HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

SIEMENS

ATM-Voice

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren:	Reto Brühwiler
	Manuela Grob
Betreuer:	Prof. Dr. P. Heinzmann
Projektpartner:	Siemens Schweiz
Experte:	Charles Zehnder
Gegenleser:	Prof. Dr. A. Doering
Start:	14. September 2009
Ende:	18. Dezember 2009

Aufgabenstellung des Betreuers

Air Traffic – Verarbeitung von Flugfunkmeldungen

Studiengang:	Informatik (I)
Semester:	HS 2009/2010 (14.09.2009-14.02.2010)
Durchführung:	Studienarbeit
Fachrichtung:	Internet-Technologien und -Anwendungen
Gruppe:	Reto Brühwiler, Manuela Grob
Betreuer:	Prof. Dr. P. Heinzmann
Koreferent:	Prof. Dr. A. Doering
Industriepartner:	Charles Zehnder, Siemens Schweiz / Ground Instructor Pilotenausbildung
Ausschreibung:	Die Air Traffic Monitoring Plattform http://atm.cnlab.ch bietet Informationen zu Flugbewegungen, welche mittels Lärmsensoren, Kameras und Radarempfänger beobachtet werden. Gegenwärtig steht zur Diskussion, die Plattform mit Flugfunkinformationen zu erweitern. In diesem Zusammenhang stellt sich die Frage, wie gut die Inhalte von Flugfunkmeldungen extrahiert werden können. Im Rahmen der vorliegenden Studienarbeit sind in einer ersten Phase folgende Teilaufgaben zu bearbeiten: Einarbeitung in den Flugfunk: Systeme, Meldungsarten, Begriffe / Wörter, Regeln / Grammatik Einarbeitung in Speech-to-Text: Umwandlungsprinzipien, Analyse und Vergleich verschiedener Tools (z.B. Google Voice, Nuance Dragon Naturally Speaking, Loquendo Automatic Speech Recognition, Microsoft Speech Recognition, Sphinx) Bewertung der Spracherkennungssysteme hinsichtlich des Einsatzes beim Flugfunk Realisierung eines Pilotsystems zur Verarbeitung von Meldungen unterschiedlicher Qualität: Lehr-CDs, Funkmeldungen von Piloten / Flugverkehrsleitern, verschiedene Aufnahmekanäle (Flugfunkempfänger, Web); Erkennung von Callsigns und Meldungsinhalten. In einer späteren Phase sollen Voice-Meldungen in atm.cnlab.ch integriert werden (Abspeichern von Meldungen mit Meta-Informationen, Zuordnung zu Flügen, Abspielen von Meldungen). Fernziel ist es, die Qualität von Flugfunkmeldungen beurteilen zu können (Verständlichkeit, Korrektheit, Einhaltung der Funkregeln).
Voraussetzungen:	Gute Kenntnisse der digitalen Signalverarbeitung und Erfahrungen im Flugverkehr (Pilotenlizenz) sind von Vorteil.

Rapperswil, den ....

Betreuer: Prof. Dr. P. Heinzmann

Erklärung der Studenten[1]

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	Person	Funktion
Kapitel: 4, 5.3-5.5, 0	Brühwiler Reto	Autor der Arbeit
Kapitel: 5.1, 5.2, 5.6-5.9, 6.2.2, 7	Grob Manuela	Autorin der Arbeit
Korrekturlesen, Hinweise zur sprachlichen Überarbeitung	Grob Marlise	Korrekturlesen
Teilnahme an Review[2]	Camichel Corsin	„Kritiker"
Teilnahme an Review	Dünser Patrick	„Kunde"
Idee, Aufgabenstellung, allgemeines Pflichtenheft, Betreuung während der Arbeit	Prof. Dr. P. Heinzmann	verantwortlicher Professor
Betreuung während der Arbeit	Andreas Maag	Assistent HSR
Betreuung während der Arbeit	Charles Zehnder, Siemens Schweiz / Ground Instructor Pilotenausbildung	Industriepartner

Ich erkläre hiermit,

dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,

dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Rapperswil, den ....
Student: Brühwiler Reto

Rapperswil, den ....
Studentin: Grob Manuela

[1] Diese Erklärung basiert auf der Muster-Erklärung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

[2] Review vom 10.12.09 gemäss Modul Projekt- und Qualitätsmanagement HS 2009/2010

Inhaltsverzeichnis

1. Abstract

Im Rahmen der Studienarbeit Air-Traffic-Monitoring (ATM) – Voice sollte abgeklärt werden, wie gut sich die Inhalte von Flugfunkmeldungen erkennen lassen. Insbesondere ob sich der Funkverkehr zwischen dem Tower und dem Flugzeug automatisch in Textmeldungen umwandeln liesse. Schlussendlich sollten diese Meldungen einer Flugnummer zugeordnet werden können.

Im ersten Teil der Arbeit wurden die Grundlagen der Spracherkennung erarbeitet. Dazu gehört das Verständnis der einzelnen Komponenten, wie zum Beispiel das Hidden Markov Model (HMM) und Begriffe wie Mel Frequency Cepstral Coefficients (MFCC) und Perceptual Linear Prediction (PLP). In der Einarbeitungsphase musste man sich auch mit den Flugfunkregeln vertraut machen. Es zeigte sich, dass Flugfunkmeldungen auf einem kleinen Vokabular und auf einer einfachen Grammatik basieren, was die Spracherkennung eigentlich erleichtern sollte. Auf der anderen Seite wirken sich die schlechte Tonqualität und die sehr unterschiedliche Sprecherqualität erschwerend auf die Spracherkennung aus.

Unter Berücksichtigung dieser Erkenntnisse wurde die Anforderungsanalyse gemacht. Darauf basierend wurden verschiedene Speech-to-Text Lösungen im Hinblick auf den Einsatz im Flugfunkbereich gesucht und verglichen. Für die Realisierung eines Prototypen-Systems wurde das Open Source Framework Sphinx 4 ausgewählt. Ausschlaggebend waren neben den Kosten (Freeware) und der Verfügbarkeit vor allem die Anpassbarkeit der einzelnen Komponenten. Bei Sphinx können gemäss Dokumentation eigene „Dictionarys" und „Grammatiken" vorgegeben werden. Somit hätte ein „Dictionary" erstellt werden können, welches speziell an den Wortschatz der Fliegerei angepasst ist und eine „Grammatik", welche die Flugfunkregeln berücksichtigt.

Bei der Realisierungsphase hat sich aber leider gezeigt, dass sich die Sphinx Komponenten nicht wie versprochen anpassen liessen. Probleme bereitete insbesondere die Kombination des geeignetsten AcousticModels und der eigenen Grammatik.

Für das angestrebte Ziel, die Funksprüche in Text umwandeln zu können, braucht es noch weitere Forschungs- und Entwicklungsarbeit. Nichtsdestotrotz konnte mit dieser Arbeit eine gute Basis geschaffen werden. Vor allem in der Erstellung des Dictionarys und der Grammatik, welche zu einem grossen Teil ausgearbeitet wurde.

2. Management Summary

2.1. Ausgangslage:

Der Flugfunk zwischen dem Tower und dem Flugzeug wird digitalisiert aufgenommen. Diese Funksprüche können bis jetzt nur erneut wiedergegeben und so abgehört werden. Viel besser wäre es, wenn diese automatisch in Text umgewandelt werden könnten und so eine Auswertung erlauben würden.

Im Rahmen der vorliegenden Arbeit soll abgeklärt werden, ob und wie Flugfunkmeldungen in Text umgewandelt werden können. Insbesondere soll zu jedem Funkspruch erkannt werden, zu welchem Flugzeug er gehört.

2.2. Vorgehen:

Im ersten Teil der Arbeit wurden die Grundlagen der Spracherkennung erarbeitet. Zusätzlich mit dem neu angeeigneten Wissen über den Flugfunk wurde definiert, was ein Produkt für die Spracherkennung von Funksprüchen erfüllen muss. Das Flugfunkvokabular ist verhältnismässig klein und die Funksprüche sollten gewissen Regeln folgen. Dies sind Vorteile, welche die Erkennungswahrscheinlichkeit erhöhen. Dagegen ist die Qualität der Audiodateien beeinträchtigt, da viele Nebengeräusche nicht entfernt werden können.

2.3. Ergebnisse:

Die Marktanalyse ergab, dass kein konventionelles Produkt dieses Soll vollständig erfüllt. Grund dafür sind vor allem die Probleme mit der Sprachqualität.

Die grösste Chance, die Funksprüche zu erkennen, rechneten wir uns mit Sphinx 4 aus. Denn in der Dokumentation steht geschrieben, dass es einfach anpassbar ist. Ein weiterer positiver Aspekt ist, dass Sphinx kostenlos zur Verfügung steht (Freeware).

Bei der Umsetzung hat sich gezeigt, dass diese Anpassungen nicht problemlos umgesetzt werden können. Gründe dafür sind die technischen Grenzen und die Schwierigkeiten bei der Kombination der einzelnen Komponenten von Sphinx.

Nichtsdestotrotz konnte mit dieser Arbeit aufgezeigt werden, was alles benötigt wird, um Funksprüche in Text umwandeln zu können. Auch wurden die Flugfunkregeln und der Wortschatz für eine Weiterbearbeitung vorbereitet.

2.4. Ausblick

Das Endziel, die Erkennung des Funkspruchs, konnte leider nicht erreicht werden. Dies vor allem deshalb, weil die Studenten viel Zeit aufwenden mussten, sich die Grundlagen in der Spracherkennung zu erarbeiten. Eventuell könnte an der HSR ein neues Informatik Aufbaumodul angeboten werden, in welchem Spracherkennung unterrichtet wird. Damit könnten gute Voraussetzungen für weiterführende Arbeiten durch anderen Studenten geschaffen werden.

Sphinx 4 ist ein sehr spannendes Framework und könnte mit Unterstützung von erfahreneren Java-Entwicklern und genügend Zeit so angepasst werden, dass Funksprüche zu einem grösseren Teil erkannt werden würden.

3. Einleitung

Die Air Traffic Monitoring Plattform[3] bietet Informationen zu Flugbewegungen, welche mittels Lärmsensoren, Kameras und Radarempfängern beobachtet werden. Gegenwärtig steht zur Diskussion, die Plattform mit Flugfunkinformationen zu erweitern. In diesem Zusammenhang stellt sich die Frage, wie gut die Inhalte von Flugfunkmeldungen extrahiert werden können.

Im Rahmen der vorliegenden Arbeit soll abgeklärt werden, ob und wie Flugfunkmeldungen in Text umgewandelt werden können. Dies bedeutet, dass sich die Studenten in die Grundlagen der Spracherkennung und Flugfunkregeln einarbeiten müssen. Anhand dieser Informationen sollen die Anforderungen definiert werden, welche von einem System, welches Funksprüche erkennt, erfüllt werden müssen. Insbesondere soll bei diesen Anforderungen das Augenmerk darauf gelegt werden, dass zu jedem Funkspruch erkannt wird, zu welchem Flugzeug er gehört. Diese Informationen sind mittelfristig in die Air Traffic Monitoring Anwendung zu integrieren.

Im nachfolgenden Kapitel werden als erstes die Grundlagen des Flugfunkes für diese Aufgabenstellung genauer beschrieben. Anschliessend wird die Basis für die Spracherkennung gelegt.

Anhand der Kapitel, 4 Funkverkehr und 5 Spracherkennung kann die Evaluation der Spracherkennungssysteme aufgebaut werden. In dem Kapitel 6 Evaluation bestehender Spracherkennungssysteme befindet sich nur eine grobe Übersicht über die durchgeführte Marktanalyse. Eine detaillierte Beschreibung befindet sich im Anhang. Die Entscheidung wird im letzten Bereich des 6. Kapitels genau erläutert.

Das Kapitel 7 Sphinx 4 beschäftigt sich insbesondere mit der Sphinx-Architektur und einer Übersicht über die einzelnen Komponenten.

Im Anhang befinden sich, wie bereits erwähnt die ausführlichen Testdokumentationen.

[3] Link zur Air Traffic Monitoring Plattform: http://atm.cnlab.ch

4. Funkverkehr[4]

4.1. Allgemein

Seit den 1920er Jahren nimmt der zivile Flugverkehr stetig zu und eine Stagnation ist nicht ersichtlich. Um eine möglichst hohe Sicherheit zu erreichen, sind sämtliche wichtige Abläufe in der Fliegerei reglementiert. Neben den Regeln trägt die Kommunikation zwischen den einzelnen Teilnehmer massgeblich zur Sicherheit bei. Ohne Kommunikation kann ein so stark frequentierter Flugbetrieb nicht geregelt werden. Die Kommunikation zwischen Boden (Tower) und Flugzeug wird mittels Funk realisiert. Für den Flugfunk ist der Frequenzbereich zwischen 117,975 und 137 MHz reserviert. Die Audioübertragung wird im Kapitel 5.4.3. erläutert.

4.2. Dictionary

Um die Verständigung auch bei einer schlechten Übertragungsqualität zu gewährleisten, wurden international gültige Diktierbegriffe definiert. Um Verwechslungen zu vermeiden, unterscheiden sich diese Begriffe so weit wie möglich in der Aussprache. So kann der Buchstabe oder die Zahl auch erkannt werden, wenn nicht das ganze Wort verstanden wurde.

Zahl	Wort
0	Zero
1	One
2	Two
3	Three (ausgesprochen “tri”)
4	Four
5	Five
6	Six
7	Seven
8	Eight
9	Nine (ausgesprochen “niner”)
100	Hundred
1000	Thousand (ausgesprochen “tausend”)

Tabelle 1: Zahlendefinition

[4] Wikipedia-Artikel über den Flugfunk im Allgemeinen http://de.wikipedia.org/wiki/Flugfunk (zuletzt besucht: 20.10.09)

Buchstabe	Wort
A	Alfa
B	Bravo
C	Charlie
D	Delta
E	Echo
F	Foxtrott
G	Golf
H	Hotel
I	India
J	Juliett
K	Kilo
L	Lima
M	Mike
N	November
O	Oscar
P	Papa
Q	Quebec
R	Romeo
S	Sierra
T	Tango
U	Uniform
V	Victor
W	Whiskey
X	X-Ray
Y	Yankee
Z	Zulu

Tabelle 2: Buchstabendefinition

Neben den Wörtern für die Zahlen und Buchstaben werden im Dictionary auch alle anderen für den Flugfunk wichtigen Wörter abgelegt.
Ein Beispiel-Dictionary mit einem Grossteil der Wörter ist im Dokument Dictionary.xlsx zu finden. Die phonetische Schrift wurde aus dem Dictionary der CMU[5] extrahiert.

[5] Dictionary der CMU: http://www.speech.cs.cmu.edu/cgi-bin/cmudict?in=welcome (zuletzt besucht am: 18.12.09)

4.3. Grammatik

Die Grammatik definiert, in welcher Konstellation die Wörter aus dem Dictionary verwendet werden dürfen. Durch die Vorschriften im Flugfunk ist klar definiert, wie die Funksprüche strukturiert sein müssen. Dies trägt zum besseren Verständnis bei.

Eine Beschreibung der Grammatik ist im Dokument „Grammatik.docx" abgelegt.

4.4. Ablauf der Landung

Der Luftraum ist in Zonen aufgeteilt. Diese Zonen sind nicht nur horizontal begrenzt, sondern haben auch eine Mindest- und eine Maximalhöhe. Diese Zonen sind in Klassen eingeteilt. Es gibt 7 Klassen A bis G. In der Schweiz gibt es nur 4 Klassen (C, D, E und G). E und G sind zur freien Zirkulation vorgesehen. Die Klasse G ist sogar unkontrolliert. Rund um grössere Flugplätze und Flughäfen sind diese Zonen der Klasse C zugeteilt. Der Flughafen selbst befindet sich in der D Klasse. Über der D Klasse befindet sich eine C Klasse. Spätestens 5 Minuten vor dem Einflug in diese Zone, muss sich der Pilot per Funk bei der entsprechenden Frequenz anmelden. Kurz vor der Control Zone (CTR) Grenze muss der Pilot dem Tower mitteilen, dass er landen will. Die CTR entspricht der Klasse D Zone. Sollte keine CTR definiert sein, muss der Pilot sich spätestens 30 km vor dem Flugplatz melden.

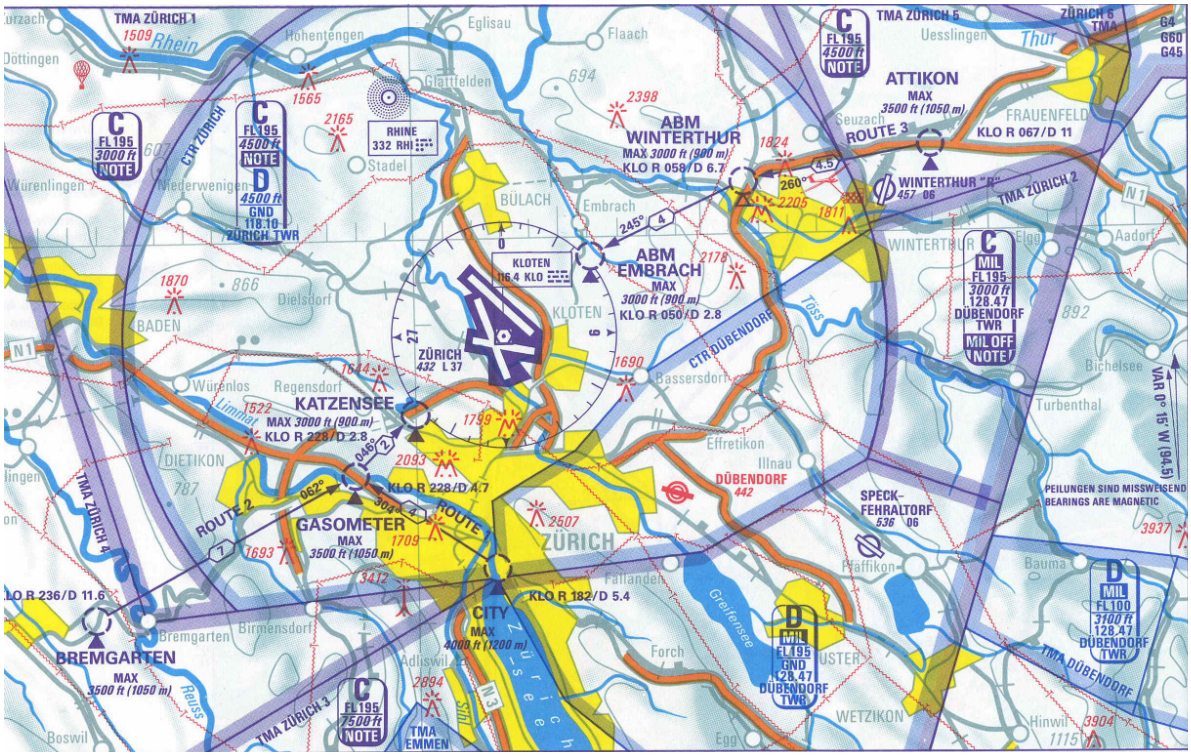

Abbildung 1 Flugkarte Flughafen Zürich(1)

Nach dem Aufrufen des Towers gibt der Pilot seine Position, die Himmelsrichtung, die Flughöhe und „FOR LANDING" durch. Der Tower quittiert den Aufruf, gibt die Anflugstrecke und die Art des Anfluges an. Der Tower gibt auch die zu verwendende Piste bekannt. Je nach Situation kann der Tower auch noch über die Windsituation informieren. Danach folgt noch die Höhenmesser Einstellung. Der Pilot bestätigt dem Tower die Anflugstrecke, die Art, die Piste und die Höhenmessereinstellung.

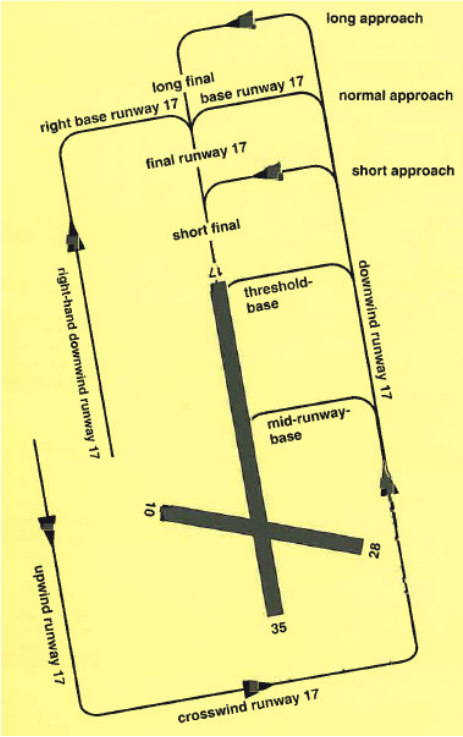

Abbildung 2 Landeanflugarten(2)

5. Spracherkennung(1) [6] [7] [8] [9]

5.1. Geschichte

Seit den 1960er Jahren wird versucht, gesprochene Wörter mit dem Computer in Text umzuwandeln. Es wurden Programme entwickelt, welche aus einem beschränkten Vokabular gesprochene Einzelwörter erkannten. Zum Beispiel führte dies zu einem Gerät, welches einzelne Ziffern erkennen konnte.

Doch die ersten grösseren Erfolge traten erst Mitte der 1980er Jahre auf. In dieser Zeit wurde erkannt, dass man aus dem Kontext schliessen kann, welches der gleich aussprechbaren Wörter gemeint ist (z.B. arm oder Arm). Dazu wird mit sogenannten N-Gramm-Statistiken[10] auch die Abhängigkeit aufeinanderfolgender Wörter beigezogen.

5.2. Aktuelle Situation

Es gibt unterschiedliche Einsatzbereiche von Spracherkennungssoftware. Zum einen werden diese in Call-Centern verwendet, wobei hier die unterschiedlichen Stimmen und ein kleines Vokabular relevant sind. Zum anderen gibt es Systemsteuerungssoftware, welche auf den Sprecher zugeschnitten sind und so ein Training verlangen.

Diese beiden Einsatzbereiche unterscheiden sich vor allem durch die Grösse des Vokabulars. Bei den Systemsteuerungsprogrammen kann die Akzeptanz eines Inputs kleiner gehalten werden, da die Aussprache einheitlicher ist. Im Gegensatz dazu steht das Call-Center mit etlichen verschiedenen Sprechern.

---

[6] Vorlesung an der ETH-Zürich über Spracherkennung http://www.tik.ee.ethz.ch/~ewendert/PPS_Spracherkennung/ (zuletzt besucht am: 16.10.2009)

[7] Wikipedia-Artikel, für die Grundlagen in der Spracherkennung in Englisch http://en.wikipedia.org/wiki/Speech_recognition (zuletzt besucht am: 16.10.2009)

[8] Wikipedia-Artikel, für die Grundlagen in der Spracherkennung auf Deutsch http://de.wikipedia.org/wiki/Spracherkennung (zuletzt besucht am: 16.10.2009)

[9] Wikipedia-Artikel zum Verständnis von Homophon http://de.wikipedia.org/wiki/Homophon (zuletzt besucht am: 16.10.2009)

[10] Erklärung von N-Gram (1 S. 375): Die Wahrscheinlichkeit eines Wortes hängt nur von den letzten N-1 Wörtern ab.

5.3. MP3 und wav im Vergleich[11] [12] (6)

5.3.1. wav

Eine wav-Datei beinhaltet ein digitalisiertes Audiosignal. Die Quantisierungsauflösung ist so fein gehalten, dass das Verfahren als verlustfrei gilt. Jede dieser Dateien besteht aus einem Header, einem Datenteil und einem optionalen Resource Interchange File Format (kurz RIFF) Datenfeld. Dieses beinhaltet z. B: Informationen über das Copyright. Die Daten werden unkomprimiert abgespeichert.

5.3.2. MP3

MP3 steht für MPEG-1 Audio Layer3 und dient der Audiodatenkompression, welche verlustbehaftet ist.

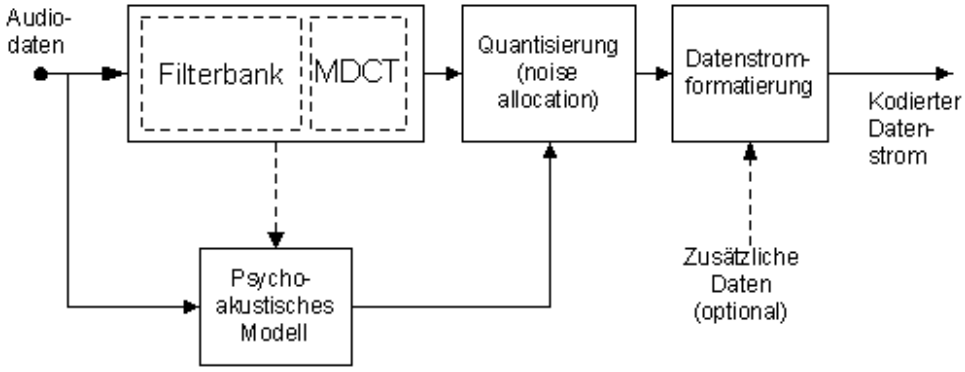

Abbildung 3: MP3 Codierung(5)

In der Filterbank wird das Audiosignal vom Zeit- in den Frequenzbereich transformiert. Es stehen dabei 32 Frequenzbänder zur Verfügung. Die Separierung funktioniert gleich wie ein Bandpass. Ein Bandpass lässt nur Frequenzen durch, welche in einem bestimmten Bereich sind. Frequenzen, die oberhalb oder unterhalb dieses Bereichs liegen, werden geblockt.

Mit dem mathematischen Verfahren Modified Discrete Cosine Transformation (MDCT) wird die Wellenform in Frequenzabfolgen umgewandelt.

Das psychoakustische Modell befasst sich mit der menschlichen Komponente. Dieses Modell bestimmt, welche Frequenzen herausgefiltert werden können, da diese zum Beispiel ausserhalb des hörbaren Bereich liegen. Des Weiteren wird ausgenutzt, dass der Mensch Frequenzen, welche dicht beieinander liegen, nicht unterscheiden kann.

---

[11] Erklärungen zu wav-Dateien
http://www.fh-friedberg.de/fachbereiche/e2/telekom-labor/zinke/nw/vp/doku/dito41.htm
(zuletzt besucht: 18.11.09)

[12] Wikipediaseite über wav: http://de.wikipedia.org/wiki/Wav (zuletzt besucht 18.11.09)

Bei der Quantisierung werden unter Berücksichtigung des psychoakustischen Models stufenlose Werte in gestufte Werte konkretisiert.

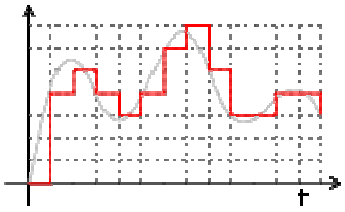

Abbildung 4: Quantisierung(6)

In der Datenstromformatierung werden den Daten die Headerinformationen vorgehängt. Diese beinhalten Informationen über die Codierung und eine Prüfsumme, welche jedoch optional ist.

5.3.3. Vergleich

Der Hauptunterschied ist der benötigte Speicherplatz. Im Normalfall hört der Mensch den Unterschied nicht. Für den Computer gibt es diesen jedoch, da viele Zwischenwerte fehlen.

5.4. Sprachproduktion und Vergleich

5.4.1. Die Stimme

Beim Sprechen presst die Lunge die Luft an den Stimmlippen vorbei, welche sich im Hals befinden. Dadurch beginnen diese zu schwingen, wobei ein Ton entsteht. Dieses Phänomen haben die meisten bereits in ihrer Kindheit kennengelernt: wenn bei einem mit Luft gefüllten Ballon die Öffnung auseinander gezogen wird und die sich im Ballon befindliche Luft ausströmt, entsteht ein Geräusch.

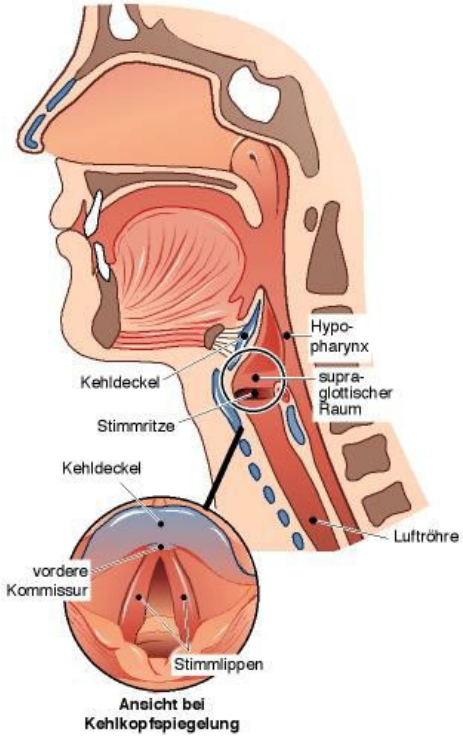

Abbildung 5 Rachenraum(7)

Bei Frauen liegt die Grundfrequenz bei ca. 250 Hz, bei Männern hingegen bei ca. 125 Hz. Der entstandene Ton wird im Resonanzraum verstärkt. Der Resonanzraum setzt sich aus dem oberen Teil des Halses, dem Rachenraum und einem Teil der Nase zusammen. Der Grundton wird durch das unterschiedliche Anspannen der Stimmlippen sowie Änderungen des Resonanzraumes, durch Bewegen der Lippen, der Zunge und des Kiefers angepasst. So lassen sich Töne zwischen 80 Hz und 12 kHz erzeugen.

5.4.2. Das Gehör

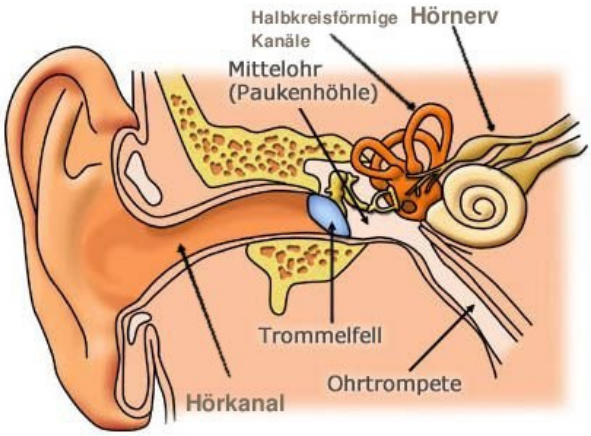

Abbildung 6: Das Ohr(16)

Schallwellen gelangen durch den äusseren Gehörgang zum Trommelfell. Die Schwingungen des Trommelfells werden durch den Hammer mittels Amboss an den Steigbügel weitergegeben. Von dort werden die Schwingungen an die Gehörschnecke übermittelt. In der Gehörschnecke befinden sich kleine Härchen, welche die Schwingungen übernehmen und in Nervenimpulse umwandeln. Diese werden ans Gehirn weitergeleitet. Dabei deckt jede Härchengruppe einen bestimmten Frequenzbereich ab. Insgesamt können Frequenzen zwischen 20 Hz und 20 kHz erkannt und verarbeitet werden. Diese Wahrnehmungsgrenzen sind im nachfolgenden Bild dargestellt.

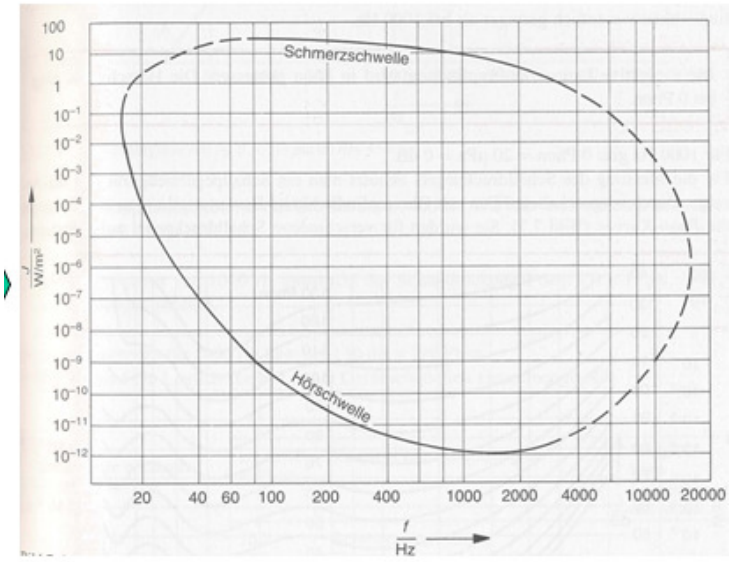

Abbildung 7 Akustische Wahrnehmungsgrenzen(4)

5.4.3. Audioübertragung

Beim Flugfunk werden die zu übertragenden Audiodaten mittels Amplitudenmodulation auf eine Trägerfrequenz (f_T) modelliert.

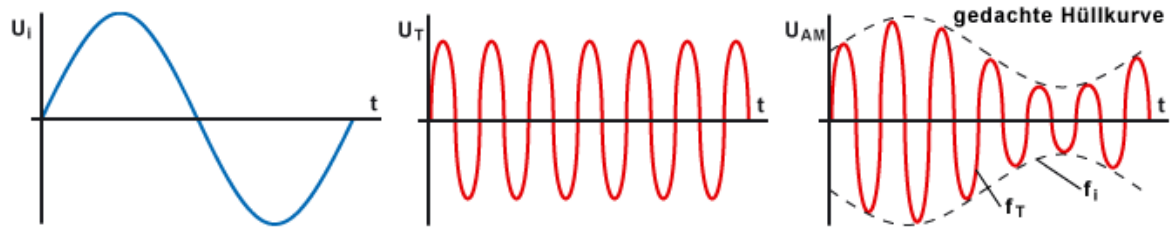

Abbildung 8: Amplitudenmodulation(8)

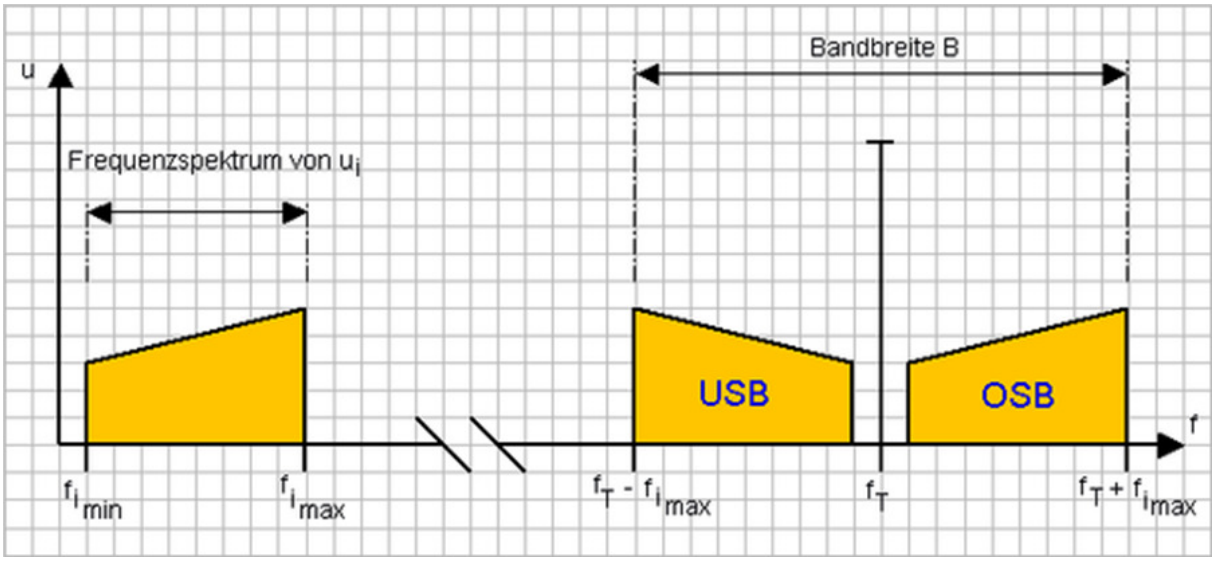

Abbildung 9: Frequenzspektrumsansicht der Amplitudenmodulation(9)

Zum Beispiel ist die Trägerfrequenz des Towers des Flughafens Zürich um Start- und Landebewilligungen einzuholen 118.100 MHz. Die Trägerfrequenzen haben einen Abstand von 25 kHz. Die nächste verfügbare Trägerfrequenz liegt dementsprechend bei 118.125 MHz. Da das Signal oberhalb und unterhalb der Trägerfrequenz abgebildet ist, steht dem Signal maximal der halbe Trägerfrequenzabstand zu Verfügung. Dadurch darf in unserem Beispiel das zu übertragende Signal maximal eine Bandbreite von 12.5 kHz besitzen.

5.4.4. Psychoakustik

Die Psychoakustik befasst sich mit der menschlichen Empfindung des Schalls. Unter anderem werden hohe Töne als lauter empfunden. Des Weiteren befasst sich die Psychoakustik mit Wahrnehmungsgrenzen. Diese wurden in Kapitel 5.4.2. beschrieben.
Tagtäglich wird der Mensch sehr vielen Reizen ausgesetzt, doch ist er nicht fähig, all diese Reize zu verarbeiten. Deshalb müssen einige Reize ausgeblendet werden. Die Psychoakustik erforscht, weshalb welche Reize ausgeblendet werden. Zum Beispiel wird ein monotoner Ton mit der Zeit als leiser empfunden.

5.5. Hidden Markov Model[13](1)

Beim Hidden-Markov-Modell (HMM) handelt es sich um ein stochastisches Modell, welches aus zwei gekoppelten Zufallsprozessen besteht. Der erste Prozess ist der Markov-Prozess und besitzt n Zustände $S_1, S_2, \dots, S_N$. Die Zustände sind verdeckt und steuern den zweiten Zufallsprozess. S_1 bezeichnet den Anfangszustand und ist eine reine Quelle. S_N den Endzustand und ist eine reine Senke. Abgesehen von diesen beiden Spezialzuständen kann von jedem Zustand in jeden anderen Zustand gewechselt werden. Einige Sprünge sind jedoch wahrscheinlicher als andere. Es ist auch möglich, zurück in den momentanen Zustand zu springen. Der Zustand zu einem bestimmten Zeitpunkt t wird als q_t bezeichnet. Wobei $q_i \in \{S_1, S_2, \dots, S_N\}$. Im zweiten Zufallsprozess werden aus einer Sequenz „Q" Beobachtungen „X" erzeugt.

$q_1, q_2, \dots, q_T \rightarrow x_1, x_2, \dots, x_T$: des Weiteren muss immer $q_0 = S_1$ und $q_{T+1} = S_N$ sein. Im Anfang- und Endzustand werden keine Beobachtungen erzeugt.

Die Zustandsübergangswahrscheinlichkeit a_{ij} bezeichnet, mit welcher Wahrscheinlichkeit vom Zustand S_i in den Zustand S_j gewechselt wurde $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$.

$b_j(x_t)$ bezeichnet die Wahrscheinlichkeit, dass die Beobachtung x_t aus dem Zustand $S_j$erzeugt wurde $b_j(x_t) = P(x_t | q_t = S_j)$.

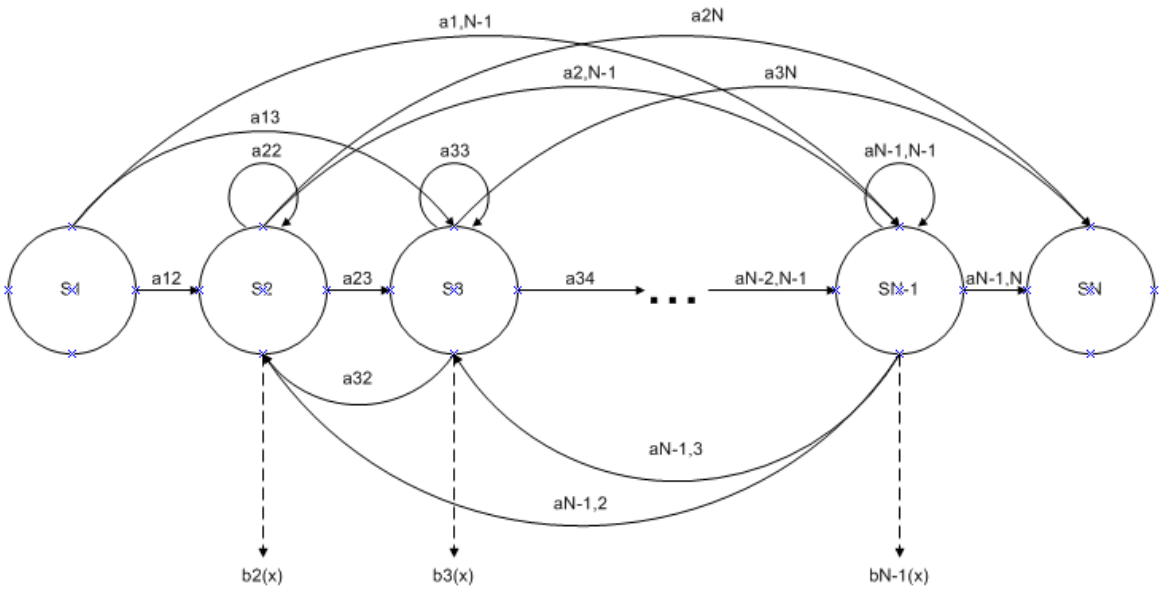

Abbildung 10: Zustandsdiagramm eines HMM mit N-Zuständen(3 S. 110)

[13] Wikipedia-Artikel zu HMM http://de.wikipedia.org/wiki/HMM (zuletzt besucht: 20.10.09)

Um den Prozess vollständig zu beschreiben, wird λ verwendet. λ setzt sich zusammen aus einem Tupel von Matrizen. $\lambda = \{A, B\}$ wobei A eine N x N Matrix und B eine (N-2) x M Matrix ist.

$$A = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1,N-1} & 0 \\ 0 & a_{22} & a_{23} & \cdots & a_{2,N-1} & a_{2N} \\ 0 & a_{32} & a_{33} & \cdots & a_{3,N-1} & a_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{N-1,2} & a_{N-1,3} & \cdots & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} b_2(1) & b_2(2) & b_2(3) & \cdots & b_2(M-1) & b_2(M) \\ b_3(1) & b_3(2) & b_3(3) & \cdots & b_3(M-1) & b_3(M) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{N-1}(1) & b_{N-1}(2) & b_{N-1}(3) & \cdots & b_{N-1}(M-1) & b_{N-1}(M) \end{bmatrix}$$

Formel 1: Matrizen A und B(3 S. 110, 112)

Hierbei handelt es sich um diskrete Beobachtungen, d. h. es können nur M verschiedene Werte resultieren. Diese Variante wird als „discrete density hidden Markov Model" kurz DDHMM bezeichnet. Können sich beliebig viele verschiedene Werte ergeben, handelt es sich um das „continuous density hidden Markov Model" kurz CDHMM.

Beim Links-Rechts-HMM darf nie zurück gesprungen werden. Es sind also nur Sprünge nach vorne oder zu sich selber erlaubt. Noch einschränkender ist das Lineare HMM. Hier darf nur in den nächsten oder den eigenen Zustand gewechselt werden. Aus dem Zustand S_i kann somit nur in den Zustand $S_i$oder S_{i+1} gewechselt werden.

Im dem nachfolgendem Beispiel sind die Sprünge beim Erkennen des Wortes „one" dargestellt. Die roten Pfeile zeigen den wahrscheinlichsten Weg (Startzustand – W – AH – N – Endzustand)

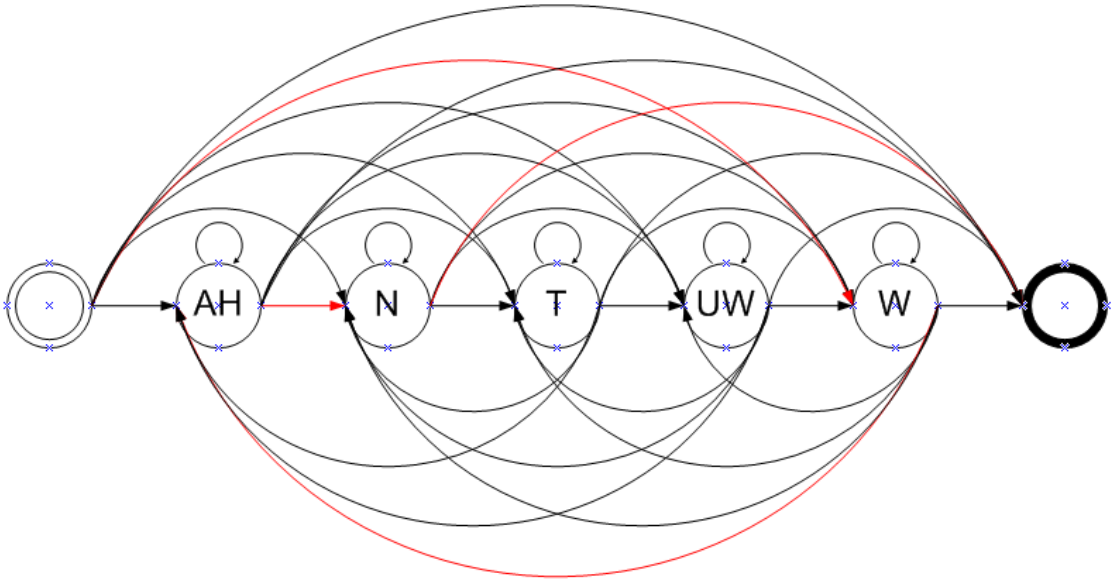

Abbildung 11: HMM mit wahrscheinlisten Weg von One

5.6. Mel Frequency Cepstral Coefficients[14] [15](1)

Eine Vorverarbeitung des digitalen Signals wird mit Hilfe des Sprachmerkmals Mel frequency cepstral coefficients (MFCC) gemacht. Hierbei wird automatisch versucht, das Signal von der Tonlage zu trennen. Dies bedeutet, dass zusätzliche Frequenzen, welche nicht die eigentlichen Informationen speichern, entfernt werden.

Für die Berechnung des MFCCs wird zuerst das zu verarbeitende Signal in Blöcke / Fenster unterteilt. Dies geschieht zum Beispiel entweder anhand der Hamming-Fensterfunktion[16] oder einer fixen Länge des Analysefensters. Diese Länge wird experimentell definiert, der Erfahrungswert liegt bei 25 ms.

Anschliessend werden für jedes dieser Fenster die Merkmalsvektoren berechnet. Dazu kann z. B. die (diskrete) Fouriertransformation[17] oder das Mel-Cepstrums verwendet werden. Unterschiede liegen in der Frequenzskala. Die Einheit Mel (Tonheit) wurde aufgrund von Erkenntnissen aus der Psychoakustik definiert. So wird die wahrgenommene Tonhöhe berücksichtigt und nicht die absolute Frequenz.

Als nächstes wird das Betragsspektrum[18] berechnet, welches anschliessend logarithmisiert wird.

---

[14] Wikipedia-Artikel zu MFCC http://de.wikipedia.org/wiki/Mfcc (zuletzt besucht am: 01.12.09)

[15] Informationen zu den Cepstrum Koeffizienten: http://www5.informatik.uni-erlangen.de/fileadmin/Persons/NiemannHeinrich/klassifikation-von-mustern/m00-www.pdf Seiten 213-216 (zuletzt besucht am: 01.12.09)

[16] Die Fenster werden mit Hilfe von Ein- bzw. Ausblenden der Anfänge bzw. Ausgänge des Fensters künstlich periodisiert; dies verhindert unendlich lange Signale.
http://de.wikipedia.org/wiki/Fensterfunktion (zuletzt besucht am: 01.12.09)

[17] Erklärung zu Fouriertransformation http://de.wikipedia.org/wiki/Fouriertransformation (zuletzt besucht am 11.12.09)

[18] Bei der Aufspaltung eines Signals in seine Frequenzanteile wird ein Spektrum generiert. Dieses kann wiederum in ein Betragsspektrum und ein Phasenspektrum zerlegt werden.
http://www.lexikon-motorentechnik.com/?I=8389&R=H (zuletzt besucht am: 17.12.09)

5.7. Perceptual Linear Prediction[19]

Perceptual Linear Prediction (PLP) ist eine Kombination der Diskreten Fourier Transformation (DFT) und der Linearen Vorhersage (engl. Linear Prediction LP). Als erstes wird das DFT- Spektrum berechnet und an die psychoakustischen Eigenschaften des Ohres angepasst.

Diese Informationen werden für die Merkmalsextraktion verwendet.

5.8. Diskrete Fourier Transformation[20]

In der Signalverarbeitung wird die Diskrete Fourier Transformation verwendet, um mehr Informationen über die vorkommenden Frequenzen zu erhalten. Zum einen sind dies alle vorkommenden Frequenzen und deren Amplituden. Zum anderen werden damit digitale Filter mit grossen Filterlängen implementiert.

5.9. Linear Prediction[21]

In der Zeitreihenanalyse gibt es das mathematische Verfahren Lineare Vorhersage (engl. Linear Prediction). Das Ziel dabei ist es, für das analysierte Signal anhand der bisherigen, die zukünftigen Werte zu schätzen.

---

[19] Erklärung zu PLP http://books.google.ch/books?id=IiI23SEwNPMC&pg=PA46&lpg=PA46&dq=plp+spracherkennung&source=bl&ots=geOasbySzQ&sig=b-gfs5kyKtHH11I_-tEHE45bIa54&hl=de&ei=IPPdSoLUO5e1sAabg5G2Dg&sa=X&oi=book_result&ct=result&resnum=3&ved=0CBQQ6AEwAg#v=onepage&q=&f=false Seiten 48, (zuletzt besucht am: 08.12.09)

[20] Informationen zu DFT: http://de.wikipedia.org/wiki/Diskrete_Fourier-Transformation (zuletzt besucht am: 15.12.09)

[21] Wikipediaseite über Linear Prediction: http://de.wikipedia.org/wiki/Lineare_Vorhersage (zuletzt besucht am 15.12.09)

6. Evaluation bestehender Spracherkennungssysteme

6.1. Vorgaben anhand unserer Aufgabenstellung

Die Spracherkennung von Funksprüchen hat andere Grundlagen als diejenige von Callcenters oder Systemsteuerungen. Aus diesem Grund sind in der nachfolgenden Liste die wichtigsten Punkte erklärt, welche bei der Erkennung von Funksprüchen relevant sind.

- Dictionary: Im Flugfunk wird nur ein beschränktes Vokabular verwendet.
- Grammatik: Die Flugfunkregeln, welche von den Piloten beachtet werden sollten, können in eine Grammatik umgewandelt werden.
- Signalbandbreite: Die verwendete Bandbreite darf höchstens 12.5 kHz betragen. Siehe Kapitel 5.4.3.
- Nebengeräusche: Beim Flugfunk gibt es viele Nebengeräusche, zum Beispiel durch die Turbinen.
- Unidirektional: Wenn unser System den Funkspruch nicht erkennen kann, da evtl. die Verbindung noch schlechter war, kann von den Funkern keine Wiederholung verlangt werden. Vermutlich können hier die Quittierungen von Nutzen sein.
- Training: Die Piloten werden kaum ein Training absolvieren, damit die Aussprache vom System besser verstanden wird.

6.2. Evaluation einzelner Systeme / Programme

Gemäss den vorher beschriebenen Punkten wurden Kriterien festgelegt, welche von in Frage kommenden Systemen erfüllt werden müssen. In den nachfolgenden Unterkapiteln werden die zwei besten sich auf dem Markt befindenden Systeme genauer beschrieben. Zum einen ist dies das konventionelle Produkt Dragon Naturally Speaking, welches in den Tests gute Ergebnisse hatte und zum anderen das Open Source Framework Sphinx 4.

Die weiteren betrachteten Systeme befinden sich im Anhang:

12.3.2. Google Voice: Ein ganzes System, bei welchem unter anderem auch Gesprochenes in Text umgewandelt wird. Die Anmeldung dauert jedoch mehrere Wochen und die Spracherkennung wird als noch verbesserbar bezeichnet.

12.3.3. Loquendo: Dieses konventionelle Produkt wird vor allem in Callcenters und Software– bzw. Systemsteuerungen eingesetzt. Leider kann es nicht wirklich getestet werden.

12.3.7. Voice Pro: Voice Pro wird vor allem unter Windows verwendet, also als Software- und Systemsteuerung. Es hat leider erhebliche Probleme mit Nebengeräuschen.

6.2.1. Dragon Naturally Speaking

Bei Recherchen im Internet stösst man früher oder später immer auf dieses Programm. Es bietet zwei Anwendungsmöglichkeiten: zum einen kann das Programm verwendet werden um Texte zu diktieren; zum anderen können damit auch das Betriebssystem und verschiedene Anwenderprogramme ferngesteuert werden. Dragon Naturally Speaking ist eine der meist verbreiteten und meist empfohlenen „out of the box solution" im Bereich der Spracherkennung und Sprachsteuerung. Das Produkt ist auch als Entwicklerversion verfügbar. Ein für uns wichtiges Feature ist, dass das Programm nicht nur Eingaben über das Mikrofon akzeptiert, sondern auch mittels Audiodateien in verschiedenen Formaten. Bei abnehmender Qualität der Soundfiles sinkt die Erkennungsrate rapid. Auch Hintergrundgeräusche bereiten dem Programm grosse Schwierigkeiten. Nachdem das Programm einige Sätze erkannt hat, versucht es einzelne Worte zu korrigieren. Erkannte Texte werden umgeformt durch Einbezug bekannter Grammatiken. Dies verfälscht das Ergebnis, sofern der zu erkennende Text nicht der Normkonvention entspricht. Leider lässt sich dieses - für den normalen Gebrauch des Programms sehr nützliche - Feature nicht ausstellen.

Weitere Angaben im Anhang unter 12.3.1. Dragon Naturally Speaking.

6.2.2. Sphinx4-1.Obeta3[22]

Dieses Open Source Produkt ermöglicht es, mittels Java Sprache zu erkennen. Der erste Test funktionierte jedoch nur mit den mitgelieferten Beispiel-Dateien zuverlässig. Parallel zu unseren Tests hat sich auch Herr Andreas Maag mit Sphinx beschäftigt. Seine Tests konnten erste Erfolge verbuchen.

Die Tests sind im Anhang im Kapitel 12.3.4. Sphinx4.0beta3 ausführlicher beschrieben.

[22] Forum über Sphinx 4.0 http://www.linux-club.de/viewtopic.php?f=18&t=69703&start=0 (zuletzt besucht am: 20.10.09)

6.3. Übersichtstabelle über die Evaluation der zwei besten Systeme

In der nachfolgenden Tabelle werden die zwei Systeme anhand folgender Kriterien verglichen.

- Spracherkennung: Wie gut können diktierte Texte in guten Umgebungssituationen erkannt werden?
- Nebengeräusche: Wie wirken sich die Nebengeräusche des Funkspruches aus?
- Buchstabieralphabet: Inwiefern wurde das Buchstabieralphabet bereits implementiert oder kann es noch eingefügt werden?
- Training: Inwieweit kann das System trainiert werden, sodass die Erkennung der Funksprüche verbessert werden kann? Muss im Training ein vorgegebener Satz gesprochen werden oder kann dieser frei gewählt werden?
- Kosten: Wie viel kostet das Produkt?
- Beeinflussbarkeit: Was kann alles verändert bzw. angepasst werden?
- Einsatzbereich: In welchen Bereichen wird dieses Produkt am meisten eingesetzt?
- Ausführlichkeit der Tests: In wie weit wurden eigene Tests durchgeführt?
- Besonderheiten: Gibt es sonst ein wichtiges Kriterium speziell für dieses Produkt?

Kriterium	Gewichtung	Dragon Natu-rally Speaking	Sphinx 4.10 Beta 3
Spracherkennung	A	9	7
Nebengeräusche	A	4	6
Buchstabieralphabet	A	10	5
Training	A	8	SphinxTrain
Kosten	B / C	Standard: ca. 100 Euro Professional: ca. 1000 Euro SDK: auf Anfrage	Open Source
Verfügbarkeit	B	Auf Anfrage	10
Beeinflussbarkeit	C	4	9
Einsatzbereich		Programm Fernsteue-rung und Text diktieren	vor allem in Projekten, bei welchen eigene Pa-rameter relevant sind
Ausführlichkeit der Tests		7	8
Besonderheiten		sehr verbreitet und häufig empfohlen.	

Tabelle 3: Evaluationsübersicht über Dragon und Sphinx

Legende:	A:	muss	10:	perfekt
	B:	soll	5:	möglich
	C:	kann	0:	gar nicht

6.4. Entscheidung

Aus Zeitgründen wurde bereits in der vierten Woche entschieden, mit welchem System weitergearbeitet wird. Die Entscheidung fiel auf Sphinx 4. Der Hauptgrund dafür ist die Möglichkeit zur Anpassung der einzelnen Komponenten, welche in den Dokumentationen angepriesen wird. Mit dieser Möglichkeit erhoffen wir uns, das System soweit anpassen zu können, dass die Erkennung von Funksprüchen zu einem grossen Prozentsatz erreicht werden kann. Ein weiterer wichtiger Grund ist, dass Sphinx Open Source und somit kostenlos ist.

Als Konsequenz werden sich die Nachforschungsarbeiten vor allem auf Sphinx und die benötigten Algorithmen fixieren.

7. Sphinx 4 (2)(3)[23]

7.1. Allgemein

Sphinx 4 ist ein Java-basiertes Open-Source Framework zur Spracherkennung, welches sich flexibel verändern lässt. Der Programmaufbau ist mit verschiedenen Modulen gelöst, so dass diese auch während der Laufzeit ausgetauscht werden können. Von dieser Option werden wir in diesem Projekt keinen Gebrauch machen. Was aber sicher verändert wird, sind die konfigurierbaren Parameter.

Die Basis für die Spracherkennung liefert der HMM-Algorithmus. Dieser ist im Kapitel 5.5. ausführlich beschrieben.

7.2. Entwicklungsgeschichte von Sphinx[24]

Die Carnegie Mellon University (CMU)[25] hat bereits im Jahre 1986 angefangen Sphinx zu entwickeln. Seitdem wurden mehrere Versionen veröffentlicht, ab 2000 als Open Source. Gegen Ende der Arbeit zeigte sich, dass Sphinx weiterentwickelt wird, denn der Internetauftritt wurde zum Teil erneuert.

Nachfolgend werden die einzelnen Projekte kurz beschrieben, welche alle den HMM und die Wahrscheinlichkeitsdichtefunktion[26] implementieren.

7.2.1. Sphinx

Die erste Version von Sphinx wurde im Jahre 1986 veröffentlicht. Es ist ein sprecherunabhängiges Spracherkennungssystem, welches auf HMM basiert und n-Gram Statistiken nutzt. Heute hat es nur noch geschichtliche Bedeutung, denn hinsichtlich der Performance ist es längst überholt.

---

[23] Offizielle CMU-Sphinx Webseite: http://cmusphinx.sourceforge.net/html/cmusphinx.php (zuletzt besucht am 08.12.09)

[24] Wikipedia-Seite über CMU Sphinx: http://en.wikipedia.org/wiki/CMU_Sphinx (zuletzt besucht am: 02.12.09)

[25] Globale Forschungsuniversität in Pittsburgh USA: http://www.cmu.edu/about/index.shtml (zuletzt besucht am: 15.12.09)

[26] Deutsch Erklärung für probability density function (PDF): http://www.babylon.com/definition/probability_density_function/German (zuletzt besucht am: 08.12.09)

7.2.2. Sphinx 2

Das Ziel dieses Recognizers war es, die Spracherkennung möglichst schnell zu erreichen, so dass Real-Time Programme möglich wurden. Sphinx 2 wurde am Linux-World Event im Jahre 2000 als Open Source herausgegeben.

Diese Version von Sphinx wurde in einige konventionelle Produkte integriert, auch in nicht kostenpflichtige wie das Betriebssystem Ubuntu. Doch wird es nicht mehr weiter entwickelt, da es von PocketSphinx abgelöst wurde.

7.2.3. Sphinx 3

Bei Sphinx 3 wurde der Fokus auf eine hohe Erkennungsrate gesetzt. Dies bedeutet, dass Real-Time nicht mehr das Ziel war. Nichtsdestotrotz kann diese Version nicht für kritische Interaktionen verwendet werden. Sphinx 3 wird noch weiterentwickelt.

7.2.4. Sphinx 4

Wie bereits erwähnt, handelt es sich bei Sphinx 4 um ein Java Framework. Die Entwicklung wurde von Sun Microsystems unterstützt. Die derzeitigen Entwicklungsziele sind: ein neuer Trainer, Verbesserung des Konfiguration-Managements und ein grafisches User Interface.

Die nachfolgende Tabelle zeigt, dass die Entwicklung mit Unterbrüchen vonstattengeht.

Version	Erscheinungsdatum	Anz. Downloads
Sphinx 4 1.0 Beta 3	17.08.09	9'269
Sphinx 4 1.0 Beta 2	21.02.09	12'919
Sphinx 4 1.0 Beta 1	28.09.04	97'878
Sphinx 4 1.0 Alpha	03.06.04	9'765

Tabelle 4: Sphinx 4 Release-Übersicht

7.2.5. PocketSphinx

Die Version von Sphinx für eingebettete Systeme ist PocketSphinx. Es ist das schnellste Spracherkennungs-System von CMU, aber nicht das fehlerfreiste. Aus diesem Grund wird weiterentwickelt, insbesondere um bessere Algorithmen zu implementieren.

7.2.6. SphinxTrain[27]

Der SphinxTrain wird verwendet, um Modelle für Sphinx 3 zu erstellen, welche auch von Sphinx 4 verwendet werden können.

Nähere Informationen befinden sich im Anhang: Kapitel 12.3.5. SphinxTrain.

7.3. Komponenten

Nachfolgend werden die wichtigsten Komponenten von Sphinx 4 beschrieben. Die englischen Original-Bezeichnungen, welche auch als Klassen oder Interfaces verwendet werden, wurden beibehalten.

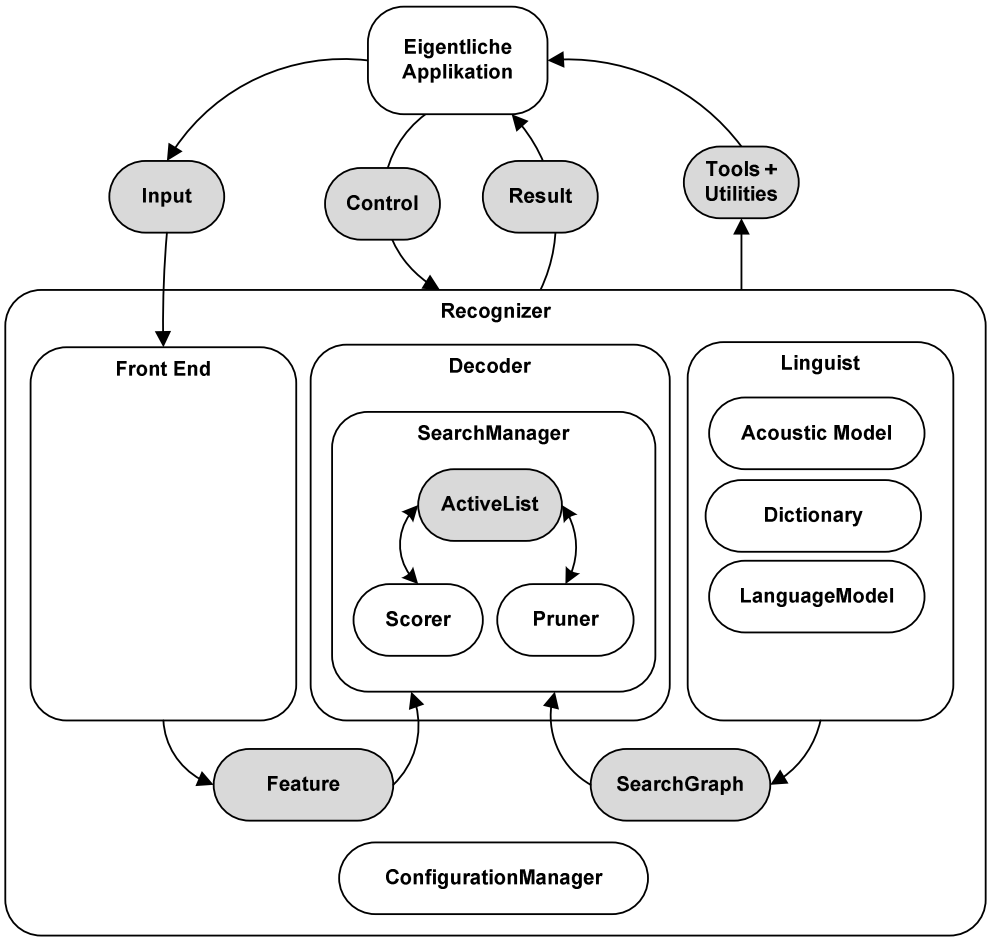

Abbildung 12:
Übersicht über die wichtigsten Komponenten von Sphinx 4(10 S. 2)

[27] Erklärung zum SphinxTrain
http://cmusphinx.sourceforge.net/sphinx4/doc/UsingSphinxTrainModels.html
(zuletzt besucht am: 20.10.09)

7.3.1. Recognizer

Unter dem Begriff Recognizer werden mehrere Module von Sphinx zusammengefasst. Diese sind verantwortlich für die Erkennung des gewünschten Sprachinputs und das Übermitteln des Resultates an die eigentliche Applikation, welche auf den ganzen Prozess Einfluss nehmen kann, indem sie Controls sendet.

7.3.1.1. FrontEnd

Das FrontEnd ist das erste Modul, welches das Sprachsignal durchläuft. Hier wird das Signal vorverarbeitet, so dass es als Sequenz von Features weitergegeben werden kann. Diese Vorverarbeitung wird durch DataProcessors ermöglicht. So kann das gleiche Signal zu verschiedenen Features führen, denn die DataProcessors können verschiedene Parameter haben, z.B. MFCC[28] oder PLP[29]. Diese DataProcessors können parallel ablaufen, auch können Outputs wieder als Inputs verwendet werden. So können die Signale mehrmals bearbeitet werden.

7.3.1.2. Decoder

7.3.1.2.1. SearchManager

Der SearchManager nutzt die Features des FrontEnds und den SearchGraph vom Linguist und macht die eigentliche Spracherkennung. Die so generierten Results werden an die Applikation zurück geliefert.

7.3.1.2.2. SearchGraph

Der SearchGraph ist ein gerichteter Graph, welcher alle möglichen Wörter und Sätze aufgrund der vordefinierten Parameter aufzeigt. Die Erstellung und die Speicherung dieses Graphen erfordert je nach Einsatzgebiet viel Speicherplatz und Rechenleistung.

7.3.1.3. Linguist

Der Linguist erstellt anhand der Informationen aus den AcousticModels, dem Dictionary und den LanguageModels einen SearchGraph. Der SearchGraph wird vom Decoder während der Suche verwendet, doch die eigentliche Komplexität wird durch den Linguist gekapselt.

[28] Siehe Kapitel 5.6. Mel Frequency Cepstral Coefficients

[29] Siehe Kapitel 5.9. Linear Prediction

7.3.1.3.1. AcousticModel

Das AcousticModel enthält strukturelle Informationen. Dies sind vor allem mit dem HMM-Algorithmus generierte Zustandsdiagramme. Miteinbezogen werden hier die Wortpositions- und Kontextinformationen.

7.3.1.3.2. Dictionary

Im Dictionary befindet sich die phonetische Schrift, d.h. die Art, wie ein Wort ausgesprochen wird. Hierzu werden alle Wörter, welche im LanguageModel zu finden sind, gemäss dem AcousticModel in Teilwörter gebrochen.

Variante 1: In dieser Variante wird die in Buchstaben umgewandelte phonetischen Schrift aneinandergereiht und so festgehalten, wie ein Wort ausgesprochen wird. So kann ein Wort auch mehrmals erfasst werden, falls es unterschiedliche Aussprechmöglichkeiten gibt. In dieser Variante gibt es bereits ein Dictionary in Englisch mit über 125‘000 Wörtern.

ONE	HH W AH N
ONE(2)	W AH N
TWO	T UW

Variante 2: Diese Variante wird vor allem für kleinere Dictionarys verwendet. Die anschliessend generierten HMMs sind so besser verständlich.

One	W_one AX_one N_one
two	T_two OO_two

7.3.1.3.3. LanguageModel

Das LanguageModel definiert für die jeweilige Sprache die Struktur der Wörter. Dies wird entweder als graphenbasierte Grammatik oder als zufallsabhängige N-Gram Modelle gelöst. Bei der graphenbasierten Grammatik befindet sich in jedem Knoten ein Einzelwort, wobei diese so angeordnet sind, dass bei einer Traversierung eine sinnvolle Kombination entsteht.

Es gibt bereits einige vordefinierte LanguageModels, welche auf unsere Problemstellung angepasst werden können.

Die Grammatik wird im Kapitel 7.4. genauer beschrieben.

7.3.2. ConfigurationManager

Im ConfigurationManager können Parameter verändert und Module während der Laufzeit ausgewechselt werden. Zum Beispiel kann hier definiert werden, welche DataProcessors wann ablaufen.

7.3.3. Tools

Hier werden während der Laufzeit Decoder Statistiken geliefert. So kann zum Beispiel die Word Error Rate (WER), die Geschwindigkeit und die CPU-Auslastung angezeigt werden. Dies ist nützlich, um die Auswirkungen von Parameterveränderungen umgehend analysieren zu können.

7.4. Grammatik

Im Linguist kann zusätzlich noch eine Grammatik verwendet werden. Da sich die Beteiligten im Flugverkehr an eine Grammatik halten sollten, werden wir eine Grammatik implementieren.

Die Grammatik wird in Sphinx in dem nachfolgend beschriebenen JSG-Format definiert. Diese Schreibweise entspricht der in Mathe 2 gelernten Backus-Naur-Form.

7.4.1. Java Speech Grammar Format[30]

Übersicht der Notation:

Notation	Backus-Naur-Form(12)	Weitere Erklärungen
<Bezeichnung>	Regel / Rule	müssen innerhalb der Grammatik eindeutig sein, können bei Bedarf mit dem Package-Namen qualifiziert werden
<NULL>	eps	entspricht dem Epsilon in den mathematischen Grundlagen der Informatik(12)
<VOID>	ERROR	macht einen Satz ungültig
Bezeichnung	Terminale	wird nicht mehr weiter aufgelöst; können auch zusammengesetzte Wörter sein werden im Dictionary gesucht und weiter aufgelöst
|	|	oder
[ ]	[ ]	optional
( )	( )	für die Gliederung; zur Kombination mit oder; für etwas aus der Klammer
public		kann ausserhalb der Grammatik verwendet werden
/x/ terminale		/x/ Gewichtung der Terminale Je grösser x, desto wahrscheinlicher. X ist ein Float und >= 0
*****	* (Kleene-Star)	0..n –Mal
+	+	1..n –Mal
{}		Alles in diesen geschweiften Klammern wird von der Grammatik nicht beachtet. Es wird im Result des Recognizers an die Applikation übergeben.

Tabelle 5: Notation von JSGF im Zusammenhang mit der Backus-Nauer-Form

[30] JSGF Spezifikationen http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html (zuletzt besucht am: 27.10.09)

7.4.2. Beispiel für Flugfunk[31]

<callsign> = <code> "-" (<letter>|<digit>) (<letter>|<digit>) (<letter>|<digit>);

<letter> = (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);

<digits> = (one | two | three | four | five | six | seven | eight | nine | zero);

<code> = (a2 | a5 | a6 | a7 | ap | b | c | c2 | c5 | c6 | cc | cn | cp | cr | cs | cu | cx | d | d2 | d4 | d6 | dm | dq | ec | ei | el | ep | es | et | ez | f | g | h4 | hA | hb | hc | hi | hk | hl | hp | hr | hs | hz | i | j2 | ja | jy | ln | lv | lx | ly | lz | n | ob | od | oe | oh | ok | om | oo | oy | p2 | p4 | ph | pj | pk | pp | pz | ra | rp | s2 | s5 | s7 | s9 | se | sp | st | su | sx | tc | tf | tg | ti | tj | tn | tr | ts | tu | uk | ur | v2 | v5 | v7 | v8 | vh | vn | vp | vr | vt | xa | ya | yi | yj | yk | yl | yr | ys | yu | yv | z | z3 | za | zk | zp | zs);

[31] Länderliste: http://aircraft-registration-country-codes.blogspot.com/ (zuletzt besucht am: 17.12.09)

8. Schlussfolgerungen

8.1. Zusammenfassung

Die Erkennung von Funksprüchen erweist sich als sehr komplex. Dies vor allem deshalb, weil die Funksprüche mit Nebengeräuschen versehen sind, welche sich nicht filtern lassen. Positiv auf die Erkennungswahrscheinlichkeit wirken sich hingegen das einfache Dictionary und die Grammatik aus.

Bei der Evaluation zeigte sich, dass zurzeit kein System auf dem Markt ist, welches die Funksprüche bereits erkennen kann. Insbesondere haben alle Produkte Probleme mit den Nebengeräuschen in den Audiodateien. Nichtsdestotrotz wurde Sphinx 4 genauer betrachtet, da in der Dokumentation die Anpassbarkeit angepriesen wurde. Dass Theorie und Praxis oft zwei verschiedene Paar Schuhe sind, hatte sich auch in diesem Fall bestätigt: was theoretisch einfach aussieht, lässt sich selten Eins zu Eins in die Praxis umsetzen. Die grosse Herausforderung lag vor allem in der Kombination der einzelnen Komponenten. Aus den Tests resultierte, dass die grösste Chance darin besteht, mit Hilfe eines eigenen Dictionarys und einer eigenen Grammatik die Funksprüche zu erkennen. Diese Dokumente wurden für eine Weiterverarbeitung bestmöglich vorbereitet.

8.2. Beurteilung der Resultate

Das Endziel, die Erkennung der Flugnummer aus den Funksprüchen, konnte nicht wie erhofft erreicht werden. Gründe dafür liegen in der Komplexität der Arbeit und des Zeitmangels. Zuerst mussten die Grundlagen in der Spracherkennung und in der Flugfunkgrammatik erarbeitet werden. Die dafür aufgewendete Zeit fehlte anschliessend in der Realisierung des Pilotsystems.

Vor allem in der Anfangsphase hat sich gezeigt, dass den Studenten die Erfahrung mit einer Research-Arbeit fehlt. Dies äusserte sich in der teils zu Beginn unstrukturierten Arbeitsweise. Anfangs war nicht ganz klar in welche Rubrik die Tests der Evaluationsphase verbucht werden müssen siehe Anhang Kapitel 12.1. Zeitplan.

9. Glossar

Abkürzung / Begriff	Beschreibung
.ch	Internetkürzel für Schweiz
.com	Internetkürzel für Commercial
.de	Internetkürzel für Deutschland
.edu	Internetkürzel für Education
.org	Internetkürzel für Organisation
API	Application Programming Interface
ATM	Air Traffic Monitoring
BAZL	Bundesamt für zivile Luftfahrt
BRU	Brühwiler
bzw.	Beziehungsweise
CD	Compact Disc (Speichermedium)
CDHMM	Continous Density HMM
CLASS	Klasse
CMU	Carnegie Mellon University
cnlab	Information Technologie Research Company
CPU	Central Processing Unit; Hauptprozessor eines Computers
CTR	Control Zone
d.h.	das heisst
DataProzessor	Signalverarbeitungsmodul
DDHMM	Discrete Density HMM
DFT	Diskreten Fourier Transformation
Dictionary(10)	wird in den Architecture Notes aus dem Jahre 2002 als „Lexi-con" bezeichnet (11)
docx	Dateiendung für Word 2007 Dateien
Dr	Doktor
ETH	Eidgenössische Technische Hochschule
FH	Fachhochschule
Freeware	Frei verfügbare Software
GRO	Grob
HEI	Heinzmann
HMM	Hidden Markov Model
HS	Herbstsemester
HSR	Hochschule Rapperswil
html	Hypertext Markup Language
http	Hypertext Transfer Protokol
Hz (kHz)	Hertz: SI-Einheit für die Frequenz (entspricht 1000 Hz)
INC	Incorporated
ISBN	International Standard Book Number
JAR	Dateiformat für Java-Programme
JSG	Java Speech Grammar
JSGF	Java Speech Grammar Format
Layer 3	Schicht im OSI-Modell
Linguist(10)	wird in den Architecture Notes aus dem Jahre 2002 als Know-ledge Base bezeichnet (11)
LP	Linear Prediction
MAA	Maag
MDCT	Modified Discrete Cosine Transformation
MFCC	Mel Frequency Cepstral Coefficients (Kapitel: 0)
Mic	Microfon
MP3	MPEG-1 Audio Layer 3; Komprimierte wav-Datei
MPEG	Verfahren zur Video- und Audiodatenkompression

ms	Millisekunden
MS	Meilenstein
N-Gramm	Die Wahrscheinlichkeit eines Wortes hängt nur von den letzten N-1 Wörtern ab. (3 S. 375)
Open Source	Öffentlicher Quelltext
OSI Model	*Open Systems Interconnection Reference Model*
Path	Weg (Verzeichnis)
PDF	Probability density function (Wahrscheinlichkeitsdichtefunktion)
PLP	Perceptual Linear Prediction (Kapitel: 0)
Prof	Professor
Quelle	Ausgangspunkt
RIFF	Resource Interchange File Format
SA	Studienarbeit
SAPI	Speech API
SDK	Software Development Kit
Senke	Endpunkt
SI	Internationales Einheitssystem; Système international d'unité
Speech-to-Text	Sprache in Text umwandeln
Stochastisches Modell	zufallsorientiertes Modell
Tupel	geordnete Werteansammlungen (eindimensionale Arrays)
u. a.	unter anderem
USA	Vereinigte Staaten von Amerika
wav	WAVE-Datei für Speicherung von Audiodaten
WER	Word Error Rate; Rate falsch erkannter Wörter
Wiki	Wikipedia (online Nachschlagewerk)
xlsx	Dateiendung für Excel 2007 Dateien
xml	Extensible Markup Language
z. B.	zum Beispiel

: Glossar

10. Quellenverzeichnis

1. **BAZL.** Situationsplan Zürich Kloten.

2. **Roos, Walter.** *Calling Tower.* Opfikon ZH : Kalt-Zehnder Druck, 1998. 3-905036-06-1.

3. **Pfister, Beat und Kaufmann, Tobias.** *Sprachverarbeitung.* ETH Zürich : Springer-Verlag Berlin Heidelberg, 2008. Buch. ISBN 978-3-540-75909-6.

4. **Rinkel, Andreas.** Vorlesung Informationstheorie. . Hochschule Rapperswil, 2007. Bd. , . .

5. [Online] 04. 12 2009. http://www.itec.uka.de/seminare/redundanz/vortrag14/.

6. [Online] 04. 12 2009. http://de.wikipedia.org/wiki/Quantisierung.

7. [Online] 16. 12 2009. http://www.skoeser.de/bilder/fachliches/glottalisierung/kehlkopf.jpg.

8. [Online] 11. 12 2009. http://www.elektronik-kompendium.de/sites/kom/0401181.htm.

9. [Online] 11. 12 2009. http://de.wikipedia.org/wiki/Amplitudenmodulation.

10. **Walker, Willie, Lamere, Paul und Philip, Kwok.** *Sphinx-4: A Flexible Open Source Framework for Speech Recognition.* s.l. : SUN MICROSYSTEMS INC., 2004. Whitepaper.

11. **Walker, Willie, Lamere, Paul und Kwok, Philip.** *Sphinx 4 for Java platform, Architecture Notes.* s.l. : SUN MICROSYSTEMS INC, 2002. Whitepaper.

12. **Wirth, Joachim und Walter, Bichsel.** *Vorlesung: Mathematische Grundlagen der Informatik 2.* HSR, 2008. .

13. **Ewender, Thomas.** Möglichkeit von Erkennung der Funksprüche. ETH Zürich : E-Mail-Kontakt, 5.-7. Oktober 2009.

14. [Online] 15. 11 2009. http://de.academic.ru/pictures/dewiki/70/Flughafen_Basel_Mulhouse.jpg.

15. [Online] 15. 10 2009. http://www.stupidedia.org/images/thumb/6/6d/Swiss_Air_Flugzeug.jpg/300px-Swiss_Air_Flugzeug.jpg.

16. [Online] 16. 12 2009. http://www.optik-brandstaetter.at/img/ohr.jpg.

11.Abbildungsverzeichnis

12. Anhang

12.1. Versionskontrolle

In der nachfolgenden Tabelle sind die Versionen aufgelistet, welche zwischenzeitlich entstanden.

Version	Datum	Beschreibung
0.08	20.10.09	1. Vorabversion; Zur ersten Beurteilung durch P. Heinzmann
0.12	24.11.09	2. Vorabversion; Zur Information von A. Doering
1.0	18.12.09	Abgabe des Berichts

Tabelle 7: Versionskontrolle des Berichts

12.2. Protokolle

Eine Liste aller Protokolle befindet sich in der nachfolgenden Tabelle. Zusätzlich zu den Verweisen auf die Dokumente werden auch kurz die jeweiligen Schwerpunkt beschrieben.

Protokoll	Schwerpunkte
20090915-SA-ATM-Voice-Protokoll-01_V1.1.docx	Startup Meeting Einführung in ATM-Plattform Zielsetzung
20090923-SA-ATM-Voice-Protokoll-02_V1.1.docx	Erste Evaluation von Systemen Kriterien für Evaluation Bericht Gliederung Definition von Meilensteinen
20090930-SA-ATM-Voice-Protokoll-03_V1.1.docx	Evaluationsvertiefung
20091006-SA-ATM-Voice-Protokoll-04_V1.1.docx	Zwischenbericht
20091014-SA-ATM-Voice-Protokoll-05_V1.0.docx	Kurzzusammenfassung
20091021-SA-ATM-Voice-Protokoll-06_V1.1.docx	Inputs des Industriepartners Grundlagen des Flugfunks
20091029-SA-ATM-Voice-Protokoll-07_V1.1.docx	Stand mit Sphinx
20091104-SA-ATM-Voice-Protokoll-08_V1.1.docx	Besprechung des Teilberichts Präsentationsinhalt
20091111-SA-ATM-Voice-Protokoll-09_V1.0.docx	Zwischenpräsentation
20091118-SA-ATM-Voice-Protokoll-10_V1.1.docx	Zwischenpräsentation Sphinx noch ohne Durchbruch
20091124-SA-ATM-Voice-Protokoll-11_V1.0.docx	Bericht
20091204-SA-ATM-Voice-Protokoll-12_V1.1.docx	Ende der Implementation

Tabelle 8: Übersicht über Protokolle

12.1. Zeitplan

12.1.1. Meilensteine

In der nachfolgenden Tabelle sind die Meilensteine definiert. Diese konnten termingerecht eingehalten werden.

Projektstart:		14.07.2009
MS1: Entscheidungsstichtag	Entscheidung der Durchführbarkeit	06.10.2009
MS2: Abgabe 1. Teil von Berichts	Teil zur Begutachtung an HEI	14.10.2009
MS3: Zwischenpräsentation		11.11.2009
MS4: Abschluss Implementation	Fokus auf Bericht setzen	03.12.2009
MS5: Abgabe Bericht	CD an Sekretariat, Mail an HEI	18.12.2009
MS6: Präsentation		22.12.2009
Projektende:		22.12.2009

Tabelle 9: Übersicht über Meilensteine

12.1.2. Stunden pro Kategorie

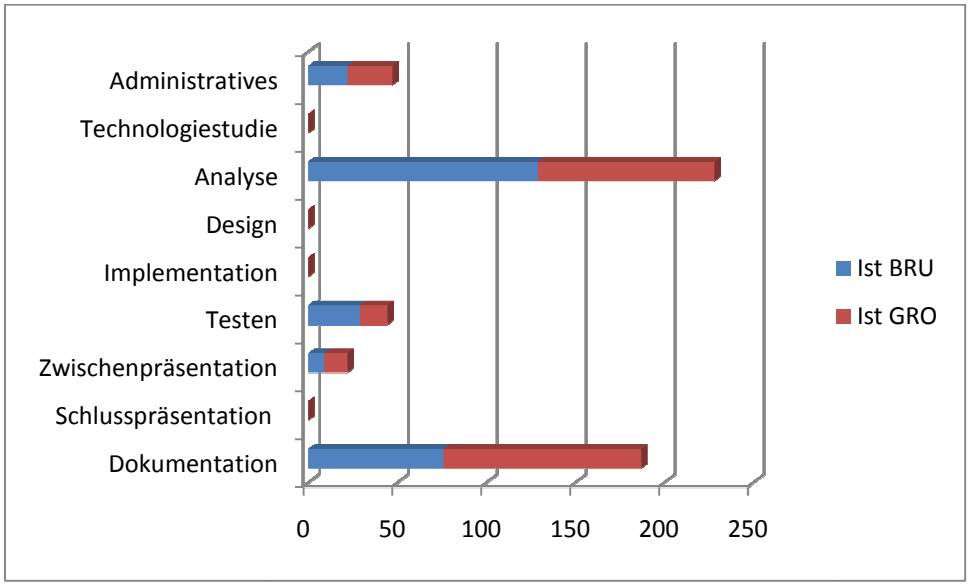

Abbildung 13: Stundenübersicht

Wie die Stundenübersicht zeigt, wurde erst gegen Ende realisiert, wie in dieser Arbeit die Stunden zu erfassen sind. Bis weit in die Semesterwochen haben wir alle Arbeiten, welche mit der Analyse von Spracherkennungssystemen in Zusammenhang standen, unter Analyse verbucht. Erst später wurde uns bewusst, dass wir höchstwahrscheinlich keine eigentliche Implementation machen werden und somit auch nicht die klassischen Test und Design Phasen haben werden. Als wir dies realisierten, änderten wir die Zeiterfassung, aber nicht rückwirkend, da diese nur auf Schätzungen beruht hätte.

12.2. Erfahrungsberichte

Die Erfahrungsberichte der Studenten befinden sich im externen Dokument: Erfahrungsberichte.docx.

12.3. Detaillierte Testberichte

12.3.1. Dragon Naturally Speaking

Versuchsaufbau

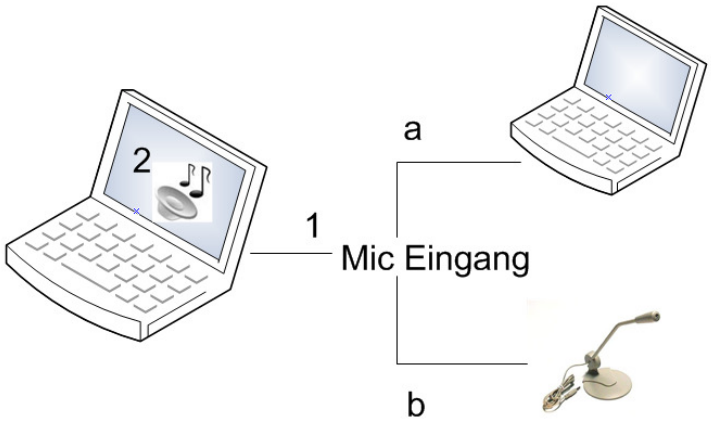

Abbildung 14: Versuchsaufbau Dragon Naturally Speaking

Für die Versuche wurden auf dem linken Notebook Dragon Naturally Speaking installiert und 2 Profile angelegt (siehe Grundeinstellungen). Das verwendete Profil ist mit der Nummer in der Zeichnung veranschaulicht. Beim 1. Profil gibt es zwei unterschiedliche Quellen, welche am Mikrofon Eingang angeschlossen werden. Die Kleinbuchstaben a/b veranschaulichen, welche Versuchsaufbauquelle verwendet wird

Grundeinstellungen

Bevor das Programm das Gesprochene in Text umwandeln kann, muss ein Profil erstellt werden. Beim Erstellen muss festgelegt werden, welches der Inputkanal ist. Das 1. Profil haben wir mit einem Input-Mikrofon angelegt. Dabei mussten auch einige wenige Sätze vorgelesen werden, dies ist jedoch noch nicht das eigentliche Training. Dieses ist optional, wird jedoch empfohlen. Es könnte auch erst zu einem späteren Zeitpunkt durchgeführt werden. Da unser Ziel das Erkennen von Funksprüchen von verschiedenen Personen ist, wollten wir das Programm nicht an unsere Stimmen gewöhnen. Deshalb wurde das Training zunächst weggelassen. Zudem lernt das System selbstständig, während es benutzt wird. Wir haben noch ein zweites Profil angelegt, welches Audiodateien als Input verwendet. Auch hier musste ein Text vorgelesen und mit Hilfe eines Aufzeichnungsprogramms als Audiodatei gespeichert werden; wir haben den „Audiorecorder" von Windows verwendet.

Versuch 1

Aufbau:

Profil: 1(Mic)
Input: Headset (b)
Output: Notepad

Im ersten Versuch haben wir einige Sätze diktiert. Diese wurden von Dragon Naturally Speaking in Text umgewandelt.

Soll:

This is a test dictation. It's the first try to interact with the software. Because of the good conditions I hope there should be no problem to understand these sentences.

Ergebnis:

This is a test dictation it's the first try to interact with this software because of the good condition I hope there should be no problem to understand the sentence.

Fazit:

Das Ergebnis stimmte uns zuversichtlich. Die Eingaben wurden sehr gut erkannt. In verschiedenen Durchläufen zeichnete sich ab, dass Hintergrundgeräusche das Ergebnis deutlich negativ beeinflussen.

Versuch 2

Aufbau:

Profil: 1(Mic)
Input: Live Stream (Tower Flughafen Genf) (a)
Output: Notepad

Im zweiten Experiment wurde auf einem Notebook die Live-Übertragung des Towers in Genf übermittelt. Auf dem anderen Notebook lief Dragon Naturally Speaking und versuchte, den Funkverkehr in Text umzuwandeln. Die beiden Notebooks wurden mittels Klinkenkabel (male/male) verbunden. Beim abspielenden Notebook wurde das Kabel in den Kopfhörerausgang gesteckt und beim aufzeichnenden Notebook in den Mikrofonein-

gang. Um trotzdem mithören zu können, wurde beim Kopfhörerausgang ein Doppelstecker angebracht, um zwei Kabel anschliessen zu können. So war es möglich, gleichzeitig einen Kopfhörer und das andere Notebook anzuhängen. Es resultierten keine hörbaren Unterschiede beim Empfänger. Um auf Nummer sicher zu gehen, wurde dasselbe Experiment auch ohne Doppelstecker durchgeführt.

Soll:

Da der Textinhalt schwer verständlich war und sehr schnell gesprochen wurde, war es uns nicht möglich mitzuschreiben. Es konnte auch nicht zurück gespult werden, da es sich um eine Liveübertragung gehandelt hat.

Ergebnis:

What I mean that it is in and it is him him him him them as a is a and I will and him and he let me than it is 5 hundred and a bad is the IP header is him and that I and pay to have him in any way to say it is a then I know it is okay when you are done in the minute I can add it to your IE is as good as it is okay if you are in a good one is that it is okay and it is you who are in him and is as good as it is about an update on what you will about him that I had to is that there is quite a

I'm in an is a day and a him and that he is the one you up to the Iowa is that it is a political line is that when it is is it is okay to benefit them if I had him about my

Fazit:

Auch ohne die Möglichkeit, mit einem Sollzustand vergleichen zu können, ist klar ersichtlich, dass das Ergebnis nicht richtig sein kann.

Das Ergebnis war ernüchternd. Es wurde praktisch nichts erkannt. Vereinzelte Worte wurden erkannt, jedoch sehr selten. Es wurde sehr häufig das Wort „is" erkannt. Dies deutet darauf hin, dass das Rauschen die Erkennung stört. Das Programm versucht nach dem Erkennen der Wörter Sätze zu bilden. Dadurch werden bereits einige Anzeichen gelöscht. Der Text beinhaltete vor der Korrektur noch mehr dem Rauschen ähnliche Wörter.

Versuch 3

Aufbau:

Profil: 2 (Audiodatei)
Input: MP3 Datei (a)
Output: Notepad

Da die Qualität des übertragenen Funkverkehrs im Versuch 2 sehr schlecht war, versuchten wir die Umwandlung in Text anhand erhaltener Funk-Beispiel-Dateien zu testen. Diese waren qualitativ besser. Mit etwas Übung waren diese Dateien für uns verständlich.

Soll:

Good Morning. Weather information Fox ... 0726 UTC Runway news 1 5 for departure please use preferable runway 6 2 expect procedure altix radar ALS runway 1 5 interception 3800ft ...

Ergebnis:

he meet me you and me and what do you think I'd like to tell you that you will make a good runtime at home and I am good at everything he will not only did he

Fazit:

Den gewünschten Erfolg brachte auch dieser Versuch nicht. Der Ansatz mit den Dateien war richtig. Es ist wichtig, dass wir uns auf die Erkennung der Dateien konzentrieren. Sollte dies funktionieren, ist es immer noch möglich, dasselbe mit einer Live-Übertragung zu versuchen.

Versuch 4

Aufbau:

Profil: 1 (Mic)
Input: MP3 Datei (a)
Output: Notepad

Das Ziel war nun, die Dateien in Text umwandeln zu lassen. Da mit dem Profil 2 bisher noch kein tolerierbares Ergebnis erzielt werden konnte, brauchten wir einen neuen Versuchsaufbau. Versuch 1 hat gezeigt, dass die Erkennung über den Mikrofoneingang funktioniert; deshalb kombinierten wir Versuch 2 und Versuch 3. So konnten wir die Dateien mit dem Profil 1 ausprobieren. Die beiden Notebooks wurden wie in Versuch 2 verbunden. Anstelle der Live-Übertragung wurde eine Datei verwendet. Dies ermöglichte auch die Wiederholung des Versuchs unter gleichen Bedingungen.

Soll:

Good Morning. Weather information Fox ... 0726 UTC Runway news 1 5 for departure please use preferable runway 6 2 expect procedure altix radar ALS runway 1 5 interception 3800ft ...

Ergebnis:

Good morning to you and hopefully to be given to you UTC grimly neutral and trying with much of your program and retreated they could compete in a Detroit out of her website into the country to 800 Jimmy, the infamous as you is the computer and 2000° for the BBQ 1900 m read aloud a few 208 and…

Fazit:

Das Diktieren mit der eigenen Stimme hat sehr gut funktioniert. Leider war kein Versuch mit Dateien oder Streams nur annähernd so erfolgreich. Dies ist auf diverse Gründe zurückzuführen. Einerseits liegt es an den Hintergrundgeräuschen, dem Rauschen. Des Weiteren ist die Qualität nicht die Beste. Ein grosser Einfluss hat auch die Korrektur, welche das Programm eigenständig vornimmt. Das Programm versucht, das Erkannte in einen sinnvollen Zusammenhang zu bringen. In unserem Fall hat dies das Ergebnis weiter verschlechtert. Leider lässt sich diese Funktion nicht abstellen. Zumindest nicht bei der Standard Version.

12.3.2. Google Voice[32]

Google Voice ist ein System, welches entwickelt wurde, um dem Benutzer eine Rufnummer zu ermöglichen, bei welcher er bestimmen kann, ob, wo und wie er den Anruf empfangen möchte. So kann er eine Nummer für das Mobiltelefon, das private und geschäftliche Festnetz und seinen Computer verwenden. Innerhalb dieses Systems können die Nachrichten auch in Text umgewandelt werden. Doch die Spracherkennung wird sogar von Google im Anpreisungs-Video als teilweise fehlerhaft und ausbaunötig bezeichnet.

Der angeforderte Account hätte erst einen Monat nach Antrag eröffnet werden können und leider nur aus den USA. Diesen Bescheid erhielten wir erst nach der Entscheidung, welches Produkt genauer betrachtet wird.

12.3.3. Loquendo[33]

Auf der Webseite von Loquendo befinden sich bedauerlicherweise nur Demos von Text-in-Sprach-Umwandlungen. Die selber eingegebenen Texte haben nicht funktioniert. Deshalb ist nicht klar erkenntlich, was wirklich klappt. In die andere Richtung (Sprache-in-Text-Umwandlung) werden nur Demonstrationsvideos zur Verfügung gestellt. Aus diesen Beispielen ist ersichtlich, dass das Hauptaugenmerk für Loquendo bei Call-Centern und Business-Customers liegt. Einen positiven Eindruck hinterliessen die Videos „News and Tracking" und „Intranet Voice Access". In diesen Darstellungen wird gezeigt, dass das Datum und gewisse Informationen aus dem Gesagten ausgelesen und weiterverarbeitet werden können. Loquendo unterstreicht, dass die Erkennung trotz Hintergrundgeräuschen gut ist. Auf keinem der Videos ist ersichtlich, wie Sprache in Text umgewandelt wird respektive wie das Erkannte in Textformat aussieht.

[32] Internetseite zur Anpreisung von Google Voice http://www.google.com/googlevoice/about.html# (zuletzt besucht am: 01.11.09)

[33] Grundwebseite von Loquendo http://www.loquendo.com/de/technology/asr.htm (zuletzt besucht am: 11.10.09)

12.3.4. Sphinx4.0beta3

Test 1: 26. September 2009

Ausgehend von den ersten erfolgreichen Tests hat uns Andreas Maag den Tipp gegeben, uns einmal mit Sphinx 4 zu beschäftigen. Aus diesem Grund wurde in einem ersten eigenen Test versucht, die mitgelieferten Demoprogramme auszuführen. Nachfolgend eine Tabelle mit den Ergebnissen und Bemerkungen zu den einzelnen Tests.

Demoprogramme und ihre Funktionalität

HelloWorld	Vokabular: klein und vorgegeben Good Morning / Hello & einige Namen Funktioniert nur bedingt
Hello Digits	Vokabular: alle Ziffern auf Englisch Funktioniert nur bedingt
Hello N-Gram Demo	Vokabular: nicht vorgegeben Funktioniert nur bedingt
ZipCity	Kombination von Landkarte und der Spracherkennung Funktioniert gar nicht
WavFile	Mitgelieferte wav-Datei: Ziffern von 1-5, funktioniert einwandfrei Eigene wav-Datei kann nicht gefunden werden, auch nicht nach dem Versuch einer Anpassung im Quellcode
Transcriber	Mitgeliefertes wav-File: funktioniert einwandfrei
JSGF Demo	Funktioniert teilweise, schaffte es nicht Watch Rain Man zu erkennen, fand entweder watch gladiator oder watch platoon, nach 17 Versuchen klappte es dann doch noch.
Dialog Demo	Funktioniert zweimal, bei 20 Versuchen

Fazit:

Der erste eigene Test war nicht sehr erfolgreich. Doch war ersichtlich, dass das Programm mit den mitgelieferten Audiodaten einwandfrei funktioniert. Es stellte sich die Frage, weshalb es mit der eigenen Stimme nicht besser funktionierte. War es ein Problem, dass die Stimme zu einer deutschsprachigen Frau gehörte? Siehe dazu Kapitel 0.

Sprachproduktion und Vergleich.

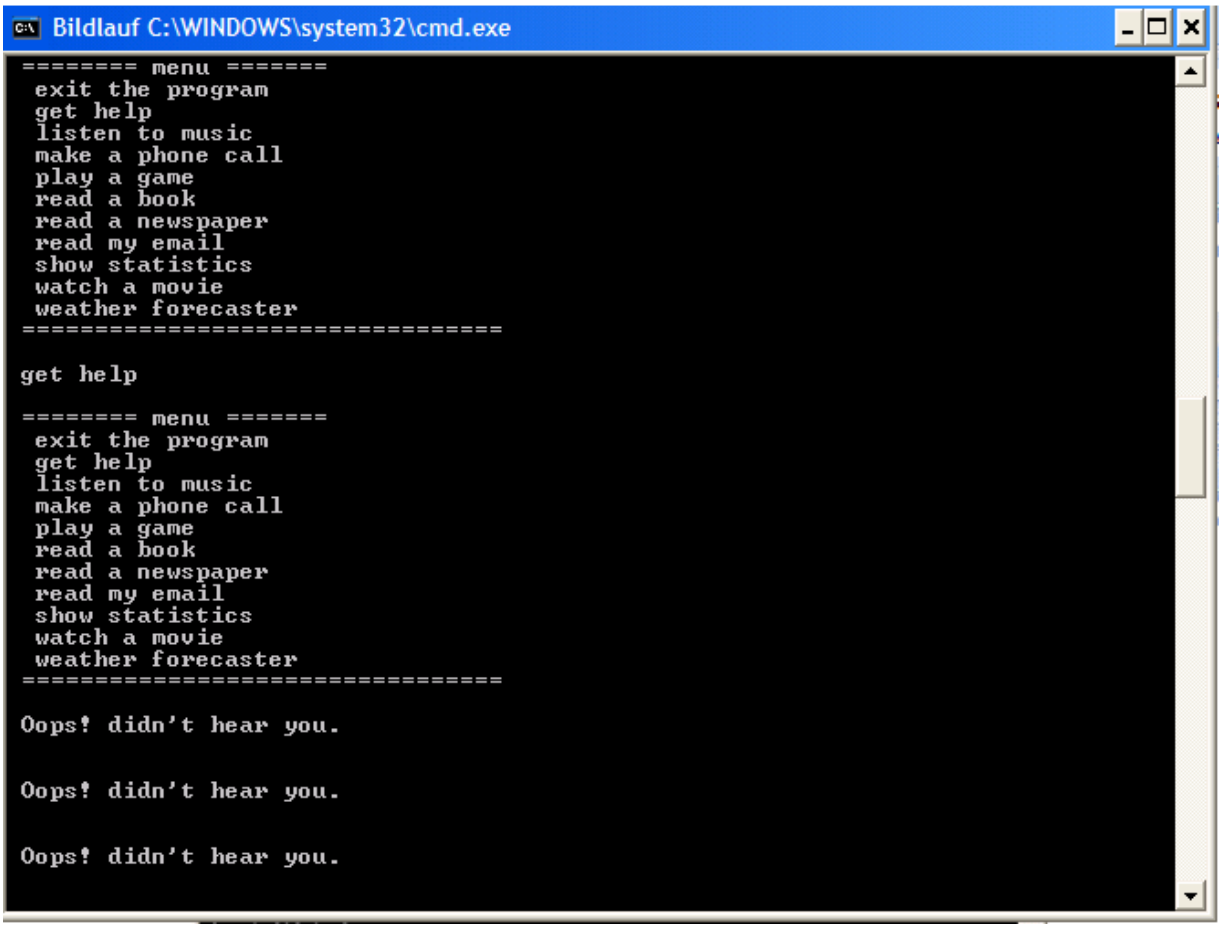

Abbildung 15: Testergebnis

Test 2: Ende Oktober 2009 (mehrere Tage)

Anhand der theoretischen Nachforschungen konnte das Wissen über Sphinx erheblich verbessert werden. Aus diesem Grund wurde nun ein neuer Versuch gestartet. Als Erstes wurde versucht, eine eigene wav-Datei ins Programm einzubinden. Die Tests liefen auf der Basis des Demo-Transcriber Programms von Sphinx.

Das Einbinden einer wav-Datei wird erreicht, indem in der Demo-XML-Datei an den Zeilen 238 und 293 jeweils der Dateiname und der jeweilige Speicherort angepasst wird. Dies ist umständlich, doch für die Tests nicht behindernd. Aus diesem Grund wird diese Einbindung vorläufig beibehalten.

Als Nächstes wurde die Grammatik angepasst. Da dies der erste wirkliche Versuch war, wurde ein kleiner Teil des vorgegebenen Funkspruches verwendet. Die einzelnen gesprochenen Wörter wurden in einer neuen

Grammatik so einfach wie möglich eingebunden. Dies bedeutet, dass eine einfache Oder-Verknüpfung erstellt wurde.

Beim Ausführen dieses veränderten Testprogramms wurde folgender Output generiert:

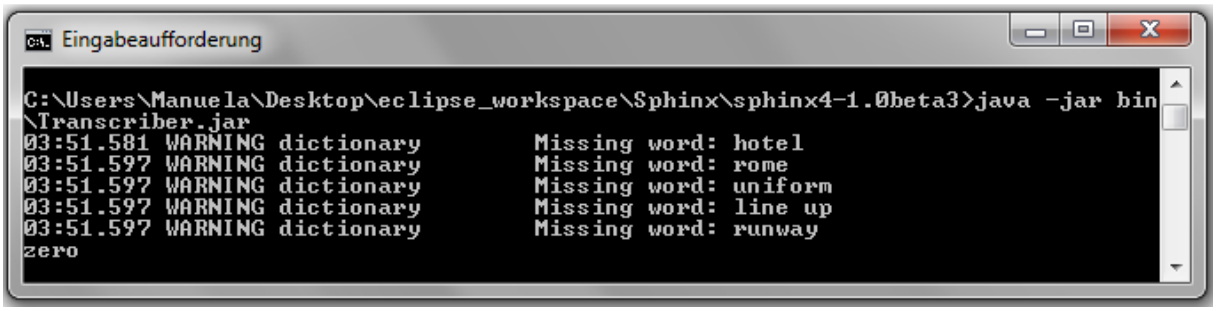

Abbildung 16: Ausgabe Sphinx Test 2

Der eigentliche Output sollte am Schluss noch ein „eight" ausgeben. Doch für den ersten Versuch ist das Resultat vielversprechend. Es bedeutet, dass das Einbinden der Grammatik funktioniert hat und dass nun das Dictionary noch angepasst werden muss.

Test 3: AcousticModels

Nach dem Starten einer Demo-Applikation wird als erstes der Linguist erstellt. Dies bedeutet, dass das Dictionary und anschliessend das AcousticModel geladen werden.

Mit Sphinx 4 werden bereits vier verschiedene AcousticModels mitgeliefert. Im AcousticModel sind die HMM-spezifischen Informationen gespeichert. Zusätzlich befinden sich auch die einzelnen Dictionarys in einem Unterordner.

In der nachfolgenden Tabelle sind diese vordefinierten AcousticModels aufgelistet, zusammen mit den Demo-Applikationen, in welchen sie verwendet werden.

AcousticModel	Verwenduns-applikation(en)
RM1_8gau_13dCep_16k_40mel_130Hz_6800Hz	-
TIDIGITS_8gau_13dCep_16k_40mel_130Hz_6800Hz	Transcriber, Wavfile
WSJ_8gau_13dCep_8kHz_31mel_200Hz_3500Hz	-
WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz	Confidence, Hellongram, Helloworld, Jsapi, Lattice

Tabelle 10: verwendete AcousticModels

Aus dieser Tabelle ist ersichtlich, dass nur zwei Modelle wirklich verwendet werden. Das Tidigits-AcousticModel beinhaltet alle Ziffern und ist deshalb für uns nicht das Gewünschte. Hingegen beinhaltet das WSJ-AcousticModel, mit seinen über 125'000 Wörtern, den Grossteil des im Flugfunk verwendeten Dictionary. Diese Wörter entsprechen dem Nordamerikanischen Englisch. Aus diesem Grund ist es unser Ziel, das WSJ-AcousticModel mit der Grammatik aus dem Transcriber zu verbinden und evtl. noch Wörter ins Dictionary aufzunehmen.

In diesen AcousticModels gibt es eine Model.class-Datei. Diese Klasse kann leider nicht mehr angepasst werden. Zum einen ist sie mit Java 1.5 kompiliert worden und kann deshalb nicht mehr geöffnet werden. Zum anderen kann eine davon dekompilierte Klasse nicht zusammen mit den anderen Dateien des AcousticModels verwendet werden, da es eine inkonsistente magische Nummer gibt. Dieses Problem konnte leider nicht behoben werden.

Test 4: Anderes Dictionary einbinden.

Ziel war es, im mitgelieferte Sphinx Demo „WavFile" das Dictionary zu ersetzten. Hierzu müssen zwei Dateien angepasst werden. Einerseits muss in der Datei „wavefile.Manifest der Class-Path angepasst werden. Des Weiteren muss das „config.xml" angepasst werden. In den beiden rot markierten Zeilen muss der Pfad angegeben werden.

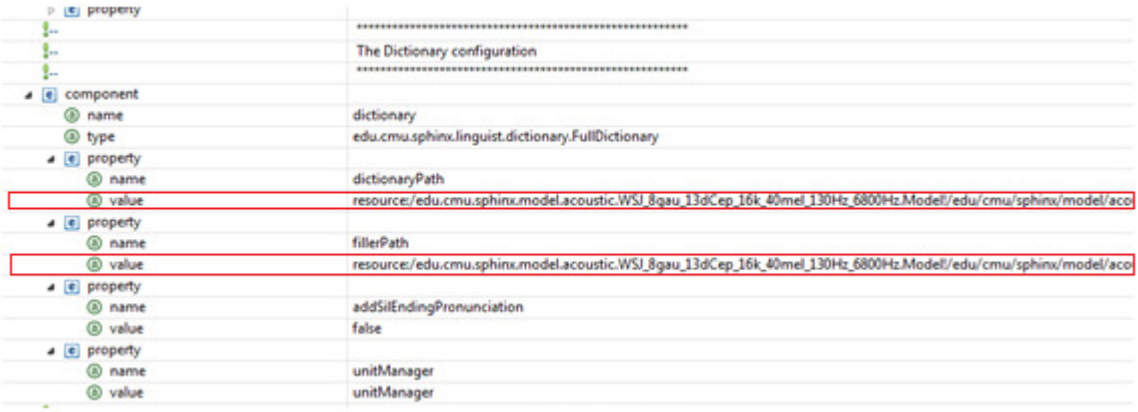

Abbildung 17: Anpassung am config.xml

Leider stürzte das wavfile.jar ab, da ein OutOfMemoryError auftritt.

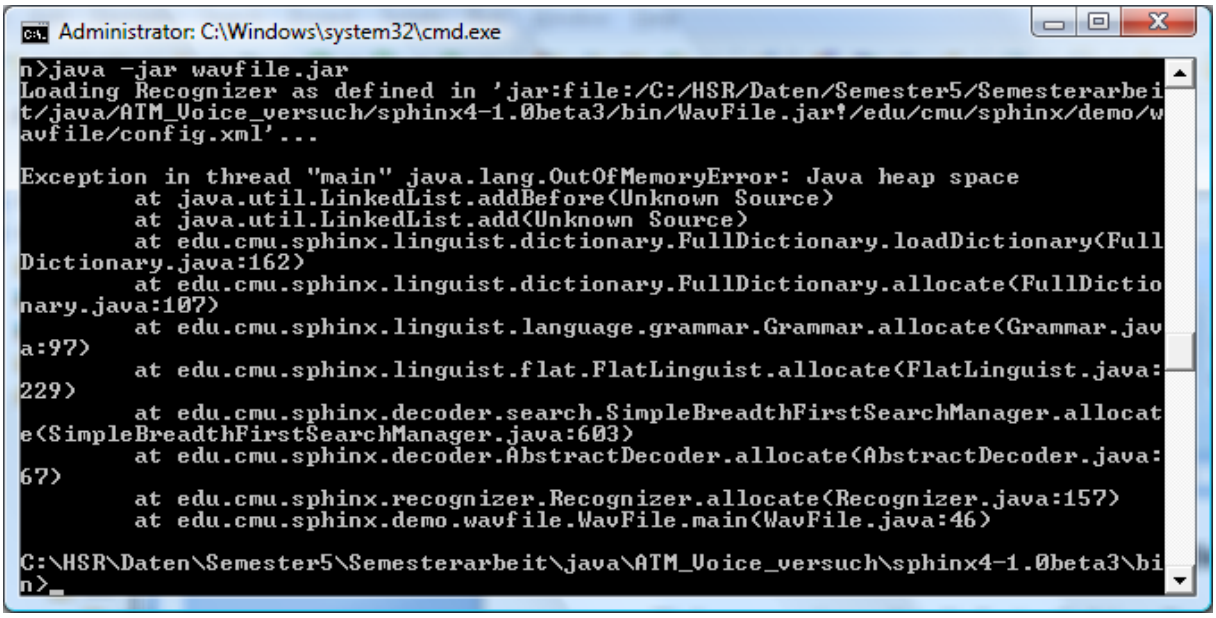

Abbildung 18: Ausgabe der Fehlermeldung

Mit Hilfe des Eclipse Memory Analyzers kann der Fehler noch weiter analysiert werden. Der Memory Analyzer braucht ein Abbild des Heapspeichers. Um dieses zu erstellen, muss der Parameter „-XX:+HeapDumpOnOutOfMemoryError" eingegeben werden. → Der Jar Aufruf sieht nun folgendermassen aus:
„java -XX:+HeapDumpOnOutOfMemoryError –jar WavFile.jar"

Nach dem Importieren in den Memory Analyzer können unter anderem die grössten Speicherfresser aufgelistet werden. In diesem Fall ist es wie erwartet die jar Datei mit dem grossen Dictionary.

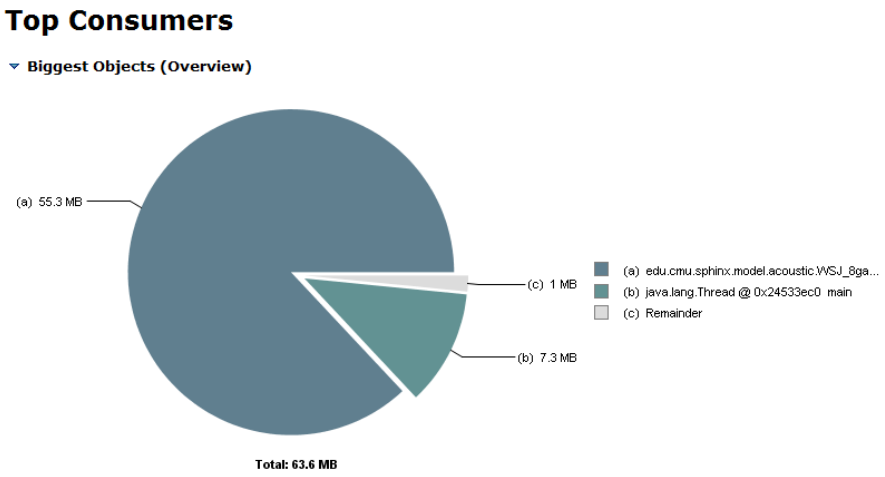

Abbildung 19: Übersicht über Speicherverbrauch

Daraus wird ersichtlich, dass das grosse Dictionary wie erwartet geladen wird. Jedoch ist diese mitgelieferte Demo Datei so nicht lauffähig.

Test 5: digits.gram

Um zu testen, ob die „digits.gram" Datei wirklich verwendet wird wurde „one" aus der Liste der Zahlen entfernt. Aus dem Ergebnis war ersichtlich, dass die Datei tatsächlich verwendet wird. Sphinx hatte versucht, das in der Audiodatei enthaltenen „one" einer anderen Zahl zuzuordnen, welche ähnlich klingt. Je nach vorangegangener Zahl war dies eine andere.

12.3.5. SphinxTrain[34]

Um eigene AcousticModels zu trainieren, kann der SphinxTrain verwendet werden. Dieser wurde eigentlich für Sphinx 3 erstellt, doch da die AcousticModels von Sphinx 4 auf denjenigen der Version 3 beruhen, kann er auch für uns nützlich sein. Für die Trainings braucht es neben genügend Ressourcen auf dem Computer folgende Elemente:

Feature Files

Für jede Audiodatei müssen Features generiert werden. Diese sollten eigentlich mit einem Pearl-Skript des SphinxTrain erstellt werden können. Doch konnte der SphinxTrain nicht wie in den Dokumentationen beschrieben kompiliert werden. So konnten keine Features aus unseren Funksprüchen generiert werden.

Control File

In dieser Datei ist eine Liste mit allen Pfaden zu den Feature Dateinamen abgespeichert.

Transcript File

Alle Feature Dateien werden kopiert nochmals abgelegt. In der Transcript Datei ist eine Liste mit den Pfaden zu diesen Dateien enthalten.

Main Dictionary

Dieses Dictionary sollte alle Wörter enthalten, welche verwendet werden. Falls selber eines generiert werden möchte, sollte Festival[35] verwendet werden. Ansonsten gibt es auf der Internetseite CMU[36] immer wieder aktuelle Dictionarys in Nordamerikanischem Englisch.

---

[34] Informationen zu SphinxTrain: http://www.speech.cs.cmu.edu/sphinxman/scriptman1.html#0 (zuletzt besucht am: 13.12.09)

[35] Download für Festival: http://festvox.org (zuletzt besucht am: 16.12.09)

[36] Internetseite für aktuelle Dictionaries von CMU: http://www.speech.cs.cmu.edu/cgi-bin/cmudict (zuletzt besucht am: 16.12.09)

Filler Dictionary

Dieses Dictionary beinhaltet alle Füllwörter. Es kann von einem anderen AcousticModel übernommen werden.

Phonelist

Die phonetische Schrift wird in der Phonelist definiert. Es dürfen im Dictionary nur diese Elemente verwendet werden.

12.3.6. Test Dokumentation von Andreas Maag

Andreas Maag hat Sphinx auch getestet, seine Testdokumentation befindet sich in einem externen Dokument: Testbericht Maag.docx.

12.3.7. Voice Pro[37]

Voice Pro 12 ist vor allem eine Lösung, um das Betriebssystem Windows und die meisten Microsoft Produkte mittels Sprachbefehlen zu steuern. Die Spracherkennung in den Filmen funktioniert einwandfrei, doch kann diese leider nicht gratis selber getestet werden. Dies führt zur offenen Frage, wie gut die Software mit schlechter Qualität umgehen kann. Ein weiterer problematischer Punkt ist, dass der Zugriff zum eigentlichen Quellcode höchstwahrscheinlich verwehrt bleibt.

12.3.8. Windows Speech API[38] [39]

Innerhalb des Betriebssystems Windows 7 wird eine Spracherkennungssoftware mitgeliefert, die Windows Speech API. Beim Schreiben dieses Berichts wurde zum Teil damit gearbeitet. Teilweise funktionierte das Diktieren ohne Probleme, vor allem, wenn fliessend und ohne Nebengeräusche gesprochen wurde. Doch muss der Text sehr aufmerksam nachbearbeitet werden, da schnell kleine Fehler passieren können. Das Programm lernt immer mehr dazu, so können zum Beispiel Sphinx und andere Wörter buchstabiert und bei weiterer Verwendung erkannt werden. Als Fazit kann gesagt werden, dass das Diktieren von Text recht gut funktioniert, doch dass die Spracherkennung von Funksprüchen ziemlich unwahrscheinlich ist.

Windows hat bereits vor Jahren angefangen, Software zur Spracherkennung zu entwickeln, unter anderem mit ehemaligen CMU-Angehörigen. Daraus resultierten mehrere Speech APIs (kurz SAPI). In Windows 7 wurde SAPI 5.3 standardmässig mit installiert, ebenfalls in Windows Vista. Die Dateien befinden sich unter C:\Windows\Speech.

[37] Shop Voice Pro 12: http://www.linguatec.de/ (zuletzt besucht am: 11.12.09)

[38] Informationen zur Spracherkennung von Windows: http://en.wikipedia.org/wiki/Windows_Speech_Recognition (zuletzt besucht am: 17.12.09)

[39] Detaillierte Informationen zu SAPI: http://en.wikipedia.org/wiki/Microsoft_Speech_API (zuletzt besucht am: 17.12.099)

12.4. Übersichtstabelle über Evaluation

Kriterium	Gewichtung	Dragon Naturally Speaking	Google Voice	Loquendo	Sphinx 4.10 Beta 3	Voice Pro	Windows Speech API
Spracherkennung	A	9	5	6	7	5	7
Nebengeräusche	A	4	3	3	6	3	3
Buchstabieralphabet	A	10	-	-	5	-	2
Training	A	8	-	5	SphinxTrain	vorhanden	6
Kosten	B/C	Standard: ca. 100 Euro Professional: ca. 1000 Euro SDK: auf Anfrage	benötigt Anmeldung aus USA, Registrierung dauert mehrere Wochen	auf Anfrage	Open Source	ab 69 Euro	im Betriebsystem Windows Vista oder 7 inbegriffen
Verfügbarkeit	B	auf Anfrage	1	1	10	1	1
Beeinflussbarkeit	C	4	1	-	9	-	-
Einsatzbereich		Programm Fernsteuerung und Text diktieren	Vor allem zum einheitlichen Gebrauch von mehreren Telefonnummern	Vor allem bei Call Centern und Business Kunden	vor allem in Projekten, bei welchen eigene Parameter relevant sind	Ansteuerung von Betriebssystem und Software	Windows Betriebsystem: Systemsteuerung und Diktieren
Test Ausführlichkeit		7	2	3	8	2	5
Besonderheiten		weit verbreitet und häufig empfohlen.	Account nur aus USA erlaubt				

Tabelle 11: Übersichtstabelle über Evaluation

Erklärung zu den einzelnen Kriterien im Kapitel 0.

Legende:

A:	muss	10:	perfekt
B:	soll	5:	möglich
C:	kann	0:	gar nicht

12.5. E-Mail-Verlauf mit ETH-Spracherkennungs-Experte(13)

Bei den Nachforschungsarbeiten, mit Unterstützung von Herrn Heinzmann, erfuhren wir, dass an der ETH Zürich eine Gruppe für Spracherkennung besteht.[40] Wir entschieden uns, jemanden dieser Gruppe per E-Mail anzuschreiben. Da wir Studenten bereits vorgängig bei der Suche im Internet auf Herrn Thomas Ewender gestossen sind, der Vorlesungen zum Thema „Sprachverarbeitung" hält, haben wir mit ihm Kontakt via Mail aufgenommen. Dabei erfuhren wir, dass die Funksprüche eigentlich erkannt werden sollten und dass Sphinx 4 ein gutes Open-Source-Framework ist.

Der gesamte Mailverlauf befindet sich in einem separaten Dokument Mail-verlauf_Ewender.docx.

[40] ETH-Internetseite der Spracherkennungsgruppe http://www.tik.ee.ethz.ch/~spr/SPGinfo/SPGinfo.html (zuletzt besucht am 16.10.2009)