C	P1	1 proper name
C	O1 – O2	2 city or street names
C	Q1 – Q2	2 questions
C	K1 – K2	2 special keyboard characters
C	W1	1 Web address
C	W2	1 email address
Y	01 – 99	core word synonyms
1	01 – 85	Basic IVR commands
2	01 – 40	Directory navigation
3	01 – 22	Editing
4	01 – 57	Output control
5	01 – 69	Messaging & Internet browsing
6	01 – 33	Organiser functions
7	01 – 39	Routing
8	01 – 12	Automotive
9	01 – 95	Audio & Video

Tabelle 1.1: Inhaltliche Kodierung der Audiodateien

Spalte	Beschreibung
Session ID	Relativer Pfad zur Audiodatei
Microphone Position Channel 0	Mikrofonposition Kanal 0
Microphone Position Channel 1	Mikrofonposition Kanal 1
Microphone Position Channel 2	Mikrofonposition Kanal 2
Microphone Position Channel 3	Mikrofonposition Kanal 3
Microphone Type Channel 0	Mikrofontyp Kanal 0
Microphone Type Channel 1	Mikrofontyp Kanal 1
Microphone Type Channel 2	Mikrofontyp Kanal 2
Microphone Type Channel 3	Mikrofontyp Kanal 3
Environment	Aufnahmeumgebung
Place	Genauer Aufnahmeort
Pos	Position relativ zu einer Wand (nah/fern)
Size	Grundfläche der Aufnahmeumgebung in qm
Audio	Hintergrundmusik an/aus

Tabelle 1.2: Zeilenformat der Datei *RecCondition.tbl*

Spalte	Beschreibung
Path	Relativer Pfad zur Audiodatei
Speaker	Identifikationsnummer Sprecher
Gender	Geschlecht
Age	Alter
Unique Number	Einzigartige Identifikationsnummer
Content	Inhaltscode
Label	Enthaltende Äußerungen und Fehlermarkierungen

Tabelle 1.3: Zeilenformat der Dateien *test.dbml* und *train.dbml*

Codewort	Bedeutung
g_fil	Füllwörter, Gedankenpausen
g_spk	Sprecher (Atem, Lippengeräusch, Husten, ...)
g_int	Zeitweise Störung (Tür, Klingel, Uhr, ...)
g_sta	Konstante Störung (Rauschen, Straßenverkehr, Menschen, ...)
*<Wort>	Unvollständiges Wort (Anfang fehlerhaft)
<Wort>*	Unvollständiges Wort (Ende fehlerhaft)
**	Nicht erkennbar
/<Buchstabe>/	Phonetische Form eines Buchstabens
w_sil	Stilleabschnitte vor, nach oder zwischen Äußerungen

Tabelle 1.4: Spezielle Codewörter für Aufnahmen

1.3 Neustrukturierung der Datenbank

Bei der vorliegenden Ordnerstruktur der Sprachdatenbank handelt es sich um eine hinsichtlich einer automatisierten Aufnahme optimierte Art des Aufbaus. Die Daten wurden in der Form */<database>/<block>/<session>* gespeichert, wobei *<block>* einer zweistelligen Zahl zwischen 00 und 99 und *<session>* einer aus der Blocknummer sowie einer Zahl zwischen 0 bis 9 zusammengesetzten dreistelligen Zahl besteht. Im Hinblick auf eine praktische Weiterverwendung der Daten für beispielsweise Sprachexperimente ist diese Struktur jedoch nicht effizient. Um die Arbeit mit den großen Datenmengen deutlich zu erleichtern, wurde eine neue Ordnerstruktur angelegt, welche zusätzlich zu den über die Dateinamen kodierten Informationen schnelle Rückschlüsse auf den Inhalt der einzelnen Audiodateien erlaubt. Diese neue Struktur ist in Abbildung 1.1 auf der nächsten Seite dargestellt.

Die beiden Hauptordner *Home* und *Mobile* gliedern sich in jeweils mehrere Unterszenarien, welche im Hinblick auf zukünftige Verwendung der Datenbank für Trainings- und Erkennungsexperimente eine schnelle Auswahl der geeigneten Dateien anhand der Aufnahmeumgebung erlauben. Die zwei folgenden Ordnerebenen separieren die Audiodateien weiterhin nach Geschlecht sowie Mikrofonkanal. Soll eine noch feinere Filterung der Daten vorgenommen werden, so wäre dies beispielsweise durch die Position des Sprechers relativ zu einer Wand oder die Grundfläche des Aufnahmeraumes zu realisieren. Aufgrund der in Hinblick auf zukünftige Experimente eher untergeordneten Rolle dieser Parameter wird jedoch bewusst eine auf eine detailliertere Ordnerstruktur verzichtet.

Um den Konvertierungsprozess der Datenbank automatisch durchzuführen, wurde das MATLAB-Skript *sort_speecon_database.m* verwendet, dessen Parameter und Flussdiagramm in Abbildung 1.2 und 1.3 auf Seite 7 dargestellt sind. Prinzipiell liest das Skript die der SPEECON-Datenbank beiliegende Informationsdatei *RecCondition.tbl* aus, um mittels der je nach Session individuellen Aufnahmebedingungen die Position der Audiodateien innerhalb der neuen Ordnerhierarchie zu bestimmen. Aufgrund der sehr hohen Anzahl an Dateien wird zur Identifikation einzelner Dateien während des Kopierprozesses eine vorgefertigte Dateiliste verwendet, welche sich aus der inhaltlichen Verschmelzung der Dateien *test.dbml* und *train.dbml* ergibt. Bei *copyfiles = 1* werden die Dateien in das Zielverzeichnis *new_folder* der neuen Datenbank kopiert, während bei *removeold = 1* am Ziel bereits vorhandene Daten im Vorfeld gelöscht werden.

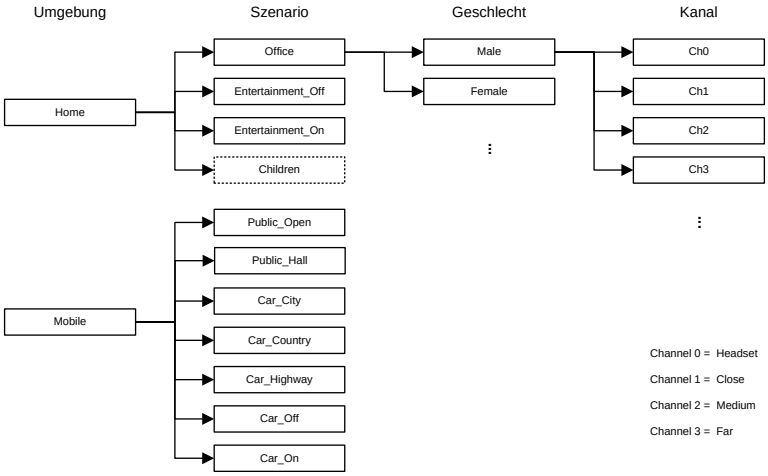

Abbildung 1.1: Ordnerhierarchie der Datenbank

Ist der letzte Parameter *lendian* gesetzt, werden die Audiodateien während des Kopiervorgangs im LE-Format abgespeichert.

Parallel zu diesem Prozess wird die Datei *filelog.txt* erstellt, welche eine aktualisierte Auflistung aller Dateipfade inklusive Informationen über deren Aufnahmeeigenschaften und Inhalte enthält. Diese Datei wird von den übrigen erstellten Skripten als Basisliste für die Suche nach Dateien innerhalb der Datenbank verwendet und muss somit innerhalb des Hauptordners verbleiben. Die zeilenweise Formatierung der Datei entspricht dem der beiden originalen **.dbml* Dateien.

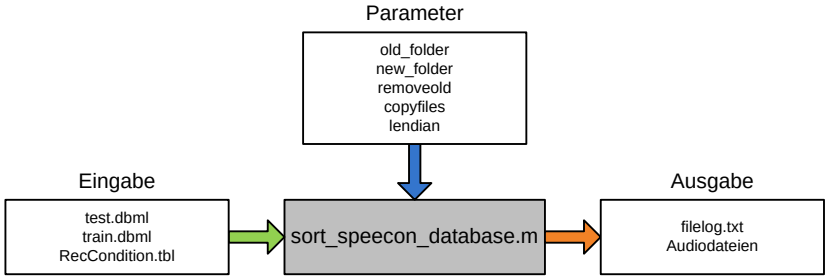

Abbildung 1.2: Parameter *sort_speecon_database.m*

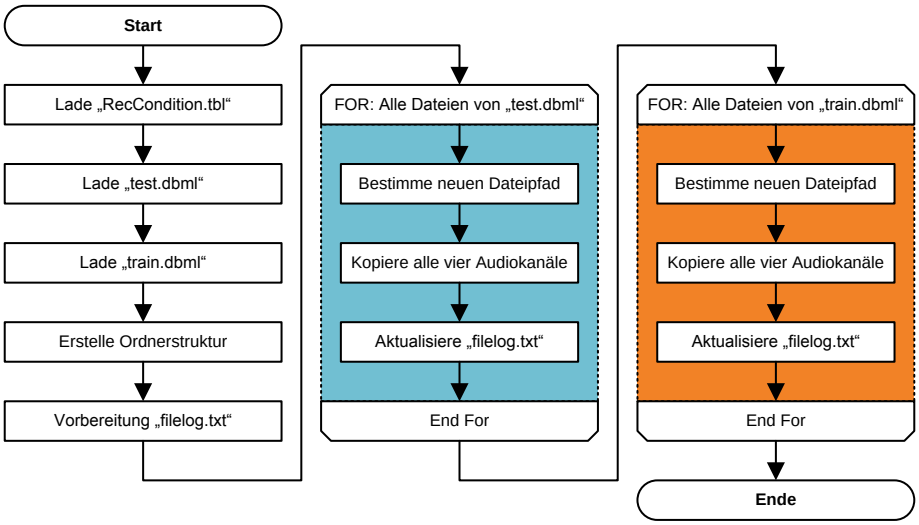

Abbildung 1.3: Flussdiagramm *sort_speecon_database.m*

Kapitel 2

Datenaufbereitung und HMM-Training

Das folgende Kapitel enthält eine detaillierte Erläuterung der Durchführung sowie der vorangehenden Datenvorbereitung des HMM-Trainingsprozesses. Anhand von Diagrammen werden die Ein- und Ausgabeparameter sowie die genaue Struktur der zu diesem Zweck entwickelten MATLAB-Skripte dargestellt.

2.1 Übersicht

Bevor mit dem Training der HMM-Dateien begonnen werden kann, müssen zunächst in mehreren Teilschritten verschiedene für den Trainingsprozess relevante Daten generiert bzw. aus der SPEECON-Datenbank ermittelt werden. Die zu diesem Zweck geschriebenen MATLAB-Skripte greifen hierzu teilweise auf bereits vorhandene HMM-Trainingsprogramme des HTK-Programmpakets (Hidden markov Model Toolkit) sowie zwei Programme von Prof. Dr.-Ing. Hirsch zurück. In Abbildung 2.1 auf der nächsten Seite ist eine Übersicht des vollständigen Prozesses dargestellt. Zu Beginn werden zunächst mit dem Skript *create_timelabels.m* Labeldateien erstellt, welche die in der jeweiligen Audiodatei gesprochenen Äußerungen sowie übrige Stillephasen samt der zugehörigen Zeitinformationen enthalten. Im nächsten Schritt werden für alle isolierten Wörter mittels *create_listfiles.m* eine Listendatei entsprechenden Namens erstellt, welche die Pfade aller Dateien enthält, in denen das Worte auftritt. Das vierte

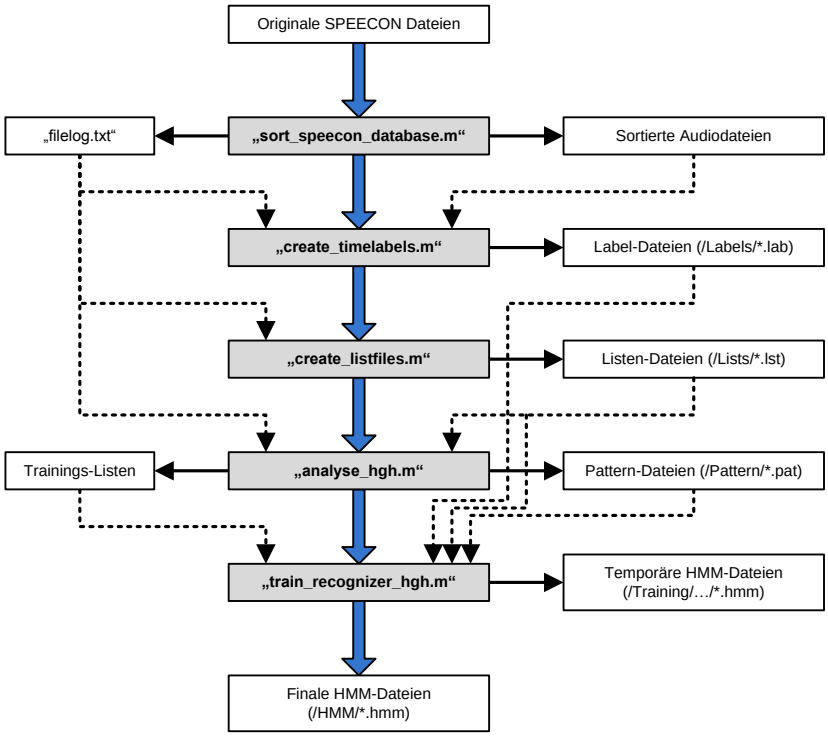

Abbildung 2.1: Übersicht des Trainingsprozesses

Skript *analyse_hgh.m* extrahiert die Eigenschaftsvektoren einer jeden Audiodatei und speichert diese als so genannte “Pattern”-Dateien ab. Zusätzlich werden hierbei Trainingslisten erstellt, die nur Äußerungen enthalten, welche öfter als 3 bzw. 10 mal auftreten. Im letzten Schritt wird nun das eigentliche Training der HMM-Dateien durchgeführt, welches sich aus mehreren Stufen zusammensetzt. Hierzu verwendet das Skript *train_recognizer_hgh.m* drei Kommandozeilenprogramme des HTK-Paketes. Die temporären HMM-Dateien werden hierbei nach jedem Prozessschritt separat im Ordner *„/Training/...”* und die finalen Modelle in *„/HMM/...”* abgespeichert. Nach dem Abschluss dieses Prozesses können die Modelle nun für Erkennungsexperimente genutzt werden.

2.2 Labeln der Audiodateien

Für das Training der HMMs sind genaue Informationen über die zeitliche Lage von Äußerungen und Stillephasen innerhalb der Sprachdateien erforderlich, wel-

che in so genannten Labeldateien gespeichert sind. Diese Aufgabe wird durch das MATLAB-Skript *create_timelabels.m* unter Verwendung des externen Programms *create_label* durchgeführt. Hierzu generiert das Skript zunächst eine Basis-Labeldatei, in der die Reihenfolge der unterschiedlichen Wort- und Stillezustände mit den zugehörigen, noch unbekannten Zeitinformationen zeilenweise im Format *<Startzeit>_<Endzeit>_<Wort>* gespeichert sind. Beim anschließenden Aufruf von *create_label* wird auf Basis der erstellten Datei eine Analyse der Sprachdatei durchgeführt und die detektierten Zeitinformationen bei erfolgreicher Durchführung in der Labeldatei aktualisiert. Im Falle einer fehlerhaften Analyse wird zum Namen der Labeldatei "<Dateiname>_failure" hinzugefügt. Dies tritt beispielsweise dann auf, wenn die vorliegende Information über die Anzahl an Wörtern einer Sprachdatei nicht den tatsächlich aufgenommenen Wörtern entspricht. Dies ist jedoch lediglich innerhalb der Kategorie *spontaneous speech* aufgetreten, welche für das HMM-Training nicht relevant sind.

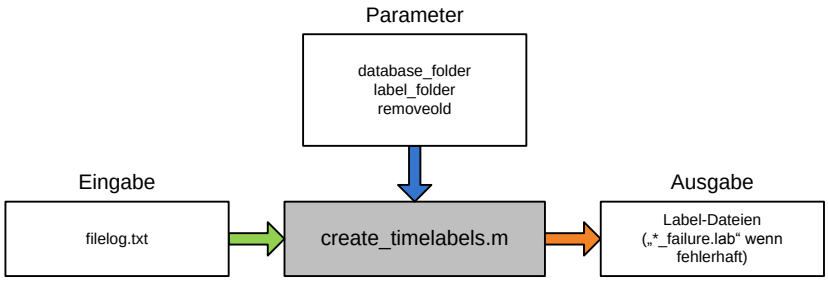

Abbildung 2.2: Parameter *create_timelabels.m*

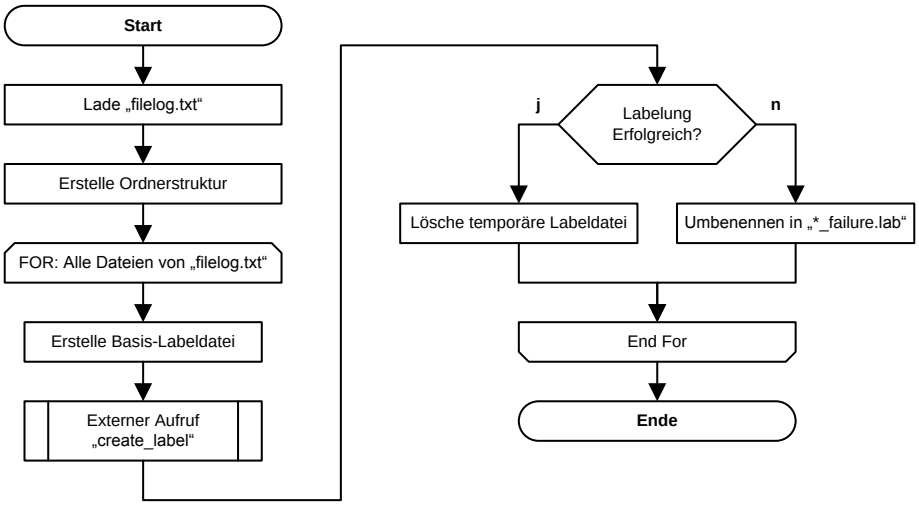

Abbildung 2.3: Flussdiagramm *create_timelabels.m*

2.3 Erstellung der Dateilisten

Für die Entwicklung des HMM-Modells eines bestimmten Wortes ist eine Dateiliste notwendig, welche die Pfade aller Audiodateien enthält, die für das Training verwendet werden können. Nutzbare Aufnahmen müssen in einer von akustischen Störeinflüssen wie beispielsweise Verzerrungen, Rauschen oder Hintergrundgeräuschen möglichst freien Umgebung aufgezeichnet worden sein („clean data”) sowie keine auf Sprecher oder Technik zurückzuführende Fehler beinhalten. Für die verwendete SPEECON-Datenbank wurden die Aufnahmeumgebungen *Office* und *Entertainment_off* als qualitativ passend ausgewählt. Innerhalb des Programms wird diese Filterung durch die erste FOR-Schleife realisiert, welche lediglich fehlerfreie und isolierte Wörter der korrekten Aufnahmeumgebung zu der entsprechenden Listendatei „*<Wortname>.lst*” im Ordner „*/clean/...*” hinzufügt. Zusätzlich werden über die zweite Schleife im Ordner „*/erroneous/...*” Listendateien erstellt, die Wörter aus allen Aufnahmeumgebungen enthalten und die weniger kritischen Fehlertypen *g_spk* und *g_int* ignorieren.

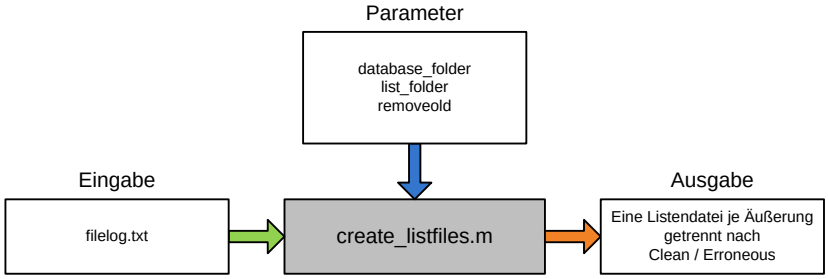

Abbildung 2.4: Parameter *create_listfiles.m*

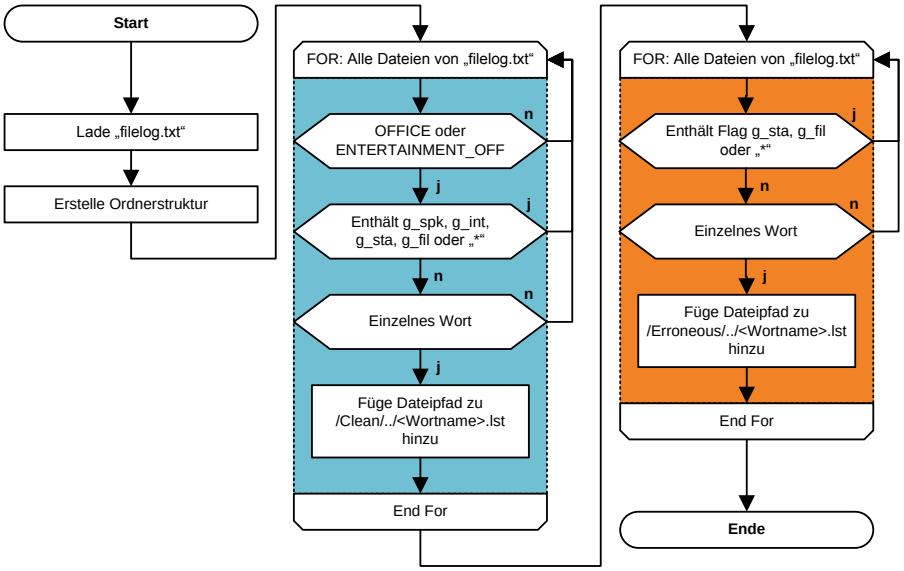

Abbildung 2.5: Flussdiagramm *create_listfiles.m*

2.4 Parameterextraktion

Im dritten vorbereitenden Schritt wird nun mittels des Skripts *analyse_hgh.m* die Extraktion der akustischen Parameter aus den Audiodateien durchgeführt. Nachdem das Programm die Listendateien der fehlerfreien Audiodateien in einer temporären Datei aufgelistet hat, wird diese an das externe Programm *rec_sim_hgh_ada_all* übergeben, welches die eigentliche Extraktion durchführt und die Parameterdatei im Ordner *pattern_folder* nach Geschlecht sepa-

riert abspeichert. Alle auf diese Weise erstellten Pattern-Dateien werden den Listendateien *"/<pattern_folder>/patterns_<Geschlecht>.lst"* hinzugefügt. Ist der Parameter *createwordlists = 1* werden zusätzlich vier Trainingslisten mit Wörtern erstellt, zu denen mindestens 3 bzw. 10 verschiedene, zum Training geeignete Sprachdateien vorliegen.

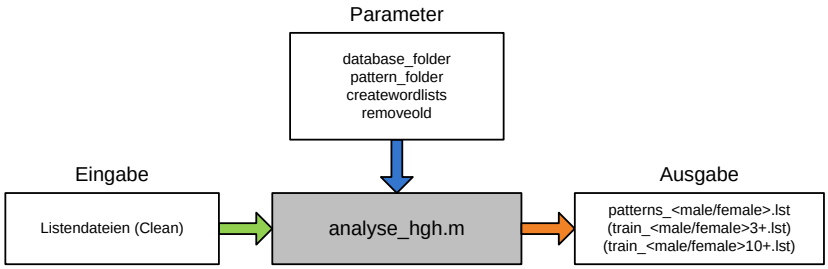

Abbildung 2.6: Parameter *analyse_hgh.m*

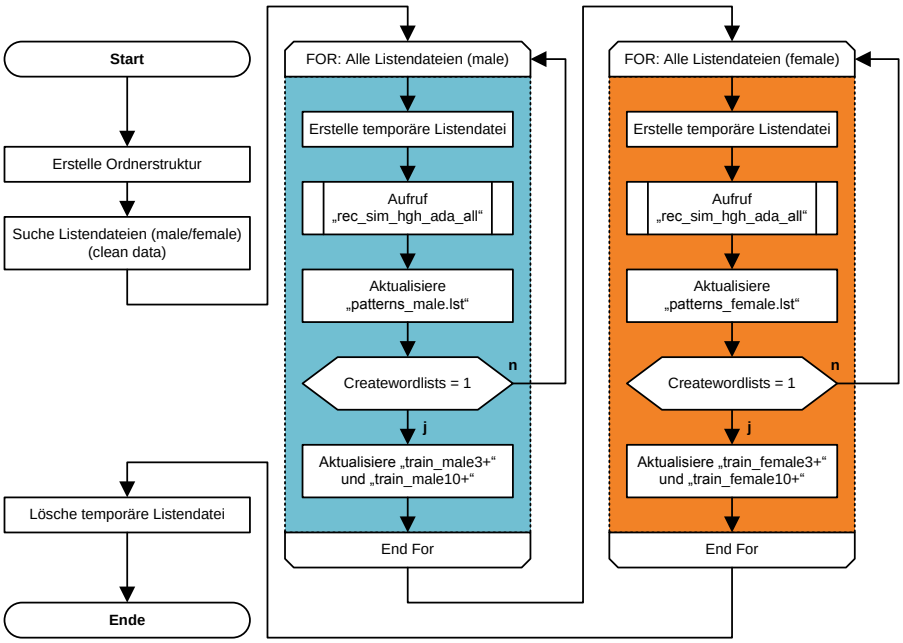

Abbildung 2.7: Flussdiagramm *analyse_hgh.m*

2.5 Training der HMM-Modelle

Das MATLAB-Skript *train_recognizer_hgh.m* greift auf drei Unterprogramme des HTK-Paketes zurück. Es handelt sich bei diesem Paket um eine portable Programmsammlung des Cambridge University Engineering Department (CUED), welche zum erstellen, manipulieren und testen von Hidden Markov Modellen sowie weiteren Aufgaben im Bereich der Spracherkennung verwendet werden kann. Im Rahmen dieser Projektarbeit wird das Paket für ein isoliertes Worttraining zur Erstellung von HMM-Modellen verwendet. Der Trainingsprozess ist in verschiedene Phasen eingeteilt, welche jeweils mit den Unterprogrammen *HInit*, *HHed* und *HRest* durchgeführt werden.

Das Skript liest zunächst die Wortlisten *train_male* und *train_female* ein, bei welchen es sich standardmäßig um *train_male3+* und *train_female3+* handelt. Um das Training des *w_sil*-Modells zeitlich zu optimieren, wird eine auf 500 Patterns beschränkte, separate Trainingsliste angelegt. Anschließend wird das HTK-Programm *HInit* ausgeführt, welches initiale Modelldateien erstellt und diese im Ordner "*/<dir_training>/...*" abspeichert. Hierzu schneidet das Programm die benötigten Abschnitte aus den gelabelten Audioressourcen heraus, unterteilt diese in einheitliche Segmente und bestimmt anschließend durch einen mehrstufigen, iterativen Algorithmus Mittelwerte und Standardabweichungen. Als Vorlage für die Modelle dient die HMM-Prototypdatei *proto_42coef*, welche ein Zweizustandsmodell beinhaltet. Nachdem das initiale Training der männlichen und weiblichen Trainingslisten abgeschlossen ist, wird das *w_sil*-Training auf Basis der Prototypdatei *proto_silence_42coef* durchgeführt.

In der zweiten Phase werden mittels *HHed* die erstellten HMM-Modelle dahingehend manipuliert, dass die Mischungen der durch *HInit* iterativ bestimmten Gaussverteilungen jeweils verdoppelt werden. Die Mischungen des Modells für *w_sil* werden hingegen um den Faktor 8 erhöht.

Im dritten und finalen Schritt werden die Wahrscheinlichkeiten der erstellten Modelle mittels des HTK-Programms *HRest* unter Nutzung der gleichen Sprachressourcen verfeinert. Der in *HInit* verwendete Viterbi-Algorithmus wird in dieser Phase jedoch durch den so genannten Baum-Welch- Algorithmus ersetzt. Diese erneute Parameterschätzung wird anschließend für das Modell von *w_sil* durchgeführt und das Training damit abgeschlossen.

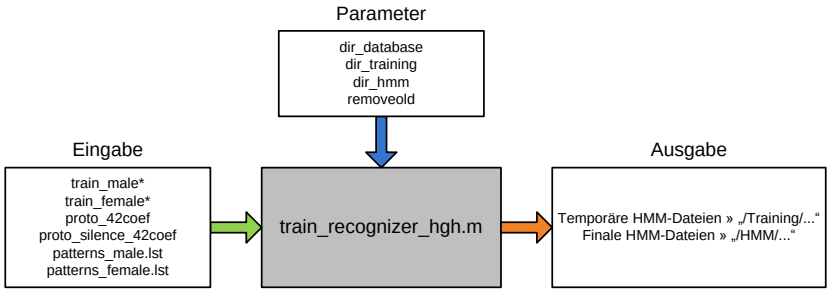

Abbildung 2.8: Parameter *train_recognizer_hgh.m*

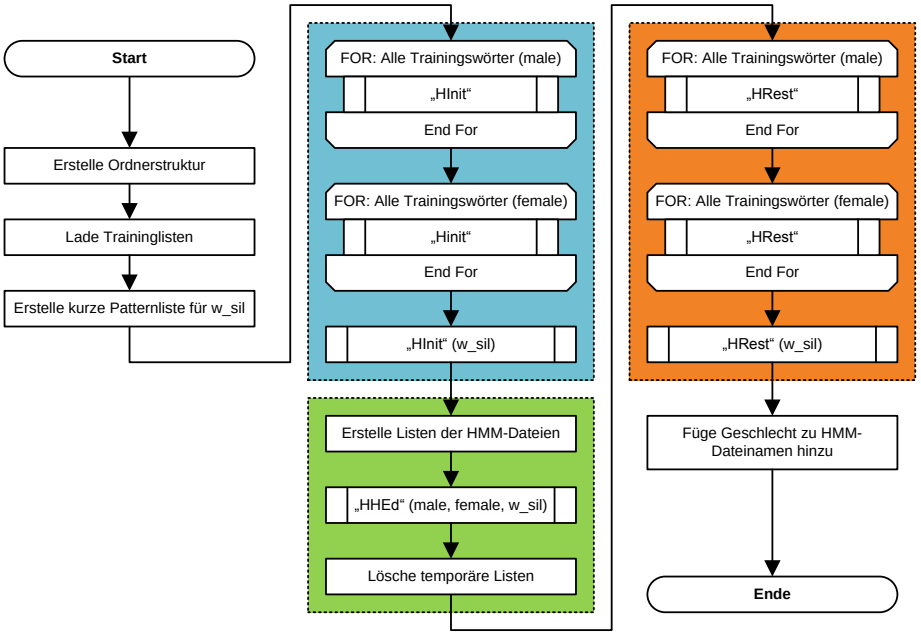

Abbildung 2.9: Flussdiagramm *train_recognizer_hgh.m*

2.6 Szenarioerstellung für Experimente

In Vorbereitung eines Erkennungsexperimentes müssen mehrere Dateilisten erstellt werden, welche über die graphische Oberfläche des verwendeten Analyse- und Erkennungsprogramms eingebunden werden. Die erste Datei stellt eine Auflistung aller Pattern- bzw. Audiodateien dar, welche zur Erkennung verwendet werden sollen. Im Falle von Audiodateien muss das Programm im Modus "Analysis & Recognition" betrieben werden, da zunächst die zugehörigen Patterndateien erstellt werden müssen. Dies ist für die im Rahmen dieses Projektes durchgeführten Experimente der Fall, da bei den mittels der Skripte erstellten Patterndateien lediglich störungsfreie Audiodateien und somit nur ein kleiner Teil der Datenbank berücksichtigt wurden. In der zweiten benötigten Datei hingegen muss eine Liste aller notwendigen HMM-Modelle gespeichert werden. Zusätzlich wird eine so genannte "Syntax"-Datei benötigt, welche dem Erkennungsprogramm über ein Zustandsmodell die Form der zu erkennenden Wortfolge mitteilt. Während die letzte Datei schnell angepasst werden kann, stellt die Ermittelung und Auflistung der benötigten Audio- und HMM-Dateien je nach Definition des Experiments ein langwieriges Unterfangen dar und wird durch das MATLAB-Skript *create_recognition_scenario.m* deutlich erleichtert.

So ermöglicht das Skript eine vollautomatische Suche von Audiodateien, die nach den Kriterien Aufnahmeumgebung und Kanal spezifisch gefiltert werden können. Zusätzlich werden die benötigte Datei- und HMM-Liste automatisch erstellt und somit die schnelle Definition eines Szenarios ermöglicht. Aufgrund des ungleich komplexeren Aufbaus und somit schwierigeren Erstellung einer Syntaxdatei wird auf eine Automatisierung an dieser Stelle verzichtet. Über den Parameter *mode* kann zwischen zwei internen Programmabläufen selektiert werden. Im Modus "0" wählt das Programm auf Basis der bereits mit *create_listfiles.m* erstellten Dateilisten Wörter aus, während im Modus "1" die Datenbankdatei "filelog.txt" erneut durchsucht wird. Der erste Modus ist aufgrund der vorliegenden Listen deutlich schneller, während der zweite Modus eine nachträglich im Quelltext hinzugefügte Filterung bezüglich weiterer Kriterien ermöglicht.

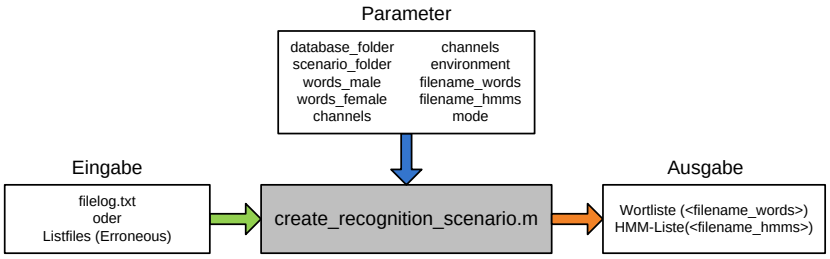

Abbildung 2.10: Schnittstelle *create_recognition_scenario.m*

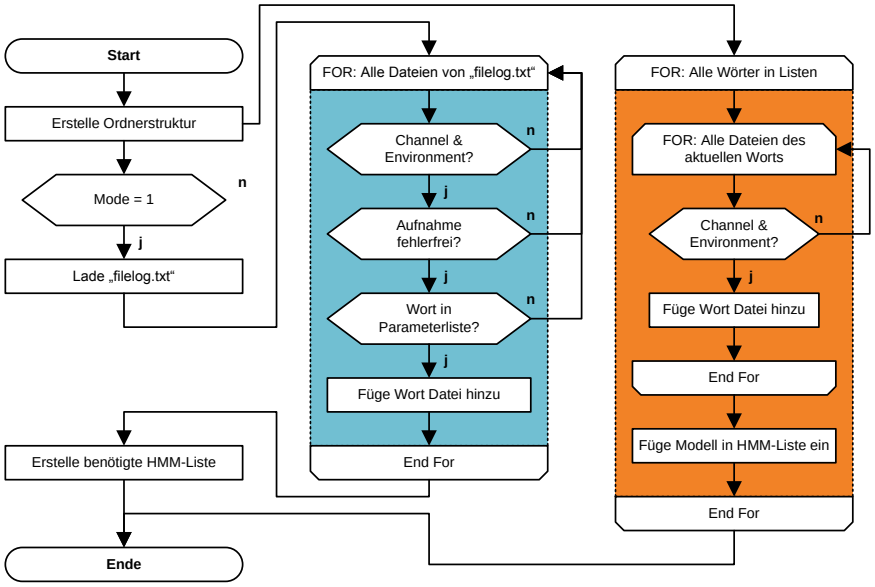

Abbildung 2.11: Flussdiagramm *create_recognition_scenario.m*

Kapitel 3

Erkennungsexperimente

Das nachfolgende Kapitel enthält eine detaillierte Beschreibung aller durchgeführten Erkennungsexperimente. Die Rahmenbedingungen der einzelnen Experimente werden definiert, die gewonnenen Daten grafisch aufbereitet und interpretiert. Die Analyse und Erkennungsprozess der zu erkennenden Sprachdaten werden dabei über ein vorliegendes MATLAB-Skript mit grafischer Oberfläche realisiert.

3.1 Experiment: Isolierte Zahlen 0 bis 9

Im ersten Experiment wird eine Erkennung von isolierten Aufnahmen der italienischen Zahlen 0 bis 9 vorgenommen. Zunächst wird eine Erkennung von Audiodateien der ungestörten Umgebungen *Office* und *Entertainment_Off* durchgeführt. Im Anschluss werden mehrere Erkennungsversuche mit gestörten Audiodaten vorgenommen, welche den Kategorien *Entertainment_On*, *Public_Open* und *Public_Hall* sowie *Car_On*, *Car_City* und *Car_Highway* entsprechen. Aufgrund der sehr geringen Zahl an verfügbaren Aufnahmen der *Car*-Kategorie, wird auf eine wortgenaue Darstellung verzichtet und lediglich die durchschnittlichen Werte in Unterabschnitt 3.1.3 berücksichtigt.

3.1.1 Erkennung in ungestörter Umgebung

Die Erkennungsergebnisse der Umgebungen *Office* und *Entertainment_Off* werden im Balkendiagramm 3.1 dargestellt. Die farblich zu unterscheidenen Entfernungskategorien *Clean*, *Close*, *Medium* und *Far* entsprechen den Aufnahmekanälen 1 bis 4. Da für das HMM-Training die Audiodateien des ersten Kanals verwendeten wurden, liegen die einzelnen Erkennungsquoten von *Clean* konstant bei 100%. Der Prozentwerte der Entfernung *Close* weisen eine sehr niedrige Varianz mit einem Minimum von 82 Prozent auf, was auf eine gute Erkennungsqualität schließen lässt. Bei den höheren Abständen *Medium* und *Far* zeigen sich im Schnitt niedrigere Trefferquoten und eine deutlich höhere Streuung der Werte. Die niedrigen Prozentzahlen sind durch die HMM-Modelle zu begründen, deren Parameter ausschließlich durch störungsfreie Audioaufnahmen entwickelt wurden und somit aufgrund des mit der Entfernung zunehmenden Raumhalls häufigere Fehlerkennungen auslösen.

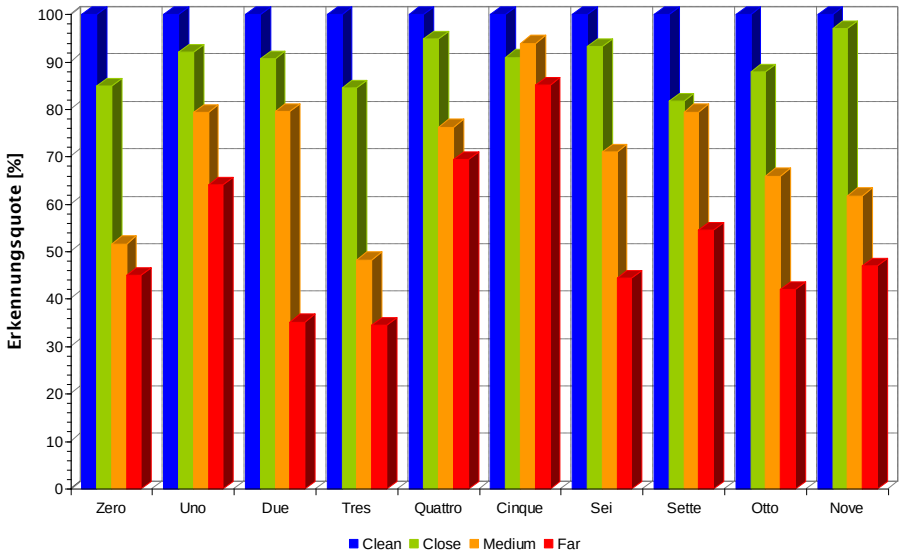

Abbildung 3.1: Erkennungsquote der Ziffern in ungestörter Umgebung

Ebenfalls lässt sich eine Abhängigkeit der Erkennungsquote vom jeweils gesprochenen Wort erkennen. So liegen insbesondere bei „Cinque" die Prozentwerte von *Medium* und *Far* mit ca. 89% deutlich über dem Durchschnitt, während diese für die Worte „Zero" und „Tres" unterhalb von ca. 50% liegen. Teilweise tritt dieser Zusammenhang auch bei den den übrigen Zahlen auf und lässt somit den Schluss zu, dass bei fehlender Adaption der HMMs die Beschaffenheit der durch Raumhall gestörten Worte sich mehr oder weniger kritisch auf die Erkennungsquote ausübt.

Qualitativ lässt sich diese Beobachtung auch in der Verwechselungsmatrix in Abbildung 3.2 erkennen, welche die durchschnittlichen Wahrscheinlichkeitswerte der korrekten bzw. fehlerhaften Detektionen aus den Entfernungen *Medium* und *Far* enthält. So weißt das Wort „Cinque" die höchste Trefferquote und eine sehr niedrige Anzahl an Fehldetektionen auf, während „Zero" und „Tres" oftmals fehlerhaft als „Cinque" und „Sette" erkannt wurden. Hieraus lässt sich folgern, dass zur Verbesserung der Erkennungsquoten ein HMM-Training mit bereits gestörten Aufnahmen oder eine kompensierende Adaption der Modelle notwendig ist.

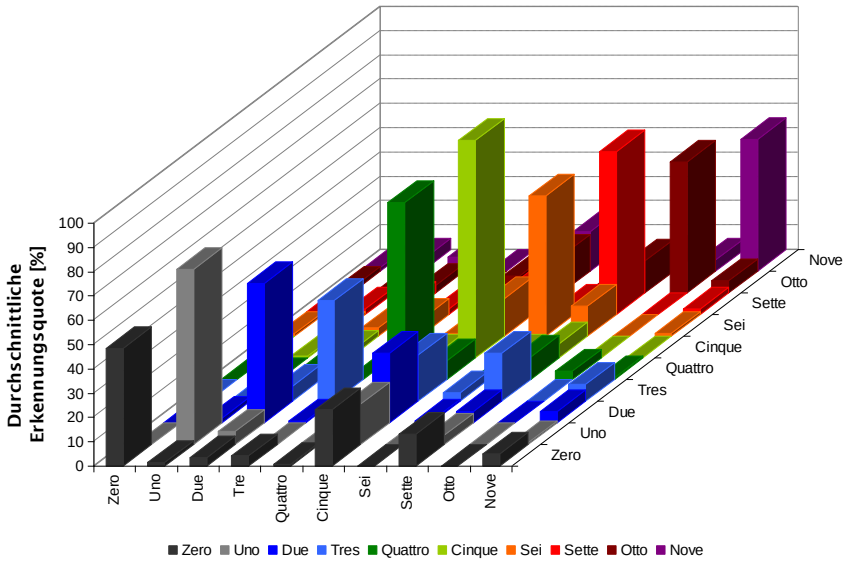

Abbildung 3.2: Gemittelte Verwechselungsmatrix von *Medium* und *Far*

3.1.2 Erkennung in gestörten Umgebungen

Im folgenden Kapitel werden die Erkennungsergebnisse der Aufnahmeumgebungen *Entertainment_On*, *Public_Open* und *Public_Hall* dargestellt. In den beiden zuletzt genannten Fällen wurde die Entfernung *Far* bzw. der vierte Kanal nicht berücksichtigt, da die vorliegenden Audiodaten dem des dritten Kanals bzw. *Medium* entsprechen.

Wie sich in Abbildung 3.3 zeigt, weichen die Ergebnisse bei präsenter Hintergrundmusik von denen mit ungestörten Daten ab. So ist die Varianz der Erkennungsquote im Abstand *Close* mit ca. 33% bereits relativ hoch und im Durchschnitt schlechter. Die höchste Detektionsquote weißt das Wort „Quattro" auf, während die Wörter „Zero" und „Sette" am seltensten korrekt erkannt wurden. Einzelne Zahlen weisen somit zwar qualitativ ähnliche Quoten wie die der ungestörten Daten auf, insgesamt betrachtet entwickeln sich die Werte jedoch unterschiedlich.

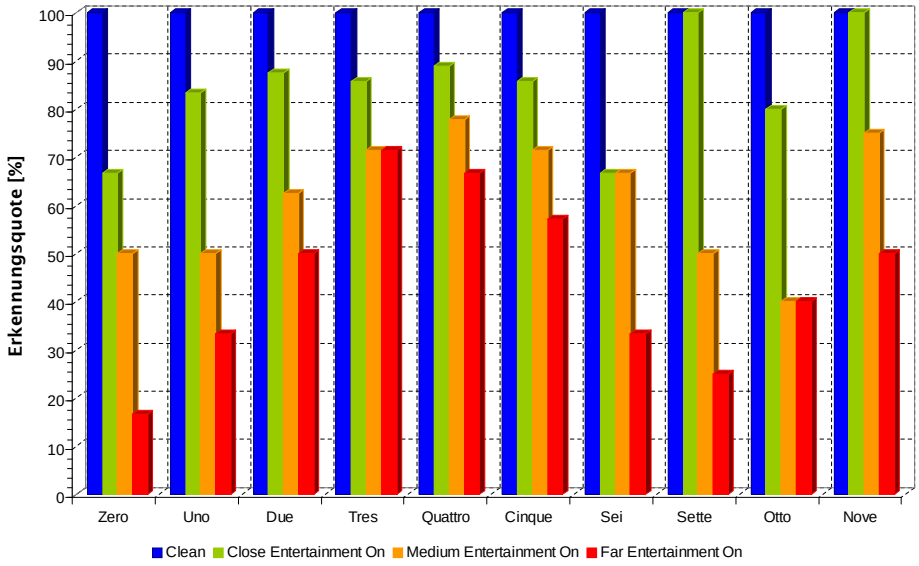

Abbildung 3.3: Erkennungsquote der Ziffern mit Hintergrundmusik

Die Abbildungen 3.4 und 3.5 weisen wie die Ergebnisse von *Entertainment_On* eine deutlich höhere Varianz im Vergleich zur ungestörten Erkennung auf. Generell besitzten die Wörter „Cinque" und „Quattro" wiederholt die besten sowie „Zero" und „Tres" die schlechtesten Erkennungswerte. Die aufgrund von Hintergrundgesprächen ausgeprägtere akustische Störung in der Umgebung *Public_Hall* lässt sich durch die niedrigeren Erkennungsquoten der Entfernung *Medium* erkennen.

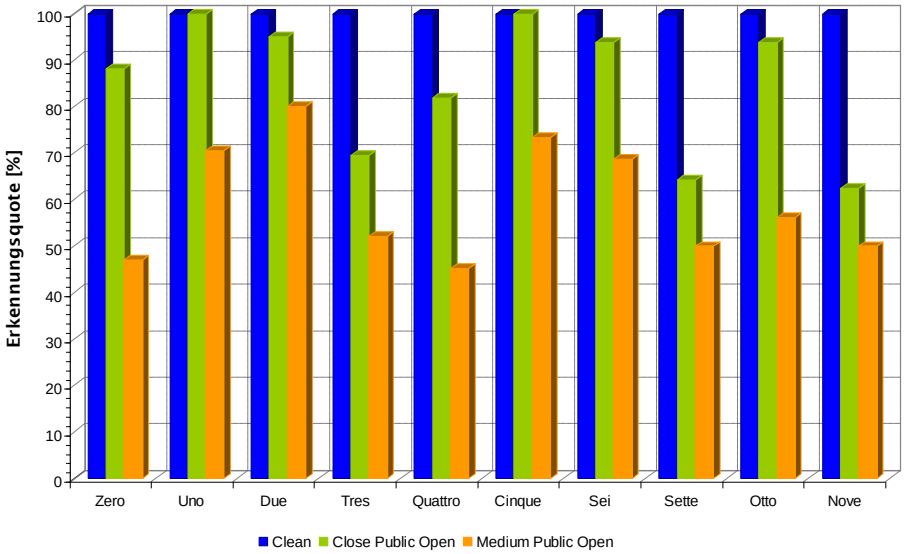

Abbildung 3.4: Erkennungsquote der Ziffern auf öffentlichem Platz

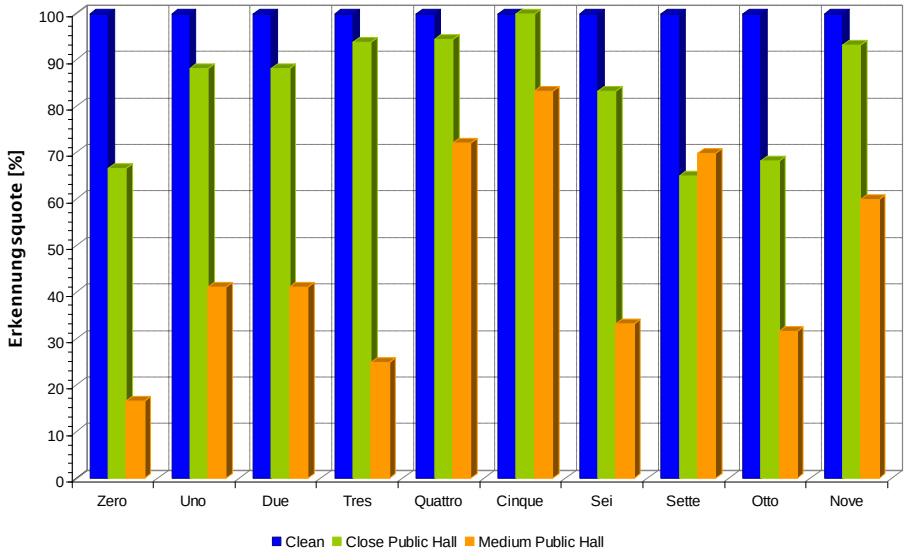

Abbildung 3.5: Erkennungsquote der Ziffern in öffentlichem Gebäude

3.1.3 Vergleich der Ergebnisse

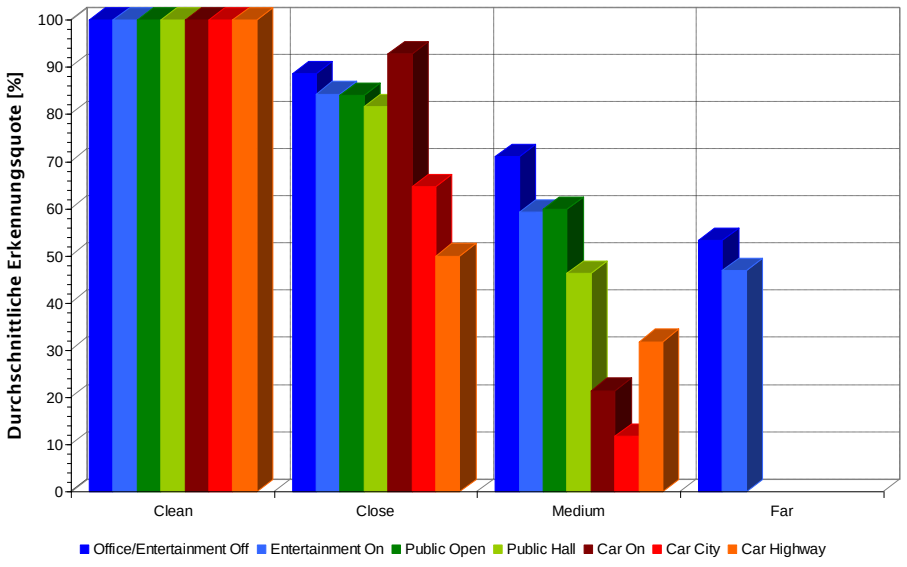

Abbildung 3.6: Durchschnittliche Ziffernerkennungsquoten

Zum Vergleich der bisherigen Ergebnisse aus ungestörten und gestörten Experimente werden in Abbildung 3.6 die durchschnittlichen Erkennungsquoten der einzelnen Umgebungen über ihrer Aufnahmedistanz aufgetragen. Sehr deutlich lässt sich hier der störende Einfluss von Raumhall und Hintergrundgeräuschen erkennen, welcher mit wachsendem Abstand zwischen Sprecher und Mikrofon zunimmt. Wie erwartet verschlechtert die Präsenz von Hintergrundmusik im Falle von *Entertainment On* die Erkennungsquote, welche über alle Distanzen niedriger als die der ungestörten Sprachdaten liegt. Des Weiteren verbleiben die Erkennungsquoten von *Public Hall* konstant unterhalb von *Public Open*, was auf den starken Einfluss der in dieser Umgebung ausgeprägten Hintergrundgespräche hinweist.

Die Ergebnisse der Fahrzeugaufnahmen sind im Rahmen dieses Experimentes kritisch zu betrachten, da nur sehr wenige Audioressourcen für eine Erkennung vorlagen. So lässt sich im allgemeinen eine erklärbare Entwicklung erkennen, der Prozentwert der Umgebung *Car Highway* in der Distanz *Medium* weicht allerdings deutlich von den übrigen Werten ab. Es kann somit nicht mit Sicherheit die Aussage getroffen werden, ob eine bessere Erkennung vorlag oder die Werte statistisch verfälscht sind. Generell sind die Ergebnisse jedoch hinreichend gut interpretierbar und lassen sich somit für eine grobe Abschätzung der Erkennungsperformanz von nicht adaptierten HMM-Modelle in gestörten Umgebungen verwenden.

3.2 Experiment: Isolierte Wörter „Si” und „No”

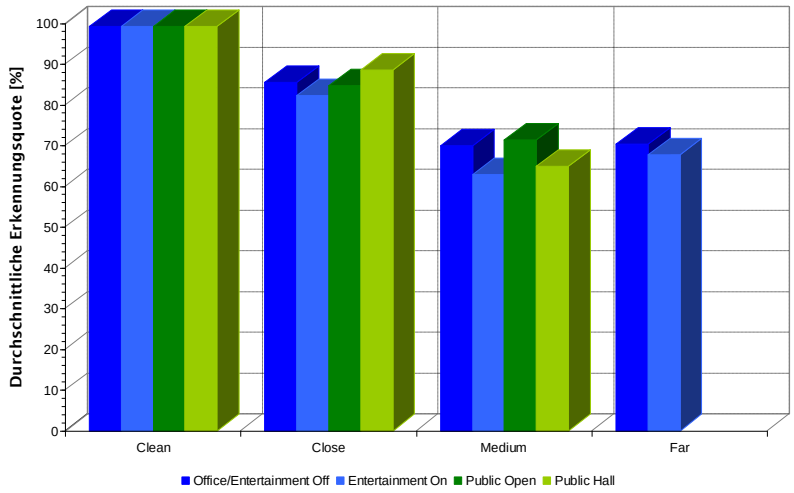

Abbildung 3.7: Durchschnittliche Si-No-Erkennungsquoten

Innerhalb des zweiten Erkennungsexperimentes wurden „Si” und „No” verwendet, welche zu den in der Datenbank am häufigsten vorkommenden Wörtern zählen. Wie im ersten Versuch folgt auf die ungestörte eine gestörte Erkennung aus den Umgebungen *Entertainment_On*, *Public_Open* und *Public_Hall*. Aufgrund der sehr geringen Wortanzahl wird im Gegensatz zum vorhergehenden Kapitel auf eine detaillierte Analyse verzichtet und in Abbildung 3.7 lediglich die Durchschnittswerte der Erkennungsquoten dargestellt.

Allgemein betrachtet ergibt sich erneut ein gegenläufiger Trend von Erkennungsquote und Aufnahmedistanz. Allerdings fällt dieses Ergebnis vergleichsweise weniger deutlich und nicht für alle Messreihen zutreffend aus, wie im vorhergehenden Experiment zu beobachten war. So liegen zum einen die Prozentwerte der Kategorien *Office/Entertainment Off* und *Entertainment On* des Abstandes *Far* höher als von *Medium*, während zum anderen die gestörten Aufnahmen von öffentlichen Plätzen und Gebäuden in der Distanz *Close* sogar bessere Werte als die der ungestörten Daten aufweisen. Da für ersten beiden Umgebungen

genügend Audiodateien vorlagen, der Erkenner jedoch nur die Wahl zwischen zwei möglichen HMMs hatte, könnte dies die vorliegenden Ergebnisse von *Far* erklären. Die relativ hohen Erkennungsquoten der beiden letzteren Kategorien könnte durch eine verfälschte Statistik begründet werden, da hier nur eine sehr geringe Anzahl von Sprachdateien genutzt werden konnte. Somit lassen die Ergebnisse dieses Experimentes keine detaillierte Interpretation zu und sollten falls möglich mit einer größeren Anzahl von Schlüsselwörtern und Audiodateien verfeinert werden.

Kapitel 4

Fazit

Im Verlauf dieses Projektseminars wurde die Struktur und Formatierung der italienischen SPEECON-Datenbank angepasst und verbessert, um eine effiziente Nutzung der Daten im Bezug auf Spracherkennungsexperimente zu ermöglichen. Die zu diesem Zweck entwickelten MATLAB-Skripte lassen sich durch eine weitreichende Parametrisierung individuell konfigurieren und somit auch für zukünftige Aufgaben weiterverwenden. Die Ergebnisse der Erkennungsexperimente stimmen weitestgehend mit den theoretischen Erwartungen überein und ermöglichen eine qualitative Abschätzung der Einflüsse von akustischen Störfaktoren auf Spracherkennungssysteme.

Abbildungsverzeichnis

Tabellenverzeichnis