
BACHELORARBEIT

Herr
Franz Bellmann

**Prototypische Systemintegration
und Evaluation von Open Source
Sprachmodellen zur
automatischen Spracherkennung
gesprochener deutscher Texte**

2020

BACHELORARBEIT

Prototypische Systemintegration und Evaluation von Open Source Sprachmodellen zur automatischen Spracherkennung gesprochener deutscher Texte

Autor:

Franz Bellmann

Studiengang:

Allgemeine und Digitale Forensik

Seminargruppe:

FO17w4

Erstprüfer:

Prof. Dr. Dirk Labudde

Zweitprüfer:

Dr. Steffen Grunert

Mittweida, September 2020

Bibliografische Angaben

Bellmann, Franz: Prototypische Systemintegration und Evaluation von Open Source Sprachmodellen zur automatischen Spracherkennung gesprochener deutscher Texte, 51 Seiten, 21 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Computer- und Biowissenschaften

Bachelorarbeit, 2020

Referat

I. Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis	II
Tabellenverzeichnis	III
Vorwort	IV
1 Einleitung	1
2 Stand der Technik: Automatic Speech Recognition-Modelle	3
2.1 Geschichtliche Entwicklung	3
2.2 Heutige Automatic Speech Recognition-Modelle (ASR)	4
2.3 Ausblick.....	5
3 Grundlagen.....	7
3.1 Korpusse (Data Sets).....	7
3.1.1 Tuda	8
3.1.2 Voxforge	8
3.1.3 The Spoken Wikipedia Corpora (SWC).....	8
3.1.4 The M-AILABS Speech Dataset	8
3.1.5 Common Voice	9
3.1.6 Forschergeist.....	9
3.2 Hidden Markov Modell.....	9
3.3 Neuronale Netze	11
3.4 Automatic Speech Recognition kurz ASR	13
3.4.1 Funktionsweise ASR.....	13
3.4.2 Qualität / Evaluation der ASR - Modelle	14
3.4.3 Probleme der ASR	16
3.4.4 Lösungen für die Forensik.....	16
3.5 Die zur Evaluation verwendeten Modelle	18
3.5.1 Kaldi TUDA Projekt	19
3.5.2 Deep Speech	20
3.5.3 Google Web Speech API (GWS API).....	21

3.5.4	wave2letter	21
4	Experimente und Evaluation	23
4.1	Übersicht	23
4.2	Testdurchlauf: Ohne Hintergrundrauschen	25
4.2.1	Testdurchlauf: Ohne Hintergrundrauschen - langsam	28
4.2.2	Testdurchlauf: Ohne Hintergrundrauschen - schnell	32
4.2.3	Vergleich	34
4.2.4	Konklusion über die Datensätze ohne Hintergrundrauschen	37
4.3	Mit Hintergrundrauschen	37
4.3.1	Überblick	38
4.3.2	Test und Evaluation	40
4.3.3	Ungefiltert und gefiltert	40
4.3.4	Fazit	42
4.4	Zusammenführen der zwei Evaluationsschritte	42
4.4.1	Fazit	44
5	Diskussion.....	45
6	Schlussfolgerung	47
	Literaturverzeichnis	49

II. Abbildungsverzeichnis

2.1	Timeline Spracherkennung.....	5
3.1	Einfaches Vorhersagemodell mit Markov-Kette	10
3.2	Stark vereinfachtes Hidden Markov Modell zur Spracherkennung.....	10
3.3	HMM mit drei Zuständen	11
3.4	Funktionsweise eines neuronalen Netzes	12
3.5	Funktionsweise ASR-Modell	13
3.6	Beispiel Wörterbuch	14
3.7	Beispiel WER-Berechnung	15
3.8	Fast Fourier Transformation	17
3.9	Energie des Signals.....	17
3.10	Verteilungsdichte	18
3.11	Deep Speech Architektur	20
4.1	Diagramm Normale Geschwindigkeit.....	27
4.2	Spektrogramm: Vergleich des Wortes "Ich"	28
4.3	Diagramm langsame Geschwindigkeit.....	30
4.4	Spektrogramm: Vergleich Langsam und Normal	31
4.5	Diagramm schnelle Geschwindigkeit	33
4.6	Diagramm Vergleich der Word Error Rate	35
4.7	Vergleich der Testdaten mit Hintergrundgeräuschen	39
4.8	Spektrogramm einer Sprachsample mit Hintergrundrauschen	39
4.9	Diagramm	44

III. Tabellenverzeichnis

3.1	Überblick über die deutschen Datensätze	7
4.1	Überblick über die verwendeten Modelle	23
4.2	Überblick über die Test - Sprachsamples ohne Hintergrundrauschen	24
4.3	Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz .	25
4.4	Gemittelte Word Error Rate über die gesamten Modelle	27
4.5	Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz im Vergleich zur normalen Geschwindigkeit	29
4.6	Gemittelte Word Error Rate über die gesamten Modelle	31
4.7	Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz mit erhöhter Geschwindigkeit im Vergleich zur normalen und zur verlangsamten Ge- schwindigkeit	32
4.8	Gemittelte Word Error Rate über die gesamten Modelle und Geschwindigkeiten	34
4.9	Überblick über die Test - Sprachsamples mit Hintergrundrauschen	38
4.10	Ergebnisse der Sprachsamples mit Hintergrundrauschen (Google Web Speech)	40
4.11	Ergebnisse der Sprachsamples mit Hintergrundrauschen (Kaldi 1000h)	41
4.12	Ergebnisse der Sprachsamples mit Hintergrundrauschen (Kaldi 630h)	41
4.13	Testergebnisse der Zusammenführung der besten beiden Evaluationsschritte (Goo- gle Web Spech API)	43
4.14	Testergebnisse der Zusammenführung der besten beiden Evaluationsschritte (Kaldi 1000h Modell)	43

IV. Vorwort

Vor Ihnen liegt die Bachelorarbeit „Prototypische Systemintegration und Evaluation von Open Source Sprachmodellen zur automatischen Spracherkennung gesprochener deutscher Texte“. Durch mein Praktikum im Bereich der Audioanalyse und Transkribierung kam ich auf die Idee, mich ausführlicher mit dem Thema zu beschäftigen.

Ziel war es, „robuste“ Spracherkennung zu finden und verschiedene Ansätze zu präsentieren, die für das Aufgabengebiet der Forensik von Nutzen sein könnten.

Die Nachforschungen waren nicht immer einfach, da die vorhandenen freien Spracherkennung Projektarbeiten via Github sind und diese kompliziert zu verstehen bzw. zu implementieren sind. Doch nach einer umfangreichen Analyse der zur Verfügung stehenden Modelle gelang es mir, die Forschungsfrage zu beantworten.

Herzlich bedanken möchte ich mich bei meinen beiden Lektorinnen, die es nicht immer leicht mit meiner Rechtschreibung gehabt haben.

Ich wünsche Ihnen viel Freude beim Lesen dieser Bachelorarbeit.

Franz Bellmann

Oederan, 03. September 2020

1 Einleitung

Wenn du abgehört und belauscht wirst, will man doch was von dir wissen. Das können nicht alle von sich sagen.

Erhard Blanck (*1942)

Die Strafverfolgung ist eines der ältesten Bedürfnisse der Menschheit. Ging es im 18. Jahrhundert nur um Rache und Sühne, ist es heute hauptsächlich der Gedanke der Gerechtigkeit in unserem Rechtsstaat, der die Behörden anweist, Straftaten zu verfolgen und aufzuklären. Hierbei ist es fundamental wichtig, Informationen bzw. Indizien zu einem Fall zu erlangen. Vor der Digitalisierung musste diese Informationsbeschaffung noch von vielen involvierten Personen, wie Spionen, Spitzel und verdeckten Ermittlern erledigt werden. Heute können Informationen u.a. von Abhörgeräten (Wanzen) beschafft werden. Dies ist weniger aufwendig, hat aber einen großen Nachteil - man muss die erlangten Hinweise umfangreich analysieren.

Erhard Blanck hat mit obigem Zitat Recht, die Strafverfolgung hat zum Ziel, möglichst viel über die "interessante" Person zu erfahren, um eine sicher Anklageschrift zu verfassen. Es kommt nun bei einer Abhör-Überwachung zu immer größeren Mengen an Daten, die aktuell einzeln durch die Ermittler ausgewertet werden müssen. Dies ist ein langwieriger und mühsamer Prozess, denn die aufgezeichneten Daten müssen angehört, akustisch verstanden und niedergeschrieben werden. Bei der Vielzahl an Informationen kann sich dies über Wochen und Monate hinziehen. Mit Hilfe von modernen Techniken ist es heute möglich, die Strafverfolgung dahingehend zu unterstützen oder zu beschleunigen. Eine Möglichkeit ist die automatische Spracherkennung (engl. automatic speech recognition, kurz ASR). Dies ist ein computergestütztes Verfahren zum Erkennen und Transkribieren von menschlicher Sprache. Die beschafften Audio-Daten liegen dabei meist in schlechtem Zustand vor. Störgeräusche, wie atmosphärische Interferenzen, Wind- oder Wettergeräusche sowie ortsabhängiger Schall, behindern die Analyse. Es bedarf dabei einer aufwendigen Vorverarbeitung, um die Informationen der Daten zu extrahieren, damit eine Spracherkennung diese Daten in Text umwandeln kann. Diese Vorverarbeitung erfordert ein manuelles Experimentieren und Aufbereiten der Daten, welches wiederum von der Subjektivität des Ermittlers abhängig ist. Deswegen stellt sich die Frage, ob es Spracherkennungsmodelle gibt, welche so robust sind, dass sie mit minimaler Vorverarbeitung diese Daten nutzbar machen können.

Aktuelle Spracherkennungsmodelle der führenden Unternehmen (Apple, Microsoft und Facebook) werden heute schon dazu genutzt, eine User-Schnittstelle in den Geräten zu implementieren. Diese können zum Beispiel für

- Diktierprogramme
- Steuerung und Bedienung von Geräten
- Unterstützung für behinderte Menschen
- Navigationssysteme
- Fremdsprachensoftware

genutzt werden. Solche Modelle sind leider nicht frei verfügbar. Es gibt eine Vielzahl von Open-Source-Modellen, die selten an die kostenpflichtigen Apps oder Programme heranreichen. Eine Nutzung dieser Open-Source-Modelle im forensischen Bereich und die Vorverarbeitung der Daten sind deshalb Gegenstand dieser Arbeit.

Um der Thematik näherzukommen, wird mit einem kleinen geschichtlichen Abriss über die Entwicklung der Spracherkennungssysteme und dem Stand der Technik heute, ein Überblick über das Thema geschaffen. Im Grundlagenteil erfolgt die Erläuterung der genutzten ASR-Systeme, der Sprachkorpusse und der verwendeten Algorithmus-Strukturen. Des Weiteren werden die Funktionsweise eines Spracherkenners und die Metriken zur Evaluation vorgestellt. Ein kurzer Exkurs über die Probleme der ASRs und die technischen Lösungen für die Forensik schließen das Kapitel ab. Bevor es zur Evaluation kommt, werden die genutzten Modelle und ihre Struktur beschrieben.

Im nächsten Teil werden dann die Modelle mit einem eigens zusammengestellten Testdatensatz ohne Störgeräusche evaluiert. Um zu testen, wie sich die ASR-Systeme verhalten, wenn es zu einer Einspeisung von vorverarbeiteten Daten kommt, werden die Testdaten verlangsamt bzw. beschleunigt und nochmals evaluiert. Als nächstes sollen die besten Modelle in realen Bedingungen, d.h. mit Störgeräuschen, getestet werden. Danach wird mit einem Audio-Analyseprogramm das Störgeräusch reduziert und nochmals evaluiert, bis zu guter Letzt die besten Ergebnisse der jeweiligen Modelle zusammengeführt und nochmals getestet werden.

Der letzte Teil besteht aus einer Bewertung der Ergebnisse und deren Diskussion.

2 Stand der Technik: Automatic Speech Recognition-Modelle

2.1 Geschichtliche Entwicklung

Das erste Bestreben, Sprache in Text zu verwandeln, kann auf das Jahr 1952 zurückgeführt werden. In diesem Jahr haben amerikanische Forscher der *Bell Labs* (ehemalige Forschungseinrichtung der Telefongesellschaft AT&T) AUDREY entwickelt. AUDREY war der erste dokumentierte Spracherkennung. Erkannt wurden mit dem System die Ziffern Null bis Neun. Somit hatte AUDREY einen Sprachkorpus von immerhin zehn Elementen und eine Genauigkeit von knapp 97%. Allerdings gelang dies nur mit der vorher trainierten Stimme. Wurde das Programm mit einer anderen Stimme getestet, fiel die Genauigkeit auf nur noch 1-3 % rapide ab.¹

Erst 1962 entwickelte IBM die *SHOEBOX*. Diese konnte nun schon mit den zehn Ziffern und mathematischen Operatoren, wie Plus und Minus, umgehen. Die Sprache wurde in elektronische Impulse umgewandelt und in verschiedene Laute klassifiziert. *SHOEBOX* war als eine "Machbarkeitsstudie" konzipiert und konnte einfache arithmetische Operationen ausführen. Eine "Machbarkeitsstudie" diente als Grundlage einer Überprüfung der Durchführbarkeit eines Projektes.²

Lange geschah nichts im Bereich Spracherkennung. Erst als das US-Militär die enormen Möglichkeiten der damals noch "Speech Understanding Research"- Programme erkannte und bezuschusste, kam wieder Aufschwung in dieses Forschungsgebiet.

1976 wurde dann die HARPY-Spracherkennung entwickelt. HARPY besitzt die rudimentären Ansätze eines Hidden Markov Modells und sucht somit den optimalen Weg durch ein Netzwerk, um eingespeiste Sätze zu erkennen. "Im HARPY-System wurde versucht, das gesamte phonetische, phonologische, lexikalische und syntaktische Wissen explizit in einem einzigen Netzwerk darzustellen; die Erkennung eines gesprochenen Satzes lässt sich damit auf eine Suche nach dem besten Weg durch dieses Netzwerk zurückführen [1, S. 106]." Dabei besaß HARPY einen Sprachkorpus von 1011 Wörtern und konnte in nahezu Echtzeit arbeiten.

In den 80er Jahren wurde das Hidden Markov Modell (HMM) populär. Es konnte anhand der hörbaren Töne (Emission) auf die Wahrscheinlichkeit der gesprochenen Laute (versteckte Schicht) schließen. Für jedes Phonem wurde ein eigenes HMM trainiert und

¹ Vgl. "From AUDREY to Siri", Pieraccini, www.icsi.berkeley.edu/pubs/speech/audreytosiri12.pdf, Zugriff 14.06.2020

² Vgl. IBM History, www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html, Zugriff 14.06.2020

durch Zusammensetzung der HMMs ermittelte das System die Wörter mit der höchsten Wahrscheinlichkeit. Doch war es wegen der zu langsamen Hardware nur möglich, einzelne Wörter zu erkennen. Ganze Sequenzen an gesprochener Sprache zu transkribieren lag noch in weiter Ferne. Durch immer schneller werdende Prozessoren konnten die HMMs weiter optimiert werden.

Daraus entstand 1997 *NATURALLY-SPEAKING* der Technologiefirma Dragon. Diese Software hatte einen Sprachkorpus von 23.000 Wörtern und konnte einfache Sequenzen verarbeiten. Genutzt wurde dieses Programm hauptsächlich als Diktierunterstützung oder zur Steuerung des Computers. Nachteilig war die Sprecherabhängigkeit. So musste jeder neue Nutzer das Programm langwierig auf sein Sprachspektrum trainieren.³

Bis in die späten 2000er Jahre stagnierte das Forschungsgebiet automatischer Spracherkennung. Das HMM galt als ausgereift und konnte keine signifikanten Verbesserungen mehr erzielen.

2.2 Heutige Automatic Speech Recognition-Modelle (ASR)

2008 kam dann mit Google Voice Search [2] das erste cloudbasierte Modell auf den Markt. Mit Hilfe der Cloud-Technik konnte Google jede gesprochene Suchanfrage sammeln und zum Trainieren ihrer künstlichen Intelligenz nutzen. Dies ermöglichte es Google 2010, bis zu 200 Milliarden Wörter, Redewendungen, Sätze und Umgangssprachen in englischer Sprache zu erkennen.

Aus einer kleinen Maschine, die nur Zahlen erkennen konnte, ist ein Markt entstanden, der durch Echtzeit- Sprachübersetzung, Sprach- und Navigationsaufgaben, Smart-Home-Steuerung und *Internet of Things* Einzug in unser Leben gefunden hat. Apple konnte als eine der ersten mit ihrer Spracherkennungssoftware eine "Assistentin" namens SIRI implementieren, die diese gesamten Vorteile bündelt und dem Endnutzer zur Verfügung stellt.

³ Vgl. http://www.dragon-medical-transcription.com/history_speech_recognition.html, Zugriff 15.06.2020

2.3 Ausblick

Mit *Android Auto*⁴ oder *Apple Car Play*⁵ ist es zur Zeit schon möglich, das “Infotainmentsystem“ von Autos zu nutzen. Dies ist erst der Anfang zum sprachgesteuerten Fahren. *Google Duplex* soll in den kommenden Jahren unser persönlicher Assistent werden, der Bestellungen tätigt oder Telefonate im Namen des Nutzers führt.

Abbildung 2.1 zeigt die Meilensteine der ASR-Entwicklung. Gut zu erkennen ist, dass zwischen den Jahren 1986 und 2006 ein langer Zeitraum liegt. Erst als das Militär und der Nachrichtendienst das Potential erkannt hatten, kam es zu einer signifikanten Weiterentwicklung.

Die immer größer werdende Rechenleistung und die Cloud-Technik ermöglichen immer perfektere Spracherkenner. Sie können mit immer mehr Daten trainiert werden und diese in Echtzeit berechnen. Ferner wird es möglich sein, aus den transkribierten Audio-Files Sprecher, Stimmen und Stimmungen sowie Informationen automatisch zu extrahieren. All diese Fortschritte können zu einer nützlichen forensischen Variante eines ASR führen.

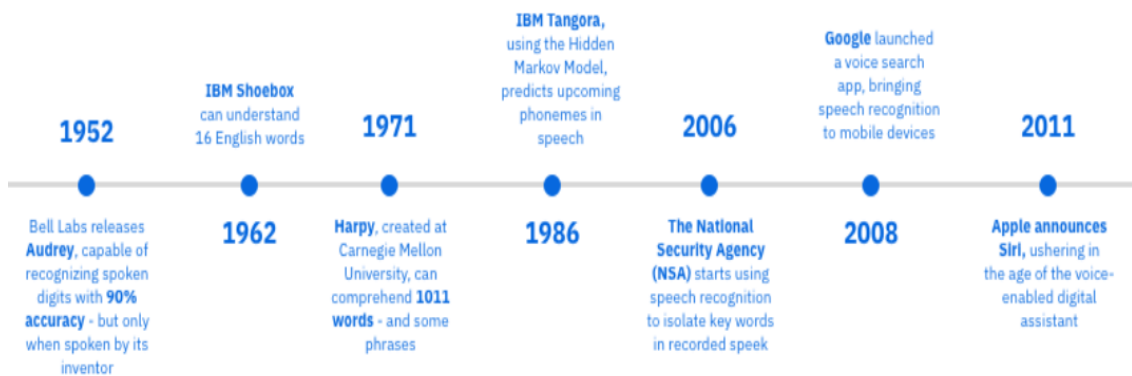


Abbildung 2.1: Timeline zeigt die wichtigsten Entwicklungen im Bereich Spracherkennung⁶

⁴ https://www.android.com/intl/de_de/auto/

⁵ <https://www.apple.com/de/ios/carplay/>

⁶ <https://codeburst.io/html5-speech-recognition-api-670846a50e92>, Zugriff 25.06.2020

3 Grundlagen

3.1 Korpusse (Data Sets)

Ein Datenset, das eine Sammlung von Audio-Daten enthält, nennt man Sprachkorpus⁷. Viele der verwendeten Korpusse besitzen für jede File eine Transkription, so dass man den vom Programm erstellten Text und den wirklichen Text vergleichen kann. Ferner ist es dadurch einfacher möglich, eigene Korpusse anhand der vorhandenen zu erstellen ohne diese von Hand zu schreiben. Es gibt nur einige wenige deutsche Datensätze, die man zum Trainieren und Evaluieren der vorhandenen Modelle nutzen kann. Trainieren heißt in diesem Falle, dass das Modell einen Teil der Daten erhält, seine Berechnungen durchführt und ein Wort wieder gibt. Ist dieses Wort ungleich dem wirklichen Wort, werden Parameter verändert und der Schritt solange wiederholt, bis das vorhergesagte Wort und das wirkliche Wort identisch sind (siehe Kapitel 3.3). Aber auch eine Berechnung der Wahrscheinlichkeiten, welches Wort als nächstes kommt (siehe Kapitel 3.2) fällt unter den Bereich Trainieren.

Hierbei gilt: Je größer der Fundus und je diverser die gesammelten Daten sind, desto besser lernt das jeweilige Modell. Dies liegt daran, dass dadurch mit mehr Wortabhängigkeiten, verschiedenen Tonlagen und Dialekten trainiert werden kann.

Nachfolgend werden die für die Modelle genutzten Korpusse vorgestellt. Einige davon sind umfangreicher als andere, enthalten dafür aber weniger spezifischere Daten. Um ein Modell zu trainieren, welches mit einer größtmöglichen Anzahl von Wörtern umgehen kann, ist eine Mischung der Korpusse erforderlich. Aber nicht jedes System nutzt die gleichen Sprachkorpusse. Tabelle 3.1 zeigt eine kurze Übersicht über die genutzten Sprachsets.

Datenset	Anzahl Sprecher	Länge gesamtes Set in Stunden
Tuda	147	127
Voxforge	180	35
SWC	299	260
M-AILABS	n.a.	237
Common Voice	8460	483
Forschergeist	n.a.	2

Tabelle 3.1: Überblick über die deutschen Datensätze

⁷ https://de.qwe.wiki/wiki/Speech_corpus

3.1.1 Tuda

Tuda ist ein Projekt der Technischen Universität Darmstadt. Der frei verfügbare Datensatz beinhaltet 147 verschiedene Sprecher, die zusammen 36 Stunden gesprochenes Material liefern. Jeder Terminus wurde gleichzeitig mit vier Mikrofonen aufgenommen, um eine Variation zu erhalten. Dadurch erhöht sich die trainierbare Stundenanzahl auf 127. Der Korpus beinhaltet Sätze aus Wikipedia, Reden aus dem europäischen Parlament sowie kurze "Command & Control"-Befehle. [3]

3.1.2 Voxforge

Der Voxforge-Datensatz ist ähnlich dem Tuda-Korpus. Beide extrahieren ihre Rohdaten aus den gleichen Quellen (Wikipedia, Reden aus dem europäischen Parlament und "Command & Control"-Befehle). Mit insgesamt 35 Stunden Sprachdaten und 180 Sprechern ist der Datensatz kleiner als der von Tuda. Das liegt daran, dass die Daten mit nur einem Mikrofon aufgenommen wurden.⁸

3.1.3 The Spoken Wikipedia Corpora (SWC)

Der deutsche SWC ist von der Universität Hamburg ins Leben gerufen worden. Das Ziel dieses Korpusses ist es, die sich immer weiter entwickelnde Wikipedia-Community zu nutzen, um die veröffentlichten Artikel als Sprachquelle zu verwenden. Dies ermöglicht eine breite Abdeckung von Themengebieten. Das deutsche Wikipedia-Datenset besteht aus über 763 Artikeln, die von 299 Sprechern gelesen werden. Dies ergibt eine Gesamtspiellänge von 260 Stunden. [4]

3.1.4 The M-AILABS Speech Dataset

Das M-AILABS Speech Dataset beinhaltet Trainingsdaten mit einer Einteilung in männlich, weiblich und gemischt. Die vorgelesenen Texte sind zwischen 1-20 Sekunden lang und stammen aus Texten eines Zeitraums von 1884 bis 1964. Daraus folgt, dass neuere Wörter und Wendungen nicht im Datensatz enthalten sind. Insgesamt besteht er aus 237 Stunden Sprachdaten.⁹

⁸ <http://www.voxforge.org>, Zugriff 18.06.2020

⁹ <https://www.caito.de/2019/01/the-m-ailabs-speech-dataset>, Zugriff 17.06.2020

3.1.5 Common Voice

Common Voice wurde als "Crowdsurfing"-Projekt konzipiert. Die Entwickler wollten damit ein Datenset erstellen, das frei zugänglich und erweiterbar ist. Mit mehr als 480 validierten Gesamtstunden und 8460 Stimmen (Stand 18.06.2020) ist dieser Korpus einer der größten deutschen Sets. Common Voice wird von Mozilla für deren Adaption Deep Speech [5] genutzt und auch mit einer eigenen Nutzerschnittstelle zum Sammeln von Sprachdaten vorangetrieben¹⁰ Auf dieser Site können User vorgegebene Texte über ein einfaches Onlineportal nachsprechen und gesprochene Texte validieren [6].

3.1.6 Forschergeist

Forschergeist ist kein Sprachdatenset im eigentlichen Sinne. Auf der Seite von Forschergeist¹¹ werden diverse transkribierte Podcasts zum freien Download angeboten. Diese können zum Training der Modelle genutzt werden.

3.2 Hidden Markov Modell

Eine Variante, die Spracherkennung auf mathematischem Weg zu lösen, gelingt über das stochastische System des Hidden Markov Modells (HMM). Das HMM ist eine komplexere Variante der Markov-Ketten (vgl. Abbildung 3.1). Bei einer Markov-Kette betrachtet man die Zusammenhänge, Abhängigkeiten und Zustände von aufeinanderfolgenden Ereignissen vollständig. Sei nun der aktuelle Zustand t , so kann man mit diesem Modell die Wahrscheinlichkeit des nachfolgenden Zustandes $t+1$ ermitteln. Demzufolge ist die Markov-Kette ein stochastisches Modell, das die Wahrscheinlichkeit des Eintreffens eines zukünftigen Zustandes beschreibt.

Sei X die Zustände, so ist die Wahrscheinlichkeit dieser Sequenz in einer Markov-Kette

$$P(\text{Die, Markov, Kette}) = P(X_1)P(X_2|X_1)P(X_3|X_2)$$

$$P(X_1 = \text{Die}, X_2 = \text{Markov}, X_3 = \text{Kette})$$

$$= P(X_1 = \text{Die}) \cdot$$

$$P(X_2 = \text{Markov} | X_1 = \text{Die}) \cdot$$

$$P(X_3 = \text{Kette} | X_2 = \text{Markov})$$

$$= (0.5 \cdot 0.5 \cdot 0.25) = \underline{\underline{0.0625}}$$

$$P(\text{Die, Kette, Die}) = P(0.5 \cdot 0.25 \cdot 0.25) = \underline{\underline{0,03125}}$$

¹⁰ <https://voice.mozilla.org/de>

¹¹ <https://forschergeist.de/>, Zugriff 07.07.2020

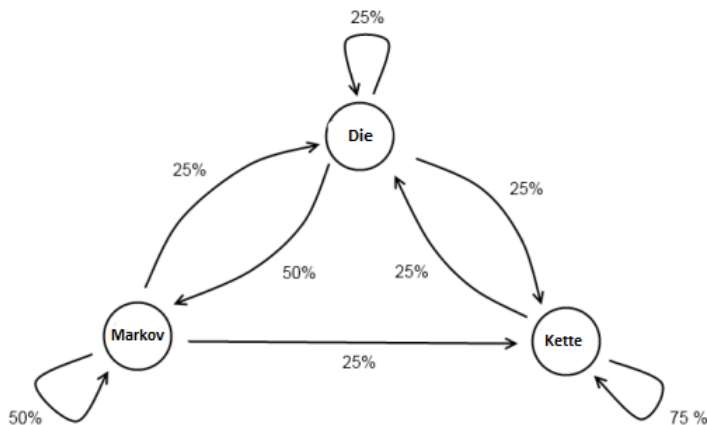


Abbildung 3.1: Einfaches Vorhersagemodell mit Markov-Kette, um die Wortgruppe "Die Markov Kette" zu bilden. Die Kreise beschreiben die jeweiligen Zustände. Die Pfeile zeigen die Übergangswahrscheinlichkeiten von einem Zustand zum anderen bzw. zu sich selbst an. In diesem Beispiel ist die Startwahrscheinlichkeit für den Zustand "Die" 50 % (nicht mit eingezeichnet).

Durch die Kombination der Zustände und deren Übergangswahrscheinlichkeiten entsteht eine Vielzahl an Ergebnissen. Das Ergebnis mit dem höchsten Wert wird bevorzugt vom Modell gewählt.

Meist ist es nicht möglich, das gesamte System zu betrachten. Sieht man bei der Markov-Kette das gesamte System, ist beim HMM nur die Ausgabebeziehung zu beobachten. Wie in Abbildung 3.2 zu sehen ist, wird nur die obige Sequenz "die auf der Bank sitzende Frau" ausgegeben. Das Verfahren darunter ist für den Nutzer nicht sichtbar.

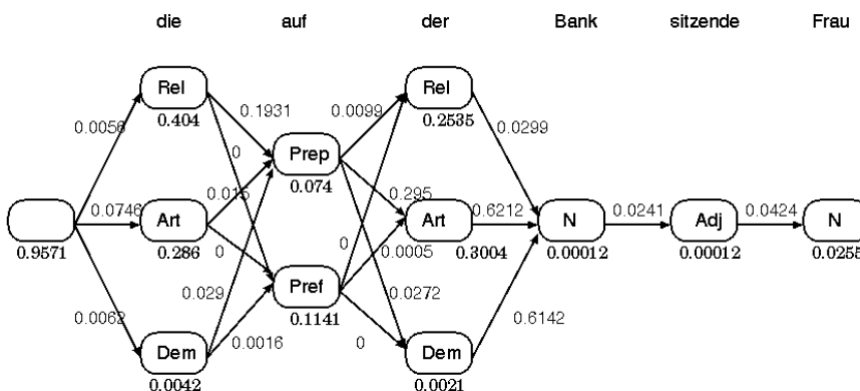


Abbildung 3.2: Stark vereinfachtes Hidden Markov Modell zur Spracherkennung¹². Das HMM errechnet anhand der Wahrscheinlichkeit der Startzustände, Zustandsübergänge und Emissionen die Möglichkeit der jeweiligen Wortarten (im 1.Zustand: Rel, Art oder Dem). Durch einen argmax - Operator wird diejenige Wortart mit der größten Wahrscheinlichkeit gewählt und daraus, durch weitere HMMs, das Wort mit der höchsten Wahrscheinlichkeit.

In der Sprachverarbeitung wurde dieses Prinzip lange Zeit genutzt. Es war aber ineffizient, da für jedes Wort ein eigenes HMM trainiert werden musste. Das Problem konnte mit der Zerteilung der Wörter in der Vorverarbeitung gelöst werden. Der eingespeiste Satz wurde in seine einzelnen Ausdrücke geteilt und diese Einzelwörter in Phoneme. In der deutschen Sprache kann in 40 verschiedene Phoneme unterschieden werden. Diese ergeben in ihren Variationen das Wort. Die HMMs wurden daher nicht mehr auf einzelne Wörter, sondern auf Phoneme trainiert (s. Abbildung 3.3). Das Hidden Markov Modell wird für eine probabilistische Spracherkennung genutzt.

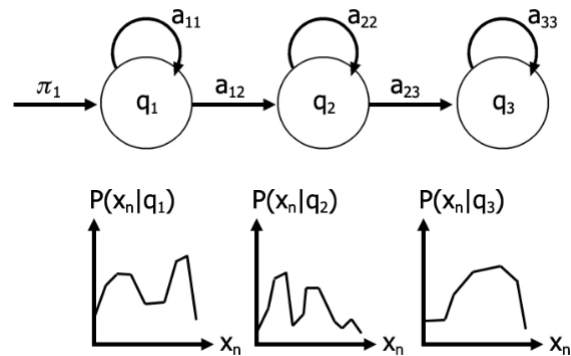


Abbildung 3.3: Ein HMM mit drei Zuständen¹³. Jeder Zustand q_1, q_2, q_3 hat eine Übergangswahrscheinlichkeit a_{ij} . Der Startzustand q_1 besitzt zusätzlich noch eine Startwahrscheinlichkeit π_1 . Die modellierten Phoneme werden den Emissionswahrscheinlichkeiten $P(x_n | q_i)$ zugeordnet, also der Wahrscheinlichkeit, dass der Zustand die Beobachtung des Phonems macht. Der untere Teil zeigt die Phoneme nach der Merkmalsextraktion.

3.3 Neuronale Netze

Ein modernerer Ansatz zur Spracherkennung ist die Nutzung von neuronalen Netzen zur Mustererkennung. Durch die gestiegene Leistung der Hardware sowie die Unterstützung durch GPU-Berechnungen kann heutzutage ein ausreichend großes Netz trainiert werden, um eine hohe Erkennungsgenauigkeit zu erreichen. Meist werden Hybride, d.h. eine Kombination von neuronalen Netzen und Hidden Markov Modellen, zum Dekodieren von Phonemen eingesetzt (vgl. Abschnitt: 3.4 “Automatic Speech Recognition“)

Vereinfacht dargestellt ist die Funktionsweise eines neuronalen Netzes in Abbildung 3.4. Den Eingabewerten $x(1)$ bis $x(n)$ werden an ihren Kanten die Gewichte $W(1)$ bis $W(n)$ zugeordnet. Die Summenfunktion multipliziert die Eingabewerte mit ihren Gewichten

¹² <https://files.ifi.uzh.ch/cl/volk/LexMorphVorl/Lexikon06.Freq.html> (Dr. Martin Volk, Vorlesungsmanskript), Zugriff 23.06.2020.

¹³ <http://www.informatik.uni-ulm.de/ni/Lehre/WS04/HSemSprache/ausarbeitungen/Hallerbach.pdf> (Andreas Hallerbach 2005), Zugriff 23.06.2020.

und kumuliert diese auf. Das Ergebnis wird zusammen mit einem Schwellwert in eine Aktivierungsfunktion gegeben. Diese erzeugt eine Ausgabe Y .

In der Trainingsphase des Netzes vergleicht die künstliche Intelligenz dann den Ausgabewert mit dem erwarteten Wert. Anhand einer Kostenfunktion wird der Fehler zwischen den Werten ermittelt. Ziel ist nun, diesen Unterschied zu minimieren. Sind diese beiden Werte gleich, werden keine Änderungen durchgeführt [7] [8].

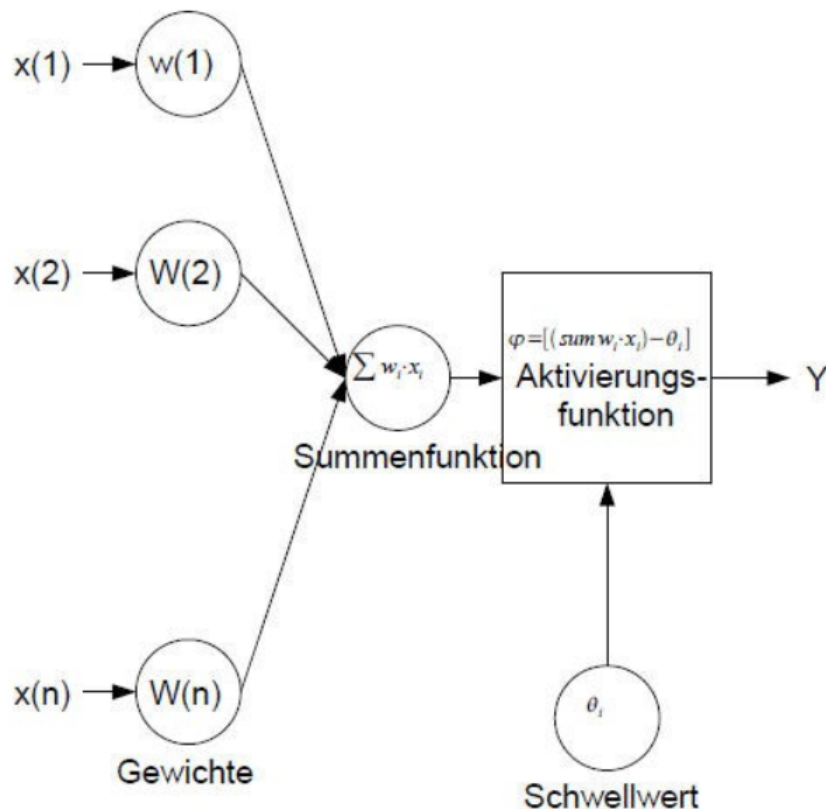


Abbildung 3.4: Funktionsweise eines neuronalen Netzes¹⁴

Die künstliche Intelligenz wird dabei nach der Merkmalsextraktion eingesetzt (siehe Abbildung 3.3 unten). Sie bestimmt anhand ihrer Topologie, welches Phonem am besten zu dem Merkmalsvektor passt und vergleicht dies mit den "gelabelten" Daten. Ist diese Validierung ausreichend gut, beendet der Algorithmus das Training. Andernfalls wird ein neuer Durchlauf mit veränderten Parametern durchgeführt.

Der Vorteil liegt in der Vielzahl an freien Parametern (Topologie des Netzes, Kosten- und Aktivierungsfunktion, Epochen, Gewichte), die das Netz beschreiben können. Durch das Variieren der unterschiedlichen Parameter kann eine geringere Fehlerrate erreicht werden.

¹⁴ <https://www.freebasic-portal.de/tutorials/neuronale-netze-grundlagen-35.html>, Zugriff 30.06.2020.

3.4 Automatic Speech Recognition kurz ASR

“Automatische Umwandlung menschlicher, gesprochener Sprache in die dazugehörige Wortsequenz in maschinenverarbeitbarer Form.“

Definition durch Dr. Sebastian Stücker

Sprechen mehrere Personen ein und denselben Satz, kommen diese nicht zu ein und demselben Ergebnis. Sprache ist abhängig von Dialekten, Sprechgeschwindigkeit, anatomischen Besonderheiten, emotionaler Lage des Sprechers und vielem mehr. Die ASR Modelle können deswegen keine exakte Bestimmung durchführen, sondern nur eine Wahrscheinlichkeit wiedergeben, dass das jeweilige Wort gesprochen wurde.

3.4.1 Funktionsweise ASR

Die Funktionsweise eines automatischen Spracherkennungsprozesses besteht grundsätzlich aus drei Hauptbestandteilen (s. Abbildung 3.5). Dazu gehören die Vorverarbeitung, das Erkennen und die Ausgabe des wahrscheinlichsten Wortes. In der Vorverarbeitung wird die analoge Sprache in digitale Signale umgewandelt und mit Hilfe der *Voice Activity Detection* (s. Kapitel 3.4.3) in kleine Abschnitte unterteilt. Des Weiteren findet in der Vorverarbeitung auch das Filtern des Signals statt (Abgrenzung des Signals zwischen Nutz- und Störgeräuschen).

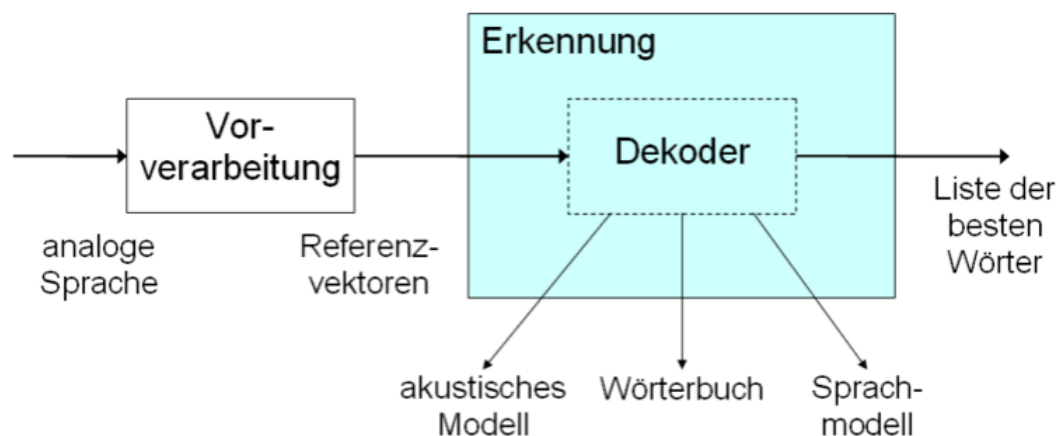


Abbildung 3.5: Funktionsweise ASR-Modell nach Prof. Dr. Alexander Waibel¹⁵

Durch die Fast Fourier-Transformation zerlegt man das zeitdiskrete Signal in seine Frequenzen. Von dem “sauberen“ und transformierten Signal wird anhand von markanten

¹⁵ <https://de.wikipedia.org/wiki/Spracherkennung>, Zugriff 28.06.2020.

Merkmalen ein Merkmalsvektor (*“feature vector“*) erstellt. Diese Merkmale lassen sich durch vordefinierte Regeln verändern bzw. gewichten.

Das akustische Modell stellt nun die Verbindung zwischen den Merkmalsvektoren und den Phonemen dar und gibt, akustisch betrachtet, die Wahrscheinlichkeit wieder, welches Phonem gesprochen wurde. Dies geschieht mit Hilfe des Wörterbuches. In diesem stehen alle dem Modell bekannten Wörter in deren Lautschrift als Baumdiagramm dargestellt (s. Abbildung 3.6). So kann sich der Algorithmus den optimalen Pfad suchen. Die daraus entstandene Phonemkombination wird mit dem Sprachmodell verglichen. Das Sprachmodell ist eine Liste von statistischen Werten, die angibt, wie groß die Wahrscheinlichkeit ist, dass die gefundene Kombination ein bestimmtes Wort ergibt. Das Wort mit der größten Übereinstimmung wird ausgegeben¹⁶.

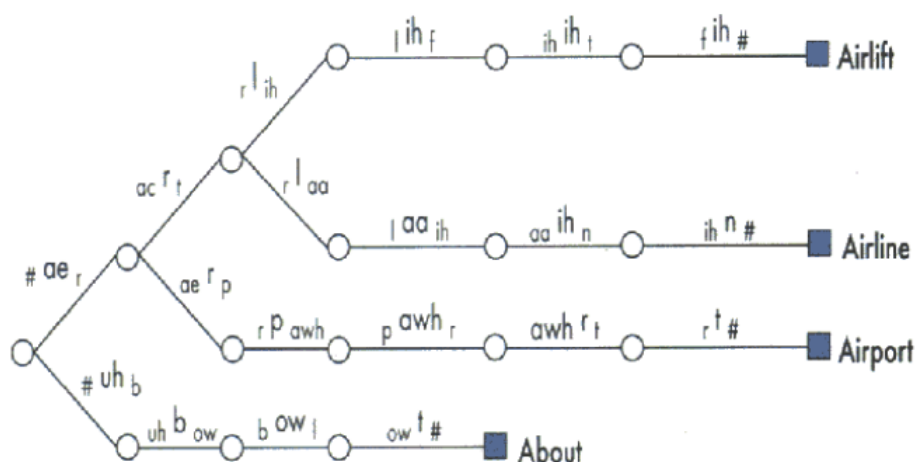


Abbildung 3.6: Beispiel Wörterbuch. Der Algorithmus sucht den Phonem-Pfad der höchsten Wahrscheinlichkeit¹³. Entlang des Pfades berechnet das Modell den stochastisch wahrscheinlichsten Wert und folgt dann den Kanten zum Wort.

3.4.2 Qualität / Evaluation der ASR - Modelle

Um die verschiedenen ASR-Modelle miteinander vergleichen zu können, wurden verschiedene Metriken eingeführt. Die *Word Error Rate* ist die am weitesten verbreitete Einheit. In dieser Arbeit werden nur die folgenden betrachtet.

¹⁶ <https://nats-www.informatik.uni-hamburg.de/pub/DIALSYS/VeranstaltungsMaterial/Sprachsignal.pdf>, Zugriff 28.06.2020

Word Error Rate (WER)

- gibt den Anteil von Fehlern in einem transkribierten Text im Vergleich zur Referenz an (in %)

$$WER = \frac{S_W + D_W + I_W}{N_W} \cdot 100$$

mit:

- I_W – Anzahl der Insertions (Einfügung)
- S_W – Anzahl der Substitutions (Vertauschung)
- D_W – Anzahl der Deletions (*Löschung*)
- N_W – Anzahl der Wörter im Referenztext

Das Beispiel in Abbildung 3.7 zeigt die Berechnung der Word Error Rate. Der Referenztext wird mit dem vom Programm erkannten Text auf Einfügungen, Vertauschungen und Löschungen verglichen. Die jeweiligen Fehler haben eine Gewichtung von 1. Diese wird aufaddiert. Der erste Fehler ist eine Insertion (Einfügung), da der Spracherkenner das Wort "ja" erkannt hat, dieses jedoch nicht im Referenztext vorkommt. Beim zweiten Fehler wird anstatt "wir" "vier" erkannt und damit erfolgt eine Vertauschung (Substitution). Wenn das Modell ein Wort aus dem Referenztext nicht erkennt, ist dies eine Löschung (Deletion) wie hier beim Wort "nach".

Die Wortfehlerrate ergibt sich dann aus den kumulierten Fehlern, geteilt durch die Anzahl der Wörter, multipliziert mit einhundert, um eine Prozentzahl zu erhalten.

Referenz :	***	Elvis	sagt	wir	fahren	nach	Memphis
Hypothese:	ja	Elvis	sagt	vier	fahren	***	Memphis
Fehlerklasse:	INS	-	-	SUB	-	DEL	-
Wortfehlerrate:	$WER = 100 \cdot \frac{1+1+1}{6} = 50$						

Abbildung 3.7: Beispiel der WER-Berechnung - zu sehen sind die drei Fehler INS (Insertion), SUB (Substitution) und DEL (Deletion)¹⁷.

Word Recognition Rate (WRR)

- ist ein Maß zur Bestimmung der Leistung eines Spracherkennungssystems

$$WRR = 1 - WER$$

¹⁷ <https://files.ifi.uzh.ch/cl/siclemat/lehre/hs09/ecl1/script/html/scriptse38.html>, Zugriff 24.08.2020.

3.4.3 Probleme der ASR

Die ASR ist ein weitgefasstes und schwieriges informationstechnisches Problem. Trotz enormer Leistungssteigerung kommen die Algorithmen noch lange nicht an die menschliche Erkennungsrate heran. Dies liegt an folgenden grundsätzlichen Problemen:

Phonetische Probleme:	unterschiedliche Aussprache der Wörter (Dialekte, schnell oder langsam)
Linguistische Probleme:	gleich klingende Wörter mit unterschiedlicher Bedeutung (z.B. das Wort "Tau") können nicht durch Kontextwissen, sondern nur durch Statistiken erkannt werden
Metawissen:	Schwierigkeiten beim Erkennen von "noch nicht gesehenen Wörtern" bzw. beim Erschließen aus dem Kontext
Voice Activity Detection:	Sprachsegmente müssen von Intervallen des Schweigens abgegrenzt werden, um zu verhindern, dass Hintergrundrauschen zur Worterkennung genutzt wird.
Wortgrenzen:	bei fließend gesprochenen Texten verwischen die Grenzen zweier Wörter

Hinzu muss das Problem der Stör- und Hintergrundgeräusche gelöst werden. Eine Stimme klingt in einem kleinen Raum anders als in einem großen Saal. Aber auch einfache Windgeräusche am Mikrophon können den Signalverlauf einer Stimme nachhaltig ändern.

3.4.4 Lösungen für die Forensik

Die jeweiligen Modelle müssen sehr flexibel gestaltet werden, da keine Eins-zu-eins-Übereinstimmung zum jeweiligen Wort vorherrschen kann. Das Modell berechnet hier eine Wahrscheinlichkeit, die das Auftreten des Wortes beschreibt. Häufig wird dabei eine Kurzzeit-Fouriertransformation (FFT) [10] genutzt. Mit ihr kann ein zeitdiskretes Signal in seine Frequenzanteile zerlegt werden. Es erfolgt eine Stückelung des Nutzsymbols in kurze, überlappende Singale. Diese sogenannten Segmente werden mit Hilfe der Fourier-Transformation in Frames umgewandelt. Mit den Frames kann dann ein Kurzzeitspektrum des Signals erstellt werden. Durch dieses Spektrum ist es besser möglich, die Ähnlichkeiten der gesprochenen Wörter zu vergleichen (s. Abbildung 3.8).

Ein wichtiger Punkt für den forensischen Einsatz ist die *Voice Activity Detection*. Da die bereitgestellten Audio-Daten sehr oft mit Störgeräuschen durchsetzt sind, muss eine klare Trennung von Nutz- und Störsignal stattfinden. Dabei sollten keine Sprachsegmente als Störsegmente klassifiziert und keine Wörter zerschnitten werden. Man benutzt dazu zwei einfache Verfahren:

¹⁸ <https://stackoverflow.com/questions/48510984/how-do-i-display-a-spectrogram-from-a-wav-file-in-c>, Zugriff 24.06.2020.

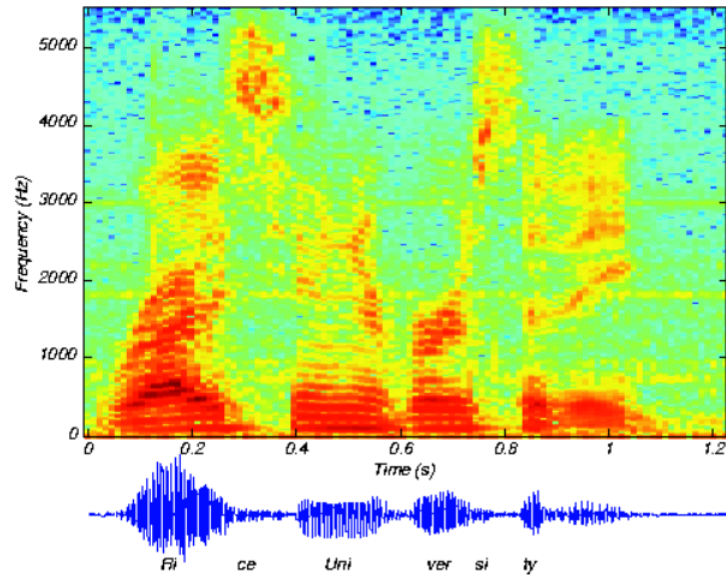


Abbildung 3.8: FFT¹⁸. Unten: der Schalldruckpegel zu dem gesprochenen Wort *University*. Oben: Das dazugehörige FFT - Spektrogramm gibt die Frequenzanteile des eingespeisten Signals zu jedem Zeitpunkt wieder.

Nulldurchgangsrate

Die Nulldurchgangsrate (engl. "zero crossing rate" (ZCR) ist die Frequenz, mit der das Signal die x-Achse pro Sekunde schneidet, dividiert durch zwei. Rauschen, Frikative und Lösungen von Plosiven haben eine hohe ZCR, Vokale und Nasale eine geringe. Mit Hilfe der ZCR kann man Audiosignale grob in Klassen einteilen.

Energie des Signals

Die Energie des Schalls ist die Fläche unter der Kurve einer Schalldruckschwingung. Es ist mit diesem Wert möglich, Sequenzen mit hoher Energie herauszufiltern und zu verarbeiten (s. Abbildung 3.9).

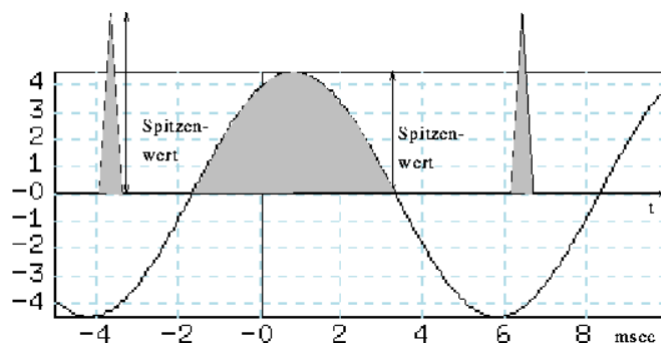


Abbildung 3.9: Exemplarisch sind hier drei Schallschwingungen zu sehen. Die grauen Bereiche geben die Energie des Signals wieder¹⁹.

¹⁹ <https://www.phonetik.uni-muenchen.de/studium/skripten/AP/APKap1.html>, Zugriff 24.06.2020.

Es herrscht ein gut-klassifizierbarer Unterschied bei der ZCR zwischen Sprachsignalen und Hintergrundrauschen vor. Man kann im einfachsten Fall die Aufnahme mit Hilfe der ZCR in drei Parts unterteilen: Sprache vokal K_1 , Sprache frikal K_2 und Hintergrundgeräusche K_3 . Das Hintergrundsignal besitzt eine geringere Energie als die beiden Sprachsignale. Durch das Hinzufügen der Signalenergie wird eine mehrdimensionale Verteilungsdichte erzeugt. Man kann somit eine eindeutige Klassifizierung von Datensätzen durchführen (s. Abbildung 3.10).

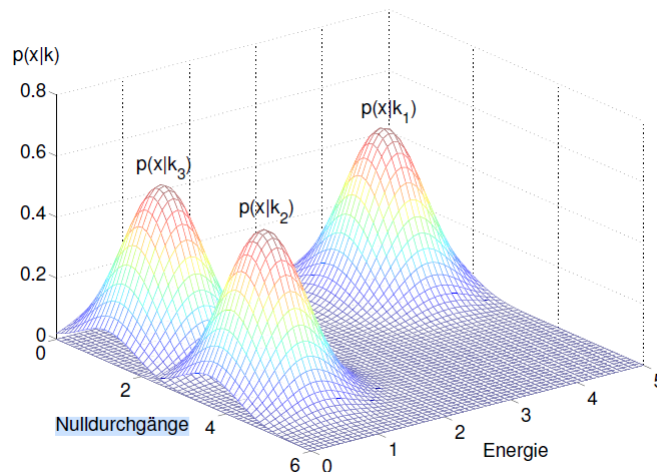


Abbildung 3.10: Die Zusammenführung der Nulldurchgänge und der Energie des Signals erzeugt eine Verteilungsdichte, um die Wahrscheinlichkeit der Zustände K_1 , K_2 und K_3 zu beschreiben¹³.

Der Einfluss der ZCR und der Energie lassen sich durch Parameter manipulieren, um das System auf die erforderliche Umgebung anzupassen. Auch das Einführen weiterer Klassifikatoren K_n sowie eines Schwellenwertes können zu einer deutlichen Verbesserung führen [11].

3.5 Die zur Evaluation verwendeten Modelle

Die in dieser Arbeit verwendeten *automatic speech recognition* Modelle sind alle Open Source und befinden sich im ständigen Wandel. Durch die rege Community werden sie stets weiter optimiert. Bis auf die Google Web Speech API können bei allen Modellen Neuerungen und Patches auf den jeweiligen Github Seiten nachvollzogen werden.

3.5.1 Kaldi TUDA Projekt

Das Kaldi-TUDA Projekt wird durch die Universität Hamburg²⁰ vorangetrieben. Über deren Github-Site²¹ erhalten die Nutzer neue vortrainierte Modelle und Veröffentlichungen zu ihrer Verfügung. Genutzt wird ein hybrides System von *Gaussian Mixture Model (GMM)*, *Hidden Markov Model* und *Time-Delayed Neural Networks (TDNNs)*. Mehr als 350.029 Phonemkombinationen befinden sich im Wörterbuch. Das Sprachmodell wurde mit einer 2-layer *Long-Short-Term Memory (LSTM)* antrainiert [12]. Mit Kaldi²² wird ein auf C++ basierendes Toolkit genutzt, welches sich auf Sprach- und Signalverarbeitung spezialisiert hat. Es ist flexibel-modular aufgebaut und kann von Sprachwissenschaftlern beliebig erweitert werden. Durch seine Fähigkeit, spezielle Merkmale aus den Sprachsignalen zu extrahieren, eignet sich Kaldi sehr gut für die automatische Spracherkennung.

Um die vortrainierten Modelle zu starten, wird ein *Kaldi GStreamer* Server und ein Client benötigt. Der GStreamer ist ein Multimedia - Toolkit. Es erleichtert den Datenfluss von Audio- oder Videodaten zwischen Host- und Clientsystemen. Das Aufsetzen eines *GStreamers* ist jedoch kompliziert und kann leicht fehlschlagen, deswegen wird von den Betreibern des Projektes empfohlen, ein konfiguriertes System aus dem Docker - Container zu nutzen (`docker-kaldi-gstream-server` von `jcsilva`²³).

Gaussian Mixture Model Das GMM ist eine statistische Technik. Mit ihr kann man unter anderem Merkmale des Stimmenspektrums in Cluster einteilen. Es leitet sich aus der Addition mehrerer Gauß-Kurven ab. Genutzt wird dieses Mix-Modell, um die Spracherkennung flexibler auf verschiedene Sprecher zu trainieren. Meist werden zwischen 5-30 Mischkomponenten (Gauß-Kurven) addiert. Diese entsprechen den extrahierten Merkmalen. Die Wahrscheinlichkeitsdichtefunktion (Fläche unter der gemischten Kurve) gibt die Wahrscheinlichkeit der Beobachtung an [11].

Time Delayed Neural Network TDNN ist ein Spezialfall der neuronalen Netzwerke. Mit ihrer Hilfe kann man eine zeitliche Abhängigkeit im Input des Netzes erzeugen. Dadurch lassen sich Abweichungen in der Aussprache von Wörtern trotzdem noch den gleichen Phonemen zuordnen [13].

Long Short-Term Memory LSTM ist eine besondere Art der neuronalen Netze. Durch die Einführung der Input-, Forget- und Output-Gates wurde eine Art "Gedächtnis" in das Netz implementiert. Damit hat das Netz die Möglichkeit, auf bekanntes Wissen besser zuzugreifen. Das ASR-Modell kann damit anhand vorheriger erkannter Wörter die ungefähre Richtung des aktuell zu bearbeitenden Wortes festlegen. Besonders bei komplexen Netzwerktopologien ist dies stark von Vorteil.

²⁰ <https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/acoustic-models.html>, Zugriff 04.07.2020

²¹ <https://github.com/uhh-lt/kaldi-tuda-de>, Zugriff 04.07.2020

²² <https://kaldi-asr.org/doc/index.html>, Zugriff 05.07.2020

²³ <https://github.com/jcsilva/docker-kaldi-gstreamer-server>, Zugriff 05.07.2020

3.5.2 Deep Speech

Deep Speech setzt auf das *Recurrent Neural Network (RNN)*. Anhand dieser Technik entfällt das akustische Modell und die HMMs. Nachteil ist aber, dass dem Netzwerk eine sehr große Anzahl von Trainingsdaten zu Verfügung gestellt werden muss und diese Daten auch eine lange Zeit zum Durchlaufen benötigen. Demzufolge mussten die Entwickler eine Parallelverarbeitung der Daten realisieren.

Deep Speech extrahiert durch die RNN-Topologie sogenannte *Mel-Frequenz-Cepstrum-Koeffizienten (MFCC)* als Merkmal. Die Aufgabe der MFCCs ist es, durch Analyse des Frequenzspektrums auf die Form des menschlichen Stimmapparates bei Abgabe des Lautes zu schließen. Daraus kann das gesprochene Phonem abgeleitet werden. Die Abbildung 3.11 zeigt die Architektur.

Zur Erstellung des Sprachmodells wird *KenLM*²⁴, ein Toolkit zum Trainieren von *N-Gramme*, genutzt. [5] [14] [15]

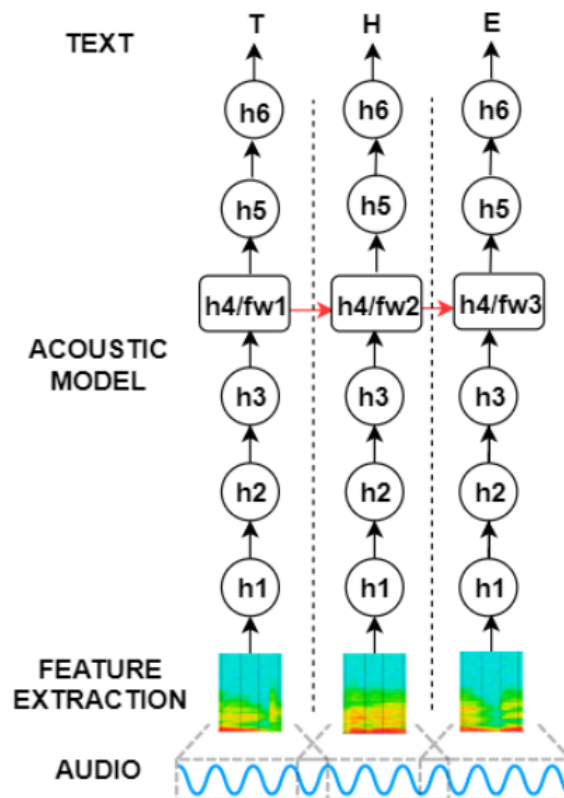


Abbildung 3.11: Deep Speech Architektur²⁵ Die Topologie besteht aus sechs Schichten (Layer), eingeteilt in drei (h1 - h3) voll verbundene Layer (Dense-Layer), gefolgt von einer RNN-Schicht (h4/fw1 - h4/fw3), wieder einem Dense-Layer (h5) und einer Ausgabeschicht(h6). Die Ausgabe ist dann die Wahrscheinlichkeit, welche Phonemkombination gesprochen wurde. Diese Architektur ersetzt das akustische Modell.

²⁴ <https://kheafield.com/code/kenlm/>, Zugriff 05.07.2020

²⁵ <https://hacks.mozilla.org/2018/09/speech-recognition-deepspeech/>, Zugriff 06.07.2020.

Zum Zeitpunkt dieser Arbeit standen zwei Versionen des vortrainierten Modells zur Verfügung.

Recurrent Neural Network RNNs sind eine Sonderform der neuronalen Netzwerke. Die Neuronen können hierbei ihre Signale in einem geschlossenen Kreis wieder an ihren Input weiterleiten. Dadurch ist es dem Netzwerk möglich, Informationen aus zeitlich auseinanderliegenden Beobachtungen zu berücksichtigen. RNNs eignen sich daher besonders für die Verarbeitung von sequentiellen Daten, wie Sprache oder Text.²⁶

N-Gramms bezeichnen die Zerlegung von Texten in Buchstaben oder Phoneme (Fragmente). Das N gibt die Anzahl der verbundenen aufeinanderfolgenden Fragmente an, die betrachtet werden. Sie dienen einer statistischen Analyse und zeigen, wie wahrscheinlich es ist, dass auf eine bestimmte Folge von N-Grammen das nächste folgt. Eine einfache Anwendung dazu ist die Satzvervollständigung in Handys.

3.5.3 Google Web Speech API (GWS API)

Die *GWS API*²⁷ ist eine JavaScript-Schnittstelle, mit der man eine Spracherkennung in Webseiten implementieren kann. Wie das Modell aufgebaut und trainiert wurde, gibt Google nicht preis. Möglicherweise orientiert sich diese Schnittstelle an der kostenpflichtigen *Google Cloud Speech*-Technologie.

3.5.4 wave2letter

wave2letter ist ein effizientes ASR-System der "Facebook AI Research"²⁸. Es basiert auf einem Convolutional Neural Network (CNN), welches mit diversen Merkmalen (MFC-Cs, Energie des Signals) angereichert wird. Durch eine Connectionist Temporal Classification (CTC) wird die Kostenfunktion intelligent geleitet und kann somit die Trainingszeit erheblich verkürzen [16]. Bereitgestellt wird das vortrainierte Modell vom "Zamia-Speech Project"²⁹.

Convolutional Neural Network Das CNN ist eine spezielle Form eines neuronalen Netzes. Es besteht grundsätzlich aus Convolutional- und Pooling Layer im Verbund. Dieser Verbund kann sich beliebig oft wiederholen. Der Convolutional-Layer faltet die Eingabe, der Pooling Layer entfernt überflüssige Informationen. Es folgt ein Dense Layer, bei dem wieder alle Eingabewerte mit allen Ausgabewerten verbunden sind.

²⁶ <https://www.statworx.com/de/blog/deep-learning-teil-1-einfuehrung/>, Zugriff 06.07.2020

²⁷ <https://wicg.github.io/speech-api/#introduction>, Zugriff 06.07.2020

²⁸ <https://research.fb.com/downloads/wav2letter/>, Zugriff 07.07.2020

²⁹ <https://github.com/goofy/zamia-speech#german-wav2letter-models>, Zugriff 07.07.2020

Connectionist Temporal Classification CTC ist eine Fehlerfunktion für neuronale Netzwerke, die eingesetzt wird, um Sequenzprobleme (Handschrifterkennung, Phonemerkennung) zu lösen.

4 Experimente und Evaluation

4.1 Übersicht

Die hier vorgestellten Modelle besitzen alle (bis auf die GWS API) ihre eigenen Test- und Trainingssets im Split von 20% zu 80%. Tabelle 4.1 zeigt die verwendeten Modelle im Kurzüberblick.

Name	Sprachmodell	Datenkorpus	Trainingsdaten
Kaldi-TUDA Projekt	GMM/TDNN-HMM	TUDA + SWC + M-Ailabs + CV	1000 h
Kaldi-TUDA Projekt	GMM/TDNN-HMM	TUDA + SWC + M-Ailabs	630 h
Deep Speech v0.6.0	RNN	TUDA + CV + Voxforge + M-Ailabs	928 h
Deep Speech v0.5.0	RNN	TUDA + CV + Voxforge	645 h
GWS API	n.a.	n.a.	n.a.
wav2letter	CNN	Voxforge + TUDA + Forschergeist	400 h

Tabelle 4.1: Überblick über die verwendeten Modelle

Um nun die jeweiligen Modelle gegeneinander zu evaluieren, müssen diese auf einheitliche Testdaten geprüft werden. Diese Testdaten sollten die ASRs vor verschiedene Herausforderungen stellen, um die jeweilige Robustheit zu überprüfen.

Es wurden daher neun verschiedene Audio-Files ausgewählt (vgl. Tabelle 4.2). Unter diesen sind gesprochene Texte mit und ohne leichtem Dialekt, gesprochen von männlichen und weiblichen Sprechern. Einzig *“chemnitz”* ist mit sehr starkem sächsischen Dialekt eingespielt worden. Diese Datei lässt Rückschlüsse darauf ziehen, wie sich die ASR-Modelle im polizeilichen Einsatz schlagen, da es dabei meist um die Transkribierung von nicht hochdeutsch sprechenden Personen geht.

Die Hälfte der Dateien beinhaltet Füllworte wie *“ähm”* und *“äh”* sowie weitere umgangssprachliche Besonderheiten und Eigenworte.

Mit *“slow”* wurde dem Testset eine File hinzugefügt, die sehr lang ist und einfache, langsame Sprache nutzt.

Die File *“lola5”* und *“ste011”* stammen von der Seite

[“https://www.audio-lingua.eu/spip.php?rubrique3”](https://www.audio-lingua.eu/spip.php?rubrique3) und besitzen das Sprachniveau A1.

Die Daten *“ste013 - 16 und ste028”* sind ebenfalls von obiger Seite. Sie besitzen das Sprachniveau A2 und haben eine durchschnittliche Länge von 16 Sekunden.

Die Datei “*slow*“ wurde von der Seite “<https://slowgerman.com/2020/05/31/aberglaube-in-deutschland-sg-209/>“ heruntergeladen. Sie ist mit Abstand die längste File mit einer langsamen deutschen Sprache.

Das Sample “*chemnitz*“ ist ein Videomitschnitt vom “Sachsen-Fernsehen“. Diesen gibt es bei YouTube unter: <https://www.youtube.com/watch?v=OsFWiJOL6fs>. Alle Dateien wurden als mp3 heruntergeladen und in das WAV - Format auf 16kHz migriert, um auch hier eine einheitliche Ausgangslage herzustellen. Bis auf *slow* mussten alle Sprachsamples selbst transkribiert werden. Insgesamt liegt ein Testset von 959 Wörtern und eine Gesamtspiellänge von 9:38 min vor. Die Ergebnisse jedes einzelnen Modells werden gemittelt, um einen Vergleichswert zu erhalten. Die Tabelle 4.2 zeigt die Samples im Überblick.

Name	Länge	Dialekt	Geschlecht	Besonderheit
lola5	0:26 min	keiner	weiblich	einfache Sprache
ste011	0:16 min	keiner	männlich	einfache Sprache
ste013	0:18 min	keiner	männlich	einige Füllwörter
ste014	0:22 min	leichter Dialekt	weiblich	einige Füllwörter
ste015	0:19 min	leichter Dialekt	weiblich	schnell gesprochen, umgangssprachlich
ste016	0:17 min	leichter Dialekt	männlich	langsam, aber mit Füllwörter
ste028	0:32 min	mittelstarker Dialekt	weiblich	umgangssprachlich
slow	6:41 min	keiner	weiblich	langsam und einfache Sprache
chemnitz	0:27 min	starker Dialekt	männlich	sehr schwer verständlich

Tabelle 4.2: Überblick über die Test - Sprachsamples ohne Hintergrundrauschen

Die durch die ASRs erstellten Textdateien werden durch eine Pipe von Vorverarbeitungsskripten für die Evaluation vorbereitet. Darunter fällt die Kleinschreibung aller Wörter und das Entfernen von Satzzeichen und Füllwörtern. Wörter werden nicht als identisch erkannt, wenn sie eine unterschiedliche Groß-/Kleinschreibung besitzen oder wenn sie mit einem Satzzeichen verbunden sind. All diese Schritte sind notwendig, um eine belastbare Evaluation durchzuführen.

4.2 Testdurchlauf: Ohne Hintergrundrauschen

Der erste Durchlauf wurde ohne Hintergrundrauschen durchgeführt. Die Ergebnisse lassen sich in Tabelle 4.3 ablesen. Klar erkennbar ist, dass die GWS API mit einer WER von gerade 16% weit vor den beiden Kaldi Modellen und der Version 0.6.0 des Deep Speech Modells liegt. Die CNN des wave2letter Modells folgt nur knapp hinter diesen. Weit abgeschlagen ist das Deep Speech 0.5.0 Modell.

Interessanterweise liegt die WRR der GWS (79%) nur knapp höher als die des Kaldi 1000h Modells (75%) und das, obwohl die WER bei Kaldi doppelt so hoch ist. Gefolgt werden diese beiden ASRs von Kaldi 630h, Deep Speech v0.6.0 und wave2letter. Weit abgeschlagen ist wieder die veraltete 0.5.0 Version von Deep Speech.

Name	WER	WRR
Kaldi 630h	34 %	66 %
Kaldi 1000h	32 %	68 %
DeepSpeech v0.5.0	76 %	24 %
DeepSpeech v0.6.0	42 %	58 %
GWS API	16 %	84 %
wav2letter	43 %	57 %
	40,5 %	59,5 %

Tabelle 4.3: Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz

Sieht man von der Google API einmal ab, da hier keine Parameter bekannt sind, zeigt sich, dass es unter den jeweiligen Modellarten ein Abflachen der WER mit zunehmenden Trainingsstunden gibt. Deep Speech v0.6.0 (928 Trainingsstunden) verbesserte sich gegenüber seiner Vorgängerversion (645 Trainingsstunden) um ganze 34 %.

Im Kaldi Modell ist diese Entwicklung jedoch nicht ganz so drastisch. Hier bringt eine Erhöhung der Trainingsstunden um 370h nur einen um 2 % verringerten WER.

Im Vergleich dazu wird wave2letter mit nur 400 Stunden trainiert, erreicht aber eine ähnliche WER wie DeepSpeech v0.5.0.

Bei der WRR liegt die GWS wieder vor den beiden Kaldi Modellen, gefolgt von DeepSpeech v0.6.0 und dem wave2letter. Weit abgeschlagen ist wieder das ältere DeepSpeech Modell. Durchschnittlich lässt sich sagen, dass die Modelle drei von fünf Wörtern richtig erkennen.

Abbildung 4.1 zeigt die jeweiligen WER der Modelle auf den einzelnen Testdaten. Man sieht, dass das beste Modell zweimal eine WER von 0 % erreicht, aber auch wie alle anderen ASRs am Sample "chemnitz" scheitert und annähernd 100 % Fehlerrate erreichte.

Man erkennt weiter, dass die beiden Kaldi Modelle nah beieinander liegen, in manchen Fällen auch die gleichen Fehlerraten erreichen, das 1000h-Modell aber trotzdem immer

ein klein wenig besser ist als das 630h-Modell.

Des Weiteren ist ein starker Ausbruch der Google API auf dem Testset "ste15" zu erkennen. Allen Modellen, bis auf Kaldi 630h, bei dem eine Erhöhung der WER sichtbar ist, bereitete das Sample "slow" weniger Probleme in der Transkription. Bei beiden DeepSpeech Systemen ist hier auch die geringste Fehlerrate zu erkennen.

Um die beiden Extreme zu nennen, gab es die geringste WER bei dem Sample "lola" und die höchste bei "chemnitz".

DeepSpeech v0.5.0 lieferte bei diesem Test die schlechtesten Werte. Im Mittel hatte sie eine Fehlerrate von 76 %. Einzig "ste16" mit 64 % konnte die sehr schlechten Werte etwas mindern. Das liegt hauptsächlich daran, dass dieses Modell im Test nur Personalpronomen und Bindewörter erkennen konnte. Je mehr solche Worte also in den Testdaten vorkamen, desto besser wurde die Erkennungsrate.

wave2letter und DeepSpeech v0.6.0 lagen dicht beieinander (vgl. Abbildung 4.1). Obwohl sie auf unterschiedlichen Systemen beruhen, hatten beide Probleme mit Zahlen. Diese wurden zwar oft erkannt, mussten dann aber in den Evaluationsdaten ausgeschrieben werden, damit es zu keinem höheren WER kam. Außerdem haben beide Modelle keine deutschen Namen in den Trainingsdaten, so dass es immer wieder zu Fehlern kam, wenn in den Testdaten Namen gesprochen wurden.

Die beiden Kaldi Modelle unterscheiden sich in nur 2 Prozentpunkten und haben demzufolge mit den gleichen Problemen zu kämpfen. Für beide ist es schwierig, Eigenworte, Namen und Stadtnamen zu erkennen. Es ist auch bei diesen Modellen nötig, die Zahlen im Evaluationstext auszuschreiben, da sie nicht automatisch in Ziffern transkribiert werden. Im Gegensatz zur 630h-Version hat die 1000h ein verbessertes Sprachmodell implementiert, so dass es für dieses Modell möglich war, unbekannte Wörter (meist umgangssprachliche) besser zu erkennen.

Die Google API ist die beste Spracherkennung aus dem ersten Test. Ihre WER liegt zwischen 0 und 55 %. Der starke Ausreißer bei "ste15" kann damit erklärt werden, dass es sich um ein schnelles und umgangssprachlich gesprochenes Sample handelt. Weitere Probleme für diese ASR sind die Länge der Files. Die Audio-Datei "slow" musste in zwei Teile geteilt werden, damit sie von der GWS erkannt und transkribiert werden konnte. Doch auch die File "chemnitz" war für das Modell nicht zu lösen.

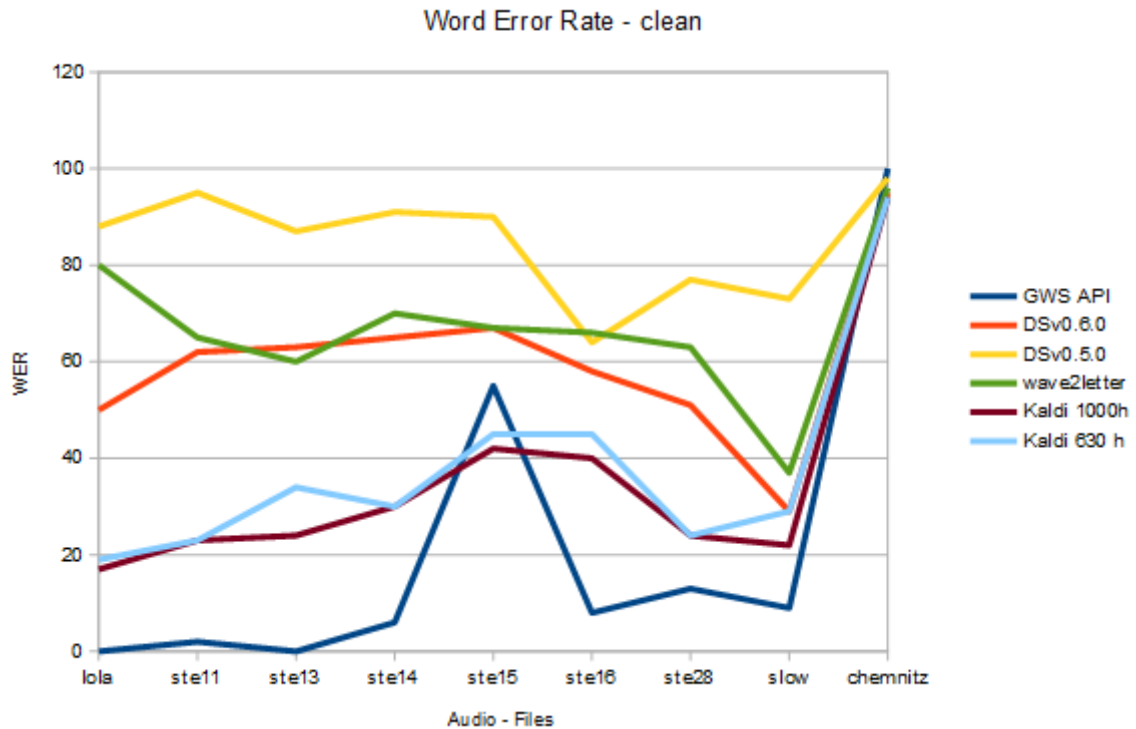


Abbildung 4.1: Evaluation der sechs verschiedenen ASR Modelle auf einen selbst erstellten Testdatensatz

Tabelle 4.4 zeigt die genaue Aufschlüsselung der Word Error Rate auf die jeweiligen Sprachdaten für die Evaluierung in normaler Sprechgeschwindigkeit. Im Test am besten konnten die Modelle mit der File "slow" (Durchschnittliche WER von 32 %) umgehen. Diese wurde langsam, ohne Dialekt und in einfacher Sprache gesprochen. Sehr schlecht konnten die ASRs mit "ste15" (61,2 %) und weit abgeschlagen "chemnitz" (96,7 %) umgehen. Daraus leitet sich die Theorie ab, dass die Erkennungsrate geringer wird, je höher die Sprechgeschwindigkeit und je mehr Dialekt in den Daten vorhanden ist.

Name	WER
lola	37,5%
ste11	44,2%
ste13	45,2%
ste14	47,7%
ste15	61,2%
ste16	45,2%
ste28	41,0%
slow	32,0%
chemnitz	96,7%

Tabelle 4.4: Gemittelte Word Error Rate über die gesamten Modelle

Um einen besseren Vergleich des Inputs der ASR Modelle zu bekommen, wurden das am besten und das am schlechtesten erkannte Sprachfile in ein Spektrogramm transferiert und hier exemplarisch das Wort "Ich" dargestellt (vgl. Abbildung 4.2). Man kann deutlich die Unterschiede der Frequenzen erkennen: im linken Spektrogramm die klar abgegrenzten Bereiche des Lautes "I" (im unteren Teil) und der typische Frequenzverlauf des stimmlosen palatalen Frikativs (oberer Bereich). Das rechte Spektrogramm zeigt hingegen fast einen homogenen Frequenzverlauf im gesamten Spektrum. Dies ist ein typisches Bild bei einem undeutlichen Sprecher.

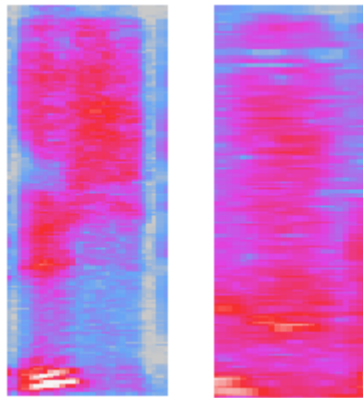


Abbildung 4.2: Spektrografischer Vergleich des Wortes "Ich" der Sprachdateien "slow" (links) und "chemnitz" (rechts)

Das gute Abschneiden des "slow" - Sets lässt vermuten, dass die Verringerung der Sprechgeschwindigkeit eine Verbesserung der Word Error Rate zufolge hat.

4.2.1 Testdurchlauf: Ohne Hintergrundrauschen - langsam

Um die obige These zu überprüfen, wurde das Testdatenset mit Hilfe des Audio-Bearbeitungsprogramms "Sonic Visualiser"³⁰ um 15 % verlangsamt. Dieser Wert unterliegt einer subjektiven Annahme, bei der eine schrittweise Verlangsamung solange durchgeführt wurde, bis sich eine hörbare Veränderung im Sprechtempo einstellte.

Auch bei den verlangsamt abgespielten Testdaten liegt die Google Web Speech API klar mit einer WER von 13 % vorn (siehe Tabelle 4.5). Durch die präparierten Daten konnte die API sich sogar noch um 3 Prozentpunkte verbessern. Bei der WRR steigerte sich die GWS sogar auf 87 %. Dies liegt hauptsächlich daran, dass schnell hintereinander gesprochene Wörter nun als solche erkannt und nicht als unbekannt gelabelt wurden.

³⁰ <https://www.sonicvisualiser.org/> , Zugriff 03.08.2020

Danach folgt, wie auch schon bei der normalen Geschwindigkeit, das Kaldi 1000h-Modell. Hier konnte sich eine Verbesserung der WER von 4% beobachten lassen. Einzig die WRR verlor an Genauigkeit. Dies lag daran, dass das Modell einige Wörter aus dem Testkorpus nicht mehr als Wörter erkannt hat und sie somit nicht dem Decoder übergab. Das Kaldi 630h-ASR konnte sich nicht verbessern und verlor einen Prozentpunkt in der WRR im Vergleich zum normalen Testdatensatz. Als Gründe sind hier anzuführen, dass Wörter wieder unbekannt gelabelt wurden.

Mit kleinen Verbesserungen folgen DeepSpeech v0.6.0 und wave2letter den Testsiegern. Beide haben in der verlangsamten Version große Schwierigkeiten Zahlen und auch Zahlenwörter zu erkennen. Trotzdem konnte auch hier die Verringerung der WER durch eine genauere Trennung der Wörter erzeugt werden.

Immer noch weit abgeschlagen, aber mit der höchsten Verbesserung, steht DeepSpeech v0.5.0. Dieses Modell konnte, im Vergleich zum normal abgespielten Test, die WER um ganze 6 Prozentpunkte verringern.

Insgesamt wurde durch die Verringerung der Sprechgeschwindigkeit eine durchschnittliche WER von 38% und eine WRR von 68% erreicht. Dies ist eine Verbesserung von 2,5 %, bzw. 2,4 %. An der Reihenfolge der getesteten ASRs hat sich nichts geändert. Weit vorn liegt immer noch die GWS und die höchste Fehlerrate hat, wie im vorherigen Test, die Version 0.5.0 von DeepSpeech.

Name	normal		langsam	
	WER	WRR	WER	WRR
Kaldi 630h	34 %	72 %	35 %	65 %
Kaldi 1000h	32 %	75 %	28 %	72 %
DeepSpeech v0.5.0	76 %	20 %	70 %	30 %
DeepSpeech v0.6.0	42 %	57 %	40 %	60 %
GWS API	16 %	79 %	13 %	87 %
wav2letter	43 %	55 %	42 %	58 %
	40,5 %	59,6 %	38 %	62 %

Tabelle 4.5: Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz im Vergleich zur normalen Geschwindigkeit

Abbildung 4.3 zeigt den jeweiligen WER Verlauf der Modelle. Es ist hier klar zu sehen, dass die Google API ihren Ausreißer vom vorherigen Test bei "ste15" nicht wiederholt hat und sogar deutlich besser abschließen konnte. Zwar erreichte die API hier nur noch einmal eine Fehlerrate von 0% ("ste13"), doch blieb die WER bei acht von neun Testsamples unter 10 %. Alle Modelle konnten aber auch in diesem Test ihre Erkennungsrate im Test-File "chemnitz" nicht steigern. Die WER approximiert hier immer noch gegen 100 %.

Das Kaldi 630h Modell konnte im Sample "lola" nichts erkennen und hat da eine Fehlerrate von 100 %. Es nähert sich dann aber, wie im vorherigen Test, wieder dem Kaldi

1000h Modell an. Bemerkenswert ist, dass die beiden Kaldi-Modelle (weniger stark) sowie das wave2letter und die DeepSpeech v0.5.0 (stärker) bei "ste15" einen Anstieg der Fehlerrate zu verzeichnen haben. Gut zu sehen ist auch hier, dass die WER bei der Datei "slow" wieder absinkt und wie bei der obigen Evaluierung durchschnittlich den niedrigsten Wert annimmt.

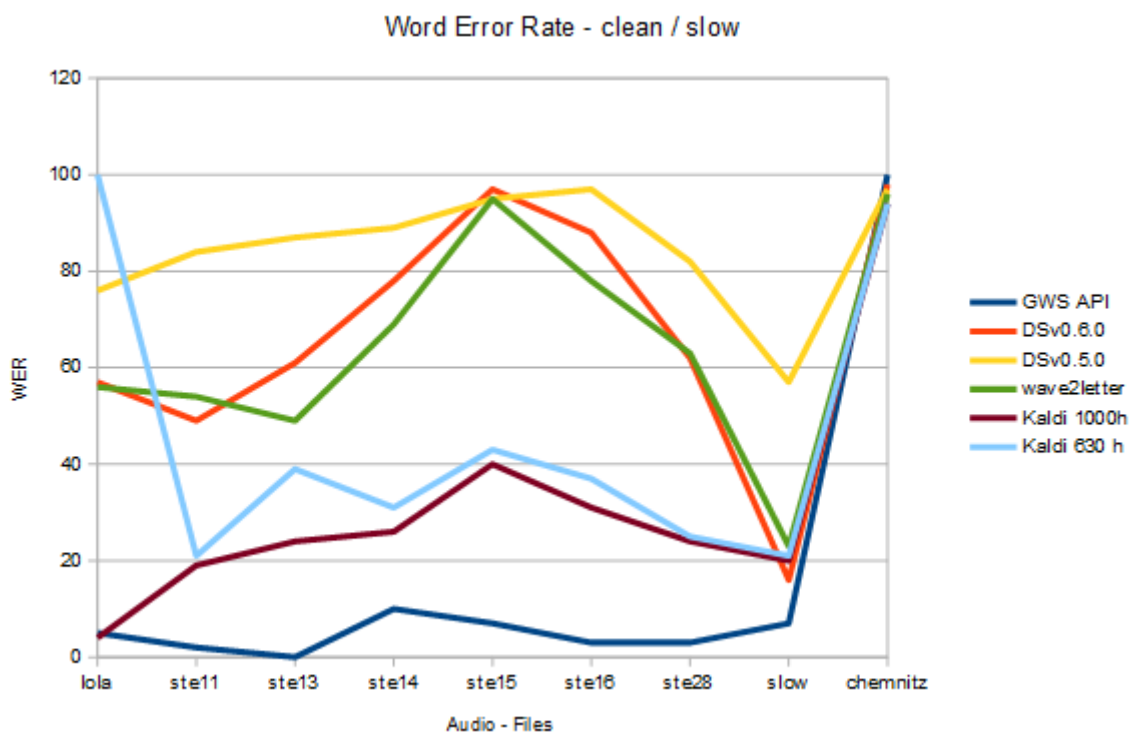


Abbildung 4.3: Evaluation der sechs verschiedenen ASR Modelle auf einen eigen erstellten Testdatensatz. Dabei wurden die Audio - Daten um 15% verlangsamt.

In Tabelle 4.6 lässt sich dies ablesen. "slow" ist mit einer durchschnittlichen WER von 24 % das Sprachsample, welches von den neun ASR Modellen am besten erkannt wurde. Mit der Verringerung der Geschwindigkeit konnte sogar die WER nochmals um 8 Prozentpunkte im Vergleich zur normalen Abspielgeschwindigkeit vermindert werden. Eine deutliche Verbesserung erlebte auch die File "ste11". Diese konnte die Fehlerrate auf 38,1 % um 5,1 Prozentpunkte verringern.

Eine stark erhöhte WER haben "lola" und "ste16" erzielt. Bei ersterer lag dies hauptsächlich daran, dass das Kaldi 630h Modell nichts erkennen konnte und somit eine WER von 100% erreichte. "ste16" war schon in ihrer normalen Sprechgeschwindigkeit langsam gesprochen und wurde durch die nochmalige Verlangsamung schwieriger für die ASRs zu erkennen.

Immer noch weit abgeschlagen liegt "chemnitz". Zwar konnte sich die Word Error Rate etwas verringern, aber es bleibt dabei, dass so gut wie jedes Wort nicht richtig erkannt wurde.

	normal	langsam
Name	WER	WER
lola	37,5%	49,6%
ste11	44,2%	38,1%
ste13	45,2%	43,3%
ste14	47,7%	50,1%
ste15	61,2%	62,8%
ste16	45,2%	55,6%
ste28	41,0%	43,2%
slow	32,0%	24,0%
chemnitz	96,7%	96,5%

Tabelle 4.6: Gemittelte Word Error Rate über die gesamten Modelle

Die Verbesserung der Fehlerrate in den jeweiligen Modellen nach der Verlangsamung der Testdaten lässt sich mit der besseren Differenzierung der Wörter sowie einer eindeutigeren Charakteristik der Frequenzen erklären. In Abbildung 4.4 sieht man im oberen Spektrogramm, dass die Trennung der Wörter (vertikale blaue Balken) eindeutig erfolgte. Dies ist nützlich, um schnell sprechende Audio-Files besser zu transkribieren.

Nachteil dieses einfachen Verfahrens ist die Abtrennung von Frequenzräumen. Man sieht im oberen Spektrogramm, dass ab einer Frequenz knapp unter 7 kHz ein Schnitt stattfindet. Im unteren Spektrogramm (normale Geschwindigkeit) wird der Frequenzbereich bis über 8 kHz genutzt. Dieser Schnitt beeinflusst das charakteristische Bild eines Wortes bzw. seiner Phoneme maßgeblich. Deswegen kann gesagt werden: Je besser ein ASR Modell damit umgehen kann, desto besser wird die WER bei einer Reduzierung der Sprechgeschwindigkeit. Dies ist aber nicht bis zu einer perfekten Erkennung möglich, da dies immer mit einer Beschneidung der Frequenzbereiche und so mit einer Unerkennbarkeit des Wortes bzw. der Phoneme einhergeht.

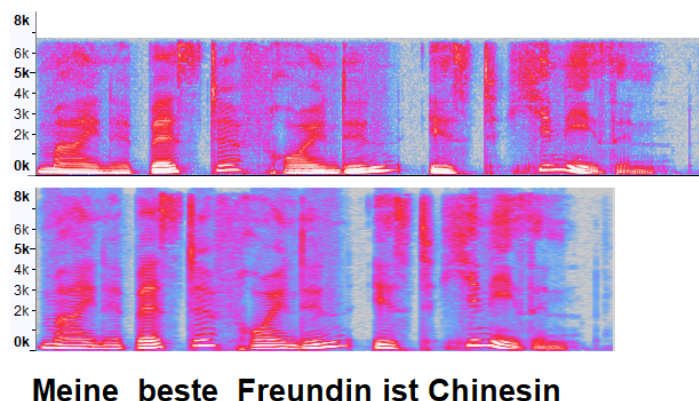


Abbildung 4.4: Dieser Vergleich der beiden Spektrogramme zeigt den Satz "Meine beste Freundin ist Chinesin." in langsamer Form (oben) und in normal gesprochener Form (unten) des Testdatensatzes "slow"

4.2.2 Testdurchlauf: Ohne Hintergrundrauschen - schnell

Auch der folgende Test liegt einer subjektiven Wahrnehmung zugrunde. Es wurde schrittweise die Geschwindigkeit der Audio - Daten erhöht. Ab einer Geschwindigkeit von plus 15 % ließ sich eine wahrnehmbare Veränderung erkennen, so dass die Daten beschleunigt und anschließend getestet wurden.

Tabelle 4.7 zeigt die Ergebnisse des Testes. Auch hier nimmt die Google Web Speech API einen Spitzenwert ein. Zwar liegt sie mit einem WER von 16 % wieder drei Prozentpunkte über dem Wert des langsamen Tests, konnte aber mit der Erkennungsrate gleichziehen. Unter erhöhter Geschwindigkeit erkannte dieses Modell acht Prozent mehr Wörter als mit normaler Geschwindigkeit.

Etwas schlechter hat das Kaldi 1000h-Modell abgeschnitten. Dieses konnte seine WER im Vergleich zur langsamen Testreihe um ein Prozent und zur normalen Testreihe sogar um 5 % steigern.

Es folgen weiter mit nahezu gleicher WER von 42 % bzw. 43 % das Kaldi 630h-Modell bzw. DeepSpeech v.0.6.0. Bei beiden verschlechterte sich die Erkennungsrate im Vergleich zu den ersten Tests. Kaldi 630-h konnte deutlich weniger Wörter richtig erkennen und auch deutlich weniger Wörter identifizieren. DeepSpeech v.0.6.0 hingegen blieb mit beiden Evaluationsmetriken relativ konstant.

Die größten Probleme beim Erkennen der schnellen Audio-Testdaten hatten die Modelle DeepSpeech v0.5.0 und wave2letter. Ist die WER bei ersterem Modell um sechs Prozent auf 82 % im Vergleich zum normalen Testsatz gestiegen, so erhöhte sich die Fehlerrate bei wave2letter signifikant auf 97 %. Dies ist eine Steigerung der Fehlerrate von bis zu 54 % im Vergleich zu den normal abgespielten Audio-Daten.

Im Schnitt wurde über den gesamten Testdatensatz nur jedes zweite Wort richtig erkannt (WER 51,2 %). Dies ist eine deutliche Verschlechterung im Vergleich zu den ersten beiden Tests. Nur das Kaldi 1000h-Modell konnte in der Evaluation mit dem schnelleren Datensatz seine WER senken.

Name	normal		langsam		schnell	
	WER	WRR	WER	WRR	WER	WRR
Kaldi 630h	34 %	72 %	35 %	65 %	42 %	45 %
Kaldi 1000h	32 %	75 %	28 %	72 %	27 %	73 %
DeepSpeech v0.5.0	76 %	20 %	70 %	30 %	82 %	19 %
DeepSpeech v0.6.0	42 %	57 %	40 %	60 %	43 %	61 %
GWS API	16 %	79 %	13 %	87 %	16 %	87 %
wav2letter	43 %	55 %	42 %	58 %	97 %	3%
	40,5 %	59,6 %	38 %	62 %	51,2 %	40,5 %

Tabelle 4.7: Gemittelte Ergebnisse ohne Hintergrundrauschen über den gesamten Testdatensatz mit erhöhter Geschwindigkeit im Vergleich zur normalen und zur verlangsamten Geschwindigkeit

In Abbildung 4.5 sind die jeweiligen WER der Modelle dargestellt. Das Sprachsample "chemnitz" ist auch in diesem Fall wieder das am schlechtesten erkannte File mit annähernd einhundert Prozent Fehlerrate. Klar zu erkennen ist der Vorsprung der GWS API, die bis auf einen Ausreißer bei "ste14" und "slow" immer unter zehn Prozent Fehlerrate liegt.

Die beiden Kaldi-Modelle befinden sich meist gleichauf, nur erkannte die 630 Stunden-Version bei "lola" nichts und erreichte da eine WER von 100 %. Diesen Wert verzeichneten bei der gleichen Testdatei auch die Modelle DeepSpeech v0.5.0 und wav2letter. Letzteres kam mit dem beschleunigten Datensatz am wenigsten zurecht und erzielte bei keiner Audio-File weniger als 95 %.

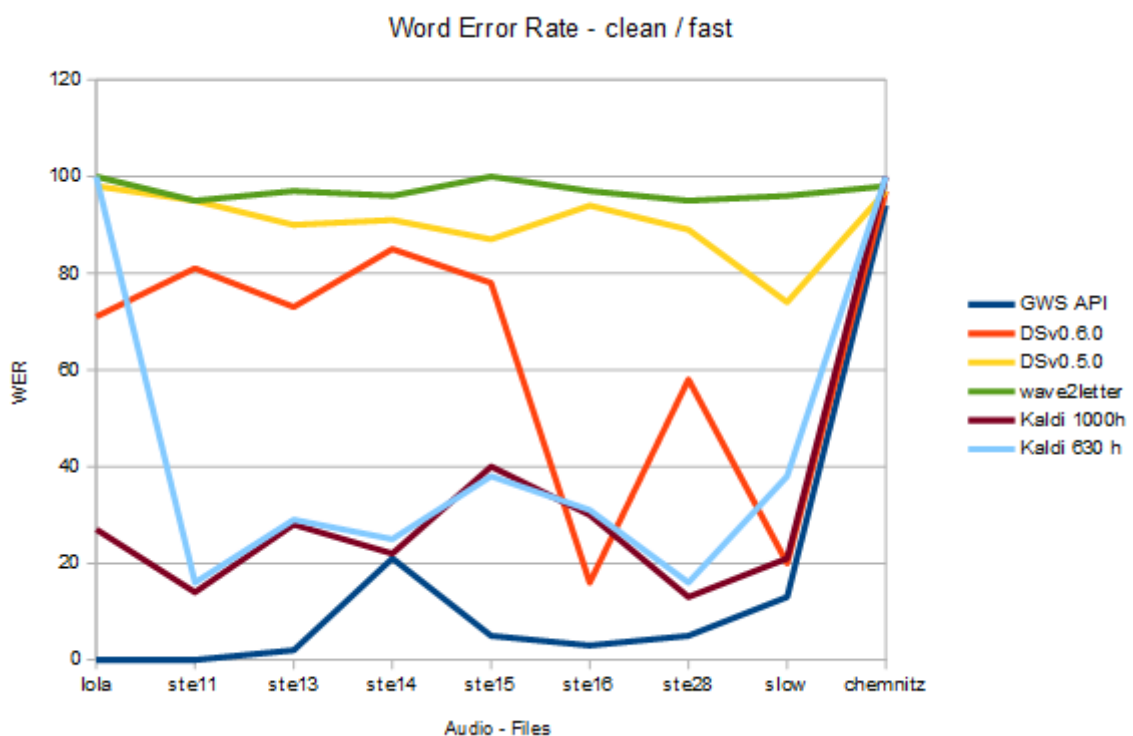


Abbildung 4.5: Evaluation der sechs verschiedenen ASR Modelle auf einen eigen erstellten Testdatensatz. Dabei wurden die Audio - Daten um 15% beschleunigt.

Tabelle 4.8 zeigt die Übersicht der Word Error Rate auf die gesamten Modelle aller Geschwindigkeiten. Zu erkennen ist hier die stark angestiegene WER. Das Testset "slow" wurde zwar immer noch am besten erkannt, aber die Fehlerrate hat sich im Vergleich zum normal abgespielten Sample um 11,2 % erhöht. Es liegt nun nur noch knapp vor "ste16". Dies hat die WER im Vergleich zum Standard beibehalten können. Sie konnte sich sogar um 10,4 % zum langsamen Test verbessern. Es folgt "ste28", deren Fehler-rate konstant zwischen 41,0 und 46,0 % liegt. Des Weiteren haben die Files "ste11" (+ 5,9 %), "ste13" (+8,0 %) und "ste14" (+9,0 %) deutlich schlechtere Werte als im normalen Testdatensatz.

Deutlich weniger Worte wurden bei "lola" richtig erkannt. Waren es noch 37,5 % Fehler-rate (normal), stieg diese um 28,5 Prozentpunkte auf 66,0 % an.

	normal	langsam	schnell
Name	WER	WER	WER
lola	37,5 %	49,6 %	66,0 %
ste11	44,2 %	38,1 %	50,1 %
ste13	45,2 %	43,3 %	53,2 %
ste14	47,7 %	50,1 %	56,7 %
ste15	61,2 %	62,8 %	58,0 %
ste16	45,2 %	55,6 %	45,2 %
ste28	41,0 %	43,2 %	46,0 %
slow	32,0 %	24,0 %	43,2 %
chemnitz	96,7 %	96,5 %	97,7 %

Tabelle 4.8: Gemittelte Word Error Rate über die gesamten Modelle und Geschwindigkeiten

4.2.3 Vergleich

Vergleicht man nun die jeweiligen Word Error Rates der Modelle mit den verschiedenen Geschwindigkeiten (vgl. Diagramm 4.6) sieht man, dass das Verlangsamen bei drei von sechs Modellen bessere Werte erzielt. Bei zwei Modellen ergibt es deutlich schlechtere Werte und bei einem Modell ein gleich gutes Ergebnis wie mit normaler Geschwindigkeit. Bei zwei Modellen bewirkt das Beschleunigen der Testdaten eine deutliche Erhöhung der WER.

Die einzelnen Modelle im Detail:

Die Google Web Speech API konnte seine Fehlerrate durch Verlangsamen aber auch durch Beschleunigen der Testsamples deutlich verringern. Gab es in normaler Geschwindigkeit noch eine WER von 21,4 %, erreichte das Modell bei der Reduzierung bzw. Erhöhung der Sprachschnelligkeit um 15 % eine Quote von 15,2 % bzw. 15,9 %. Durch die Reduzierung und Erhöhung der Geschwindigkeit konnte also eine Verbesserung der Erkennungsrate erreicht werden. Zwar sind bei dem langsamen Test weniger Files mit null Fehlern als bei dem normalen oder schnellen, doch bewegen sich die Fehlerquoten, außer bei "chemnitz", unter 10 %. Am besten kam die GWS mit dem Set "ste13" zurecht. Im Schnitt erreichte dieses Testsample eine Fehlerquote von 0,67 %. Es folgen "ste11" mit 1,33 % und "lola" mit 1,67 %. Alle drei Files sind langsam und ohne Dialekt gesprochen. Werden die Sprachaufnahmen nun mit Dialekt durchgesetzt, erhöht sich die WER. Bei "ste16" und "ste28" ist eine Steigerung auf 4,67 % bzw. 7,0 % erkennbar. Wenn nun zu dem Dialekt noch eine schnellere Sprache hinzukommt, erhöht sich die WER weiter. "ste14" (12,3 %) und "ste15" (22,3 %) geben einen guten Aufschluss darüber.

Die Ausnahme dieser Regel bildet "slow". Der mit Abstand umfangreichste Datensatz erreichte eine durchschnittliche Gesamtfehlerrate von 9,67 %.

Weit abgeschlagen ist "chemnitz". Mit einer Fehlerrate von 98 % ist dies für die GWS ein nicht zu erkennender Testsatz, bei dem Dialekt und Schnelligkeit eine Transkription verhindern.

Durch ihre sehr geringe Fehlerrate ist die Google Web Speech API das beste hier getestete ASR-Modell im Bereich der Sprache ohne Hintergrundgeräusche.

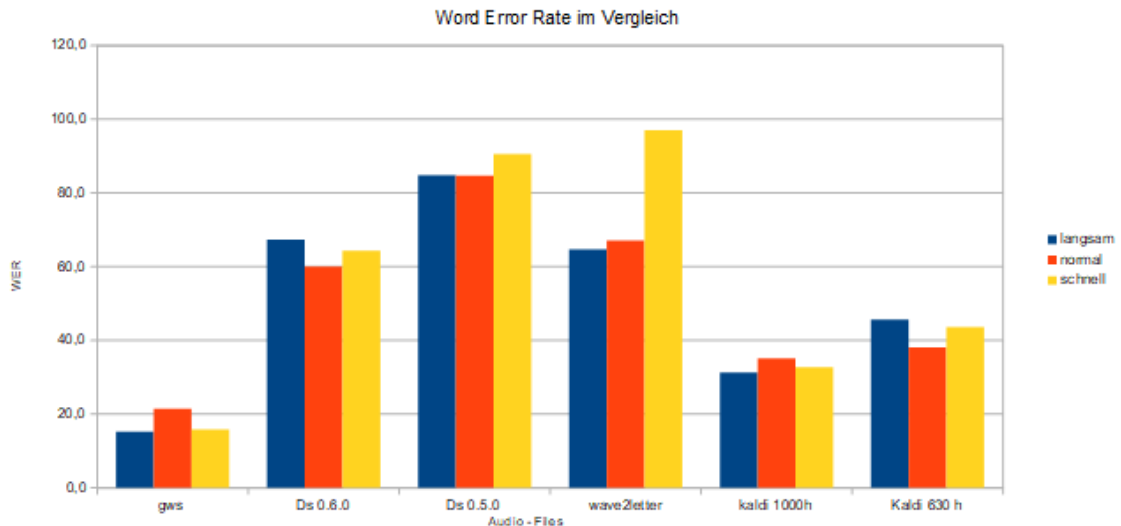


Abbildung 4.6: Vergleich der Word Error Rate bei verschiedenen Geschwindigkeiten

Das Kaldi 1000h Modell konnte ebenfalls durch Reduzierung, aber auch durch Erhöhung der Sprachgeschwindigkeit seine Fehlerrate verbessern. War diese im normalen Datensatz noch bei 35,1 %, konnte sie auf 31,3 % (langsam) bzw. 32,8 % (schnell) gesenkt werden. Auch hier erzielte man mit einer Änderung der Geschwindigkeit eine Verbesserung der WER um bis zu 4,2 %.

Mit einer Fehlerrate von 16,0 % kam dieses Modell gut mit dem Sample "lola" zurecht - am besten in der langsamen Variante (4 % Fehlerrate). In der schnellen Variante konnte Kaldi 1000h hier nur eine Quote von 27 % vorweisen. Es folgt "ste11" mit 18,67 % und "ste28" mit 20,33 %. Wenn man jetzt noch die Datei "slow" mit einem Fehlerwert von durchschnittlich 21,0 % und "ste13" (25,33 %) betrachtet, kann man annehmen, dass die Werte besser werden, je langsamer in der normalen File gesprochen wird. Der Dialekt scheint dem Modell nur dann Probleme zu bereiten, wenn dieser, wie bei "chemnitz", zu undeutlich ist oder bei den schneller werdenden, normalen Sprachsamples zu stark wird. Kaldi 1000h ist mit dem Modell von Google das einzige, welches seinen WER-Wert im beschleunigten Test gegenüber dem normalen verbessern konnte.

Gleich im Anschluss daran folgt das kleinere Kaldi 630h Modell. Dieses hat seine besten Fehlerquoten im normal abgespielten Testsatz. Waren es da noch durchschnittlich 38,1 %, verschlechterten sich die Werte auf 43,7 % (schnell) bzw. 45,7 % (langsam). Der hohe Anstieg der WER liegt hauptsächlich daran, dass in beiden bearbeiteten Tests das Sample "lola" nicht erkannt werden konnte und somit eine 100 %-ige Fehlerrate erzielt wurde. Auch dieses Modell von Kaldi konnte "ste11" (20,0 %) und "ste28" (21,67

%) gut erkennen. Beide Testdaten wurden am besten im schnellen Test erkannt. Ansonsten nähert sich dieses Modell dem 1000 Stunden Modell an, hat aber immer eine etwas höhere Fehlerrate.

Es folgt die Version 0.6.0 von DeepSpeech. Dieses Modell konnte seine Fehlerquote als einziges nicht in den bearbeiteten Testvarianten verbessern, sondern blieb mit 60 % am besten bei der normalen Geschwindigkeit. Deutlich schlechter wurde dieser Wert sogar bei dem Test mit dem langsamen Tempo. Er stieg um 7,3 %. Der Testsatz "slow" konnte von DeepSpeech am besten erkannt werden. Hier machte das Modell im Schnitt nur bei 21,67 % der Wörter Fehler. Mit großem Abstand folgten "ste16" (54,0 %) und "ste28" (57,0 %). Die weiteren Werte lagen zwischen 60 % und 96,67 %. Interessanterweise befand sich die Fehlerrate von "ste15" mit 86,7 % annähernd in der gleichen Höhe wie die Rate des Samples "chemnitz" (96,67 %).

Das gute Abschneiden des "slow"-Datensatzes und die danach folgende starke Verschlechterung der Fehlerquote lässt darauf schließen, dass DeepSpeech v0.6.0 Probleme bekommt, wenn es um umgangssprachliche Äußerungen geht.

Das wav2letter - Modell erreichte eine durchschnittliche Fehlerrate von 64,8 % und erkannte somit fast drei von fünf Wörtern nicht korrekt. Auch hier zeigt sich, dass eine Verminderung der Geschwindigkeit die Spracherkennung begünstigt. Erzielte wav2letter im normalen Tempo eine WER von 67,1 %, konnte dies bei einer Verlangsamung um 2,3 % reduziert werden. Mit 52 % erkannte das Modell die Datei "slow" mit Abstand am besten, gefolgt von "ste13" (68,67 %) und "ste11" (71,33 %).

Beim Test mit den beschleunigten Daten brach die Erkennung der Texte komplett ein und pendelte sich zwischen 95 % bis 100 % Fehlerrate ein. Am besten konnte dieser Spracherkennung also noch mit Sprachdaten umgehen, die langsam und ohne Dialekt aufgenommen wurden.

Am unterem Ende der Liste befindet sich das DeepSpeech v0.5.0 Modell. Es erreichte im normalen Datensatz seinen besten Wert von 84,8 %. Nur knapp darüber liegt der langsam abgespielte Datensatz mit 84,9 %. Konnte "slow" noch mit einer Fehlerquote von 68,0 % erkannt werden, liegen die weiteren Daten über einer WER von 80 %.

4.2.4 Konklusion über die Datensätze ohne Hintergrundrauschen

Durch die beobachteten Messergebnisse kann geschlussfolgert werden, dass die Word Error Rate durch Veränderung der Abspielgeschwindigkeit verbessert werden kann. Dies ist abhängig vom verwendeten Modell und den Rohdaten.

Die Google Web Speech API ist in der Lage, diese Daten am besten zu verarbeiten, wenn sie in normaler oder schnellerer Sprechgeschwindigkeit aufgenommen und dann in einem Vorverarbeitungsschritt verlangsamt werden. Dabei ist diese API robust im Bezug auf im Dialekt gesprochene Daten. Ausgenommen davon ist starker Dialekt wie in "chemnitz".

Anders sieht dies beim Kaldi 1000h-Modell aus. Bei diesem kann beobachtet werden, dass langsam eingesprochene Daten bei einer Verlangsamung und schnell eingesprochene Daten bei einer Beschleunigung besser erkannt werden. Der Dialekt schlägt zwar hier stärker auf die WER nieder als bei der Google API, doch auch hier ist eine Vorverarbeitung der Daten nützlich, um die Fehlerrate niedriger zu halten.

Einzig das ASR-Modell wave2letter konnte nur mit einer Verlangsamung bessere Ergebnisse erreichen. Wurden die Sprachsamples aber schneller abgespielt, kam es hier zu einer signifikanten Verschlechterung der Fehlerrate.

Zusammenfassend wurde bei der Bearbeitung der Abspielgeschwindigkeiten bei drei von sechs Modellen eine Verbesserung der WER erzielt. Im Schnitt liegt diese Verbesserung vor, wenn man die Sprachdaten verlangsamt. Diese Erkenntnis soll im Folgenden auf Sprachsamples mit Hintergrundrauschen angewandt werden.

4.3 Mit Hintergrundrauschen

Um die Spracherkenner polizeilich zu nutzen, ist es wichtig zu testen, wie sie sich verhalten, wenn in den Sprachdaten Hintergrundgeräusche zu hören sind. Damit werden die Modelle vor eine schwierige Aufgabe gestellt. Sie müssen erkennen, welche Laute zu den Nutz- oder welche zu den Stördaten gehören und die wichtigen Nutzdaten extrahieren (siehe Abschnitt Lösungen für die Forensik 3.4.4)

Um den ASRs diesen Vorgang zu erleichtern, kann eine Vorverarbeitung stattfinden, mit deren Hilfe das Hintergrundrauschen herausgefiltert werden soll. In dieser Evaluierung wurde dazu das freie Programm AUDACITY genutzt. Der Einfachheit halber kann hier ein Bereich, indem das Störgeräusch eindeutig identifiziert wurde, manuell markiert und als Filter definiert werden. Der AUDACITY-Algorithmus wendet dann automatisch den Filter auf das gesamte File an und kann somit die Hintergrundgeräusche minimieren.

4.3.1 Überblick

Zu den Evaluationsdaten gehören drei Sprachsamples, die einen Dialog in einer Bar und in einem Auto mit eingeschaltetem Radio sowie ein Monolog in einer belebten Fußgängerpassage umfassen. Tabelle 4.9 gibt eine Übersicht über die verwendeten Samples. Es wurden hier bewusst zwei längere Dialoge gewählt, um realistische Testbedingungen zu schaffen. Die erste Datei "autoDialog" kann ein Mitschnitt einer Abhöreinrichtung sein, die in ein Fahrzeug installiert wurde. Auf ihr sind unter anderem laute Fahrgeräusche wie das Abrollen der Reifen, Blinken und Windgeräusche zu hören. Ungefähr in der Mitte des Sets wird das Autoradio eingeschaltet. Es läuft eine Nachrichtensendung, die dem ASR weitere Wortfragmente zum Bearbeiten sendet.

Die zweite Datei "barDialog" stellt das Szenario einer installierten Abhöreinrichtung in der Nähe eines Tisches dar. An diesem Tisch sitzen zwei Frauen und unterhalten sich. Im Hintergrund sind typische Geräusche aus einer Bar wie Gläserklirren, Unterhaltungen und Stühlerücken zu hören. Die Störgeräusche in diesem Testset sind leiser als im ersten Set.

Das letzte Testset "passMono" passt in das Szenario eines abgehörten Handygespräches, während die Zielperson durch eine belebte Fussgänger-Passage geht. Die Gegenstimme im Handy ist nicht zu hören. Als Störgeräusche sind lautes Kindergeschrei und vorbeischiebende Personen sowie Atemgeräusche der Sprecherin zu hören.

Name	Länge	Personen	Besonderheit
autoDialog	1:59 min	Mann und Frau	starke Fahr- und Radiogeräusche, Radiosendung
barDialog	2:40 min	zwei Frauen	leichte Bargeräusche, beide sitzen etwas abseits
passMono	0:26 min	eine Frau	spaziert durch Passage und redet mit Handy

Tabelle 4.9: Überblick über die Test - Sprachsamples mit Hintergrundrauschen

Vergleicht man den Schalldruck der Daten mit Hintergrundrauschen mit dem der Daten ohne (vgl. Abbildung 4.7), kann man mehrere Punkte feststellen. Zum einen ist an den Beispielen links zu sehen, dass es bei den Audio-Dateien keinen Moment gibt, in dem es zu einem Nullpunkt des Schalldruckes kommt. Erkennbar wird dies durch den permanenten Ausschlag der blauen Linien (Schalldruck). Im Vergleich dazu kommt es, rechts im Bild zu sehen, nach jedem Anstieg des Schalldrucks zu einem kurzen Moment, in dem es keinen Schalldruck gibt. Damit kann aber das ASR System, wie oben schon erwähnt, die Worttrennung durchführen.

Ein weiterer Punkt, der beim Betrachten ins Auge fällt, ist die hohe Volatilität der Amplituden. Im rechten Beispiel ist der jeweilige Schalldruck in seinen Ausschlägen relativ homogen. In den linken Beispielen hingegen gibt es keinen gleichmäßigen Druck. Das liegt daran, dass sich der Schalldruck, wie auch Frequenzen, addiert und mit größerer

Stärke gemessen werden kann.

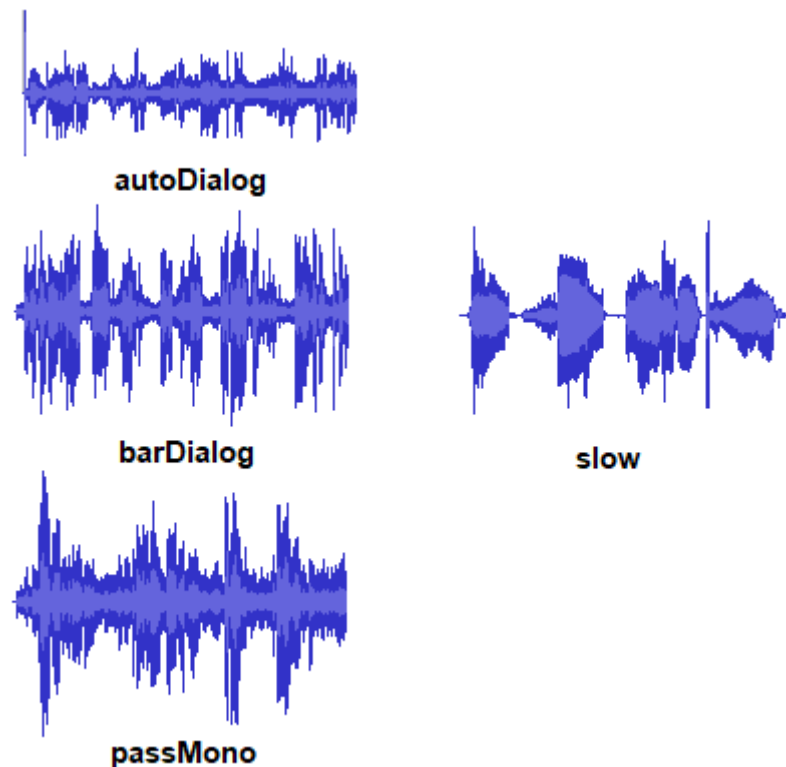


Abbildung 4.7: Schalldruck der drei Testdaten im Vergleich. Erkennbar ist der permanente Ausschlag der Kurven (links). Dies ist charakteristisch für Audio-Files mit Nebengeräuschen. Rechts der Vergleich mit der Datei "slow", welche ohne Hintergrundrauschen aufgenommen wurde. Hier ist gut erkennbar die Sequenzen ohne Schalldruck (= Wortgrenzen).

Auch im Spektrogramm (vgl. Abbildung 4.8) ist eindeutig zu erkennen, dass sich das Hintergrundsignal mit dem Nutzsignal verknüpft hat (keine Worttrennung mehr ersichtlich). Zusätzlich ist sichtbar, dass starke Signale zwischen 0 - 1kHz liegen, die eine Identifizierung der Wörter bzw. Phoneme schwieriger machen.

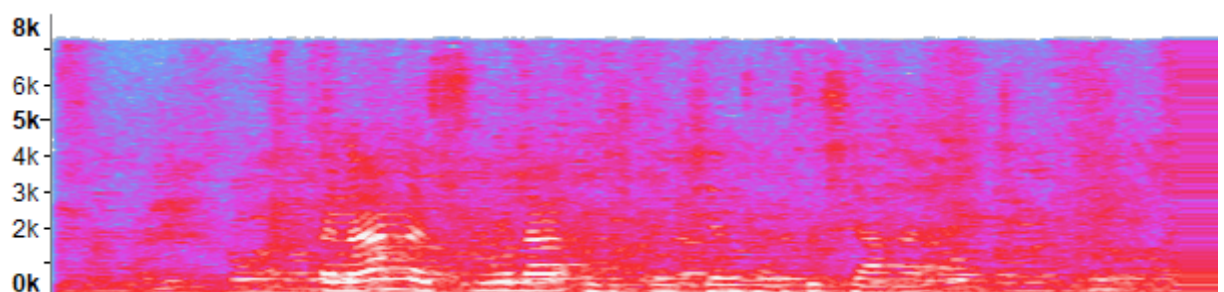


Abbildung 4.8: Spektrogramm des Sprachsamples "passMono" dient zur Verdeutlichung des ständigen Rauschens im Hintergrund des Samples. Man sieht ein permanentes und starkes Frequenzspektrum um die 1kHz.

4.3.2 Test und Evaluation

Die drei Testsets werden mit den ASR-Modellen: Google Web Speech, Kaldi 1000h und Kaldi 630h transkribiert. Diese Modelle wurden gewählt, weil sie im vorherigen Test eine durchschnittliche Word Error Rate von weniger als 50 % erreicht haben. Bei den anderen Spracherkennern scheint es eher unwahrscheinlich, dass diese unter realen Bedingungen besser abschneiden als bei Sprachsamples ohne Hintergrundrauschen. Die Transkription der Testdaten wird dann mit den von den ASRs transkribierten verglichen, um somit eine WER und WRR zu ermitteln.

Mit Hilfe von AUDACITY wurde im nächsten Schritt versucht, das Hintergrundgeräusch herauszufiltern. Die bearbeiteten Sets werden nochmals durch die ASR-Modelle transkribiert und dann evaluiert. Zum Schluss erfolgt ein Vergleich der beiden Teilschritte miteinander. Somit kann evaluiert werden, ob die ASRs eine bessere oder schlechtere WER bzw. WRR erreichen, wenn sie vorher bearbeitet wurden.

Zum Schluss werden die beiden Evaluationsschritte (langsam/schnell bzw. mit oder ohne Filterung) zusammengeführt.

4.3.3 Ungefiltert und gefiltert

Die Google Web Speech API (Tabelle 4.10) konnte in diesem Test ihre niedrige WER nicht halten, jedoch sich (zumindest im ungefilterten Datensatz) mit einer WER von 53,6 % deutlich vor den anderen beiden ASRs platzieren. Am besten ging die GWS mit den originalen Daten um. Eine signifikante Verschlechterung der Werte trat ein, als mit den gefilterten Daten getestet wurde. Im File "barDialog" kam es sogar soweit, dass die GWS hier gar nichts mehr erkennen konnte. Bei "passMono" erhöhte sich die WER um 37 %. Große Probleme gab es bei dem Set "autoDialog". Wurde das Gespräch am Anfang noch relativ gut erkannt, kam es zu sehr großen Diskrepanzen, nachdem weitere Stimmen im Radio dem Signal hinzugefügt wurden.

Ein weiterer Nachteil der GWS war die Dateilänge. Die beiden größeren Dateien mussten auf eine Länge von 1 min geschnitten werden, damit das Modell überhaupt damit umgehen konnte.

Name	original		gefiltert	
	WER	WRR	WER	WRR
autoDialog	91 %	8 %	98 %	2 %
barDialog	53 %	46 %	100 %	0 %
passMono	17 %	83 %	54 %	47 %
	53,6 %	45,6 %	84 %	16,3 %

Tabelle 4.10: Ergebnisse der Sprachsamples mit Hintergrundrauschen (Google Web Speech)

Die Kaldi 1000h-Version verringerte mit Hilfe der Filterung seine Fehlerrate (Tabelle 4.11). War es im originalen Datensatz noch eine WER von 82,3 %, ging diese im gefilterten um 6,3 Prozentpunkte nach unten. Verbesserungen konnten in "barDialog" (16 % geringere WER) und "bassMono" (3 % geringere WER) erzielt werden. Unverändert blieb hingegen das Set "autoDialog". Im Gegensatz zu der GWS API mussten aber bei diesem und dem kleineren Kaldi 630h-Modell die Audio-Daten nicht geschnitten werden.

Name	original		gefiltert	
	WER	WRR	WER	WRR
autoDialog	95 %	5 %	95 %	5 %
barDialog	82 %	18 %	66 %	34 %
passMono	70 %	30 %	67 %	33 %
	82,3 %	17,67 %	76 %	24 %

Tabelle 4.11: Ergebnisse der Sprachsamples mit Hintergrundrauschen (Kaldi 1000h)

Mit einer WER von 84 % im originalen und 77,67 % im gefilterten Datensatz konnte das Kaldi 630h-Modell die WER stark verringern (siehe Tabelle 4.12). Den größten Sprung konnte Kaldi 630h hier bei der File "barDialog" machen. Von einer WER von 82 % (original) sank die Fehlerrate auf 67 %. Aber auch bei "passMono" wurde die WER um vier Prozentpunkte gesenkt. Doch wie bei der GWS API gab es große Probleme bei der File "autoDialog". Diese konnte so gut wie gar nicht erkannt werden (97 % WER).

Name	original		gefiltert	
	WER	WRR	WER	WRR
autoDialog	97 %	3 %	97 %	3 %
barDialog	82 %	18 %	67 %	33 %
passMono	73 %	27 %	69 %	30 %
	84 %	16 %	77,67 %	22 %

Tabelle 4.12: Ergebnisse der Sprachsamples mit Hintergrundrauschen (Kaldi 630h)

Die originalen Daten erkannte die Google Web Speech am besten. Mit einer durchschnittlichen WER von 66 % konnte der beste hier getestete Spracherkennung nur ein Drittel der gegebenen Wörter richtig erkennen. Auf den nächsten Plätzen folgen das Kaldi 1000h und knapp dahinter das Kaldi 630h-Modell.

Gut ermitteln konnten die Spracherkennung das Set "passMono". Am schwierigsten hatten es die ASRs mit dem Set "autoDialog". Hier konnte keins der getesteten Modelle überzeugen. Sie erreichten eine WER von 91 % - 97 %.

Wurden die Sets vorher durch eine Geräuschreduzierung bearbeitet, stellte sich bei den Kaldi-Systemen Verbesserungen ein. Die GWS API konnte die File "barDialog" nicht transkribieren und erreichte eine WER von 100 %.

4.3.4 Fazit

Insgesamt konnte keines der ASR-Systeme überzeugen. Das beste hier getestete System konnte nur bei einem Set eine WER von 17 % erreichen. Alle weiteren Sets und Modelle kamen nicht unter eine Fehlerrate von 53 %.

Eine Zusammenführung der beiden Evaluationsschritte soll zeigen, ob sich die Word Error Rate verbessern lässt, wenn man die besten Verfahren der jeweiligen Modelle verbindet.

4.4 Zusammenführen der zwei Evaluationsschritte

Es soll nun getestet werden, ob die beiden jeweils besten Evaluationsschritte zusammengeführt werden können, um eine verbesserte WER zu erreichen.

Das heißt, es werden aus den oben genutzten Modellen (Google Web Speech API, Kaldi 1000h und Kaldi 630h) die Testschritte gewählt, die die beste WER erreicht haben und diese dann verbunden.

Die GWS API kam im ersten Evaluationsschritt besser mit Sets zurecht, die künstlich verlangsamt wurden, im zweiten Schritt mit Sets, bei denen das Hintergrundrauschen nicht vermindert wurde. Das Kaldi 1000h Modell hingegen konnte seine besten Werte mit dem schnelleren und gefilterten Set erreichen, Kaldi 630h indes mit den gefilterten, in normaler Geschwindigkeit abgespielten Daten. Letzteres wurde in Tabelle 4.12 schon evaluiert.

Die verbundenen Evaluationsschritte wurden nochmals auf ihre WER geprüft und dann verglichen.

Zuerst erfolge eine Verbindung der beiden Evaluationen der Google Web Speech API. Die GWS konnte in den beiden Schritten mit verlangsamt und ungefilterten Daten ihre besten Werte erzielen. Es wurden demzufolge die Samples mit dem Hintergrundrauschen um 15 % verlangsamt und der GWS API übergeben. Tabelle 4.13 gibt die Testergebnisse wieder.

Die niedrigste Fehlerrate konnte bei "passMono" (26 %) erreicht werden, das schlechteste Ergebnis bei "autoDialog" (35 %). Mit einer durchschnittlichen WER von 30,6 % wurden hier zwei Drittel der Wörter richtig erkannt. Ein weiterer Vorteil der Verlangsamung war, dass die Testsets nicht mehr geteilt werden mussten.

	verlangsamt & ungefiltert	
Name	WER	WRR
autoDialog	35 %	65 %
barDialog	31 %	69 %
passMono	26 %	74 %
	30,6 %	69,34 %

Tabelle 4.13: Testergebnisse der Zusammenführung der besten beiden Evaluationsschritte (Google Web Speech API)

Tabelle 4.14 gibt die Ergebnisse mit dem Kaldi 1000h Modell wieder. Da bei diesem Modell im ersten Evaluationsschritt die geringste Fehlerrate erreicht wurde, nachdem man die Audio-Dateien um 15 % beschleunigte, wurde dies auch hier getan. Danach erfolgte die Verringerung der Hintergrundgeräusche.

Mit einer WER von 85 % konnte hier das Sample "autoDialog" am besten und "barDialog" mit 100 % am schlechtesten erkannt werden. Das Kaldi 1000h Modell erreichte bei der Zusammenführung der beiden Evaluationsschritte nur eine durchschnittliche WER von 92,67 %.

	beschleunigt & gefiltert	
Name	WER	WRR
autoDialog	85 %	15 %
barDialog	100 %	0 %
passMono	93 %	7 %
	92,67 %	7,34 %

Tabelle 4.14: Testergebnisse der Zusammenführung der besten beiden Evaluationsschritte (Kaldi 1000h Modell)

Das Kaldi 630h Modell hatte im ersten Evaluationsschritt die beste WER in der normalen Geschwindigkeit und im zweiten Schritt mit den gefilterten Daten. Eine Zusammenführung war daher nicht notwendig.

4.4.1 Fazit

Den Vergleich der jeweiligen Fehlerraten kann man in Diagramm 4.9 ablesen. Es zeigt ein sehr heterogenes Bild. Die Google Web Speech API konnte mit den originalen Daten noch eine WER knapp über 50 % erreichen. Wurden nun die Daten vorverarbeitet und das Hintergrundrauschen reduziert, sprang die WER auf über 80 % nach oben. Eine einfache Unterdrückung der Störgeräusche führte hier also nicht zum Erfolg. Anders sah es bei der Zusammenführung der zwei besten Evaluationsschritte aus. Verband man die verlangsamte und nicht gefilterte Version, kam es zum Absinken der WER auf 30 %.

Das Kaldi 1000h-Modell hingegen erzielte mit der Zusammenführung der besten Schritte eine Verschlechterung der WER. Dieses ASR konnte besser mit einer Verminderung der Störgeräusche umgehen und erreichte hier seine günstigste Fehlerrate.

Ebenso verhielt sich das Kaldi 630h-Modell - hier konnte mit einer Reduzierung des Hintergrundsignals eine Verbesserung erzeugt werden. Eine Zusammenführung wurde nicht getestet, da hier schon die besten Schritte eingeflossen waren.

Die Google Web Speech API konnte auch im Test mit Hintergrundrauschen die Konkurrenz von Kaldi hinter sich lassen und den Spitzenplatz einnehmen. Mit einer Fehlerrate von 30 % liegt sie über 40 Prozentpunkte niedriger als die Spracherkennung von Kaldi.

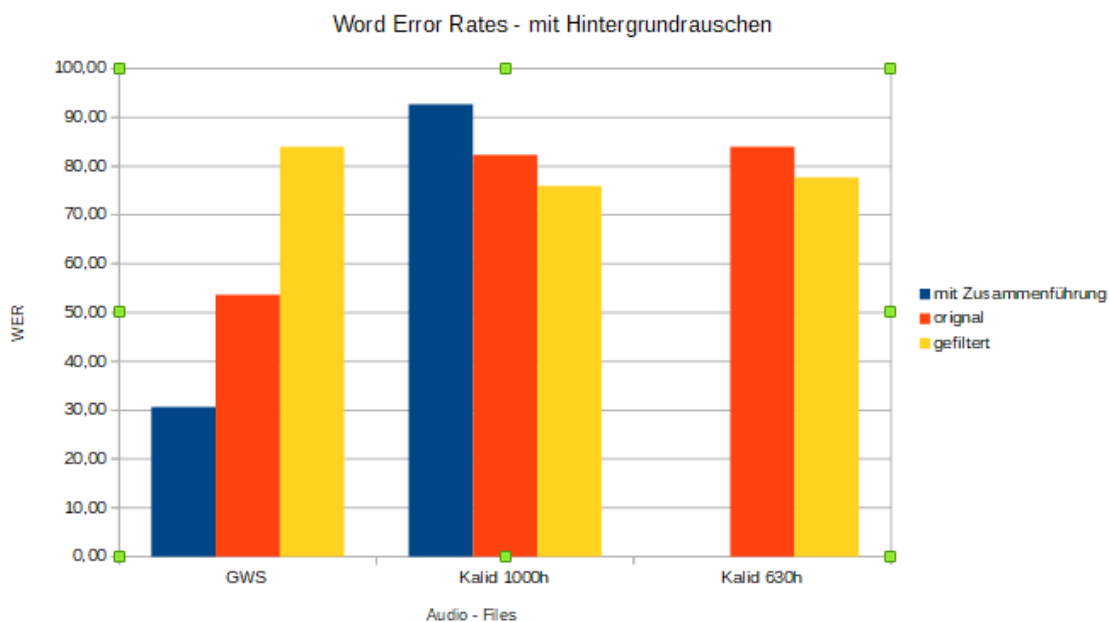


Abbildung 4.9: Das Diagramm zeigt die gemittelten Word Error Rates über die drei Testsets der jeweiligen Modelle. Es beinhaltet die Werte der originalen, der gefilterten sowie der zusammengeführten Sets.

5 Diskussion

In dieser Arbeit wurden sechs verschiedene, frei verfügbare Spracherkennungsmodelle evaluiert. Diese Modelle waren in ihren Ansätzen und Strukturen divergent gewählt, um ein größtmögliches Spektrum der Systeme zu erhalten (vgl. Kapitel 3.5). Allen sechs ASRs wurden neun Testdaten übergeben. Diese Daten kamen mit der WER-Metrik zur Evaluierung. Danach erfolgte eine Beschleunigung und eine Verlangsamung der Audio-Files. Anschließend fand eine nochmalige Messung statt.

Die besten drei Modelle wurden für einen weiteren Test ausgewählt (Google Web Speech API, Kaldi 1000h-Modell und Kaldi 630h-Modell), ihnen drei weitere Samples mit Hintergrundrauschen übergeben und evaluiert. Mit Hilfe des frei verfügbaren Multimedia-programms AUDACITY wurde eine Verringerung des Störsignals in einem Vorverarbeitungsschritt erreicht. Nach dem Messen der WER kam es zu einem Vergleich mit den originalen Daten. Die Tests mit den besten Ergebnissen wurden zusammengeführt. Mit den erhaltenen Werten konnte dann der robusteste Spracherkennungsermittler ermittelt werden. Wenn man die ASRs vom Namen her betrachtet, liegt nahe, dass die Google Web Speech API und die DeepSpeech weit vorn liegen müssten. Ersteres wird eben von Google betrieben und letzteres Modell ist die Grundlage der Mozilla Common Voice. Doch die in dieser Arbeit durchgeführten Tests konnten dies nur teilweise bestätigen.

Die Google Web Speech API bewies hier gute Robustheit und setzte sich in beiden Test weit vor die anderen ASR-Modelle. Die durchschnittliche WER von gerade einmal 16 % im Datensatz ohne Hintergrundrauschen wurde sogar noch mit einer Verlangsamung der Daten auf 13 % verringert.

Wurden der GWS Datensätze mit Hintergrundrauschen übergeben, stieg die Fehlerrate stark auf 53,6 %. Ein einfaches Verringern der Störsignale, um das Erkennen der wichtigen Signale zu begünstigen, schlug bei der ASR fehl. Aufgrund dieser Vorverarbeitung stieg hier die WER auf 84 %. Es lässt sich also feststellen, dass die Google Web Speech API schlecht mit "beschnittenen" Frequenzspektren umgehen kann.

Bei einer Verlangsamung der originalen Daten (mit Hintergrundrauschen) konnte die WER dennoch signifikant verringert werden und betrug noch durchschnittlich 30,6 %. Somit blieb die GWS API weit unter den anderen Modellen. Auch die einfache Implementierung, z.B. über einen Python-Script, ist ein Vorteil dieses Modells. Allein die unterschiedliche Länge der Files, die das Modell verarbeitet, kann als Nachteil angesehen werden. Als einzige ASR mussten hier öfter die Testdaten geteilt und danach der GWS übergeben werden.

Die beiden Kaldi-Modelle kommen auf einen guten zweiten und dritten Platz. Durch eine Vergrößerung des Trainingsdatensatzes um 270 Stunden und eine Verbesserung des LM-Modells konnte die Kaldi 1000h Version sich gegen die "kleinere" Version durchsetzen. Im Test ohne Störsignale hielten beide Spracherkennungsermittler ihre WER unter 40 %.

Durch die ständige Weiterentwicklung und Vergrößerung der Trainingsdaten kann davon ausgegangen werden, dass die Project Kaldi Entwickler irgendwann ein noch besseres Modell zur Verfügung stellen werden. Bei den Testdaten mit Hintergrundrauschen stieg die Fehlerrate aber signifikant auf 76 % (Kaldi 1000h) bzw. 77,67 % (Kaldi 630h). Damit sind die beiden ASR-Modelle zur Transkription von Audio-Dateien mit Störgeräuschen nicht praktikabel.

Enttäuschend schlossen in dieser Arbeit die beiden DeepSpeech ASRs ab. Keines der zwei Modelle konnte im Test ohne Hintergrundrauschen überzeugen. Sie erreichten eine Fehlerrate von 61 % (v0.6.0) bzw. 84,8 % (v0.5.0). Aufgrund dieser Werte wurden die beiden Modelle nicht mehr für eine Betrachtung mit Hintergrundrauschen gewählt. Interessanterweise steht DeepSpeech der am schnellsten wachsende Sprachkorpus (Common Voice) zur Verfügung. Es bleibt zu klären, ob die verwendete Modell-Struktur den Anforderungen noch gerecht wird.

Ein großes Problem der ASRs ist aber das Erkennen von einzelnen Sprechern. Ist es für einen Menschen noch relativ trivial, in einer gehörten Audio-Datei verschiedene Sprecher zu identifizieren, kann dies ein Spracherkennungsmodell nicht. Es verarbeitet Frequenzen bzw. Phoneme und transkribiert diese. Einem menschlichem Betrachter muss demzufolge erst die Semantik des erzeugten Textes bewusst werden, um Schlüsse daraus ziehen zu können.

Fragwürdig sind die auf den jeweiligen Github-Seiten evaluierten Ergebnisse der Modelle. Diese stehen in keinem Verhältnis zu den Tests hier. Eine Antwort darauf können die in dieser Arbeit verwendeten Bedingungen sein. Wurden die Modelle von Entwicklerseite in "Laborbedingungen" getestet, erfolgte die Evaluation im Rahmen dieser Bachelorarbeit unter realitätsnahen Bedingungen. Darunter fallen mehr oder weniger schwierige Dialekte und Hintergrundrauschen.

6 Schlussfolgerung

Wie gezeigt werden konnte, heben sich die getesteten, freien Spracherkennungsmodelle stark voneinander ab. Es wurden in dieser Arbeit nicht nur die verschiedenen Modelle und ihre Funktionsweise beleuchtet, sondern diese auch gegeneinander evaluiert. Mit Vorverarbeitungsschritten, wie das Manipulieren der Abspielgeschwindigkeit sowie das Reduzieren von Störsignalen, konnte gezeigt werden, dass es möglich ist, die Fehlerrate weiter zu verringern. Aufgrund dieser Erkenntnis ist es wichtig die dem ASRs verschiedene Versionen der Audio-Dateien zu übergeben um eine bestmögliche Fehlerrate zu erreichen. Zudem ist es sinnvoll weiter an der Manipulation der Geschwindigkeit der Daten zu arbeiten um ein optimales Ergebnis zu erhalten.

In Abschnitt 4.3 wurde bewiesen, dass eine Zusammenführung der obigen Vorverarbeitungsschritte zu einer weiteren Verbesserung der Erkennungsrate führt. Dies ist aber nicht so bei jedem der Modelle. Nur die Google Web Speech API konnte damit die Testdaten mit Hintergrundrauschen deutlich besser erkennen. Da die GWS signifikant geringere Fehlerraten ablieferte als alle anderen ASR-Modelle ist es sinnvoll sie als Unterstützung einer Transkription zu nutzen.

Eine weitere Forschung im Bereich der Semantik und des Text Retrieval ist existentiell notwendig, um die Word Error Rate weiter zu verringern. Mit Hilfe dieser Themengebiete können nicht erkannte Wörter aus den vorherigen erschlossen werden. Auch können weitere Vorverarbeitungsschritte, beispielsweise klareres Herausheben der Stimmen, dazu beitragen, die Fehlerrate noch weiter zu minimieren.

Bezugnehmend auf die Forschungsfrage lässt sich nach Beschäftigung mit den sechs hier behandelten Spracherkennern und der Vorverarbeitung der Testdatensätze sagen, dass nach gegenwärtigen Stand der Entwicklung nur eine der ASRs für die Verwendung im forensisch / polizeilichen Bereich zur Transkription von Audio-Daten infrage kommen kann. Dies ist die Google Web Speech API. Durch ihre einfache Implementierung und ihre Robustheit ist sie das beste hier getestete Spracherkennungsmodell.

Literaturverzeichnis

- [1] Alfons Gottwald. *Automatische Spracherkennung: Methoden der Klassifikation und Merkmalsextraktion*. De Gruyter, 2018.
- [2] Shahenda Sarhan. Smart voice search engine. *International Journal of Computer Applications*, 90:40–44, 03 2014.
- [3] Stephan Radeck-Arnetz, Benjamin Milde, Arvid Lange, Evandro Gouvêa, Stefan Radomski, Max Mühlhäuser, and Chris Biemann. Open source german distant speech recognition: Corpus and acoustic model. volume 9302, pages 480–488, 09 2015.
- [4] Arne Köhn, Florian Stegen, and Timo Baumann. Mining the spoken wikipedia for speech data and beyond. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).
- [5] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.
- [6] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus, 2019.
- [7] T. Rashid and F. Langenau. *Neuronale Netze selbst programmieren: Ein verständlicher Einstieg mit Python*. Animals. O'Reilly, 2017.
- [8] David Kriesel. *Ein kleiner Überblick über Neuronale Netze*. 2007.
- [9] Iain Mccowan, Darren Moore, John Dines, Daniel Gatica-Perez, Mike Flynn, Pierre Wellner, and Herve Bourlard. On the use of information retrieval measures for speech recognition evaluation. 01 2004.
- [10] Mertins A. *Kurzzeit-Fourier-Transformation*. In: *Signaltheorie*. Vieweg+Teubner, 2010.
- [11] Dorothea Kolossa. Einführung in die automatische spracherkennung, 03 2014.

-
- [12] Benjamin Milde and Arne Köhn. Open source automatic speech recognition for german. In *Proceedings of ITG 2018*, 2018.
- [13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.
- [14] Aashish Agarwal and Torsten Zesch. German end-to-end speech recognition based on deepspeech. In *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 111–119, Erlangen, Germany, 2019. German Society for Computational Linguistics & Language Technology.
- [15] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.
- [16] Qiantong Xu Jeff Cai Jacob Kahn Gabriel Synnaeve Vitaliy Liptchinsky Ronan Collobert Vineel Pratap, Awni Hannun. wav2letter++: The fastest open-source speech recognition system. *CoRR*, abs/1812.07625, 2018.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 03.09.2020