Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Implementation und Evaluation automatischer Mehrkanal-Spracherkennung für das Konferenzsystem BigBlueButton

vorgelegt von

Robert Georg Geislinger

MIN-Fakultät

Fachbereich Informatik

Arbeitsbereich Language Technology

Studiengang: B. Sc. Informatik

Matrikelnummer: 6947836

Abgabedatum: 28.09.2021

Erstgutachter: Dr. Timo Baumann

Zweitgutachter: Benjamin Milde, M.Sc.

Kurzfassung

Ziel der vorliegenden Arbeit ist es, eine Software zu erstellen, die für Teilnehmer individuelle Untertitel automatisch generiert und direkt in BigBlueButton-Konferenzen integriert. Dazu werden nach einem Überblick über verwandte Arbeiten sowie Grundlagen der automatischen Spracherkennung zunächst eine bereits bestehende Lösung für die Live-Untertitelung in der Konferenzsoftware-BigBlueButton sowie auch Möglichkeiten zu deren Verbesserung besprochen. Anschließend werden verschiedene Audioquellen hinsichtlich ihrer Einsatzmöglichkeiten in der Software miteinander verglichen.

Darauf basierend wird ein Prototyp zur Live-Untertitelung von BigBlueButton-Konferenzen vorgestellt, der ohne Erfahrungen im Bereich der Sprachverarbeitung in eine bestehende BigBlueButton-Installation integriert werden kann. Dabei soll eine individuelle Untertitelung bei Überschneidungen von Wortmeldungen eine gleichbleibend niedrige Wortfehlerrate ermöglichen und zudem eine gute Sprecheradaption für jeden Teilnehmer bieten. Durch die Nutzung des serverseitigen Audiomaterials können die Untertitel unabhängig vom Endgerät generiert und angezeigt werden.

Im Anschluss an die Vorstellung des Prototyps der Untertitelungssoftware bbb-live-subtitles wird dieser in zwei verschiedenen Entwicklungsphasen hinsichtlich unterschiedlicher Praxiseigenschaften betrachtet. Es werden Einflussfaktoren wie Audio-Codec und Sprecherdaten auf die Erkennungsrate berücksichtigt und die Ergebnisse miteinander verglichen. Die Lösung wird mit einem Modell zur Erkennung deutscher Sprache getestet und die Ergebnisse evaluiert. Die entwickelte Software bietet die Möglichkeit, auch mit weiteren Modellen eingesetzt zu werden, die auf Kaldi nnet3 basieren und ist unter freier Lizenz verfügbar: `https://github.com/uhh-lt/bbb-live-subtitles`

Abstract

The aim of this thesis is to create a software that automatically generates individual subtitles for participants and integrates them directly into BigBlueButton conferences. For this purpose, after an overview of related work as well as basics of automatic speech recognition, first an already existing solution for live subtitling in the conference software-BigBlueButton as well as possibilities for its improvement are discussed. Afterwards, different audio sources are compared with respect to their possible applications in the software.

Based on this, a prototype for live subtitling of BigBlueButton conferences will be presented, which can be integrated into an existing BigBlueButton installation without any experience in the field of speech processing. Here, individual subtitling is intended to provide a consistently low word error rate in the event of overlapping spoken messages and also provide good speaker adaptation for each participant. By using server-side audio, the subtitles can be generated and displayed independently of the end device.

Following the presentation of the prototype of the subtitling software bbb-live-subtitles, it will be considered in two different development phases with regard to different practical characteristics. Factors such as audio codec and speaker data influencing the recognition rate are considered and the results are compared. The solution is tested with a model for the recognition of German speech and the results are evaluated. The developed software offers the possibility to be used with other models based on Kaldi nnet3 and is available under free license: `https://github.com/uhh-lt/bbb-live-subtitles`

Inhaltsverzeichnis

1 Einführung

In der vorliegenden Arbeit sollen zunächst das Forschungsziel erklärt und mögliche Vorteile durch die Integration automatischer Spracherkennung in die Konferenzsoftware BigBlueButton besprochen werden. Anschließend wird eine automatische Spracherkennung für mehrere unabhängige Sprecher in BigBlueButton integriert. Der Einfluss der verwendeten Software und des VOIP-Audio-Codecs auf die Qualität der Spracherkennung wird außerdem evaluiert.

1.1 Motivation

Aufgrund der aktuellen Covid-19-Pandemie werden viele persönliche Treffen verschoben oder abgesagt. Auch Veranstaltungen werden, wenn möglich, nicht mehr mit einem realen Treffen abgehalten, sondern online, um die Verbreitung des Virus einzuschränken. Um Veranstaltungen trotzdem durchführen zu können, weichen viele Unternehmen und Bildungseinrichtungen auf Onlinekonferenzsysteme aus. Sie sind im Beruf, in der Universität und in der Freizeit eine gute Alternative zu herkömmlichen Telefonaten, da sie durch die Möglichkeit der Videotelefonie persönlicher sind. In Bildungseinrichtungen sind sie oft die einzige Möglichkeit für interaktive Lehre. Im Jahr 2020 stieg so die Notwendigkeit von Konferenzlösungen sehr stark an (Rabe, 2021).
Sprache ist die natürlichste Art der Kommunikation. In einem persönlichen Gespräch beinhaltet sie das gesprochene Wort, Mimik und Ausdruck der sprechenden Personen. Onlinekonferenzen stellen für Menschen mit Hördefiziten jedoch eine größere Hürde dar. Die Möglichkeit, Informationen wie Mimik, Mimik oder Lippenbewegungen anderer Teilnehmer zu lesen, wird häufig erschwert (Maria Geipel, 2021). Ursachen dafür können eine fehlende oder abgeschaltete Webcam, Kameras mit niedriger Auflösung, eine schlechte Ausleuchtung des Teilnehmers, zu großer Abstand zur Kamera, eine niedrige Bildrate oder Verbindungsschwierigkeiten sein, die das Bild für diesen Einsatzzweck stark einschränken (Spiegl, 2015; Neuhetzki, 2021). Somit fallen eingeübte Techniken der Informationsgewinnung, die für akustisch eingeschränkte Menschen wichtig sind, weg. Unterstützende Maßnahmen wie Untertitel können helfen, körperlich eingeschränkten Menschen einen besseren und natürlicheren Zugang zu Online-Konferenzen zu ermöglichen (Bundesfachstelle Barrierefreiheit, 2021).

Aber auch für Menschen ohne Einschränkungen beim Hören können Untertitel in Onlinekonferenzen hilfreich sein. Um Audio- und Videosignale in Konferenzen zu übertragen, wird eine stabile Internetverbindung mit ausreichend großer Bandbreite benötigt. Wenn eine solche Bandbreite nicht gegeben ist, sinkt die Qualität der Verbindung und damit die Verständlichkeit des Gesagten. Die Gründe hierfür können Latenzen der Internetverbindung oder hohes Datenaufkommen beim Internetanschluss des Nutzers oder aufseiten des Anbieters sein (Wup-

permann, 2021). Auch zu veraltete oder inkompatible Endgeräte, fehlerhafte Bedienung oder falsche Konfiguration können die Verbindung stark beeinträchtigen. Betroffene Teilnehmer können dann nur in geringem Maße selbst per Sprache oder Video zur Konferenz beitragen. Außerdem können sie so auch nur eingeschränkt andere Teilnehmer verstehen, da eine instabile Verbindung und geringe Bandbreite dazu neigen, das übertragene Audio- und Videosignal mit Artefakten oder Abbrüchen unverständlich zu machen. Auch Komplettausfälle sind nicht auszuschließen. Gleichzeitig sind die Einschränkungen eines Teilnehmers für andere Anwesende nicht immer sichtbar. So ist zum Beispiel für Schüler eine aktive Mitarbeit in Unterrichtseinheiten dann schwierig. Die mündliche Mitarbeit wird erschwert und Fragen zu Themen können nur schwer gestellt werden, während der Lehrer nicht direkt auf die Probleme des Schülers aufmerksam gemacht wird. Lehrer können aufgrund fehlender Informationen Schülern dann nicht immer die nötige Hilfestellung geben oder deuten die fehlende Mitarbeit im schlimmsten Fall als Arbeitsverweigerung (Hellweger Anzeiger, 2020). Verzögerungen des Audio- und Videosignales führen zudem dazu, dass sich Redebeiträge mit denen anderer Teilnehmer zeitlich überschneiden (Kettinger, 2021).

Die Konferenzsoftware **B**ig**B**lue**B**utton[1] (kurz BBB) bietet die Möglichkeit, in Onlinekonferenzen Untertitel einzublenden. Untertitel dienen dazu, gesprochene Inhalte für Hörgeschädigte erfassbar zu machen (Wikipedia, 2021). Es wird dabei zwischen offenen Untertiteln und geschlossenen Untertiteln unterschieden. Geschlossene Untertitel (englisch Closed Captions, CC) können optional eingeblendet werden, während offene Untertitel direkt in das Video mit eingelassen sind. Die vorliegende Arbeit bezieht sich ausschließlich auf geschlossene Untertitel.

Die Möglichkeit, Untertitel von Teilnehmern händisch während einer Konferenz erstellen und veröffentlichen zu lassen, stellt einen kosten- und zeitintensiven Vorgang dar. Pro Minute Videomaterial benötigen sogar trainierte Redakteure ohne technische Hilfe das bis zu 18-Fache der Sprechzeit (B. C. Roy und D. K. Roy, 2009). **A**utomatic **S**peech **R**ecognition, kurz ASR (dt. Automatische Spracherkennung), ermöglicht die automatische Untertitelung von Audio- und Videomaterial von Livekonferenzen, Veranstaltungen und auch von aufgezeichnetem Material. Die Daten werden dazu mit einem ASR-Toolkit und -Sprachmodell verarbeitet. Details dazu werden in Kapitel 3 erläutert.

BBB verfügt von Herstellerseite aus nur über eine Unterstützung für manuelle Untertitel. Diese dient in der vorliegenden Arbeit als Basis, um eine stabilere und direktere automatische Untertitelung zu ermöglichen. Die hier vorgestellte Lösung nutzt eine individuelle Verarbeitung des Audiomaterials eines jeden Teilnehmers und nicht das abgemischte Audiomaterial. Abgemischt bezeichnet in diesem Zusammenhang die durch die Konferenz verarbeiteten Audiosignale mit allen Verarbeitungsschritten sowie die Zusammenführung der einzelnen Teilnehmerbeiträge zu einem neuen Signal. Dieses enthält sowohl den Ton aller Teilnehmer als auch alle Audiomitteilungen durch die Konferenz, wie z.B. Benachrichtigungen, wenn jemand der Konferenz beitritt oder sie wieder verlässt.

Um die Fehlerrate der Spracherkennung zu senken, sollen in dieser Arbeit die Nutzer einzeln untertitelt werden. Dazu werden die Audiostreams der einzelnen Teilnehmer mitgeschnitten. Diese einzelnen Datenströme erhalten jeweils eine unabhängig laufende Spracherkennungs-

[1] `https://bigbluebutton.org/`

instanz. Eine Erkennungsinstanz enthält alle benötigten Verarbeitungsschritte, ohne Einfluss anderer Audiostreams. So wird eine hohe Erkennungsrate sowohl bei überlappenden Sprechern als auch bei kontinuierlichem Sprecherwechsel ermöglicht. Diese Arbeit basiert auf Software und Modellen mit einem offenen Lizenzmodell, welche frei und ohne Lizenzkosten eingesetzt werden können. Zur Spracherkennung wird das Kaldi Toolkit zusammen mit einem deutschen Modell, welches an der Universität Hamburg entwickelt wurde, genutzt.

1.2 Ziel

Das Ziel der vorliegenden Arbeit ist es, eine stabile, verständliche und einfach zu nutzende Untertitelung anzubieten, die ohne Kosten für Software und Lizenzen eingesetzt werden kann. Durch Sprecheradaption soll sich die Erkennung während der Verarbeitung auf jeden Teilnehmer individuell ausrichten. Die Spracherkennung soll mit den vorgestellten Methoden eine geringe Fehlerrate bieten und bestehende Lösungen ersetzen. Es soll somit die Möglichkeit geschaffen werden, auch Nutzern, die wenig bis keine Erfahrungen im Bereich der automatischen Sprachverarbeitung haben, ein einfaches Hilfsmittel zu bieten, um die eigenen Konferenzen automatisch zu untertiteln. Anpassungen und zusätzlicher Wartungsaufwand an bestehenden BBB-Lösungen sollen so gering wie möglich sein, um die Schwelle für automatische Untertitel in Konferenzen zu senken. Der Quellcode der Arbeit wird frei verfügbar gemacht. Ziel der Thesis ist zudem die Besprechung und Evaluation der erzielten Ergebnisse.

2 Verwandte Arbeiten und Werke

Für das Erstellen von Untertiteln für Fernsehprogramme werden schon seit Jahrzehnten verschiedenste Techniken genutzt (Marsh, 2006). Seit dem Anfang der zweitausender Jahre wird die automatische Spracherkennung zur Erstellung von Untertiteln in Teilen eingesetzt. Dazu wurde zunächst das Nachsprechen als eine Methode zur semi-automatischen Erstellung von Untertiteln verwendet (Romero-Fresco, 2020; Lambourne, 2006; Holter et al., 2000). Bei dieser Lösung wird die Sprache von trainierten Sprechern nachgesprochen und dann mit einem ASR-System zu Text verarbeitet. Damit ist es möglich, sowohl bei Aufzeichnungen als auch bei auch Live-Veranstaltungen wie Fernsehübertragungen oder Konferenzen eine geringe Wortfehlerrate (siehe Grundlagen 3.8) bei gleichzeitig niedriger Verzögerung zu ermöglichen.

In der Arbeit von Obach et al. wurden automatische Systeme zur Untertitelung von spanischen Fernsehprogrammen erprobt. Dafür wurden mehrere kommerzielle Systeme analysiert und miteinander verglichen. Eine Aufteilung der Tonspuren auf einzelne Sprecher wurde nicht untersucht, so dass Überschneidungen der Sprecher möglich waren. Die Erkennungsrate war dabei am besten, wenn nur eine Person sprach (Eizmendi et al., 2007).

Die Entwicklung im Bereich ASR ist bereits so weit fortgeschritten, dass die Sprache vor der Verarbeitung nicht immer erneut nachgesprochen werden muss, sondern auch direkt verwendet werden kann (Staš et al., 2015). Es wurde untersucht, ob eine automatische Spracherkennung auch bereits genutzt werden kann, um Lehrvideos automatisch mit Untertiteln zu versehen. Dabei wurden gute Ergebnisse erzielt (Qiu, 2019; Milde, Geislinger et al., 2021).

Auch bei der automatischen Untertitelung von tschechischen Parlamentsreden wurde ASR eingesetzt. Die Erkennungsrate betrug dabei bis zu 80 Prozent (Pražák et al., 2006).

Wenn die Sprache direkt von einem Video bzw. aus einer Konferenz verwendet wird, ist der Audio-Codec ein Einflussfaktor auf die Erkennungsrate. Der frei verfügbare Opus Codec, der Teil von BBB ist, wurde mit anderen Codecs verglichen, die für Echtzeit-Übertragungen wie im Mobilfunknetz oder bei VoIP verwendet werden. In Vergleichen mit verschiedenen Audio-Codecs wurde festgestellt, dass der Opus LP Modus (Linear Prediction, dt. lineare Vorhersage) bei geringer Bitrate eine konkurrenzfähige Sprachqualität bietet. Außerdem bietet der OPUS CELT Modus (Constrained-Energy Lapped Transform, dt. überlappende Transformation mit vorgegebener Energie) bei höherer Rechenleistung eine gute Alternative mit besserer Sprachqualität (Rämö und Toukomaa, 2011).

In der Arbeit von Maruschke et al. wurde der Opus Codec in einem WebRTC Umfeld getestet (Maruschke et al., 2015). WebRTC ist eine Sammlung von offenen Kommunikationsprotokollen unter freier Lizenz, die Multimedia und Datenverbindungen mit dem Webbrowser ermöglichen und nach den Richtlinien des W3C (World Wide Web Consortium) und der IETF (Internet Engineering Task Force) standardisiert sind.

Dabei zeigte sich, dass alle drei in der Arbeit untersuchten Betriebsmodi des Opus Codec (SILK, CELT oder eine hybride Form) geeignet sind, Sprache zu übertragen. Außerdem liegt die Latenz von Opus im Bereich von 2,5 ms bis 60 ms und ist damit für eine Echtzeit-Übertragung geeignet. Im Vergleich zu anderen Codecs bietet der Opus Codec auf allen Qualitätsstufen eine geringe Latenz (Wikimedia Commons, 2020).

Um eine Spracherkennung für eine Zielsprache zu ermöglichen, wird ein passender Datensatz mit Modell benötigt. Die Definition von Modellen zu Spracherkennung wird in Abschnitt 3.3.2 und 3.3.4 eingeführt. Für die englische Sprache gibt es viele Modelle, welche auch zum Teil kostenfrei verfügbar sind[1]. In dieser Ausarbeitung wird die Erkennung von deutscher Sprache untersucht. Ein eignes Modell zu entwickeln und zu trainieren, mit dem eine gute Erkennungsrate erreicht wird, ist nur mit hohem Aufwand möglich (Maas et al., 2017). Dies zu leisten, würde den Rahmen der vorliegenden Thesis übersteigen. Als vortrainiertes Modell wird daher das Kaldi-tuda-de Modell der Universität Hamburg eingesetzt. Dieses Modell wurde mehrmals erweitert und bietet mit 1000 Stunden Audio und über 100 Millionen Zeilen Text als Trainingsmaterial eine Wortfehlerrate von 11,85% (Milde und Köhn, 2018; Milde und Köhn, 2021).

[1] `https://kaldi-asr.org/models.html`

3 Grundlagen

In diesem Kapitel werden die Grundlagen automatischer Sprachverarbeitung, Audio-Codecs und Konferenzsoftware erläutert. Es wird sowohl auf Elemente der Sprache eingegangen als auch auf einzelne Schritte bei der Verarbeitung von Sprache zur Spracherkennung.

3.1 Audiosignale

Bei Audiosignalen wie gesprochener Sprache handelt es sich um analoge Signale. Diese müssen zur Verarbeitung in digitale Signale umgewandelt werden. Um die Sprache mit einem Gerät aufzuzeichnen, wird ein Mikrofon verwendet. Die aufgenommenen zu verarbeitenden Signale setzen sich aus verschiedenen Elementen zusammen wie Grundton, Obertöne und Rauschen und werden von unterschiedlichen Merkmalen beeinflusst: Umgebungsfaktoren wie Nebengeräusche oder Raumakustik und technische Aspekte wie Abtastrate, Kompression, Mikrofonart oder Auflösung. Das Ergebnis wird außerdem von Störgeräuschen wie Hesitationen (Ähm, Öhm), Rascheln oder Atemgeräuschen negativ beeinflusst.

Für die Umwandlung werden die analogen Signale mit einem Analog-Digital-Wandler von einem kontinuierlichen Signal in ein diskretes Signal transformiert (Walden, 1999). Kontinuierliche Signale besitzen einen stufenlosen und unterbrechungsfreien Verlauf und diskrete Signale sind nur zu bestimmten Zeitpunkten definiert. Der Analog-Digital-Wandler tastet das analoge Signal ab, wobei die Auflösung (in Bits gemessen) und Abtastfrequenz (in Hertz gemessen) die wesentlichen Parameter sind, um eine größtmögliche Genauigkeit zwischen dem Eingangs- und dem rekonstruierten Signal zu erhalten. Die Abtastfrequenz beschreibt, wie oft das Eingangssignal abgetastet wird, und die Auflösung, wie viele verschiedene Werte ein Wert annehmen kann. Zur vollständigen Rekonstruktion des ursprünglichen Signals muss die Abtastfrequenz mindestens dem Doppelten der Frequenz des Eingangssignals entsprechen, da es sonst zu einer Unterabtastung kommt. Bei einer Unterabtastung kommt es zu fehlerhaften Frequenzen im rekonstruierten Signal (Shannon, 1949). Der Mensch hört, je nach Alter, in einem Bereich von etwa 20 Hertz bis 20 kHz (Purves et al., 2004). So muss die Abtastfrequenz bei über 40 kHz liegen um alle Frequenzen des menschlichen Hörens abzutasten (Shannon, 1949).

3.2 Phoneme

Als Phoneme werden in der gesprochenen Sprache die abstrakten Klassen aller sprachlichen Laute (Phone) bezeichnet, die die gleiche bedeutungsunterscheidende Funktion besitzen (Müller, 1984). Morpheme bezeichnen die kleinste funktionstragende Einheit. Die deutsche Sprache besteht aus bis zu 60 Phonemen (Wiktionary, 2020).

Eine weitere Variante der Phoneme sind die Allophone. Sie beschreiben eine lautliche Variante eines Phonems, das aber nicht den Inhalt des Gesprochenen verändert. Allophone können aufgrund von Sprecherdaten wie Alter, Geschlecht, Gefühlslage, Sprechstil und Akzent auftreten oder durch die Position des Phonems im Wort entstehen (Peperkamp et al., 2003). Ein Beispiel für ein Allophon ist die Aussprache des Wortes "Rot". Es kann mit gerolltem R ('roːt) oder geriebenem R ('ɹoːt) gesprochen werden.

Die am weitesten verbreitete Darstellung von Phonemen ist das Internationale Phonetische Alphabet (kurz: IPA), welches von der International Phonetic Association entwickelt wurde. Es enthält eine Sammlung von Zeichen, mit deren Hilfe alle Laute der menschlichen Sprache beschrieben werden können (Association, Staff et al., 1999). Aufbauend auf dem IPA gibt es die X-SAMPA Notation. Diese bildet das IPA auf den ASCII-Zeichensatz ab und vereinfacht so die Nutzung von Phonemen mit Computern (Wells, 1995). Im Open-Source-Toolkit für Spracherkennung Kaldi wird deshalb X-SAMPA zur Notation verwendet. Eine Liste an Beispielwörtern und ihrer Notation in IPA und X-SAMPA-Schreibweise ist in Tabelle 3.1 dargestellt.
Das in dieser Arbeit verwendete Modell zur Spracherkennung, Kaldi-tuda-de, verwendet ebenfalls die X-SAMPA-Notation.[1]

Wort	IPA	X-SAMPA
Katze	'katsə	"kats@
Faultier	'faʊ̯ltiːɐ̯	"faU_^lti:6_^
Körbchen	'kœrpçən	"k9rpC@n
Baum	'baʊ̯m	"baU_^m
Höhle	'høːlə	"h2:l@

Tabelle 3.1: Wörter aus dem Deutschen dargestellt im IPA und X-SAMPA Format

3.3 Aufbau automatischer Sprachverarbeitung

Im Aufbau automatischer Sprachverarbeitung wird zwischen mehreren Systemen unterschieden. Im Zusammenhang mit neuronalen Netzen gibt es zwei grundlegende Systeme: Zum einen gibt es reine neuronale Netze, die häufig als End-to-End bezeichnet werden. Hierbei werden Netze mit Daten trainiert, wodurch gewichtete Modelle erzeugt werden, die später zur Spracherkennung eingesetzt werden. Zum anderen gibt es Mischsysteme, die aus mehre-

[1] https://github.com/uhh-lt/kaldi-tuda-de/blob/master/s5_r2/local/de_extra_lexicon.txt

ren Komponenten bestehen und neuronale Netze mit stochastischen Modellen wie dem HMM (Hidden Markov Model, dt. verborgenes Markowmodell) kombinieren.

In dieser Arbeit wird ein TDNN-HMM-Mischsystem verwendet, welches eine geringe Fehlerrate bietet und es gleichzeitig ermöglicht, auch einzelne Komponenten wie akustisches Modell oder Sprachmodell separat zu trainieren, nachdem das Training aller anderen Komponenten bereits abgeschlossen ist. TDNN (Time Delayed Neuronal Network, dt. zeitverzögertes neuronales Netzwerk) werden in Kapitel 3.3.2.2 eingeführt.

3.3.1 MFCC

Um das Audiosignal zu verarbeiten, muss dieses in ein unkomprimiertes PCM-Signal (Puls-Code-Modulation-Signal) Signal mit 16khz mit einem monophonen Audiokanal umgewandelt werden, insofern es nicht bereits in diesem Format vorliegt. Dieses Eingangssignal wird mit dem MFCC-Verfahren verarbeitet. MFCC (Mel Frequency Cepstral Coefficients, dt. Mel-Frequenz-Cepstrum-Koeffizienten) sind Koeffizienten, die aus dem ursprünglichen Signal in mehreren Schritten zu kurzen Feature-Vektoren verarbeitet werden (Davis und Mermelstein, 1980). Als Formanten bezeichnet man den Teil der Stimme, der im Vokaltrakt gebildet wird. Der Vokaltrakt beim Menschen umfasst Rachenraum, Mundraum und Nasenraum. Ein Grund für die Umwandlung mittels MFCC-Verfahren ist, dass Formanten, die hilfreich zur Unterscheidung von Vokalen sind, besser im Frequenzbereich (eng. frequency domain) charakterisiert werden können als im Zeitbereich (eng. time domain) (Huang et al., 2001). Der Zeitbereich beschreibt einen linearen Verlauf der Frequenzen über die Zeit (siehe Abbildung 3.1). Im Frequenzbereich sind die Frequenzen aufsteigend an der x-Achse angeordnet (statt dem Auftreten nach Zeit) und die y-Achse beschreibt die Intensität.

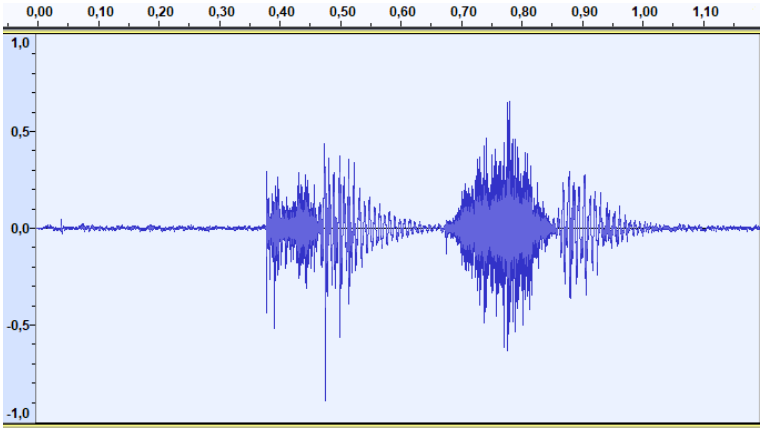

Abbildung 3.1: Das Wort Katze dargestellt als Wellen. Die x-Achse visualisiert die Amplitude und die y-Achse den zeitlichen Verlauf des Signals. (Screenshot Audacity)

Beim Sprechen wird das periodische Anregungssignal (Stimmbänder) durch einen linearen Filter (Mund, Zunge, Nasenhöhle, ...) gebildet. Diese Filter werden auch Impulsantwort genannt. Das MFCC-Verfahren filtert das Signal, um die Impulsantwort und das Anregungssignal voneinander zu trennen und die Impulsantwort weiter zu verwenden. Informationen wie

Geschlecht, Hintergrundgeräusche oder Emotionen werden aus dem Signal entfernt. Um aus dem ursprünglichen Signal die MFC-Koeffizienten zu generieren, sind folgende Schritte notwendig:

1. In der Sprache enthalten niedrigere Frequenzen mehr Energie als hohe Frequenzen. Um die Informationen hoher Frequenzen in späteren Schritten besser nutzen zu können, werden diese mit einem Hochpassfilter verstärkt. Dieses Verfahren wird Pre-Emphasis genannt (Jurafsky und Martin, 2009).

2. Um einzelne Phone zu klassifizieren, müssen diese aus dem Signal erkannt werden. Die Verteilung der Phone über das Signal ist jedoch nicht stationär über das Signal verteilt. Das bedeutet, dass ihre statistische Verteilung über die Zeit nicht konstant ist. Die Zerlegung des Signals in einzelne kurze Abschnitten, genannt Fenster, soll die Phone stationär in den jeweiligen Abschnitten verteilen (Jurafsky und Martin, 2009). Eine Dauer von 15 bis 35 ms je Fenster wurde als ideale Länge zur Unterscheidung einzelner Phoneme festgestellt (Paliwal et al., 2010). Ein Fenster hat eine Länge von 25 ms und die Fenster überlappen sich alle 10 ms (Jurafsky und Martin, 2009).

3. Um spektrale Informationen aus dem Fenster zu extrahieren, wird eine diskrete Fourier-Transformation (kurz DFT) angewendet. Die Fourier-Transformation dient dazu, das Signal vom Zeitbereich in den Frequenzbereich umzuwandeln (Jurafsky und Martin, 2009).

4. Das menschliche Gehör ist bei niedrigen Frequenzen besser in der Lage, zwischen zwei Tönen zu unterscheiden als bei höheren Frequenzen über 1000 Hz. Bis 1000 Hz ist der wahrgenommene Abstand zwischen den Tonhöhen linear. So verdoppelt sich die empfundene Tonhöhe zwischen 100 Hz und 200 Hz. Ab 1000 Hz wächst der Abstand zwischen den empfundenen Tönen nicht mehr linear, sondern logarithmisch. Die wahrgenommene Tonhöhe wird in der Mel-Skala ausgedrückt; das Mel ist die Maßeinheit für die psychoakustische Größe Tonheit. Im Bereich unter 1000 Hz werden Dreiecksfilter in einem Abstand von 100 Hz angeordnet und oberhalb 1000 Hz in einem logarithmischen Abstand verteilt. Das reduziert die zu betrachtenden Features auf insgesamt 40 und erhöht damit die Performance für spätere Berechnungen (Jurafsky und Martin, 2009).

5. Aus dem bestehenden Signal wird nun das Cepstrum berechnet. Das Cepstrum ist das Ergebnis der Berechnung mit der Fourier-Transformation, der Logarithmierung der Frequenzen und der anschließenden inversen DFT. Hierbei werden Anregungssignal und lineare Filter voneinander getrennt, um den Filter zu erhalten. Das Anregungssignal wird in diesem Schritt aus dem Signal entfernt, da es nicht notwendig ist, um Phone voneinander zu unterscheiden. Das Ergebnis ist eine Transformation vom Frequenzbereich in den Zeitbereich (Jurafsky und Martin, 2009).

6. Im letzten Schritt wird die Energie des Fensters berechnet. Die Energie korreliert mit der Phone-Identität und dient zur Unterstützung bei der Unterscheidung der Phone voneinander.

So haben Vokale mehr Energie als Knacklaute. Ein Beispiel für ein Wort mit Knacklaut oder glottalem Plosiv (IPA:ʔ) ist das Wort „Acht“. Energie in diesem Zusammenhang ist beschrieben als die Summe der Kraft über die Zeit in einem Zeitabschnitt. Außerdem werden für alle Features (40 Cepstra plus Energie) ein Delta und die Beschleunigung Delta/Delta berechnet. Das Delta repräsentiert die Änderung zwischen den einzelnen Fenstern bezogen auf die Features während die Beschleunigung die Änderung der Deltas zwischen den Fenstern abbildet (Jurafsky und Martin, 2009).

Das Ergebnis dieser Verarbeitungsschritte sind die MFC-Koeffizienten (Jurafsky und Martin, 2009). Diese sind dargestellt in der Abbildung 3.2. Die MFC-Koeffizienten bilden die Eingabe für das akustische Model.

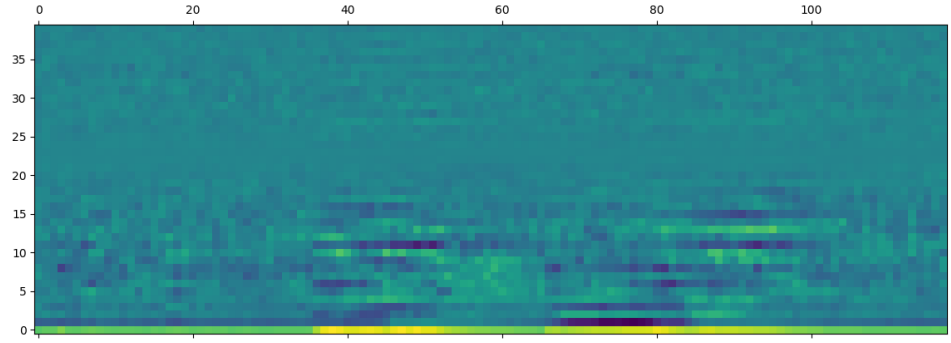

Abbildung 3.2: Die x-Achse repräsentiert den zeitlichen Verlauf, die y-Achse das Feature und die Farbe die Energie des Features. Die Zeile 0 repräsentiert die Energie aller Features zum Zeitpunkt t. Das Farbspektrum visualisiert die Energie der Signale: von grün (wenig Energie) bis gelb (hohe Energie)

3.3.2 Akustisches Modell

Das Acoustic Model (dt. akustisches Modell) wird verwendet, um Feature-Vektoren, die aus dem Eingangssignal generiert wurden, auf Phoneme oder Phonemketten abzubilden. Es umfasst sowohl das gelernte Wissen über die Dialekte, Mikrofon-Variabilität und Akustik als auch Aussprachemodelle für größere Einheiten wie Wörter oder Phrasen (Huang et al., 2001). Für die Signale werden Wahrscheinlichkeiten berechnet, die den jeweiligen bekannten Phonemen entsprechen. Hierbei lernt das akustische Modell die statistische Repräsentation der Eigenschaften der Audiosignale zur Sprache.

Akustische Modelle werden häufig mit HMM oder TDNN modelliert (Jelinek, 1976; Waibel et al., 1989). Aufgrund einer geringeren Fehlerrate bei der Verwendung von TDNN Modellen und steigender Rechenkapazität wurden die HMM-Modelle von neuronalen Netzen abgelöst (Georgescu et al., 2019; Milde und Köhn, 2018).

3.3.2.1 HMM

Das HMM ist ein stochastisches Modell, bestehend aus Markowketten. Das System wechselt zufällig von einem Zustand in den nächsten. Die Übergangswahrscheinlichkeit zwischen den Zuständen hängt nur vom aktuellen Zustand ab und nicht wie bei endlichen Automaten von vorher eingenommenen Zuständen. Die eigentlichen Zustände sind verborgen und können nicht observiert werden. Stattdessen können nur die ursprünglichen Eingaben und die Ausgaben beobachtet werden (Rabiner und Juang, 1986). Zur Dekodierung wird der Viterbi-Algorithmus verwendet. Der Algorithmus dient dazu, für jeden Zeitpunkt t den Pfad mit der höchsten Wahrscheinlichkeit zu wählen. Diese Sequenz von Zuständen wird als Viterbi-Pfad bezeichnet. Der Viterbi-Algorithmus gehört zu den Methoden der dynamischen Programmierung (Forney, 1973).

Da die Werte meist nicht symmetrisch und unimodal, also in gleichem Abstand mit einem eindeutigen Maximum verteilt sind, wird die einzelne Normalverteilung im HMM durch ein GMM ersetzt. Ein GMM (Gaussian Mixture Model, dt. Gaußsches Mischmodell) ist ein statistisches Mischmodell. Mischmodelle enthalten sowohl feste Effekte als auch zufällige Effekte. Das soll das Modell gegenüber Unterschieden zwischen den Sprechern, Geschlecht oder Akzent robust machen (Gales und S. Young, 2007). HMM erreichen im Vergleich zu TDNN-HMM Modellen eine geringere Performance (Shahin et al., 2014; Gaida et al., 2014).

3.3.2.2 TDNN

Das TDNN ist eine Art von neuronalen Netzen und bietet im Bereich der automatischen Sprachverarbeitung die Möglichkeit, eine große Variabilität von Sprache zu verarbeiten und zu erkennen (Waibel et al., 1989).

Ein weiterer Netzaufbau, der im Bereich der automatischen Sprachverarbeitung genutzt wird und mit den Daten mehrerer Zeitpunkte t arbeitet, ist das RNN (Recurrent Neuronal Network, dt. rückgekoppeltes neuronales Netz). Hier wird der Ausgang der Neuronen des Layers (dt. Schicht) n als Eingang für Neuronen des Layers $n-1$ als zusätzliche Eingabe verwendet (Kacprzyk und Pedrycz, 2015). Die Trainingszeit bei RNN-Netzen liegt aufgrund des sequenziellen Trainings höher als bei Feedforward-Netzen (dt. Vorwärts-Netze), zu denen das TDNN zählt. Bei Feedforward-Netzen bilden die Verbindungen zwischen den Neuronen einen azyklischen, gerichteten Graphen. Der Ausgang von Neuronen im Feedforward-Netz von Layer n ist somit der Eingang der Neuronen in Layer $n+1$. Diese Verbindungen können nicht auf sich selbst oder vorherige Layer zeigen. Weiterhin lassen sich Feedforward-Netze besser parallelisieren und die Trainingszeit ist besonders bei der Verwendung von Grafikprozessoren geringer.

Bei TDNN werden mehrere Zeitpunkte gleichzeitig als Eingangssignal für das Netz verwendet. So werden zum Beispiel die Zeitpunkte $t-2$ bis $t+2$ als Eingang verwendet. Um Sprache zu verarbeiten, eignen sich TDNN dadurch gut, da Phoneme keine einheitliche Länge besitzen. Das macht eine präzise Segmentierung sehr schwierig und würde einen großen Aufwand bedeuten. Bei der Verarbeitung mit einem TDNN werden die Zusammenhänge zwischen den Zeitpunkten bei höheren Layern über größere zeitliche Distanzen verbunden. So sind im

Eingangslayer die Eingänge der Neuronen mit den Zeitpunkten $t-2$ bis $t+2$ verbunden und in höheren Layern zum Beispiel $t-3$ und $t+3$ (Peddinti et al., 2015). Durch die Verarbeitung eines Signals über Zukunft und Vergangenheit kann das TDNN die Kernelemente dieses Signals auf eine zeitverschiebungsinvariante Weise konstruieren. Die Erkennung ist somit gegen Verschiebung der Elemente über die Zeit robust (Waibel et al., 1989).

3.3.2.3 TDNN-HMM

Das TDNN-HMM Modell arbeitet zweistufig mit einem Frontend (dt. Oberbau) und einem Backend (dt. Unterbau). Als Frontend wird das TDNN genutzt, das als Eingabe die MFCC-Features enthält. Als Backend wird das HMM verwendet, das als Eingang die Klassifikationen beinhaltet. Dieser Modellaufbau senkt die Fehlerrate im Vergleich zu einem reinen TDNN (C. Jang und Un, 1996). Das HMM berechnet die finalen Wahrscheinlichkeiten und gibt als Ausgang den jeweiligen Wert an. Die Verarbeitung von Daten mittels TDNN-HMM ist in Abbildung 3.3 dargestellt.

Das in der vorliegenden Arbeit genutzte akustische Modell ist nnet3, welches als TDNN-HMM modelliert ist (Povey, Zhang et al., 2015).

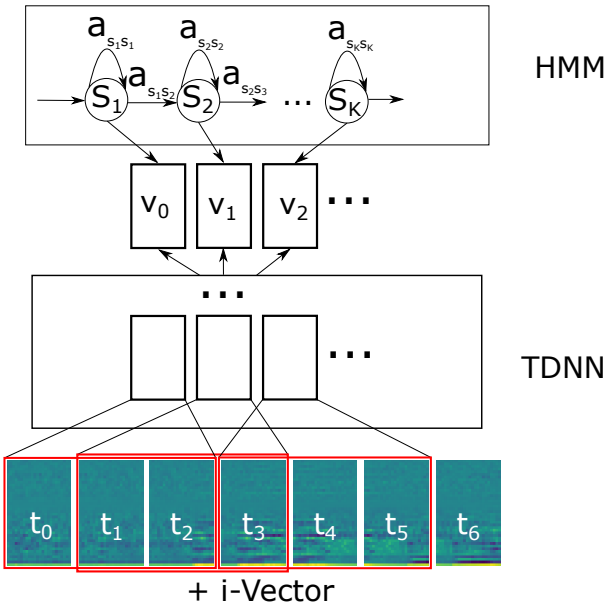

Abbildung 3.3: Aufbau eines TDNN-HMM

3.3.3 Sprecheradaption und -identifikation

Die Sprecheradaption wird in ASR-Systemen eingesetzt, um sich an die akustischen Features eines Nutzers (wie z.B. Sprechtempo und Stimmlage) anzupassen (Shinoda, 2005). Die Sprecheridentifikation ist ein Verfahren zur Identifikation und Verifikation einzelner Sprecher in einem ASR-System (Reynolds, 1995). Ziel der Sprecheradaption ist es, Eigenschaften der Sprecher wie Sprachschatz oder auch Sprachmuster abzubilden.

Die Sprecheradaption wird in Kaldi über den Identity-Vector (kurz i-Vector, dt. Identitätsvektor) abgebildet. Der i-Vector ist ein hundertdimensionaler Vektor, der die Spracheigenschaften des Sprechers und umgebungsspezifische Informationen über seine akustische Umgebung enthält (Dehak et al., 2011). Die schrittweise Anpassung des i-Vectors an den Sprecher während der Dekodierung erhöht die Erkennungsrate. Der Vektor ist von links nach rechts aufgebaut. Neuere Daten werden rechts an den bestehenden Vektor angehängt. Das bedeutet, dass zu einem Zeitpunkt t die Informationen vorheriger Zeitpunkte des Sprechers genutzt werden (Karafiát et al., 2011). Dieser Vektor wird an jedes Frame angefügt (Saon et al., 2013). Eine Visualisierung von i-Vectoren ist in Abbildung 3.4 dargestellt.

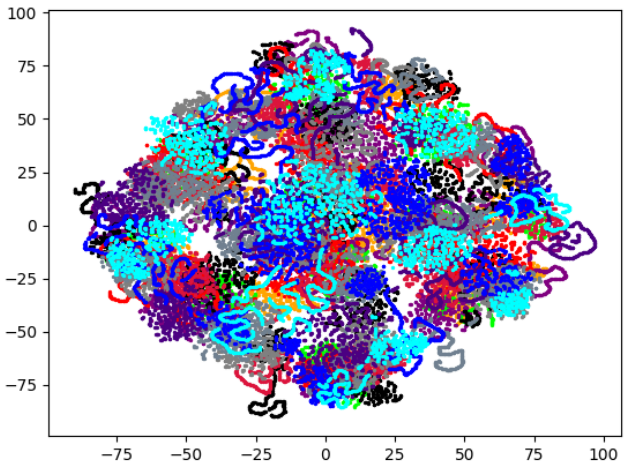

Abbildung 3.4: Visualisierung von i-Vectoren am Beispiel des Tuda-De Trainingsdatensatzes. Die Farben repräsentieren die einzelnen Sprecher.

3.3.4 Sprachmodell

Ein Language Model (dt. Sprachmodell) ist eine statistische Wahrscheinlichkeitsverteilung über Sequenzen von Wörtern. Sprachmodelle werden in verschiedenen Bereichen eingesetzt, besonders wenn Text generiert werden soll, wie zum Beispiel bei OCR (Optical Character Recognition, dt. optische Zeichenerkennung), bei maschineller Übersetzung und Spracherkennung. Ein Sprachmodell enthält die Wahrscheinlichkeiten einer Folge von Wörtern in der

Sprache. Außerdem gibt es den Kontext an, um zwischen Phrasen und Wörtern zu unterscheiden, die ähnlich ausgesprochen werden, aber andere Bedeutungen haben. Sprachmodelle werden als n-gram mit Hidden Markov Modell modelliert (Bahl et al., 1983). Repräsentiert werden sie als endlicher Automat (Mohri et al., 2002). N-gramme bezeichnen Sequenzen von Buchstaben oder Wörtern. In der Spracherkennung werden n-gram Modelle zur Abbildung von Sequenzen von Wörtern eingesetzt. Ein Beispiel für eine Sequenz von Wörtern ist „die Katze legt sich in ihr Körbchen".

Aufgrund der Vielzahl möglicher Wortkombinationen und Sätze, die eine Sprache ermöglicht, kommen im Trainingsdatensatz viele mögliche Sätze und Wortfolgen nicht vor. Eine Lösung, um trotzdem mit einem verhältnismäßig kleinen Datensatz eine große sprachliche Abdeckung zu erreichen, ist ein n-gram-Modell. In einem n-gram-Modell werden zu einem Wort die n vorherigen Wörter im Satz betrachtet und es wird berechnet, welches Wort am wahrscheinlichsten auf das betrachtete Wort folgt.

3.3.4.1 Finite-state Automaton

Ein FSA (Finite-state Automaton, dt. endlicher Automat) ist ein Maschinenmodell, das unter Eingabe eines Symbols des Eingabealphabetes vom aktuellen Zustand in einen Folgezustand wechselt (Rabin und Scott, 1959).

In der Spracherkennung werden FST (Finite-State Transducer, dt. Transduktor) mit gewichteten Kanten zur Repräsentation von verschiedenen Elementen wie Sprachmodellen, akustischen Modellen, Aussprachewörterbüchern oder Lattices (dt. Gitter) verwendet. Ein FST ist eine spezielle Form des endlichen Automaten, der nicht nur akzeptiert oder ablehnt, sondern auch eine Ausgabe erzeugt. Eine Erweiterung um Wahrscheinlichkeiten beim Zustandsübergang bildet der WFST (Weighted Finite-State Automata, dt. gewichteter Transduktor). Er enthält gewichtete Kanten wobei das Gewicht der Kanten die Wahrscheinlichkeiten repräsentiert, mit denen die Zustände aufeinander folgen (Mohri et al., 2008).

In Abbildung 3.5 ist der Ausschnitt eines beispielhaften WFST abgebildet. Die Kreise repräsentieren die Zustände, während die Pfeile/Kanten Transitionen von einem Zustand in den nächsten darstellen. Die Wahrscheinlichkeit, dass auf den Zustand **1** der Zustand **3** mit dem Phonem */a/* folgt, ist durch die Transition *0,8:/a/* mit 0.8 bzw. 80% angegeben. Zur besseren Übersichtlichkeit wurden im Beispielautomaten nur die Kanten mit der höchsten Wahrscheinlichkeit dargestellt, die zusammen das Wort „Katze" ergeben.

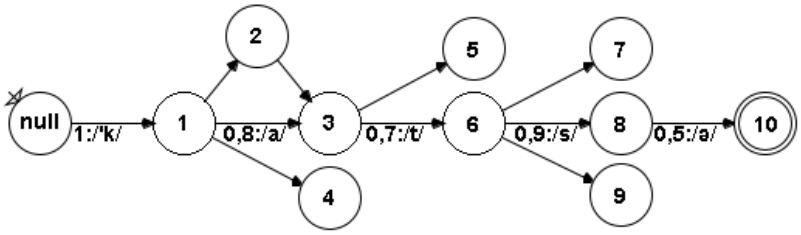

Abbildung 3.5: Ein Ausschnitt eines WSFT. Die Kantenbeschriftungen setzten sich aus der Übergangswahrscheinlichkeit und dem Phonem zusammen.

3.3.5 Online- und Offline-Dekodierung

Bei der Spracherkennung wird zwischen Online- und Offline-Dekodierung unterschieden. Bei der Offline-Dekodierung geht der Dekodierer davon aus, dass das Audiomaterial zum Beginn der Dekodierung komplett vorliegt. Das können zum Beispiel Audiodateien oder Videos sein. Bei der Online-Dekodierung erhält der Dekodierer das Audiomaterial des kontinuierlichen Signals in kleinen Chunks (dt. Brocken) (Plátek und Jurcicek, 2014). Hierbei handelt es sich um live erzeugtes Material, dessen Laufzeit zu Beginn der Verarbeitung noch nicht feststeht. Dies tritt unter anderem bei Live-Übertragungen, Konferenzen oder Telefonaten auf, die während der Dekodierung stattfinden. Um die Latenz zwischen Audiomaterial und Dekodierung konstant zu halten, wird der Real-Time-Faktor (dt. Echtzeitfaktor) betrachtet. Der Real-Time-Faktor berechnet sich aus der Zeit für die Dekodierung und der Dauer der Rede.

$$Real\text{-}time\ Faktor = \frac{Zeit(Dekodierung(a))}{Länge(a)}$$

Für die Online-Dekodierung wird im Mittel ein Real-Time Faktor von unter Eins benötigt (Cheng et al., 2011).

Einer der Unterschiede zwischen Online- und Offline-Dekodierung liegt im unterschiedlichen Einsatz der Beam Search (dt. Strahlensuche). Die Beam Search ist ein heuristischer Suchalgorithmus, der die Nachfolger vom Zustand t mit den höchsten Wahrscheinlichkeiten in einen Suchbaum aufnimmt. Ziel ist es, den Suchraum für den Viterbi-Algorithmus zu verkleinern und damit die Rechenzeit zu minimieren (J.-S. R. Jang und Lin, 2002). Die Anzahl der Blätter pro Stufe im Suchbaum wird als Beam Width (dt. Strahlenbreite) bezeichnet und als β notiert. Umso größer die Strahlenbreite ist, desto mehr Blätter werden in jeder Stufe dem Baum hinzugefügt. Je breiter der Baum ist, umso weniger mögliche Nachfolger aus dem Suchraum werden ausgeschlossen. Das Entfernen der Zustände wird als prunen (dt. Stutzen) bezeichnet. Beam Search ist ein nicht optimaler Algorithmus und findet nicht immer den optimalen Weg (Lowerre, 1990). Sollte der Suchraum bei der Online-Dekodierung zu viele aktive Hypothesen enthalten, werden unwahrscheinliche Hypothesen gestutzt, um so den Real-Time-Faktor zu senken (Cheng et al., 2011). Die Breite und Tiefe des Suchbaums können bei der Offline-Erkennung größer gewählt werden, da die Verarbeitung nicht in Echtzeit passieren muss und auch Daten aus nachfolgenden Zeitpunkten gewählt werden können. In dieser Arbeit wird zur Verarbeitung der Daten eine Online-Spracherkennung mit der Kaldi-Software verwendet.

3.3.6 Dekodergraph

Um die Audiosignale den Sprachelementen zuzuweisen, wird ein Dekodergraph genutzt. Dieser wird durch einen Dekodierer (z.B. Viterbi-Algorithmus) gesteuert.

Der Dekodergraph ist als FSA aufgebaut und als eine Komposition aus vier Elementen zusammengesetzt, die den HCLG-Dekodergraphen bilden (Mohri et al., 2002).
H enthält die Definitionen des HMM. Eingabesymbole sind Daten über den aktuellen Zustand im HMM, das Phonem und Informationen über die Transitionen. Die Definitionen des HMM bestehen aus den Transitionswahrscheinlichkeiten des aktuellen Phonems, dem vorherigen Phonem, der Wahrscheinlichkeit des nächsten Zustandes, der Wahrscheinlichkeit einer reflexiven Transition (auf sich selbst) und dem Index des Übergangs vom aktuellen zum nächsten

Zustand (Kaldi Team, 2020). Die Ausgabesymbole sind kontextabhängige Phoneme.

C ist der dynamisch erstellte Context-FST.

Der Context-FST verarbeitet als Eingabe kontextabhängige Phoneme und wandelt sie zu kontextunabhängigen Phonemen um. So soll die Anzahl möglicher Phoneme im Kontext durch Dekontextualisierung verringert werden. Bei nachfolgenden Schritten wird so vermieden, alle Phoneme im Kontext aufzuzählen, was bei großer Kontextbreite oder einer hohen Anzahl von Phonemen aufwändig wird. Ziel ist es außerdem, dass im Training nicht enthaltene Triphone abgebildet werden können (S. J. Young et al., 1994). Triphone sind eine Folge von drei aufeinander folgenden Phonemen (Jurafsky und Martin, 2009). Ausgabesymbole des Context-FST sind kontextunabhängige Phoneme.

L enthält das Lexikon. Es erhält als Eingabesymbole Phoneme und bildet diese auf Wörter ab. So werden die Phoneme /'k/ /a/ /t/ /s/ /ə/ auf das Wort „Katze“ abgebildet.

G ist der Akzeptor und enthält das Sprachmodell (Abschnitt 3.3.4). Dieser gibt an, ob eine Eingabe akzeptiert wird oder nicht. Wenn sich nach Eingabe alle Eingabesymbole in einem akzeptierenden Zustand befinden, wird die Eingabe akzeptiert, ansonsten wird sie abgelehnt. Ein- und Ausgabesymbole sind die gleichen, die das Grammar- oder Sprach-Modell kodieren (Mohri et al., 2002).

Um zur Online-Erkennung auch FST mit großem Sprachmodell (größer als mehrere Millionen Verbindungen)(Kaldi Team, 2021b) verwenden zu können, werden Lattices generiert. Sie stellen eine Repräsentation von wahrscheinlichsten Wort-Sequenzen einer Utterance (dt. Äußerung). Sie dienen der Verringerung der zu betrachtenden Möglichkeiten durch Entfernen von unwahrscheinlichen Zustandsübergängen des HCLG (Povey, Hannemann et al., 2012).

3.4 Normalisierung

Die Normalisierung dient dazu, Wörter, Abkürzungen und Zahlen in eine normalisierte, also standardisierte Form zu bringen. Abkürzungen werden bei der Normalisierung ausgeschrieben (Adda et al., 1997). Größere Zahlen sind meist zusammengesetzt aus kleineren Zahlen. Um auch Zahlen zu erkennen, die nicht im Trainingsdatensatz enthalten waren, werden die Zahlen zerlegt (Lamel et al., 1995). So wird die Zahl 1978 in mehreren Schritten in ihre Bestandteile zerlegt. Diese können unter Berücksichtigung von Artikeln („acht und siebzig“) in Wörter umgewandelt werden. Beispiele für mögliche Regeln sind in Tabelle 3.2 dargestellt.

Wenn es mehrere Schreibweisen für eine nicht normalisierte Form gibt, die auf dieselbe normalisierte Form abgebildet wird, sind diese mit einem senkrechten Strich (|) gekennzeichnet. So wird aus 1978 -> 1000 + 978 -> 1000 + 900 + 78 -> 1000 + 900 + 70 + 8 -> eintausend neunhundert acht und siebzig (Sproat und Bedrick, 2018). Das verringert die Anzahl der möglichen Wörter und verbessert so die Erkennungsrate. Das bedeutet, dass Zahlen und Nummern ausgeschrieben werden. Im Bereich der Zahlen von 0 bis 999 muss das Netz nicht 1000 verschiedene Zahlen kennen, sondern kann die Zahl in einzelne Bereiche aufteilen.

Nicht Normalisierte Form	Normalisierte Form
1978	Eintausend Neunhundert acht und siebzig
1000	Eintausend
900	Neunhundert
z.B. | zB | z.b.	Zum Beispiel
o.Ä | oÄ	oder Ähnliches

Tabelle 3.2: Regeln zur Ersetzung von Zahlen und Abkürzungen

Um den Lesefluss zu erhöhen, werden nach der Erkennung Zahlen wieder denormalisiert. Der Prozess der Normalisierung und Denormalisierung kann mit einer kontextfreien Grammatik durchgeführt werden (Aho und Ullman, 1971).

3.5 Kaldi / Kaldi-Model-Server / PyKaldi

Kaldi ist ein unter freier Lizenz stehendes Softwarepaket zur Spracherkennung und Signalverarbeitung, das in C++ geschrieben wurde. Es bietet eine Vielzahl von Werkzeugen zur Erstellung von Features wie z.B. MFCC. Mit Kaldi können eigene Spracherkennungsmodelle trainiert und für die Dekodierung verwendet werden. Kaldi unterstützt sowohl Online- als auch Offlinespracherkennung (Povey, Ghoshal et al., 2011). Um Kaldi direkt in anderen Programmiersprachen wie Python verwenden zu können, werden Wrapper (dt. Umhüllungen) verwendet. PyKaldi ist ein Wrapper, der Kaldi-Methoden direkt in Python-Syntax anbietet (Can et al., 2018).

Zur Verwendung mit Python und REDIS wird PyKaldi zusammen mit dem Kaldi-Model-Server verwendet.

Der Kaldi-Model-Server (kurz KMS) ist ein Python-Programm, das es ermöglicht, Kaldi per Mikrofon oder REDIS-Pubsub-Channel Audiodaten zur Verarbeitung zu übertragen und die Ergebnisse auch an einen REDIS-Pubsub-Channel auszugeben.[2]

REDIS ist eine unter freier Lizenz stehende nicht-relationale In-Memory-Datenbank. In-Memory-Datenbanken nutzen als Speicherort den Arbeitsspeicher und nicht Festplatten oder SSD-Speicherplatz (Garcia-Molina und Salem, 1992).

REDIS bietet die Möglichkeit, Informationen sowohl als Key-Value-Pair (dt. Schlüssel-Wert-Paar) zu speichern als auch Informationen über die Pubsub-Schnittstelle mit dem Publish-and-Subscribe-Paradigma zu verteilen. Hierbei können die Kanäle abonniert (eng. subscribe) und die Informationen des Kanals empfangen werden (Birman, 1993). Dies ist über Programmiersprachen hinweg möglich.

Der in dieser Arbeit entwickelte Prototyp zur automatischen Untertitelung wurde in Python geschrieben. Hier wurde PyKaldi verwendet, weil es eine einfache Integration von Kaldi in Python bietet. Sowohl die interne Kommunikation in BigBlueButton als auch die Kommunikation der einzelnen Bestandteile des Prototyps wird mit REDIS verarbeitet.

[2] `https://github.com/uhh-lt/kaldi-model-server`

3.6 BigBlueButton

BigBlueButton[3] (kurz BBB) ist ein quelloffenes und kostenlos verfügbares Videokonferenz-System. Es bietet die Möglichkeit der Integration in Lern- und Inhaltsverwaltungssysteme wie zum Beispiel Moodle, OpenOlat oder NextCloud (BigBlueButton Inc., 2021a). Konferenzteilnehmer können hier in Konferenzräumen per Audio, Video und Chat kommunizieren. BBB unterstützt das Teilen des Bildschirminhalts, das Hochladen von Präsentationen oder die gemeinsame Nutzung eines Whiteboards zum gemeinschaftlichen Skizzieren von Ideen und Inhalten. Außerdem ist BBB mit Etherpads ausgestattet, die das gleichzeitige Bearbeiten von Texten ermöglichen und für händische Untertitelung der Konferenz genutzt werden können.

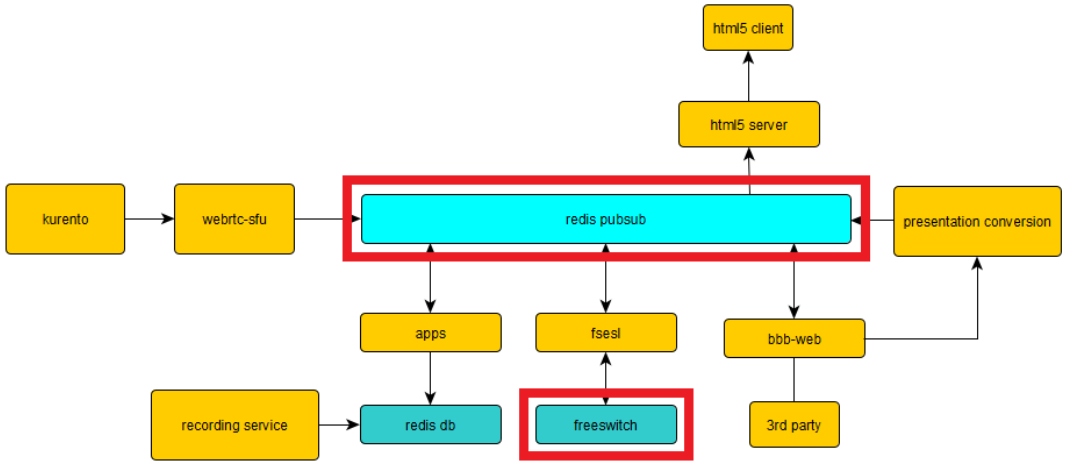

Abbildung 3.6: Architektur von BBB(BigBlueButton Inc., 2021d). Elemente mit besonderer Bedeutung für diese Arbeit sind rot umrandet.

Genutzt werden kann BBB mit einem PC, Notebook, Tablet, Smartphone oder per Telefon. Die Audio- und Videoübertragung wird über eine WebRTC Verbindung aufgebaut. WebRTC ermöglicht es, Audio- und Videoübertragungen zwischen Frontend und Browser des Nutzers durchzuführen (BigBlueButton Inc., 2021e). BBB besteht aus verschiedenen Modulen (siehe Abbildung 3.6), die untereinander kommunizieren (BigBlueButton Inc., 2021f).

In dieser Arbeit liegt der Fokus auf folgenden Bestandteilen der Software: FreeSWITCH als Telekommunikationslösung, Greenlight für die Client-Oberfläche und REDIS bzw. MongoDB als Datenbank zur Verarbeitung und Verbindung der Komponenten.

FreeSWITCH ist eine kostenfreie und quelloffene Telefonanlagen-Software[4]. FreeSWITCH wird für die Audioverwaltung der Konferenzen und die Verbindung zwischen den Nutzern eingesetzt. Sie ist modular um Audio-Codecs und Funktionen erweiterbar. Außerdem bietet

[3] https://bigbluebutton.org/
[4] https://freeswitch.org/confluence/

FreeSWITCH die Möglichkeit, das Audiosignal einzelner Teilnehmer oder einer Konferenz aufzuzeichnen. FreeSWITCH mischt bei Konferenzen die Audiosignale aller Nutzer neu ab und sendet den jeweiligen Benutzern das abgemischte Audiosignal der anderen Benutzer zu (SignalWire Inc., 2020).

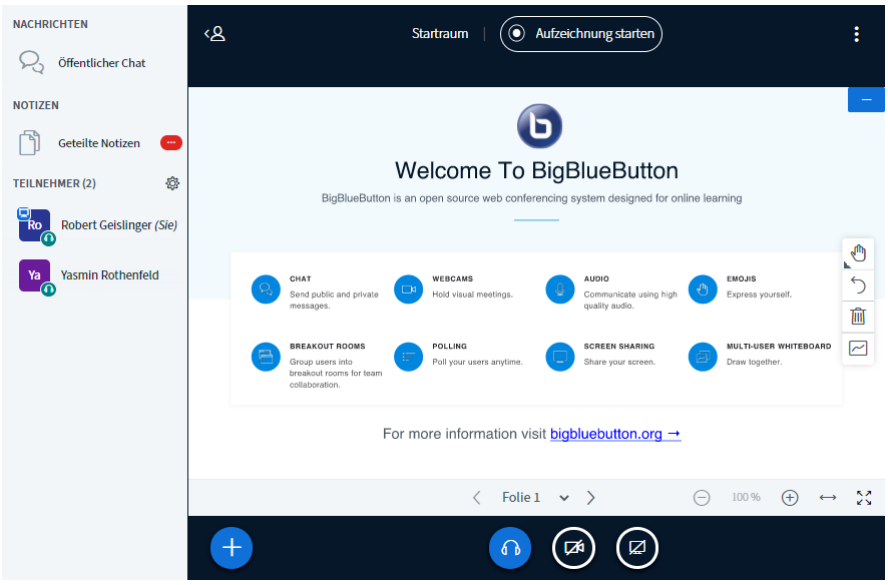

Abbildung 3.7: BigBlueButton Konferenz

MongoDB ist ein dokumentenorientiertes NoSQL-Datenbanksystem. In BBB wird MongoDB hauptsächlich für die Verwaltung der Untertitel, der Chatfunktion und der Etherpads verwendet. Zur Kommunikation zwischen den Modulen wird die REDIS-Datenbank verwendet. Die verwendeten Module sind in Abbildung 3.6 rot umrandet.

Greenlight bildet das Frontend bzw. die Oberfläche von BBB. Es ermöglicht sowohl Konferenzräume zu erstellen und zu verwalten als auch Aufzeichnungen anzusehen oder zu löschen. Es bildet die Schnittstelle zwischen den Teilnehmern und dem BBB-Server (BigBlueButton Inc., 2021b). Eine Beispiel-Konferenz ist in Abbildung 3.7 dargestellt.

3.7 Audio-Codec

Ein Codec bezeichnet eine Software oder Schaltkreise, die Daten oder Signale in ein neues Format kodieren und dekodieren. Ein Audio-Codec besteht hauptsächlich aus zwei Computerprogrammen, dem Kodierer und dem Dekodierer. Während der Kodierer die Signale in das gewünschte Format kodiert, wird der Dekodierer genutzt, um die Daten wieder abzuspielen (Cutler, 1952). Bei Codecs wird generell zwischen verlustfreien und verlustbehafteten Codecs unterschieden. Verlustfreie Codecs verringern den Speicherplatz bzw. die Bandbreite bei der Datenübertragung und können ohne Qualitätsverlust wieder in das ursprüngliche Signal umgewandelt werden. Verlustbehaftete Codecs verändern das Signal, sodass eine Rekonstruktion

in das Ursprungsformat nicht ohne Informationsverlust möglich ist (Sayood, 2002).

Eine der bekanntesten Audio-Codecs ist der MP3-Codec. Der verwendete Audio-Codec in BBB ist der quelloffene und kostenfreie Opus-Codec. Opus ist ein verlustbehaftetes Audiokompressionsformat, das für interaktive Echtzeitkommunikation über das Internet ausgelegt ist. Opus wird von der IETF (Internet Engineering Task Force, dt. Internettechnik-Arbeitsgruppe) im RFC 6716 definiert und als Standard für verlustbehaftete Audiodatenkomprimierung im Internet empfohlen (Wikimedia Commons, 2020).

3.8 Wortfehlerrate

Die WER (Word Error Rate, dt. Wortfehlerrate) bezeichnet eine gängige Metrik, um die Performance eines ASR-Systems zu bewerten. Zur Berechnung der Wortfehlerrate wird die Transkription der Aufzeichnung mit der Ausgabe des ASR-Systems verglichen. Die Ausgabe des ASR-Systems wird kategorisiert nach den korrekten Wörtern (C) und den fehlerhaften Wörtern, die in drei Kategorien aufgeteilt werden: Substitutionen (S), entfernte Wörter (D) und zusätzlich eingefügte Wörter (I). Substitutionen sind Wörter, die fehlerhaft vom System erkannt und durch ein anderes Wort ersetzt wurden (Huang et al., 2001). Beispiel:
Der Satz aus der Transkription lautet „Das Faultier krault die kleine Katze“.
Das ASR-System erkennt aus dem Audiomaterial „Das Faultier kaut die Katze ausgiebig“ Substitutionen sind im Beispiel blau, entfernte Wörter rot und zusätzliche Wörter grün markiert. Eine niedrigere WER zeigt eine bessere Erkennungsrate an.

$$\text{WER} = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C}$$

4 Durchführung

In diesem Kapitel wird die bestehende Lösung analysiert, mögliche Ansatzpunkte zum Kopieren einzelner Audiostreams, die Prototypen und deren Entwicklung so wie die notwendigen Erweiterungen von BigBlueButton besprochen. Außerdem werden die ausgeführten Experimente und deren Aufbau erläutert.

BBB enthält bereits in der Version 2.2 eine Möglichkeit, Untertitel anzuzeigen. Die Untertitel der Teilnehmer können während der Konferenz vom Moderator aktiviert werden und über die Benutzeroberfläche von allen Teilnehmern geschrieben und angezeigt werden. Eine automatische Untertitelung ist bei BBB nicht implementiert. Ein Beispiel für Untertitel in einer Konferenz ist in Abbildung 4.1 dargestellt.

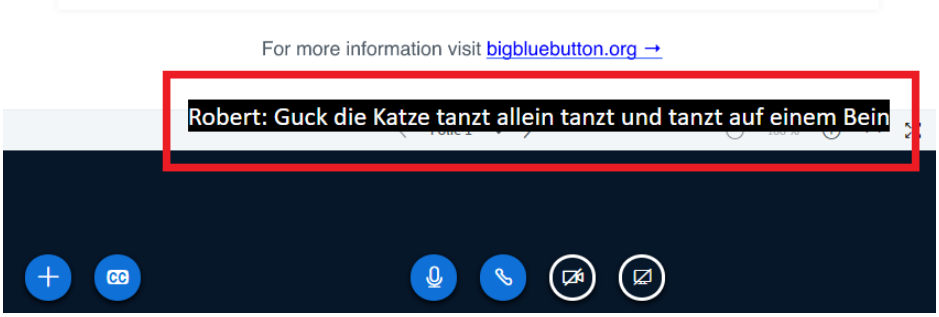

Abbildung 4.1: Untertitel in einer BBB Konferenz (roter Rahmen)

4.1 Bestehende Lösung

Eine erste Live-Untertitelung für BBB wurde im Sommersemester 2020 an der Universität Hamburg im Rahmen des Masterprojekts „Web Interfaces for NLP SS2020“ erstellt. Ziel des Projekts war es, eine Untertitelung für BBB zu entwickeln. Der Prototyp wurde in der Programmiersprache Golang geschrieben und verwendet zur Verbindung mit der Konferenz einen VoIP-Client. Dieser läuft im Hintergrund, verbindet sich beim Erstellen einer neuen Konferenz und erscheint als zusätzlicher Benutzer. Für jede Konferenz wird eine Kaldi-Model-Server-Instanz (kurz KMS-Instanz) gestartet, welche ihr Audiomaterial zur Erkennung über den VoIP-Client erhält. Das bedeutet, dass sich alle Teilnehmer eine Instanz teilen. Eine Instanz beinhaltet ein vortrainiertes Modell sowie auch alle Parameter. In den Audiodaten aus der Konferenz sind neben dem abgemischten Audiosignal aller Teilnehmer auch die akustischen Statusmeldungen aus der Konferenz enthalten. Dazu gehören Mitteilungen wie der Beitritt oder das Verlassen von Teilnehmern der Konferenz. Zwischen KMS und der Software werden

die Daten über einen REDIS-Pubsub Channel geteilt (Wille, 2020). Eine Sprecheradaption ist in diesem Prototyp nicht möglich, da die Sprechertrennung durch das abgemischte Signal verloren geht. Das Modell kann sich während eines Gesprächs mit mehreren Sprechern beim Dekodieren nicht auf jeden Sprecher individuell einstellen. Bei der Verarbeitung des Audiosignals kann es außerdem zu Problemen kommen, wenn sich Teilnehmer mit ihren Wortmeldungen zeitlich überschneiden. Dabei kommt es zum Übersprechen. Diese Faktoren können die Fehlerrate steigern, weil die Audiosignale falsch interpretiert werden können (Wrigley et al., 2004). Experimente mit der bestehenden Lösung sind leider nur eingeschränkt möglich: Die Verbindung zwischen Konferenz und VoIP-Client bricht nach etwa 20 Sekunden Laufzeit zusammen und verbindet sich einige Male nach kurzer Zeit wieder. Dies macht die Lösung in der Praxis nur eingeschränkt nutzbar und es würde einen großen Aufwand bedeuten, die WER zu berechnen. Daher wurde für diese Arbeit ein neuer Prototyp entwickelt.

Architektur, Ideen und Erkenntnisse der bestehenden Lösung wurden bei der Entwicklung eines neuen Prototyps berücksichtigt. Um die angesprochenen Nachteile der bestehenden Lösung auszuräumen, wurde evaluiert, wie die Audiosignale der verschiedenen Teilnehmer kopiert werden können, um eine stabile und gleichzeitig anwenderfreundliche Lösung zu entwickeln. Als erster Schritt dahin wurde betrachtet, ob das Audiosignal auf Client- oder Serverseite kopiert werden soll, um es später zu verarbeiten. Nachfolgend werden Vor- und Nachteile des jeweiligen Ansatzes besprochen.

4.2 Client- und Server-Lösung

Es gibt zwei grundsätzliche Möglichkeiten, die Audiosignale der Konferenz bzw. der Teilnehmer in Echtzeit für die Verarbeitung zu erhalten. Eine Möglichkeit ist die Client-Lösung, die eine Schnittstelle beim Nutzer voraussetzt, um das Signal lokal beim Teilnehmer abzugreifen. Diese Schnittstelle kann entweder per Software (z.B. Aufzeichnungssoftware) oder Hardware (z.B. Y-Kabel) realisiert werden. Bei einer Softwarelösung kann entweder auf bestehende Programme zurückgegriffen werden oder es müssen eigene Lösungen entwickelt werden.

Der Vorteil einer Clientlösung ist die bessere Audioqualität, da das Signal vor der Verarbeitung durch Audio-Codec oder Transportprotokoll abgegriffen werden kann. Die Latenz zwischen Mikrofon und ASR-System kann bei einer Clientlösung niedriger sein, wenn die weitere Verarbeitung auch lokal auf dem Teilnehmer-Endgerät ausgeführt wird. Die Verarbeitung wäre auch bei einer langsamen oder schwankenden Internetverbindung weiterhin möglich.

Ein Nachteil einer Clientlösung ist, dass die Software mit Geräten verschiedener Bauarten wie Desktop oder mobilen Geräten, verschiedenen Browsern als auch mit den gängigsten Betriebssystemen kompatibel sein muss, um eine Vielzahl an Benutzern und deren individuelle Konfiguration abbilden zu können. Für jede Kombination aus Betriebssystem (Windows, Android, Linux, iOS, WebOS), Browser (Google Chrome, Mozilla Firefox, Opera) und Hardware (Smartphone, Notebook, Tablet) muss eine eigene Software entwickelt, getestet und gepflegt werden. Die Pflege der Schnittstelle kann von Änderungen am Browser, wie einer neuen Version, oder auch von Änderungen am Betriebssystem, wie einer neuen Betriebssystemversion, neuen Treibern oder neuen Softwarevoraussetzungen abhängen. Das würde einen erheblichen

Aufwand bedeuten, da die Anzahl der möglichen Kombinationen mit jedem weiteren unterstützten Browser oder Betriebssystem steigt (Van Genuchten, 1991; Hammershøj et al., 2010).

Eine zweite Möglichkeit ist die Serverlösung. Hier werden die Daten direkt im Server kopiert und verarbeitet. Der Vorteil dieser Lösung ist die Unabhängigkeit von verwendeten Clients, da das Signal immer an derselben Stelle mit denselben Einstellungen anliegt. Dies bedeutet, dass die Lösung sowohl für mobile Geräte als auch für Desktops verwendet werden kann, ohne dass spezielle Anpassungen für eine Geräteklasse oder einen Browser vorgenommen werden müssen (Dhillon et al., 2018). Die Wartung und Weiterentwicklung über längere Zeit bleibt so übersichtlicher und ist weniger zeitintensiv. Der Nutzer muss auch keine Änderungen an seinem eigenen System vornehmen. Nur der BigBlueButton-Administrator muss bei Änderungen diese einmalig einpflegen.

Ein Nachteil der Serverlösung ist die höhere Latenz bei der Verarbeitung auf dem Server. Das Audiosignal muss über das Internet an den Server übertragen werden und kann dann erst dort verarbeitet werden (Cáceres und Chafe, 2010). Außerdem kann der Audio-Codec die Audioqualität negativ beeinflussen (Łalovarda et al., 2004). Weiterhin funktioniert diese Lösung nur, wenn der Sprecher eine stabile Internetverbindung besitzt (Cáceres und Chafe, 2010).

In der vorliegenden Arbeit wird auf eine Serverlösung gesetzt, um bei geringem Wartungsaufwand Teilnehmern aller Geräteklassen eine einheitliche Lösung bieten zu können.

4.3 Änderungen an BigBlueButton

Da BigBlueButton nur die Möglichkeit bietet, händisch Untertitel zu erstellen, jedoch keine Möglichkeit zur automatischen Erstellung von Untertiteln, mussten dafür Änderungen vorgenommen werden. Dabei beschränkten sich die Anpassungen hauptsächlich auf die Konfigurationsdateien von FreeSWITCH. Das Ziel war es, auf Serverseite eine stabile Audioquelle der Teilnehmer zu erhalten. Weiterhin sollte erreicht werden, dass nicht das abgemischte Audiosignal der Konferenz, sondern die Audiosignale aller Teilnehmer getrennt voneinander verarbeitet werden, um bei einer möglichen Übersprechung einen Einbruch der Erkennungsrate zu verhindern. Um dies zu erreichen, wurden verschiedene Methoden getestet und die jeweiligen Schritte sowie die möglichen Verbesserungen einzeln dokumentiert und besprochen.

4.4 Erster Prototyp

In einem ersten Prototyp wurde die Methode record_session[1] verwendet. Sie ermöglicht es, eine Aufnahme jedes Audiosignals unabhängig voneinander anzufertigen (SignalWire Inc., 2018). Um die Aufnahme beim Betreten der Konferenz zu starten, wurden Anpassungen an FreeSWITCH vorgenommen. Beim Betreten von Konferenzen durchläuft jeder Nutzer Dialplans (dt. Wählpläne). Dialplans werden von FreeSWITCH genutzt, um den Ablauf beim

[1] `https://FreeSWITCH.org/confluence/display/FreeSWITCH/mod_dptools%3A+record_session`

Betreten der Konferenz festzulegen und Einstellungen wie Audioqualität zu setzen. Beim Beitreten einer BBB-Konferenz wird der Nutzer über drei Dialplans in die Konferenz geleitet. Der erste Dialplan startet für den Nutzer einen Selbsttest. Hier kann er Mikrofon und Lautsprecher vor Beitritt zur Konferenz auf Funktion und Sprachqualität testen. Der zweite Dialplan übergibt Parameter zur Audioqualität und der dritte baut die Verbindung zur Konferenz auf (BigBlueButton Inc., 2021c). Im zweiten Dialplan wurden für den Prototyp Änderungen vorgenommen, um während des Beitritts zur Konferenz die Aufzeichnung zu starten. Bei der Kommunikation mit FreeSWITCH wird zwischen zwei call legs (dt. Anrufverbindungen) unterschieden: A leg und B leg. Aus Sicht der Konferenz bezeichnet A leg die eingehende Verbindung vom Anrufer zur Konferenz und B leg die ausgehende Verbindung zum Anrufer (SignalWire Inc., 2019a). Für die Aufzeichnung in dieser Arbeit wird das A leg verwendet. Damit erhält der Prototyp zur Verarbeitung ausschließlich das Audiosignal der Nutzer, welches in die Konferenz kommt. Die Aufzeichnung wird in eine Datei auf den BBB-Server geschrieben und durch ein Modul des Prototyps bei Änderungen gelesen. Änderungen an der Datei werden laufend per pubsub-Channel über die REDIS Datenbank gesendet und per KMS mit PyKaldi verarbeitet. Es wird während der Konferenz für jeden Benutzer eine separate Aufzeichnung auf die Festplatte/SSD geschrieben. Der benötigte Speicherplatz und die Speicherbandbreite steigen bei mehreren Benutzern und Konferenzen an. Ein Flaschenhals entsteht hier also bei dem Datendurchsatz, der Größe der Speichermedien und den Speichercontrollern.

4.5 Zweiter Prototyp

In einem zweiten Prototyp wurde auf eine Speicherung der Audiodaten verzichtet. Hier wurde stattdessen mit dem zusätzlichen Modul „mod_audio_fork“ von drachtio[2] FreeSWITCH um die Möglichkeit eines WebSocket-Clients erweitert. Das Modul basiert auf dem WebSocket-Protokoll und sendet die Audiodaten für jeden Benutzer über eine WebSocket-Verbindung an einen Server. WebSocket ist ein Netzwerkprotokoll, das auf TCP basiert und zur bidirektionalen Verbindung zwischen Server und Webanwendung genutzt werden kann (Fette und Melnikov, 2011). Der WebSocket-Server wird vom Prototyp bereitgestellt. Dieses Modul in FreeSWITCH zu nutzen, hat im Vergleich zur Aufzeichnung als Datei mehrere Vorteile: Es werden mit dem Modul keine Daten auf den Server-Festplatten/SSD gespeichert, sondern nur im Arbeitsspeicher des Servers, was das Altern der Speichermedien durch den Wegfall der Schreib- und Lesevorgänge verringert und performanter ist. Es eröffnet außerdem die Möglichkeit, den Prototypen in der Zukunft um eine bedarfsgerechte Untertitelung zu erweitern. Hierbei wäre es möglich, nur einzelne Teilnehmer - je nach Bedarf - mit einer Untertitelung auszustatten und nicht die komplette Konferenz. Das könnte helfen, den beschränkten Platz zur Anzeige von Untertiteln sinnvoll zu nutzen. Vor allem, wenn nur geringe Ressourcen zur Verfügung stehen, bietet sich diese Vorgehensweise an. Außerdem wäre es dann auch möglich, die Untertitelung für einzelne Benutzer im Laufe der Konferenz ein- und auszuschalten. Der Prototyp kann für mehrere Konferenzen gleichzeitig verwendet werden und die zu untertitelnde Teilnehmerzahl richtet sich nach der Anzahl der zur Verfügung stehenden Prozessorkerne.

[2] `https://github.com/drachtio/drachtio-FreeSWITCH-modules/tree/master/modules/mod_audio_fork`

Bei beiden vorgestellten Prototypen handelt es sich um Serverlösungen.

Im Vorfeld für einen Praxistext wurde das Sprachmodell neu trainiert und das Vokabular manuell um Wörter erweitert. Zusätzlich zu Quellen mit Bezug zu BigBlueButton wurden dazu die aktuellen Daten der bereits im Vorfeld verwendeten Datenquellen genutzt. Dies sollte in einem Praxisumfeld eine bessere Erkennung erreichen, um auch aktuelle Wörter sauber erkennen zu können. Ziel war hier, eine hohe Anwenderfreundlichkeit zu erreichen. Beim Praxistest waren zehn Teilnehmer in der Konferenz und es zeigte sich, dass weitere Anpassungen nötig waren, um auch bei Konferenzen mittlerer Größe eine geringe Latenz bieten zu können. Es stellte sich heraus, dass die Verarbeitung zwar stabil lief, sich aber eine steigende Latenz für die Untertitelung aufbaute. Nach etwa 30 Minuten in der Konferenz betrug die Latenz bereits zehn Minuten. Der hierfür Grund war die Verarbeitung der Daten aller Teilnehmer während ihrer kompletten Teilnahme an der Konferenz unabhängig davon, wie viele Teilnehmer gerade sprachen. So stieg die Last für den REDIS-Server und den Server mit den KMS-Instanzen mit jedem Teilnehmer, was darauf hinauslief, dass die Latenz für alle Teilnehmer im Laufe der Konferenz immer weiter stieg. Um auch bei Konferenzen mit mehr als fünf Teilnehmern eine geringe Latenz der Verarbeitung zu gewährleisten, wurden Änderungen am Websocket-Server vorgenommen. FreeSWITCH sendet über einen REDIS-Channel Informationen, ob ein Nutzer gerade spricht oder nicht. Dazu wird der Boolean-Wert „isTalking“ gelesen, der repräsentiert ob ein Nutzer spricht. Wenn ein Nutzer als „sprechend“ markiert ist, wird sein Audiosignal in seinen dazugehörigen Kanal übertragen. Beendet der Teilnehmer seine Wortmeldung, setzt FreeSWITCH den Wert von „isTalking“ auf False. Dann werden noch eine kurze lang Zeit weiter Daten gesendet (BigBlueButton Inc., 2015). So kann Kaldi die Erkennung der Utterance sauber finalisieren.

4.6 Weitere Verarbeitungsschritte

Bei den weiteren Verarbeitungsschritten unterscheiden sich die beiden Prototypen nicht. Die Verbindung zu FreeSWITCH baut das Programm „esl_to_redis“ auf und greift dafür auf die FreeSWITCH-API (Application Programming Interface, dt. Programmierschnittstelle) ESL zu. ESL (Event Socket Layer, dt. Ereignis Socket Schicht) ermöglicht die Überwachung von FreeSWITCH-internen Ereignissen wie zum Beispiel den Beitritt eines Teilnehmers in die Konferenz und das Auslösen von Aktionen, wie den Start der Aufnahme mit Plugins. Über die API erhält das Skript die Information, wenn ein Benutzer die Konferenz betritt und sendet diese an den Informations-Channel (SignalWire Inc., 2019b). Beim Informations-Channel handelt es sich um einen REDIS-Channel, den alle Skripte des Prototyps abonnieren und worüber sie Statusinformationen wie den Beginn einer Audioübertragung oder das Ende einer Verarbeitung miteinander austauschen. Wenn ein Teilnehmer die Konferenz betritt, werden seine Meta-Informationen wie Name, ID und Konferenznummer an den Informationschannel gesendet. Der WebSocket-Server erhält von FreeSWITCH im gleichen Schritt eine Datenübertragung über das WebSocket-Modul „ws_receiver“. Dieser verarbeitet das Audiomaterial wie in Abschnitt 4.5 beschrieben und sendet es an den individuellen Audiochannel des Teilnehmers. In der Architektur der Prototypen werden neben dem Informations-Channel drei weitere REDIS-Channel unterschieden: Audio, Text und Kontrolle. Der Name des jeweiligen

Channels setzt sich aus dem Namen des Teilnehmers in der Konferenz, der Konferenznummer und dem Einsatzzweck des Channels (Audio, Text, Kontrolle) zusammen. Für Teilnehmer werden diese Channel erstellt. Der Audiochannel wird genutzt, um das vom WebSocket-Server empfangene Audiomaterial des Teilnehmers im Format 16bit, Mono unkomprimiert dem KMS zur Verfügung zu stellen. Am KMS wurden zudem Änderungen vorgenommen, um KMS und REDIS-Server auch auf zwei voneinander getrennten Servern nutzen zu können.[3] Wenn auf dem Informations-Channel Information über einen neuen Audiochannel gesendet wird, startet das Skript „kaldi_starter“ eine neue KMS-Instanz und verbindet diese mit den drei Channels des Teilnehmers. Der Textchannel dient als Rückkanal vom KMS und beinhaltet die erkannten Utterances.

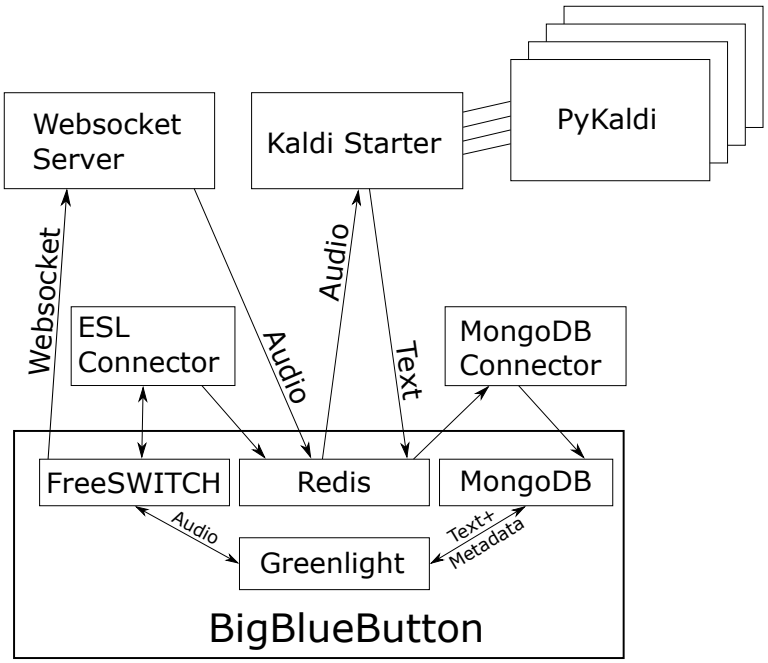

Abbildung 4.2: Architektur des zweiten Prototyps

Das Skript „mongodbconnector“ abonniert den Textchannel und bereitet die Utterances zu Untertiteln auf. Hierbei werden doppelte Leerzeichen und Hesitationen, sowie Wörter, die als nicht erkannt markiert wurden, entfernt. Dies soll den Lesefluss der finalen Untertitel verbessern. Außerdem wird den Untertiteln der Name des Sprechers vorangestellt. Wenn ein Teilnehmer die Konferenz wieder verlässt oder seine Audioverbindung trennt, wird die Information vom „esl_to_redis“ in den Informations-Channel geschrieben. Der „kaldi_starter“ beendet daraufhin die zugehörige KMS-Instanz.

Wenn der Moderator der Konferenz die integrierte Untertitelfunktion aktiviert hat, können sich die Teilnehmer die Untertitel auf ihrem Endgerät anzeigen lassen. In Abbildung 4.1 sind eingeschaltete Untertitel in einer BBB-Konferenz dargestellt.

[3] `https://github.com/uhh-lt/kaldi-model-server/commit/d99283eea2e950c0cbe9eee68418567b961b58cb`

Die Architektur des zweiten Prototyps ist in der Abbildung 4.2 mit den jeweiligen Datenverbindungen abgebildet.

4.7 Abtastratenprobleme mit FreeSWITCH

Bei der Verwendung unterschiedlicher Browser kam es aus nicht offensichtlichen Gründen immer wieder zu massiven Einbrüchen der Erkennungsrate. Die WER stieg dabei auf 100% und eine Nutzung der Audiodaten war praktisch nicht möglich. Diese Problematik bestand bei beiden Prototypen, so dass zumindest ein Zusammenhang mit der Aufzeichnung bzw. dem WebSocket-Client als unwahrscheinlich eingestuft werden konnte. Es zeigte sich nach vielen Tests, dass die Verarbeitung je nach gewählter Einstellung entweder mit Mozilla Firefox-Browsern[4] oder mit Google Chrome-Browsern[5] funktionierte, aber nicht mit beiden Browsern mit derselben Einstellung. Um das Audiomaterial mit Kaldi verarbeiten zu können, muss es in 16 kHz Einkanal vorliegen (Kaldi Team, 2021a). Diese Werte werden per Konfiguration in den FreeSWITCH-Modulen eingestellt. Unter Verwendung des Google Chrome-Browsers war eine Erkennung und damit Untertitelung möglich, wenn die Frequenz auf 16 kHz eingestellt war. Wenn diese Einstellung mit dem Firefox-Browser verwendet wurde, verdoppelte sich die Frequenz aufgrund eines Fehlers im Modul auf 32 kHz. Der Fehler betraf das Opus Modul in FreeSWITCH. Dieses erweitert FreeSWITCH um den Audio-Codec Opus und wird bei der WebRTC Verbindung zwischen Browser und Konferenz eingesetzt. Die Aufzeichnungen bei der Verwendung von Google Chrome wiesen keine Veränderung der eingestellten Frequenz auf. Dementsprechend konnte mit dem Mozilla Firefox-Browser bei einer angepassten Konfiguration mit 8 kHz ein nutzbares Ergebnis erzielt werden. Nach Kontakt mit Entwicklern von FreeSWITCH und Besprechung von Details konnte der Fehler reproduziert und auch korrigiert werden.[6] Der Fehler wird in der kommenden FreeSWITCH Version behoben.

4.8 Datensätze

Die verwendeten Datensätze, mit denen die WER des Prototyps evaluiert wurde, sind die Verbmobil-Datensätze. Sie bestehen aus Verbmobil-I (kurz VM1) und Verbmobil-II (kurz VM2). Sie enthalten 1454 deutsche Dialoge mit Terminvereinbarungen von über 400 Sprecherinnen und Sprechern aus verschiedenen Regionen Deutschlands (Burger et al., 2000). Die aufgezeichneten Konversationen enthalten Planungen und Terminfindungen zu Reisen und die Vereinbarungen von Treffen (Hess et al., 1995). Das Vokabular besteht aus Daten, Uhrzeiten, Vor- und Nachnamen, Tagen, Monaten, Nummern und Orten. Beide Datensätze enthalten zusammen 96 Stunden gesprochene Sprache als Spontansprache (Soltau et al., 2001). Das Audiomaterial wurde mit verschiedenen Mikrofonen mit einer Samplerate von 16 kHz und 16 Bit Tiefe aufgenommen (Burger et al., 2000). Alle Gespräche in den Datensätzen sind transkribiert und liegen zusätzlich im SAMPA-Format vor (Jekat et al., 1995; Alexandersson

[4] `https://www.firefox.com`
[5] `https://www.google.de/chrome/`
[6] `https://github.com/signalwire/freeswitch/issues/1112`

et al., 1997). Der VM2-Datensatz liegt sowohl als ungeschnittene Variante, in der beiden Gesprächspartner ein vollständiges Gespräch führen, als auch in kurzen Abschnitten vor (Schiel, 2003).

4.8.1 Aufbereitung vollständiger Gespräche

Für den VM2-Datensatz lagen die Gespräche zusätzlich zu den nur mehrere Sekunden langen Abschnitten auch in ungeschnittenen mehrere Minuten langen Format vor - genannt Full turns (dt. ganze Durchgänge). Bei den Aufnahmen saßen beide Sprecher zur selben Zeit mit jeweils einem eigenen Mikrofon im selben Raum. Jedes Mikrofon liefert eine Tonspur, die einzeln als eine eigene Audiodatei verfügbar ist. Beide Sprecher sind durch die Raumakustik und den Versuchsaufbau auf beiden Aufnahmen klar und deutlich zu hören. Bei einem ersten Experiment mit den vollständigen Gesprächen erreichte der Aufbau eine ungewöhnlich hohe WER von 50-80% je nach Sprecher. Der Grund dafür war, dass beide Sprecher vom ASR-System als ein Sprecher erkannt wurden, da beide Sprecher eine ähnliche Lautstärke aufwiesen. Die Daten mussten für die Experimente noch manuell angepasst werden.

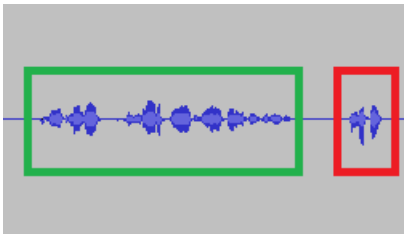

Abbildung 4.3: Das grüne Rechteck ist Sprecher 2, das Rote Sprecher 1. Beide Sprecher haben eine ähnliche Lautstärke.

Eine automatische Filterung, um den weiter im Raum entfernten zweiten Sprecher zu entfernen, brachte keine zufriedenstellenden Ergebnisse. Der Lautstärkeunterschied zwischen den Sprechern war nicht immer ausreichend als dass eine einfache Filterung zweifelsfrei zwischen ihnen unterscheiden konnte. Bei der manuellen Bearbeitung mit Audacity[7] konnte im Spektrogram nicht anhand der Amplitude eine zweifelsfreie Unterscheidung zwischen beiden Sprechern getroffen werden. In Abbildung 4.3 ist ein Tonabschnitt abgebildet, der beide Sprecher enthält. Die Amplitude beider Wortmeldungen ist auf der Tonspur ähnlich ausgeprägt. Ein Grund für die ähnliche Lautstärke können die Sprachmuster der Sprecher oder die Raumakustik im Aufnahmeraum sein. So unterschieden sich die Sprecher durch eine lautere oder deutlichere Aussprache. Viele Sprecher senkten im Laufe des Gespräches zunehmend ihre Stimme.

[7] https://www.audacityteam.org/

4.9 Experimentaufbau

Es werden mehrere Experimente miteinander verglichen. Als Referenz werden VM1 und VM2 in Kaldi mit Sprecherdaten offline dekodiert. Hier wird Kaldi ohne PyKaldi verwendet. Die Offline-Dekodierung bietet eine geringere WER und soll so als Vergleich zu Experimenten mit Online-Dekodierung und über BBB dienen. Um eine Vergleichbarkeit zwischen Online- und Offline-Dekodierung herzustellen, werden bei der Offline-Dekodierung Experimente sowohl mit als auch ohne Sprecherdaten durchgeführt. Beim Aufbau mit Sprecherdaten wird dem ASR-System zur Sprecheradaption die Information übermittelt, welcher Sprecher welche Audiodateien gesprochen hat.

Um eine Referenz für die Online-Dekodierung zu erhalten, werden die Datensätze in einem weiteren Experiment direkt mit dem KMS verarbeitet. Die Daten werden ohne Sprecherdaten über einen REDIS-Channel an den KMS gesendet und verarbeitet. Die Daten liegen auf dem REDIS-Server und der KMS-Server ist über das lokale Netzwerk verbunden. Bei der Verwendung von BBB wird ausschließlich Online-Dekodierung verwendet, so dass hier der Einfluss von BBB auf die WER gemessen werden kann.

Für ein weiteres Experiment werden die Daten mit dem Opus-Codec unter Verwendung der freien Software FFMPEG[8] kodiert. Es werden dazu die Dateien mit Opus kodiert und wieder zurück in ihr ursprüngliches Format zurückkodiert. Die Parametrisierung des Codecs entspricht der in BBB verwendeten und wird im Anhang aufgeführt. Übertragen werden die Daten per REDIS-Channel an den KMS über das lokale Netzwerk. Es werden dabei keine Sprecherdaten verwendet.

Um die Auswirkungen von BBB auf die Ergebnisse zu testen, werden die Daten über eine Konferenz verarbeitet. Dafür wird mit dem Google Chrome-Browser eine Verbindung zu einer Konferenz aufgebaut. Die Verbindung zwischen Client und Server wurde über das Internet aufgebaut. Die Daten werden mit dem VLC Multimediaplayer[9] abgespielt und direkt als softwareseitiger Mikrofoneingang verwendet. Der Browser überträgt die Daten per WebRTC in die Konferenz. In diesem Experiment werden keine Sprecherdaten verwendet.

Zur Verarbeitung der Full turns in der BBB-Konferenz wurde jeder Sprecher als eigener Teilnehmer behandelt. Für die zwölf Sprecher des Datensatzes wurde jeweils eine Verbindung aufgebaut und die Daten in VLC abgespielt. So konnte sichergestellt werden, dass die Sprecherdaten erhalten bleiben.

4.9.1 Verwendetes Modell

Für dieses Projekt wurde das Kaldi-tuda-de Modell der Universität Hamburg, Fachbereich Language Technology, verwendet. Das Modell kann als Rezept für ein eigenes Training oder als vortrainiertes Modell verwendet werden. Es basiert auf dem Kaldi nnet3-Modell. Zum Training wurden über 1000 h Audiomaterial und 102 Millionen Textzeilen in deutscher Sprache aus frei verfügbaren Quellen verwendet. Zu den Textquellen zählen Nachrichtenartikel der Tagesschau, Untertitel der ARD, Artikel der deutschen Wikipedia und Reden aus dem

[8] `https://ffmpeg.org/`
[9] `http://www.videolan.org/`

Europäischen Parlament. Als Audioquellen wurden der Tuda-De Datensatz, SWC (Spoken Wikipedia Corpus, dt. Gesprochene Wikipedia-Korpora), der Common Voice Korpus und der M-AILABS Datensatz verwendet. Das Modell erreicht eine WER von 11,85% (Radeck-Arneth et al., 2015; Milde und Köhn, 2021; Milde und Köhn, 2018; Baumann et al., 2019).

5 Evaluierung

Umgebung	VM1	VM2	Sprecherinformation
Offline Kaldi	27,40%	30,37%	Ja
Offline Kaldi	26,48%	29,72%	Nein
Online KMS	30,13%	33,83%	Nein
Online KMS Opus	30,34%	34,07%	Nein
Online KMS BBB	31,66%	35,19%	Nein
Online KMS BBB	-	34,77%	Ja (Full turns)

Tabelle 5.1: Übersicht über die in den verschiedenen Experimenten erreichten WER

In diesem Abschnitt werden die Ergebnisse der Experimente als WER, wie in der Tabelle 5.1 dargestellt, aufgelistet und evaluiert. Einige der folgenden Ergebnisse wurden bereits auf der „Interspeech 2021“ vorgestellt (Geislinger et al., 2021).

Als Referenzwert wurde ein Experiment durchgeführt, bei dem das Audiomaterial mit Kaldi mittels Offline-Dekodierung verarbeitet wurde. Hier wurden mit einer WER von 27,40% für VM1 und einer WER von 30,37% für VM2 die geringsten WER erreicht. Einer der Gründe für die hohe Fehlerrate liegt im Trainingsmaterial des Kaldi-tuda-de Modells. Für das Training des akustischen Modells wurden zu großen Teilen vorgelesene Texte verwendet. Der Verbmobil-Datensatz enthält jedoch Gespräche in Spontansprache. Die Erkennung von Spontansprache ist anspruchsvoller als die von vorgelesenen Texten. Die Gründe dafür können falsch ausgesprochene Wörter, Wiederholung von Wörtern, Hesitationen, grammatikalische Fehler oder das Verwenden von Umgangssprache sein (Ward, 1989).

Das Material wurde sowohl mit als auch ohne Sprecherdaten offline dekodiert. Bei der Verarbeitung ohne Sprecherdaten wurde eine niedrigere WER erreicht als beim Experiment mit Sprecherdaten. Die WER verbessert sich dabei um 3,5% beim VM1-Datensatz und um 2,2% beim VM2-Datensatz. Der Grund dafür kann wie beim vorherigen Ergebnis im Unterschied zwischen der vorgelesene Sprache des Kaldi-tuda-de-Trainingsdatensatzes und der Spontansprache der Verbmobil-Datensätze liegen. Die Visualisierung der Vektoren ist in Abbildung 5.1 dargestellt.

Um die Experimente mittels Online-Dekodierung durchzuführen, wurde der KMS verwendet. Ziel war es zunächst, den Qualitätsunterschied zwischen Online- und Offline-Dekodierung zu ermitteln. Der Unterschied zwischen Offline- und Online-Dekodierung ohne Sprecherdaten resultiert in einer Steigerung der WER von 13,8% relativ bei beiden Datensätzen. Einer der möglichen Gründe für die Erhöhung der WER ist der Real-time-Faktor bei der Online-Dekodierung, der nicht größer als Eins sein sollte, damit eine Verarbeitung in Echtzeit garan-

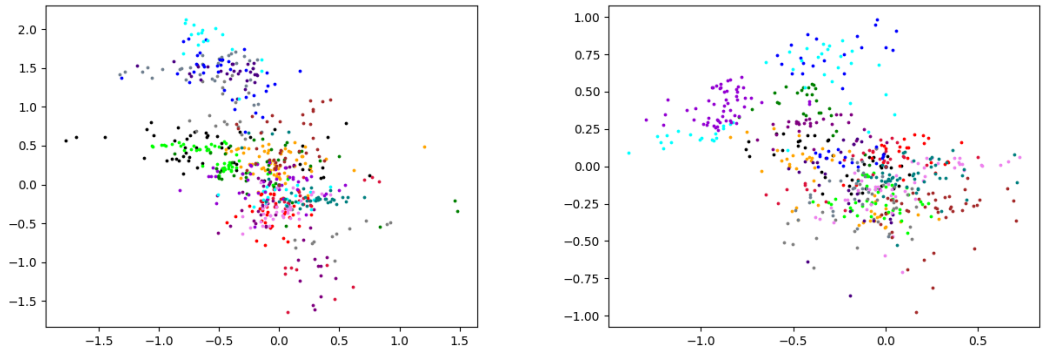

Abbildung 5.1: Visualisierung der i-Vectoren für den VM1-Datensatz (links) und VM2-Datensatz (rechts).

tiert werden kann. Auch kann bei der Offline-Dekodierung das Material im Ganzen durch das akustische Modell verarbeitet werden und nicht erst in kleinen Abschnitten. Um den Einfluss des Opus-Codecs auf die Erkennungsrate zu untersuchen, wurde das Audiosignal mit denselben Parametern kodiert, wie sie auch in BBB genutzt werden. Die WER erhöht sich durch das zuvor kodierte Material um 0,67% relativ von 30,13% auf 30,34% beim VM1-Datensatz und um 0,89% relativ von 33,83% auf 34,07% beim VM2-Datensatz. Der Grund für die höhere WER können Veränderungen des Audiomaterials wie Audiokompression durch den Opus Codec sein. Für die Wahl der Parameter gilt es, einen Mittelweg zwischen geringem Ressourcenverbrauch wie Prozessorzeit und Bandbreite und ausreichender Audioqualität zu finden. Bei der Kodierung mit einem verlustbehafteten Audio-Codec wie Opus kann es zum Verlust von Informationen kommen, was die Sprache verändert und die Erkennungsrate senkt. Die Auswirkungen durch die Kodierung mit dem Opus Codec von unter einem Prozent sind aber geringer als angenommen.

Für die Experimente mit BBB in einer Konferenz wurde der Google Chrome-Browser verwendet. Der Browser überträgt die Audiodaten über WebRTC an die BBB-Konferenz. Die Nutzung von BBB im Vergleich zum direkten Online-Dekodieren erhöht die WER relativ um 5,1% beim VM1-Datensatz und 4,0% beim VM2-Datensatz. Neben der Kodierung durch den Opus Codec kann es hier zu Qualitätseinbußen durch die Verwendung von WebRTC, Google Chrome und FreeSWITCH kommen. WebRTC baut eine UDP-Übertragung zwischen Server und Client auf, um eine möglichst geringe Latenz bei der Übertragung zu bieten (Perkins et al., 2021). UDP-Übertragungen können durch ihren verbindungslosen Aufbau zu Paketverlusten führen. Außerdem werden zu alte und ungültige Pakete verworfen, um keine Latenz aufzubauen. Trotz stabiler Internetverbindung können so immer wieder Pakete verworfen werden. Sowohl WebRTC als auch der Opus-Codec besitzen eine Fehlerkorrektur, um die Auswirkungen gering zu halten (Uberti, 2021). Laut verwendeten Parametern ist die Fehlerkorrektur in Opus aktiviert und bis zu 15% eines Audioframes können wiederhergestellt werden (siehe

Anhang). Weiterhin können weitere Verarbeitungsschritte in FreeSWITCH die Erkennungsrate senken. Außerdem verarbeitet der Browser das Audiomaterial mit Algorithmen zur Echo- oder Rauschunterdrückung, was auch zu einer gesteigerten Fehlerrate führen kann (Grigorik, 2016).

Beim Vergleich zwischen von Opus kodiertem Material und BBB wurde eine Steigerung der WER von 4,3% bzw. 3,3% gemessen. Die Steigerung der WER wird sich auf die bereits genannten Vermutungen wie zusätzliche Verarbeitungsschritte in FreeSWITCH und Browser zurückführen lassen, da in diesen beiden Experimenten das Audiomaterial mit identischem Codec und Parametern kodiert wurde.

5.1 geschlossenes vs. offenes Vokabular

Bei einem Experiment mit closed vocabulary (dt. geschlossenem Vokabular) sind nahezu alle Wörter aus dem Testdatensatz auch im Trainingsdatensatz enthalten. Dies ist häufig bei älteren Experimenten wie z.B. auch mit der Verbmobil-Engine JANUS der Fall. Bei der Verarbeitung mit JANUS wurde eine WER von 13,8% erreicht. Das Ziel des Verbmobil-Projekts war es, ausschließlich die Erkennung im Bereich der Terminvereinbarung und keine generelle Erkennung zu realisieren. Aufgrund deutlich niedrigerer Leistung der damaligen Computer wurden Modelle außerdem mit einem kleineren Vokabular trainiert als heute. Verbmobil arbeitete mit einem geschlossenen Vokabular von 6000 Wörtern (Finke et al., 1997). Bei aktuelleren Experimenten aus dem Jahr 2014 wurde das Modell mit dem VM1-Datensatz trainiert und ein geschlossenes Vokabular verwendet. Dabei wurde ein DNN-HMM-Modell trainiert, welches als Datensatz VM1 verwendet. Dieses Modell erreichte eine WER von 12,7% (Gaida et al., 2014). Bei einem offenen Vokabular ist nicht sichergestellt, dass alle Wörter des Testdatensatzes auch Teil des Trainingsdatensatzes sind. Die Ergebnisse des Kaldi-tuda-de Modells wurden mit einem offenen Vokabular von 683.000 Wörtern durchgeführt (Milde und Köhn, 2021).

Das Kaldi-tuda-de Modell erreichte mit einem offenen Vokabular von 350.000 Wörtern eine WER von 38,23% auf dem VM1-Datensatz. Bei einem eingeschränkten Vokabular von etwa 7.000 Wörtern konnte eine WER von 20,04% erreicht werden (Milde und Köhn, 2018).

In den in dieser Arbeit vorliegenden Ergebnissen ist eine deutliche Erhöhung der WER im Vergleich zu den Ergebnissen mit der Verbmobil-Engine JANUS zu erkennen. Bei den in dieser Arbeit durchgeführten Experimenten wurde jedoch ein offenes Vokabular verwendet. Ein Vergleich der Ergebnisse bei denselben Datensätzen ist deswegen nur bedingt möglich.

5.2 Vollständige Gespräche vs. kurze Audiosequenzen

Die Audiodateien, die für die Experimenten im Rahmen der vorliegenden Arbeit genutzt wurden, bestehen nicht wie in anderen Experimenten aus kurzen Audiosequenzen, sondern beinhalten realistische Gespräche. Dazu gehören auch Pausen zwischen den Wortmeldungen. Die Ergebnisse des Experiments spiegeln somit ein Praxisszenario wider, in dem jeder Sprecher in einem stillen Raum sitzt und sein eigenes Mikrofon benutzt. Das trifft zum Beispiel bei Videokonferenzen zu. Die Ergebnisse der Experimente mit den vollständigen Gesprächen (Full

Turns) brachten eine relative Verbesserung der WER von 1,2% im Vergleich zu den Experimenten ohne Sprecherinformationen über BBB. Gründe für die Verbesserung können hier zum einen die Verwendung von Sprecherinformationen und zum anderen die Pausen zwischen den Wortmeldungen sein. Im Vergleich zur Online-Dekodierung von mit Opus kodiertem Audiomaterial erhöht sich die WER bei den vollständigen Gesprächen nur um etwa 1%.

6 Fazit und Ausblick

Mit der Arbeit sollte gezeigt werden, dass sich eine in der Praxis nutzbare Untertitelung mit kostenfreier Software und Modellen realisieren lässt. Durch Verarbeitung der Teilnehmer unabhängig voneinander, ließ sich die Erkennungsrate steigern. Die durch die Arbeit entstandene Lösung lässt sich in bestehende BBB-Installationen mit geringem Aufwand und ohne Fachwissen im Bereich Sprachverarbeitung integrieren.

Es gibt viele Möglichkeiten, die Software zu erweitern - sowohl im Funktionsumfang als auch in der Nutzbarkeit. Die Software könnte erweitert werden, um Teilnehmern im Anschluss an die abgeschlossene Konferenz eine Abschrift als PDF-Dokument zuzusenden. So müssten Teilnehmer keine komplette Konferenz erneut ansehen, um die in der Konferenz ausgetauschten Informationen zu erhalten. Die Abschrift könnte dann mit Hilfe von Schlagwörtern nach bestimmten Details durchsucht oder anderweitig zusammengefasst werden. Meetings könnten auch aufbereitet werden, um Funktionen wie Wortwolken oder Tagesordnungspunkte zur besseren Strukturierung und Nachvollziehbarkeit eines Meetings anzubieten (Milde, Fischer et al., 2021). Eine direktere Integration in BBB könnte die Untertitel auch in die bereits bestehende Aufzeichnungsfunktion von BBB integrieren, um so Aufzeichnungen mit Untertiteln anbieten zu können. Außerdem könnten die Untertitel um Einstellungen in BBB erweitert werden, sodass der Moderator die Möglichkeit bekommt, Teilnehmer gezielt zu untertiteln. Eine visuelle Optimierung der Untertitel wäre außerdem denkbar, um die Untertitel besser in der Konferenz sichtbar zu machen. Dies könnte durch mehrere individuelle Gestaltungsmöglichkeiten passieren. Unterschiedliche Textfarben je Teilnehmer könnten die Unterscheidbarkeit verbessern.

Auch die Entwicklung einer Schnittstelle für Screenreader wäre denkbar, um Menschen mit gleichzeitiger Seh- und Hörschwäche eine bessere Möglichkeit der Interaktion zu bieten. Außerdem könnte der Text aufbereitet und verarbeitet werden, um Satzzeichen und Zeilenumbrüche automatisiert einzufügen. Dies würde den Lesefluss und Verständlichkeit erhöhen (Milde, Geislinger et al., 2021; Suter und Novak, 2021). Die Texte könnten automatisiert übersetzt werden, um Konferenzen zu ermöglichen, in denen nicht alle dieselbe Sprache sprechen.

Es sollte zudem untersucht werden, inwiefern sich die erhöhte Wortfehlerrate mit einem speziell angepassten Modell senken lässt. Dafür sollten die Trainingsdaten vorab mit dem Opus-Codec kodiert werden, um zu evaluieren, ob sich so die Erkennungsrate weiter verbessern lässt. Auch könnten Audiodateien verschiedenster Kodierungen getestet und verglichen werden. Hier könnte ein Teil der Trainingsdaten mit dem Opus-Codec kodiert werden. Es wäre zu testen, wie sich der Datensatz auf die Erkennungsrate auswirkt. Auch die Kodierung mit anderen Sprachcodecs sollte überprüft werden, um den idealen Codec für die Spracherkennung zu finden. Das könnte Auswirkungen auf den in Zukunft verwendeten Codec oder die verwendeten Codecparameter in BBB haben.

Die Software ist mit Anleitungen frei verfügbar, damit andere Organisationen und Einzelpersonen das Projekt kostenfrei nutzen, anpassen und erweitern können.[1].

Im Bereich der Konferenzuntertitelung besteht noch viel Verbesserungs- und Forschungspotenzial für kommende Projekte.

[1] https://github.com/uhh-lt/bbb-live-subtitles

7 Anhang

7.1 Opus Parameter

Die Parameter sind aus den Konfigurationsdateien`conference.conf.xml`[1] und `opus.conf.xml`[2].
Sample Rate = 48000
Variable bitrate = Ja
DTX = False
Comfort noise = 1400
Complexity = 5
Packet loss percent = 15
Forward Error Correction = True
Jitter Buffer lookahead = True
Interval = 20
Energy Level = 100
Die Kodierung wurde mit ffmpeg 4.3.2-2021-02-27-full_build für Windows durchgeführt.

[1] `https://github.com/bigbluebutton/bigbluebutton/blob/v2.2.x-release/bbb-voice-conference/config/freeswitch/conf/autoload_configs/conference.conf.xml#L156-L176`
[2] `https://github.com/bigbluebutton/bigbluebutton/blob/develop/bbb-voice-conference/config/freeswitch/conf/autoload_configs/opus.conf.xml`

Literatur

Adda, Gilles, Martine Adda-Decker, Jean-Luc Gauvain und Lori Lamel (1997). „Text normalization and speech recognition in French“. In: *Proc. 5th European Conference on Speech Communication and Technology (Eurospeech 1997)*. Rhodos, Griechenland, S. 2711–2714 (siehe S. 17).

Aho, Alfred V und Jeffrey D Ullman (1971). „Translations on a context free grammar“. In: *Information and Control* 19.5, S. 439–475 (siehe S. 18).

Alexandersson, Jan, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Elisabeth Maier, Norbert Reithinger, Birte Schmitz und Melanie Siegel (1997). „Dialogue acts in VERBMOBIL-2“. In: (Siehe S. 29).

Association, International Phonetic, International Phonetic Association Staff et al. (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press. ISBN: 9780521637510 (siehe S. 8).

Bahl, Lalit R, Frederick Jelinek und Robert L Mercer (1983). „A maximum likelihood approach to continuous speech recognition“. In: *IEEE transactions on pattern analysis and machine intelligence* 2, S. 179–190 (siehe S. 15).

Baumann, Timo, Arne Köhn und Felix Hennig (2019). „The Spoken Wikipedia Corpus collection: Harvesting, alignment and an application to hyperlistening“. In: *Language Resources and Evaluation* 53.2, S. 303–329 (siehe S. 32).

BigBlueButton Inc. (2015). *GitHub Repository BigBlueButton*. Abgerufen am 20. Juli 2021. URL: https://github.com/bigbluebutton/bigbluebutton/blob/v2.2.x-release/akka-bbb-fsesl/src/main/java/org/bigbluebutton/freeswitch/voice/events/VoiceUserTalkingEvent.java (siehe S. 27).

– (Juni 2021a). *BigBlueButton LMS Integrations: Designed for Schools*. Abgerufen am 11. Mai 2021. URL: https://bigbluebutton.org/schools/integrations/ (siehe S. 19).

– (2021b). *Front Ends*. Abgerufen am 12. September 2021. URL: https://docs.bigbluebutton.org/support/faq.html#does-bigbluebutton-come-with-a-front-end (siehe S. 20).

– (2021c). *GitHub Repository BigBlueButton*. Abgerufen am 20. Juli 2021. URL: https://github.com/bigbluebutton/bigbluebutton/tree/develop/bbb-voice-conference/config/freeswitch/conf/dialplan/default (siehe S. 26).

– (2021d). *High-level architecture*. Abgerufen am 15. September 2021. URL: https://docs.bigbluebutton.org/2.2/architecture.html (siehe S. 19).

– (2021e). *HTML5 client*. Abgerufen am 19. August 2021. URL: https://docs.bigbluebutton.org/2.2/architecture.html#html5-client (siehe S. 19).

– (2021f). *Redis PubSub*. Abgerufen am 18. August 2021. URL: https://docs.bigbluebutton.org/2.2/architecture.html#redis-pubsub (siehe S. 19).

Birman, Kenneth P (1993). „The process group approach to reliable distributed computing“. In: *Communications of the ACM* 36.12, S. 37–53 (siehe S. 18).

Bundesfachstelle Barrierefreiheit (Aug. 2021). *Barrierefreie Webkonferenzen*. Abgerufen am 2. Juli 2021. URL: https://www.bundesfachstelle-barrierefreiheit.de/DE/Praxishilfen/Informationstechnik/Barrierefreie-Webkonferenzen/barrierefreie-webkonferenzen_node.html (siehe S. 1).

Burger, Susanne, Karl Weilhammer, Florian Schiel und Hans G Tillmann (2000). „Verbmobil data collection and annotation“. In: *Verbmobil: Foundations of speech-to-speech translation*. Springer, S. 537–549 (siehe S. 29).

Cáceres, Juan-Pablo und Chris Chafe (2010). „Jacktrip/Soundwire meets server farm“. In: *Computer Music Journal* 34.3, S. 29–34 (siehe S. 25).

Can, Dogan, Victor R Martinez, Pavlos Papadopoulos und Shrikanth S Narayanan (2018). „PyKaldi: A python wrapper for Kaldi“. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. Calgary, Alberta, Kanada, S. 5889–5893 (siehe S. 18).

Cheng, Octavian, Waleed Abdulla und Zoran Salcic (2011). „Hardware–Software Codesign of Automatic Speech Recognition System for Embedded Real-Time Applications“. In: *IEEE Transactions on Industrial Electronics* 58.3, S. 850–859 (siehe S. 16).

Cutler, Cassius C (Juli 1952). *Differential quantization of communication signals*. US Patent 2,605,361 (siehe S. 20).

Davis, Steven und Paul Mermelstein (1980). „Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences“. In: *IEEE transactions on acoustics, speech, and signal processing* 28.4, S. 357–366 (siehe S. 9).

Dehak, Najim, P. Kenny, R. Dehak, P. Dumouchel und P. Ouellet (2011). „Front-End Factor Analysis for Speaker Verification“. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19, S. 788–798 (siehe S. 14).

Dhillon, A, Shikharesh Majumdar, Marc St-Hilaire und Ali El-Haraki (2018). „MCEP: A mobile device based complex event processing system for remote healthcare“. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE. Halifax, Kanada, S. 203–210 (siehe S. 25).

Eizmendi, G et al. (2007). „Automatic speech recognition for live TV subtitling for hearing-impaired people“. In: AAATE 2007 9th European Conference for the Advancement of Assistive Technology. Bd. 20. San Sebastian, Spanien: IOS Press, S. 286 (siehe S. 5).

Fette, I. und A. Melnikov (Dez. 2011). *The WebSocket Protocol*. RFC 6455. RFC Editor. URL: http://www.rfc-editor.org/rfc/rfc6455.txt (siehe S. 26).

Finke, M., P. Geutner, H. Hild, T. Kemp, K. Ries und M. Westphal (1997). „The Karlsruhe-Verbmobil speech recognition engine“. In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Bd. 1. München, Deutschland, 83–86 vol.1 (siehe S. 35).

Forney, G David (1973). „The viterbi algorithm“. In: *Proceedings of the IEEE* 61.3, S. 268–278 (siehe S. 12).

Gaida, Christian, Patrick Lange, Rico Petrick, Patrick Proba, Ahmed Malatawy und David Suendermann-Oeft (2014). „Comparing open-source speech recognition toolkits“. In: *11th*

International Workshop on Natural Language Processing and Cognitive Science. Venedig, Italien (siehe S. 12, 35).

Gales, Mark und Steve Young (2007). „The Application of Hidden Markov Models in Speech Recognition“. In: *Signal Processing* 1.3, S. 195–304 (siehe S. 12).

Garcia-Molina, H. und K. Salem (1992). „Main memory database systems: an overview“. In: *IEEE Transactions on Knowledge and Data Engineering* 4.6, S. 509–516 (siehe S. 18).

Geislinger, Robert, Benjamin Milde, Timo Baumann und Chris Biemann (2021). „Live Subtitling for BigBlueButton with Open-Source Software“. In: *Proc. Interspeech 2021*. Brünn, Tschechien, S. 3319–3320 (siehe S. 33).

Georgescu, Alexandru-Lucian, Horia Cucu und Corneliu Burileanu (2019). „Kaldi-based DNN Architectures for Speech Recognition in Romanian“. In: *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. Timisoara, Rumänien, S. 1–6 (siehe S. 11).

Grigorik, Ilya (Mai 2016). *Browser APIs and Protocols: WebRTC - High Performance Browser Networking (O'Reilly)*. Abgerufen am 2. Juli 2021. URL: `https://hpbn.co/webrtc/#audio-and-video-engines` (siehe S. 35).

Hammershøj, Allan, Antonio Sapuppo und Reza Tadayoni (2010). „Challenges for mobile application development“. In: *2010 14th International Conference on Intelligence in Next Generation Networks*. IEEE. Berlin, Deutschland, S. 1–8 (siehe S. 25).

Hellweger Anzeiger (Aug. 2020). *Schlappes Internet und Home Schooling: Schüler (16) sorgt sich um seinen Abschluss*. Abgerufen am 15. Juli 2021. URL: `https://gesamtschulefroendenberg.de/2020/08/schlappes-internet-und-home-schooling-schueler-16-sorgt-sich-um-seinen-abschluss/` (siehe S. 2).

Hess, Wolfgang J, Klaus J Kohler und Hans-Günther Tillmann (1995). „The Phondat-Verbmobil speech corpus“. In: *Fourth European Conf. on Speech Communication and Technology*. Madrid, Spanien, S. 863–866 (siehe S. 29).

Holter, Trym, Erik Harborg, Magne Johnsen und Torbjørn Svendsen (Jan. 2000). „ASR-based subtitling of live TV-programs for the hearing impaired.“ In: *Sixth International Conference on Spoken Language Processing*. Beijing, China, S. 570–573 (siehe S. 5).

Huang, Xuedong, Alex Acero, Hsiao-Wuen Hon und Raj Reddy (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR. ISBN: 978-0130226167 (siehe S. 9, 11, 21).

Jang, CS und Chong-Kwan Un (1996). „A new parameter smoothing method in the hybrid TDNN/HMM architecture for speech recognition“. In: *Speech Communication* 19.4, S. 317–324 (siehe S. 13).

Jang, Jyh-Shing Roger und Shiuan-Sung Lin (Jan. 2002). „Optimization of Viterbi beam search in speech recognition“. In: Taipei, Taiwan, S. 114–117 (siehe S. 16).

Jekat, Susanne, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast und J Joachim Quantz (1995). „Dialogue acts in VERBMOBIL“. In: S. 11–35 (siehe S. 29).

Jelinek, Frederick (1976). „Continuous speech recognition by statistical methods“. In: *Proceedings of the IEEE* 64.4, S. 532–556 (siehe S. 11).

Jurafsky, Daniel und James H. Martin (2009). *Speech and Language Processing (2nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0131873210 (siehe S. 10, 11, 17).

Kacprzyk, Janusz und Witold Pedrycz (2015). *Springer handbook of computational intelligence*. Springer, S. 460–471. ISBN: 978-3-662-43504-5 (siehe S. 12).

Kaldi Team (2020). *HMM topology and transition modeling*. Abgerufen am 13. September 2021. URL: http://kaldi-asr.org/doc/hmm.html#transition_model_identifiers (siehe S. 17).

– (2021a). *Feature extraction*. Abgerufen am 12. September 2021. URL: https://kaldi-asr.org/doc/feat.html (siehe S. 29).

– (2021b). *The Decodable interface*. Abgerufen am 13. September 2021. URL: http://kaldi-asr.org/doc/decoders.html (siehe S. 17).

Karafiát, Martin, Lukáš Burget, Pavel Matějka, Ondřej Glembek und Jan Černocký (2011). „iVector-based discriminative adaptation for automatic speech recognition". In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE. Big Island, Hawaii, USA, S. 152–157 (siehe S. 14).

Kettinger, Marlene (Sep. 2021). *Die 10 schlimmsten No-Gos bei Telefonkonferenzen*. Abgerufen am 2. Juli 2021. URL: https://blog.yuutel.at/no-gos-bei-telefonkonferenzen (siehe S. 2).

Łalovarda, Marija, Ivan Bolkovac und Hrvoje Domitrović (2004). *Comparison of audio codecs using PEAQ algorithm*. Techn. Ber. Faculty of EE und Computing, Dept. of Electroacoustics (siehe S. 25).

Lambourne, Andrew (2006). „Subtitle respeaking". In: *Intralinea, Special Issue on Respeaking* (siehe S. 5).

Lamel, Lori, Martine Adda-Decker und Jean-Luc Gauvain (1995). „Issues in Large Vocabulary, Multilingual Speech Recognition." In: *EUROSPEECH*. Madrid, Spanien, S. 185–188 (siehe S. 17).

Lowerre, Bruce (1990). „The HARPY speech understanding system". In: *Readings in speech recognition*, S. 576–586 (siehe S. 16).

Maas, Andrew L, Peng Qi, Ziang Xie, Awni Y Hannun, Christopher T Lengerich, Daniel Jurafsky und Andrew Y Ng (2017). „Building DNN acoustic models for large vocabulary speech recognition". In: *Computer Speech & Language* 41, S. 195–213 (siehe S. 6).

Maria Geipel, Bayerischer Rundfunk (Jan. 2021). *Kommunikation und Sprache : Sender-Empfänger-Modell*. Abgerufen am 15. Juli 2021. URL: https://www.br.de/alphalernen/faecher/deutsch/2-kommunikation-sender-empfaenger-modell102.html (siehe S. 1).

Marsh, Alison (2006). „Respeaking for the BBC". In: *Eugeni, C. e Mack, G.(a cura di) Proceedings of the First International Seminar on Real Time Intralingual Subtitling. InTRAlinea, Special Issue on Respeaking* (siehe S. 5).

Maruschke, Michael, Oliver Jokisch, Martin Meszaros und Viktor Iaroshenko (2015). „Review of the Opus Codec in a WebRTC Scenario for Audio and Speech Communication". In: *Speech and Computer*. Hrsg. von Andrey Ronzhin, Rodmonga Potapova und Nikos Fakotakis. Cham: Springer International Publishing, S. 348–355. ISBN: 978-3-319-23132-7 (siehe S. 5).

Milde, Benjamin, Tim Fischer, Steffen Remus und Chris Biemann (2021). „MoM: Minutes of Meeting Bot". In: *Proc. Interspeech 2021*. Brünn, Tschechien, S. 3311–3312 (siehe S. 37).

Milde, Benjamin, Robert Geislinger, Irina Lindt und Timo Baumann (2021). „Open Source Automatic Lecture Subtitling“. In: *Proceedings of ESSV 2021*. Berlin, Deutschland, S. 128–134 (siehe S. 5, 37).

Milde, Benjamin und Arne Köhn (2018). „Open Source Automatic Speech Recognition for German“. In: *Proceedings of ITG 2018*. Oldenburg, Deutschland, S. 251–255 (siehe S. 6, 11, 32, 35).

– (2021). *Open source speech recognition recipe and corpus for building German acoustic models with Kaldi*. Abgerufen am 20. Juli 2021. URL: `https://github.com/uhh-lt/kaldi-tuda-de` (siehe S. 6, 32, 35).

Mohri, Mehryar, Fernando Pereira und Michael Riley (2002). „Weighted finite-state transducers in speech recognition“. In: *Computer Speech & Language* 16.1, S. 69–88 (siehe S. 15–17).

– (2008). „Speech recognition with weighted finite-state transducers“. In: *Springer handbook of speech processing*. Springer, S. 559–584 (siehe S. 15).

Müller, Bernd S (1984). „Was sprechen Sprechmaschinen?“ In: *Folia Linguistica* 18, S. 87–101 (siehe S. 8).

Neuhetzki, Thorsten (Juli 2021). *DSL, Kabel & Co: So schnell muss dein Internetanschluss wirklich sein*. Abgerufen am 2. Juli 2021. URL: `https://www.inside-digital.de/ratgeber/dsl-kabel-internet-anschluss-leitung-wie-schnell` (siehe S. 1).

Paliwal, Kuldip K, James G Lyons und Kamil K Wójcicki (2010). „Preference for 20-40 ms window duration in speech analysis“. In: *2010 4th International Conference on Signal Processing and Communication Systems*. IEEE. Gold Coast, Australien, S. 1–4 (siehe S. 10).

Peddinti, Vijayaditya, Daniel Povey und Sanjeev Khudanpur (2015). „A time delay neural network architecture for efficient modeling of long temporal contexts“. In: *Proc. Interspeech 2015*. Dresden, Deutschland, S. 3214–3218 (siehe S. 13).

Peperkamp, Sharon, Michèle Pettinato und Emmanuel Dupoux (2003). „Allophonic variation and the acquisition of phoneme categories“. In: *Proceedings of the 27th Annual Boston University Conference on Language Development*. Boston, Massachusetts, USA: Citeseer, S. 650–661 (siehe S. 8).

Perkins, C., M. Westerlund und J. Ott (Jan. 2021). *Media Transport and Use of RTP in WebRTC*. RFC 8834. URL: `https://rfc-editor.org/rfc/rfc8834.txt` (siehe S. 34).

Plátek, Ondřej und Filip Jurcicek (2014). „Free on-line speech recogniser based on Kaldi ASR toolkit producing word posterior lattices“. In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Philadelphia, Pennsylvania, USA, S. 108–112 (siehe S. 16).

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer und Karel Vesely (Dez. 2011). „The Kaldi Speech Recognition Toolkit“. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Hawaii, USA: IEEE Signal Processing Society (siehe S. 18).

Povey, Daniel, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motliček, Yanmin Qian et al. (2012). „Ge-

nerating exact lattices in the WFST framework“. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. Kyoto, Japan, S. 4213–4216 (siehe S. 17).

Povey, Daniel, Xiaohui Zhang und Sanjeev Khudanpur (2015). „Parallel training of Deep Neural Networks with Natural Gradient and Parameter Averaging“. In: *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*. San Diego, USA (siehe S. 13).

Pražák, Aleš, J. V. Psutka, Jan Hoidekr, Jakub Kanis, Luděk Müller und Josef Psutka (2006). „Automatic Online Subtitling of the Czech Parliament Meetings“. In: *Text, Speech and Dialogue*. Hrsg. von Petr Sojka, Ivan Kopeček und Karel Pala. Brünn, Tschechien, S. 501–508. ISBN: 978-3-540-39091-6 (siehe S. 5).

Purves, D., G. Augustine, D. Fitzpatrick, W. C. Hall, A. LaMantia, J. McNamara und S. M. Williams (2004). *Neuroscience, 3rd ed.* S. 284. ISBN: 0-87893-725-0 (siehe S. 7).

Qiu, Xiangkai (2019). „Study on Automatic Generation of Teaching Video Subtitles Based“. In: *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceedings of the 15th International Conference on IIH-MSP in conjunction with the 12th International Conference on FITAT, July 18-20, Jilin, China, Volume 1*. Bd. 156. Springer, S. 309 (siehe S. 5).

Rabe, L. (Sep. 2021). *ZOOM Cloud Meetings - Anzahl der Downloads im Apple App Store weltweit 2021*. Abgerufen am 13.6.2021. URL: https://de.statista.com/statistik/daten/studie/1113689/umfrage/anzahl-der-downloads-von-zoom-ueber-den-apple-app-store-weltweit/ (siehe S. 1).

Rabin, Michael O und Dana Scott (1959). „Finite automata and their decision problems“. In: *IBM journal of research and development* 3.2, S. 114–125 (siehe S. 15).

Rabiner, Lawrence und Biinghwang Juang (1986). „An introduction to hidden Markov models“. In: *IEEE ASSP Magazine* 3.1, S. 4–16 (siehe S. 12).

Radeck-Arneth, Stephan, Benjamin Milde, Arvid Lange, Evandro Gouvêa, Stefan Radomski, Max Mühlhäuser und Chris Biemann (2015). „Open source german distant speech recognition: Corpus and acoustic model“. In: *International Conference on Text, Speech, and Dialogue*. Springer. Pilsen, Tschechien, S. 480–488 (siehe S. 32).

Rämö, Anssi und Henri Toukomaa (2011). „Voice quality characterization of IETF Opus codec“. In: *Twelfth Annual Conference of the International Speech Communication Association*. Florenz, Italien (siehe S. 5).

Reynolds, Douglas A. (1995). „Speaker identification and verification using Gaussian mixture speaker models“. In: *Speech Communication* 17.1, S. 91–108. ISSN: 0167-6393. URL: https://www.sciencedirect.com/science/article/pii/016763939500009D (siehe S. 14).

Romero-Fresco, Pablo (2020). *Subtitling through speech recognition: Respeaking*. Routledge. ISBN: 978-1138473744 (siehe S. 5).

Roy, Brandon Cain und Deb K Roy (Jan. 2009). „Fast transcription of unstructured audio recordings“. In: *Proc. Interspeech 2009*. Brighton, Vereinigtes Königreich, S. 1647–1650 (siehe S. 2).

Saon, George, Hagen Soltau, David Nahamoo und Michael Picheny (2013). „Speaker adaptation of neural network acoustic models using i-vectors“. In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. Olomouc, Tschechien, S. 55–59 (siehe S. 14).

Sayood, Khalid (2002). *Lossless compression handbook*. Elsevier. ISBN: 978-0-12-620861-0 (siehe S. 21).

Schiel, Florian (2003). *BASBayerisches Archiv für Sprachsignale Verbmobil II - VM2*. Abgerufen am 2. Juni 2021. URL: https://www.phonetik.uni-muenchen.de/Bas/BasVM2deu.html (siehe S. 30).

Shahin, Mostafa, Beena Ahmed, Jacqueline McKechnie, Kirrie Ballard und Ricardo Gutierrez-Osuna (2014). „A comparison of GMM-HMM and DNN-HMM based pronunciation verification techniques for use in the assessment of childhood apraxia of speech“. In: *Proc. Interspeech 2014*. Singapur, S. 1583–1587 (siehe S. 12).

Shannon, Claude Elwood (1949). „Communication in the presence of noise“. In: *Proceedings of the IRE* 37.1, S. 10–21 (siehe S. 7).

Shinoda, Koichi (2005). „Speaker adaptation techniques for speech recognition using probabilistic models“. In: *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)* 88.12, S. 25–42 (siehe S. 14).

SignalWire Inc. (2018). *mod_dptools: record_session*. Abgerufen am 15. August 2021. URL: https://freeswitch.org/confluence/display/FREESWITCH/mod_dptools%5C%3A+record_session (siehe S. 25).

– (2019a). *Call Legs*. Abgerufen am 16. September 2021. URL: https://freeswitch.org/confluence/display/FREESWITCH/Call%20Legs (siehe S. 26).

– (2019b). *Event List*. Abgerufen am 15. August 2021. URL: https://freeswitch.org/confluence/display/FREESWITCH/Event+List (siehe S. 27).

– (2020). *mod_conference*. Abgerufen am 15. August 2021. URL: https://freeswitch.org/confluence/display/FREESWITCH/mod_conference (siehe S. 20).

Soltau, Hagen, Thomas Schaaf, Florian Metze und Alex Waibel (2001). „The ISL evaluation system for Verbmobil-II“. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*. Bd. 1. IEEE. Salt Lake City, Utah, USA, S. 65–68 (siehe S. 29).

Spiegl, Thomas (Nov. 2015). *Videomeetings: 7 Tipps zur perfekten Videokonferenz*. Abgerufen am 2. Juli 2021. URL: https://www.tecchannel.de/a/7-tipps-zur-perfekten-videokonferenz,2079523 (siehe S. 1).

Sproat, Richard und Steven Bedrick (2018). *Text Normalization*. Abgerufen am 15. Juli 2021. URL: https://docplayer.net/79742188-Text-normalization-richard-sproat-steven-bedrick.html (siehe S. 17).

Staš, Ján, Peter Viszlay, Martin Lojka, Tomáš Koctúr, Daniel Hládek, Eva Kiktová, Matúš Pleva und Jozef Juhár (2015). „Automatic subtitling system for transcription, archiving and indexing of Slovak audiovisual recordings“. In: *Proceedings of the 7th Language & Technology Conference, LTC 2015*. Poznań, Polen, S. 186–191 (siehe S. 5).

Suter, Benjamin und Josef Novak (2021). „Neural Text Denormalization for Speech Transcripts“. In: *Proc. Interspeech 2021*. Brünn, Tschechien, S. 981–985 (siehe S. 37).

Uberti, Justin (Jan. 2021). *WebRTC Forward Error Correction Requirements*. RFC 8854. URL: https://rfc-editor.org/rfc/rfc8854.txt (siehe S. 34).

Van Genuchten, Michiel (1991). „Why is software late? An empirical study of reasons for delay in software development". In: *IEEE Transactions on software engineering* 17.6, S. 582 (siehe S. 25).

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano und K.J. Lang (1989). „Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3, S. 328–339 (siehe S. 11–13).

Walden, R.H. (1999). „Analog-to-digital converter survey and analysis". In: *IEEE Journal on Selected Areas in Communications* 17.4, S. 539–550 (siehe S. 7).

Ward, Wayne (1989). „Understanding spontaneous speech". In: *Speech and Natural Language: Proceedings of a Workshop*. Philadelphia, Pennsylvania, USA (siehe S. 33).

Wells, John C (1995). „Computer-coding the IPA: a proposed extension of SAMPA". In: 4.28. URL: http://www.phon.ucl.ac.uk/home/sampa/ipasam-x.pdf (siehe S. 8).

Wikimedia Commons (2020). *File:Opus bitrate+latency comparison.svg — Wikimedia Commons, the free media repository*. Abgerufen am 13. September 2021. URL: https://commons.wikimedia.org/w/index.php?title=File:Opus_bitrate%5C%2Blatency_comparison.svg%5C&oldid=459303401 (siehe S. 6, 21).

Wikipedia (2021). *Untertitel — Wikipedia, die freie Enzyklopädie*. Abgerufen am 2. Juni 2021. URL: https://de.wikipedia.org/w/index.php?title=Untertitel&oldid=214841962 (siehe S. 2).

Wiktionary (2020). *Verzeichnis:Deutsch/Phoneme und Grapheme — Wiktionary, Das freie Wörterbuch*. Abgerufen am 13. September 2021. URL: https://de.wiktionary.org/w/index.php?title=Verzeichnis:Deutsch/Phoneme_und_Grapheme&oldid=8285877 (siehe S. 8).

Wille, Frederik (2020). *GitHub Repository BBB Kaldi Connector*. Abgerufen am 20. Juli 2021. URL: https://github.com/3wille/bbb-kaldi-connector/ (siehe S. 24).

Wrigley, Stuart N, Guy J Brown, Vincent Wan und Steve Renals (2004). „Speech and crosstalk detection in multichannel audio". In: *IEEE Transactions on speech and audio processing* 13.1, S. 84–91 (siehe S. 24).

Wuppermann, Michael Michael (Apr. 2021). *Technische Probleme bei Videokonferenzen vermeiden - eScience-Büro: Digitale Forschung*. Abgerufen am 13.9.2021. URL: https://escience-ew.blogs.uni-hamburg.de/technische-probleme-bei-videokonferenzen-vermeiden/202/ (siehe S. 1).

Young, S. J., J. J. Odell und P. C. Woodland (1994). „Tree-Based State Tying for High Accuracy Acoustic Modelling". In: HLT '94. Plainsboro, NJ, USA: Association for Computational Linguistics, S. 307–312. ISBN: 1558603573 (siehe S. 17).

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 28.09.2021

Robert Georg Geislinger

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 28.09.2021

Robert Georg Geislinger