Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Projektarbeit (Informatik)

Speech Separation als Anwendung für die Automatic Speech Recognition

Autoren	Michael Isler Marco Zala
Hauptbetreuung	Prof. Dr. Mark Cieliebak
Nebenbetreuung	Daniel Neururer

Datum	24.12.2021

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum: Winterthur, 23.12.2021

Unterschriften: M. Isler

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Zusammenfassung

Das menschliche Gehör ist fähig, mehrere überlappende Stimmen zu trennen und zu verstehen. Dieselbe Aufgabe stellt für die Automatic Speech Recognition (ASR) eine grosse Herausforderung dar. Diese Lücke könnte mittels Methoden des Gebiets der *Speech Separation* geschlossen werden. In dieser Arbeit werden zweierlei Aufgaben behandelt. Zum einen erfolgt eine Literaturübersicht über den aktuellen Stand der Technik und zum anderen werden erste Experimente durchgeführt, anhand welcher abgeschätzt werden sollte, wie gut sich Speech Separation-Methoden für einen Einsatz in ASR-Systemen eignen. Die Experimente wurden am Beispiel des Modells SepFormer mit Daten des LibriSpeech Korpus und eigenen Aufnahmen umgesetzt. Die verschiedenen Sequenzen wurden sowohl physisch als auch virtuell zusammengemischt, was unterschiedlichen Aufnahmequalitäten entspricht. Evaluiert wurde jeweils anhand der Word Error Rate (WER) zwischen den Transkripten vor und nach der Separation der Sprechenden. Auch wenn die Experimente aufgrund zu kleiner Datenmenge *keinen* Anspruch auf Wissenschaft erheben, lassen die Resultate einige Vermutungen zu: Mit durchschnittlichen WER-Werten von circa 30% stellen Speech Separation-Methoden durchaus eine sinnvolle Erweiterung für ASR-Systeme dar. Für eine erfolgreiche Separation und anschliessende Transkription scheint die Audioqualität der Aufnahmen von enormer Wichtigkeit zu sein. Ebenfalls haben die Experimente angedeutet, dass die Anzahl Mikrofone und Unterschiede in der Länge der überlappenden Sequenzen das Ergebnis nicht massgeblich zu beeinflussen scheinen.

Abstract

The human ear is capable of separating and understanding multiple overlapping speakers. The same task poses a great challenge for Automatic Speech Recognition (ASR). This gap could be closed by methods from the field of *Speech Separation*. In this thesis, two tasks are addressed. Firstly, a literature review of the current state of the art and secondly, first experiments are conducted to assess how well speech separation methods are suited for use in ASR systems. The experiments were implemented using the SepFormer model as an example, with data from the LibriSpeech corpus and own recordings. The different sequences were mixed together both physically and virtually, corresponding to different recording qualities. The evaluation was based on the Word Error Rate (WER) between the transcripts before and after the separation of the speakers. Even though the experiments do not claim to be scientific due to the small amount of data, the results allow some assumptions to be made: With average WER values of about 30%, speech separation methods are certainly a useful extension of ASR systems. For successful separation and subsequent transcription, the audio quality of the recordings seems to be of enormous importance. The experiments also indicated that the number of microphones and differences in the length of the overlapping sequences do not seem to influence the result significantly.

Inhaltsverzeichnis

1. Einleitung

1.1. Ausgangslage

Das ZHAW- und ETH-Spin-Off *SpinningBytes* bietet die Webapplikation *Interscriber* [1] an, mit welchem Audioaufnahmen von Interviews, Besprechungen und Diskussionen zu Text transkribiert werden können (siehe Abbildung 1.1). Dies funktioniert für mehrere Sprecher[1] in den Sprachen Englisch und Deutsch. Wenn sich die Sprecher in Gesprächssequenzen jedoch gegenseitig ins Wort fallen, kommt das bestehende System an seine Grenzen, da es die überlappenden Stimmen nicht den verschiedenen Sprechern zuordnen kann. Das Entwicklerteam konnte bereits ein Teilproblem lösen, indem es einen Prototyp erstellt hat, der Sequenzen mit überlappenden Sprechern erkennen kann. Was fehlt, ist ein Verfahren, das die Sprecher in den entsprechenden Sequenzen separiert, um daraus zusammenhängende Transkripte der einzelnen Sprecher generieren zu können.

Diese Lücke soll mittels geeigneter *Speech Separation*-Methoden geschlossen werden: Speech Separation bezeichnet die Aufgabe, sich überlappende Sprecher auf einem Input-Audiosignal maschinell in einzelne Audiosignale zu trennen. Was das menschliche Gehörsystem problemlos beherrscht, in diesem Kontext auch als "Cocktailparty-Effekt" [2] bekannt, stellt für Maschinen eine grosse Herausforderung dar. Ursprünglich wurde das Problem vor allem im Gebiet der Signalverarbeitung untersucht, doch in den letzten Jahren konnten durch das Hinzuziehen von supervised Deep Learning-Methoden vielversprechende Fortschritte erzielt werden [3].

1.2. Aufgabenstellung

Die ursprüngliche Aufgabe in der Ausschreibung der Projektarbeit (siehe Anhang C) war bewusst allgemein und offen formuliert. Deshalb musste zunächst in einem iterativen Prozess zwischen den Betreuenden und den Studierenden ermittelt werden, was ein sinnvoller Rahmen für diese Projektarbeit sein könnte. Es haben sich zwei Aufgaben herauskristallisiert:

[1] In Anlehnung an den englischen Begriff *speaker* steht Sprecher im weiteren Verlauf der Arbeit stets für Sprecherinnen und Sprecher.

- Das Gebiet der Speech Separation entwickelt sich sehr rasch, doch es existieren keine aktuellen Übersichtspapers. Aus diesem Grund soll eine Literaturrecherche durchgeführt werden, die einen umfassenden Überblick über das Gebiet gibt und auf State-of-the-Art-Modelle eingeht. Es soll insbesondere auch auf die grundlegende Funktionsweise der Modelle eingegangen werden. Dies soll künftigen Projekt- und Forschungsgruppen an der ZHAW den Einstieg in die Materie erleichtern.
- Es ist schwierig abzuschätzen, ob Speech Separation-Modelle genügend leistungsstark für einen Einsatz in produktiven Speech Recognition-Systemen sind. Um dieser Frage nachzugehen und einzelne Faktoren genauer zu beleuchten, sollen Experimente durchgeführt werden. Es sei darauf hingewiesen, dass die Experimente *keinen* wissenschaftlichen Anspruch erheben, da die Datenmenge und der Zeitrahmen dieser Arbeit dafür zu knapp sind. Vielmehr geht es um erste Eindrücke, die als Basis für Folgeexperimente oder eine Umsetzung in Interscriber dienen könnten.

Abbildung 1.1.: Screenshot der Webapplikation Interscriber.

1.3. Aufbau

In Kapitel 2 wird auf ausgewählte Grundlagen im Bereich Machine Learning und Automatic Speech Recognition eingegangen. Im darauffolgenden Kapitel 3 erfolgt die

Literaturübersicht über den Bereich Speech Separation. In den Kapiteln 4 und 5 werden die Methodik, bzw. die Resultate der Experimente beschrieben. Abschliessend wird in Kapitel 6 ein Fazit über die gesamte Arbeit sowie ein Ausblick abgehandelt.

2. Theoretische Grundlagen

Zum Verständnis der nachfolgenden Ausführungen werden Vorkenntnisse über die Funktionsweise von Convolutional Neural Nets (CNN) und Recurrent Nerual Nets (RNN) vorausgesetzt. In Kapitel 2.1 werden für Speech Separation relevante Methoden aus dem Gebiet Machine Learning vorgestellt. Das Kapitel 2.2 soll einen *groben* Überblick über das Gebiet der Automatic Speech Recognition (ASR) geben.

2.1. Machine Learning

2.1.1. Softmax-Funktion

In künstlichen neuronalen Netzen wird häufig die logistische Sigmoid-Funktion $\sigma(x) = \frac{1}{1+e^{-x}}$ als Aktivierungsfunktion verwendet. Sie kann als Wahrscheinlichkeit interpretiert werden, dass eine binäre Zufallsvariable $X \in \{0, 1\}$ den Wert 1 annimmt. Soll hingegen eine Zufallsvariable mit K statt nur zwei möglichen Werten repräsentiert werden, greift man auf die Softmax-Funktion zurück, die eine Verallgemeinerung der Sigmoid-Funktion ist. Sie ist definiert über:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.1}$$

wobei der Outputvektor $\hat{y}$ an der Stelle $\hat{y}_i$ jeweils die Wahrscheinlichkeit für den i-ten Wert enthält für $i = 1, ..., K$. Softmax wird meist als Aktivierungsfunktion des Output-Layers zur Klassifikation von K verschiedenen Klassen eingesetzt [4].

2.1.2. Long Short-Term Memory (LSTM)

Das Long Short-Term Memory (LSTM) Modell [5, 6] ist eine häufig eingesetzte RNN-Architektur. Im Gegensatz zu herkömmlichen RNNs sind LSTMs fähig, Langzeitabhängigkeiten zu lernen, da sie weniger anfällig für explodierende oder verschwindende Gradienten sind. Ein Layer in einem LSTM-Netz besteht aus sequentiell verbundenen Gedächtnis-Zellen. Der Index t entspricht im Folgenden dem Zeitpunkt bzw. der Position innerhalb der Inputsequenz. Jede Zelle hat einen Cell State c_t und einen Hidden State h_t und besteht aus drei hintereinander geschalteten Gates (siehe Abbildung 2.1). Das

Forget Gate f_t bestimmt, ob der Cell State der vorangehenden Zelle c_{t-1} in c_t mitgespeichert werden soll. Das Input Gate, bestehend aus i_t und $\tilde{c}_t$, reguliert, ob und wie der Inputvektor x_t im Cell State gespeichert werden soll. Der Hidden State der Zelle wird durch das Output Gate o_t bestimmt. Die für RNN charakteristische Rekurrenz zeigt sich also sowohl zwischen als auch innerhalb der Zellen. Für die Forward Propagation ergibt sich:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.2}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.3}$$
$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.4}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_f) \tag{2.5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.6}$$
$$h_t = o_t \odot tanh(c_t), \tag{2.7}$$

wobei W die Gewichtsmatrix der Zelle, b die Biases und $\odot$ die elementweise Multiplikation bezeichnen.

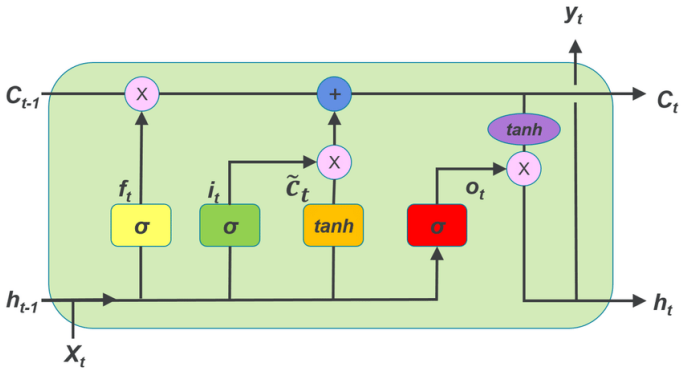


Abbildung 2.1.: Typischer Aufbau einer LSTM-Zelle, wobei $\otimes$ für die elementweise Multiplikation, $\oplus$ für die Addition und σ für die Sigmoid-Funktion stehen. Der Hidden State der vorangehenden Zelle h_{t-1} und der Input-Vektor x_t werden miteinander verkettet (übernommen von [7]).

2.1.3. Transformers

Ein Nachteil von LSTMs (und RNNs im Allgemeinen) ist die notwendige sequentielle Berechnung, da der State an Position t vom vorangehenden State an Position t-1 abhängt. Die 2017 von Vaswani et. al [8] vorgestellte Transformer-Architektur löst dieses Problem. Transformer sind ebenfalls fähig, Langzeitabhängigkeiten zu lernen, kommen jedoch ohne rekurrente Verbindungen aus, was eine effiziente Berechnung durch GPU-Parallelisierung erlaubt. Vaswani et. al schlagen eigentlich eine Encoder-Decoder-Architektur vor, doch

in der vorliegenden Arbeit ist nur der Encoder relevant (linke Hälfte auf Abbildung 2.2). Der Encoder besteht aus mehreren identischen Layern, welche sich jeweils aus einem Self-Attention-Sublayer und einem Fully-connected Feedforward-Sublayer zusammensetzen.

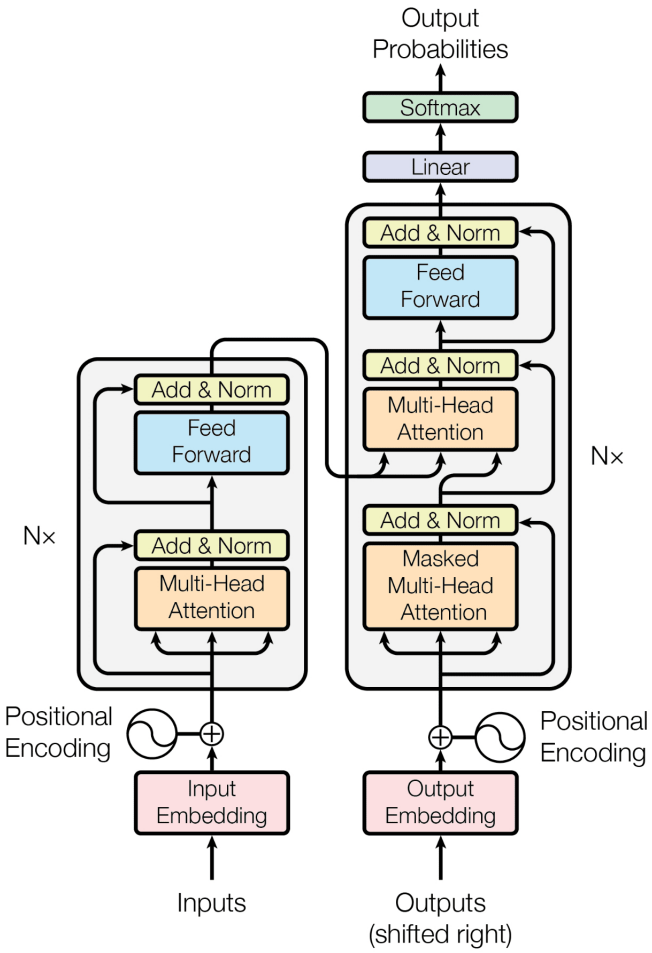


Abbildung 2.2.: Transformer Architektur, hier mit nur einem Encoder-Layer (linke Hälfte) und einem Decoder-Layer (rechte Hälfte). Die Skip Connections und Normierungen verbessern das Resultat (aus [8]).

Positional Encoding

Vor dem ersten Encoder-Layer findet das Positional Encoding statt. Da Transformer Inputsequenzen nicht sequentiell verarbeiten, muss die Information über die Position in der Sequenz auf eine andere Weise modelliert werden. Die Dimensionen der Inputvektoren werden als Sinusoide unterschiedlicher Frequenzen kodiert:

$$PE_{t,2i} = \sin(\frac{t}{10000^{\frac{2i}{d_{model}}}}) \tag{2.8}$$

$$PE_{t,2i+1} = \cos(\frac{t}{10000^{\frac{2i}{d_{model}}}}), \tag{2.9}$$

wobei d_{model} für die Dimension der Input-Vektoren steht. Die PE-Vektoren werden anschliessend zu den Inputvektoren[1] addiert.

Self-Attention-Mechanismus

Ein Attention-Modul lernt drei Gewichtsmatrizen: Query-Gewichte W_q, Key-Gewichte W_k und Value-Gewichte V_k. Die Vektoren einer Inputsequenz werden jeweils je ein Mal mit jeder Gewichtsmatrix transformiert und in den entsprechenden Matrizen Q, K und V zusammengefasst. Aus diesen Matrizen lässt sich die Scaled Dot-Product Attention berechnen:

$$Attention(Q, K, V) = softmax(\frac{QK^\intercal}{\sqrt{d_k}})V, \tag{2.10}$$

wo $d_k(= d_q)$ für die Dimension der Key-Vektoren steht. Die intuitive Idee dahinter ist, dass eine Query mit den verschiedenen Keys verglichen wird. Die Multiplikation von Q mit $K^\intercal$ ergibt eine Ähnlichkeitsmatrix, welche als Gewichtung für die Values dient. Die Queries, Keys und Values beziehen sich alle auf dieselbe Inputsequenz (daher kommt die Bezeichnung *Self*-Attention). Der Attention-Score eines Vektors abstrahiert also die Beziehung zu den anderen Vektoren der Sequenz.

Beim Transformer werden die Queries, Keys und Values zuerst in h gleichdimensionale Vektoren aufgeteilt. Die Attention-Berechnung wird dann h Mal parallel durchgeführt, was als Multi-Head Attention bezeichnet wird. So werden bei jedem *Head* eigene Gewichtsmatrizen gelernt, wodurch unterschiedliche Aspekte erfasst werden sollten. Die Outputs der Heads werden anschliessend wieder verkettet und weiterverarbeitet.

Feedforward-Netz

Das Feedforward-Netz besteht aus zwei linearen Transformationen mit einer Rectifier-Aktivierungsfunktion (ReLU) dazwischen:

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2. \tag{2.11}$$

[1] Eigentlich wird auf den Inputvektoren zuvor noch eine Einbettung (Input Embeding) durchgeführt. Bei den in dieser Arbeit diskutierten Modellen ist dies jedoch nicht erforderlich.

2.2. Automatic Speech Recognition (ASR)

2.2.1. Einleitung

Automatic Speech Recognition (ASR), auch bekannt als Speech-to-Text, ist ein interdisziplinäres Forschungsgebiet, das sich damit beschäftigt, Systeme zu entwickeln, die gesprochene Sprache erkennen und in Text umwandeln. Die Variabilität der Aussprache zwischen verschiedenen Sprechern und die Komplexität der menschlichen Sprache machen das Problem anspruchsvoll [9].

Formal soll die Wortsequenz $W = [w_1, ..., w_L]$ mit der höchsten Plausibilität gegeben eines akustischen Sprecherinputs $X = [x_1, ..., x_t]$ ermittelt werden:

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|X). \tag{2.12}$$

Durch Anwendung des Satzes von Bayes kann die Formel umgeschrieben werden zu:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \frac{P(X|W)P(W)}{P(X)} = \underset{W}{\operatorname{argmax}} P(X|W)P(W). \tag{2.13}$$

Die Wahrscheinlichkeit $P(X)$ kann ignoriert werden, da sie konstant ist.

2.2.2. Architektur

Konventionelle Systeme bestehen aus statistischen Modellen für Akustik, Aussprache und Sprache, die separat trainiert werden, sowie einem Decoder. Das Aussprache-Modell ist klassischerweise ein statistisches Wort-Aussprache-Wörterbuch, welches Phonem-Sequenzen für Wörter enthält. Phoneme stellen die kleinste bedeutungsunterscheidende Einheit einer gesprochenen Sprache dar. Das grobe Zusammenspiel zwischen den Komponenten funktioniert wie folgt: Das Inputsignal wird zunächst segmentiert und in einen Frequenzbereich transformiert (X). Gemeinsam mit dem Aussprache-Modell kann das Akustik-Modell die Wahrscheinlichkeiten $P(X|W)$ berechnen. Zusätzlich liefert das Sprach-Modell Wahrscheinlichkeiten für verschiedene Wortsequenzen $P(W)$. So kann zwischen phonetisch ähnlich klingenden Wortsequenzen mit verschiedenen Bedeutungen unterschieden werden. Am Schluss ermittelt der Decoder mittels Such-Algorithmen die plausibelste Wortsequenz $\hat{W}$ als Outputtext.

Lange Zeit dominierten Hidden Markov Models (HMM), die in Kombination mit Deep Neuronal Networks (DNN) noch heute in den meisten Systemen eingesetzt werden [10]. Als alternative Architektur wird aktuell an sogenannten *end-to-end*-Modellen geforscht,

welche beinahe gleich gute Resultate erzielen [11], in ihrem Aufbau jedoch deutlich weniger komplex sind. Sie bestehen aus einem einzigen statt drei separaten Modellen, welches für ein Inputaudio direkt einen Outputtext lernt.

2.2.3. Word Error Rate (WER)

Die Standard-Evaluationsmetrik für ASR-Systeme ist die Word Error Rate (WER). Sie drückt aus, wie stark eine Wortsequenz von einem Referenz-Transkript abweicht:

$$WER = 100 \cdot \frac{I + D + S}{N}, \tag{2.14}$$

wobei I die Anzahl Einfügungen, D die Anzahl Löschungen, S die Anzahl Ersetzungen und N die totale Anzahl Wörter im Referenz-Transkript bezeichnen. Der obere Teil des Bruchs kann als Levensthein-Distanz auf Wort- statt auf Buchstaben-Ebene interpretiert werden. Man beachte, dass die WER auch Werte über 100% annehmen kann, beispielsweise wenn die Wort-Sequenz mehr Wörter als das Referenz-Transkript enthält.

3. Speech Separation: Literaturüberblick

3.1. Einleitung

Single-Channel (d. h. monaurale) Speech Separation bezeichnet das Problem, zwei oder mehrere überlappende Stimmen einer Input-Audiodatei voneinander zu trennen. Das Ziel ist eine möglichst genaue Abschätzung der Ursprungssignale. Der Begriff wird in der Literatur nicht ganz eindeutig verwendet: Manche Autoren (vgl. bspw. [3]) verwenden Speech Separation auch als Überbegriff für sämtliche Methoden, welche eine Stimme von einem Hintergrundrauschen trennen. Der in dieser Arbeit behandelte Fall der Separation von verschiedenen Stimmen wird dann als *Speaker Separation* präzisiert.

Ursprünglich war Speech Separation im Bereich der digitalen Signalverarbeitung angesiedelt. Huang et. al [12] haben 2014 als erste Gruppe Deep Learning als Problemlösung vorgeschlagen. Seither dominieren im Bereich Speech Separation supervised Deep Learning-Ansätze. Das Gebiet entwickelt sich rasch und kontinuierlich: Sämtliche State-of-the-Art (SOTA) Modelle inklusive ihrer Vorgänger sind höchstens zwei bis drei Jahre alt. Es existieren Übersichtpapers, doch die Literaturrecherche der vorliegenden Arbeit hat ergeben, dass das aktuellste (Wang et. al [3]) aus dem Jahr 2018 stammt und somit bereits veraltet ist.

Für das bessere Verständnis wird zunächst eine Einführung in die Evaluationsmetriken und die wichtigsten Datensätze gegeben. Im darauffolgenden Kapitel wird auf die wichtigsten Meilensteine eingegangen, welche die Grundlagen für den SOTA gelegt haben. Dabei soll insbesondere am Beispiel von TasNet [13] eine Intuition für die typische Funktionsweise von Speech Separation-Modellen vermittelt werden. Anschliessend wird der SOTA vorgestellt. Das letzte Kapitel behandelt die neu entstehende Schnittmenge zwischen Speech Separation und ASR.

3.2. Grundlagen

3.2.1. Evaluation

Speech Separation lässt sich formal als Problem formulieren, bei dem C Ursprungssignale $s_1(t), ..., s_c(t)$ von einem diskreten Inputsignal[1] $x(t)$ abgeschätzt werden sollen:

$$x(t) = \sum_{i=1}^{C} s_i(t). \tag{3.1}$$

Wie nahe eine Schätzung $\hat{s}$ am Ursprungssignal s liegt, kann anhand der Evaluationsmetriken Source-to-Distortion Ratio (SDR) ermittelt werden. Diese Metrik wird in Dezibel (dB) ausgedrückt und gibt ein Verhältnis zwischen dem Ursprungssignal s und dem Fehler der Schätzung e_{noise} $(= s - \hat{s})$ an [14]:

$$SDR(s, \hat{s}) = 10 \log_{10} \left(\frac{||s||^2}{||s - \hat{s}||^2} \right). \tag{3.2}$$

Le Roux et. al [15] wenden ein, dass sich die SDR künstlich verstärken lasse, indem $\hat{s}$ neu skaliert wird. Sie schlagen deshalb vor, in Zukunft die robustere Variante Scale-invariant SDR (SI-SDR) zu verwenden, die sich seither als die relevanteste Evaluations-Metrik etabliert hat. Bei der SI-SDR wird s mit einem Faktor skaliert, so dass der neue Vektor s_{target} bei der Orthogonalprojektion von $\hat{s}$ auf s endet (siehe Abbildung 3.1).

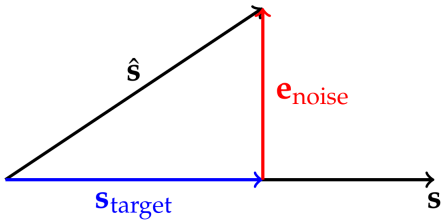


Abbildung 3.1.: Orthogonalprojektion von $\hat{s}$ auf s mit s_{target} und dem neuen e_{noise} als Resultat (aus [16]).

Also [13]:

$$s_{target} = \frac{\langle \hat{s}, s \rangle \cdot s}{||s||^2}, \tag{3.3}$$

[1] Wenn im Folgenden von einem Signal gesprochen wird, ist damit die Diskretisierung des Schwingungsverlaufs in Form eines Vektors, bzw. eines 1D-Arrays von Zahlen, gemeint.

und somit:

$$SI\text{-}SDR(s, \hat{s}) = 10 \log_{10} \left(\frac{||s_{target}||^2}{||s_{target} - \hat{s}||^2} \right). \tag{3.4}$$

Bei Perfomance-Tests von Modellen wird die SI-SDR als SI-SDR Improvement (SI-SDRi) angegeben. Dies ist die Differenz zwischen der durchschnittlichen SI-SDR der Schätzungen $\hat{s_1}, ..., \hat{s}_C$ zu der durchschnittlichen SI-SDR vom Inputsignal x (jeweils bezüglich den Ursprungssignalen $s_1, ..., s_C$):

$$SI\text{-}SDRi(s, \hat{s}, x) = \frac{1}{C} \sum_{i=1}^{C} SI\text{-}SDR(s_i, \hat{s}_i) - SI\text{-}SDR(s_i, x). \tag{3.5}$$

Es sei darauf hingewiesen, dass die SI-SDR von vielen Autoren auch als SI-SNR bezeichnet wird [15]. Dies liegt vermutlich an den Methoden-Namen des Toolkits *bss_eval* [17], welches üblicherweise für die Evaluation verwendet wird.

3.2.2. Datensätze

Es gibt zwei Gruppen von Standard-Datensätzen im Bereich Speech Separation. Die erste und zurzeit dominante Gruppe basiert auf dem lizenzpflichtigen WSJ0-Korpus [18], auch bekannt als CSR-I. Dieser enthält Aufnahmen in Studio1ualität von Texten des Wall Street Journals. Daraus werden die WSJ0-mix-Datensätze [19] generiert, indem zufällig Korpus-Aufnahmen künstlich zusammengemischt werden. Die Zahl vor dem "mix" gibt an, wie viele Sprecher übereinandergelegt wurden (bspw. wsj0-3mix bei drei Sprechern). WHAM! [20] ist eine Erweiterung von wsj0-2mix, bei welcher reale Hintergrundsgeräusche durch Aufnahmen aus Restaurants, Bars, etc. beigemischt werden. WHAMR! [21] ist WHAM! mit Nachhall-Effekt.

Die zweite Gruppe heisst LibriMix [22] und wird aus dem lizenzfreien Korpus LibriSpeech [23] generiert, welcher aus Hörbuch-Aufnahmen besteht. Diese Aufnahmen werden analog zu den WSJ0-mix-Datensätzen zufällig übereinandergelegt und je nach Variante mit WHAM-Rauschen hinterlegt. LibriMix gibt es als Libri2Mix mit zwei und Libri3Mix mit drei parallelen Sprechern. Diese Versionen können jeweils als Variante mit Rauschen (mix_both) und ohne Rauschen (mix_clean) generiert werden.

Die verschiedenen WSJ0-mix- und die LibriMix-Datensätze gibt es jeweils als *min*- und *max*-Variante. Bei der *min*-Variante werden die längeren Aufnahmen auf die Länge der kürzesten Aufnahme gekürzt. Umgekehrt werden bei der *max*-Variante die kürzeren Aufnahmen mit Null-Werten (Stille) auf die Länge der längsten Aufnahme ausgedehnt. Ein Grossteil der Autoren verwendet die *min*-Variante und trainiert somit die Modelle auf den eher unrealistischen Fall von sich (nahezu) vollständig überlappenden Sequenzen [20]. Des Weiteren werden die Datensätze aus Performance- und Speicher-Gründen

typischerweise auf 8 kHz heruntergetaktet, wodurch ein grosser Teil des Sprachspektrums ignoriert wird [20].

Die populärste Benchmark ist das Resultat der Evaluationsmetrik SI-SDR auf dem WSJ0-2mix-Datensatz in der *min*-Variante mit 8 kHz Samplingrate.

3.3. Meilensteine

3.3.1. Time-Domain Audio Separation Network (TasNet)

Einer der grossen Meilensteine der letzten Jahre im Bereich Speech Separation war das Time-Domain Audio Separation Network (TasNet) [13]. Die Methoden vor TasNet führten typischerweise eine Short-Time Fourier Transformation (STFT) auf dem Inputsignal durch ([12, 24, 25]). Dies bringt verschiedene Nachteile mit sich, beispielsweise dass der Separations-Algorithmus sowohl mit der Amplitude als auch mit der Phase eines transformierten Signals umgehen muss. Beim TasNet wird das Inputsignal hingegen mittels eines Autoencoders direkt im Zeitbereich modelliert. Die meisten SOTA-Modelle orientieren sich seither an dieser charakteristischen Encoder-Masker-Decoder-Architektur (siehe Abbildung 3.2): Der Encoder lernt eine Repräsentation des Inputsignals. Das Masker-Netzwerk schätzt optimale Masken für die Ursprungssignale ab und der Decoder rekonstruiert daraus die Ursprungssignale.

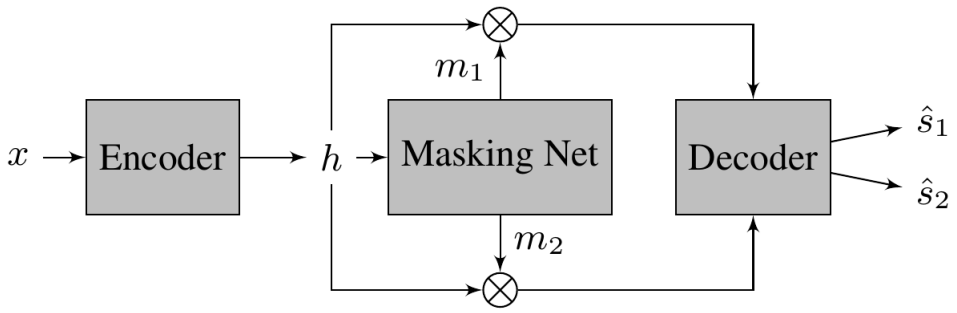


Abbildung 3.2.: Typische Architektur von Speech Separation-Modellen, hier mit zwei Sprechern (aus [26]).

Problem

Zunächst wird das Inputsignal x in K nicht-überlappende Vektoren s_i der Länge L aufgeteilt und normiert (K variiert je nach Inputlänge). Jedes Segment (des Inputs- und der Ursprungssignale) kann als nicht-negative Summe von N Basissignalen $B = [b_1, ..., b_N] \in \mathbb{R}^{N \times L}$ dargestellt werden:

$$\begin{cases} x = wB \\ s_i = d_i B \end{cases} \tag{3.6}$$

wobei $w \in \mathbb{R}^{1\times N}$ der Gewichtevektor des Inputvektors und $d_i \in \mathbb{R}^{1\times N}$ die Gewichtevektoren der Ursprungssignale sind. Das Ziel der Separation ist also, die Gewichte d_i der Ursprungssignale abzuschätzen.

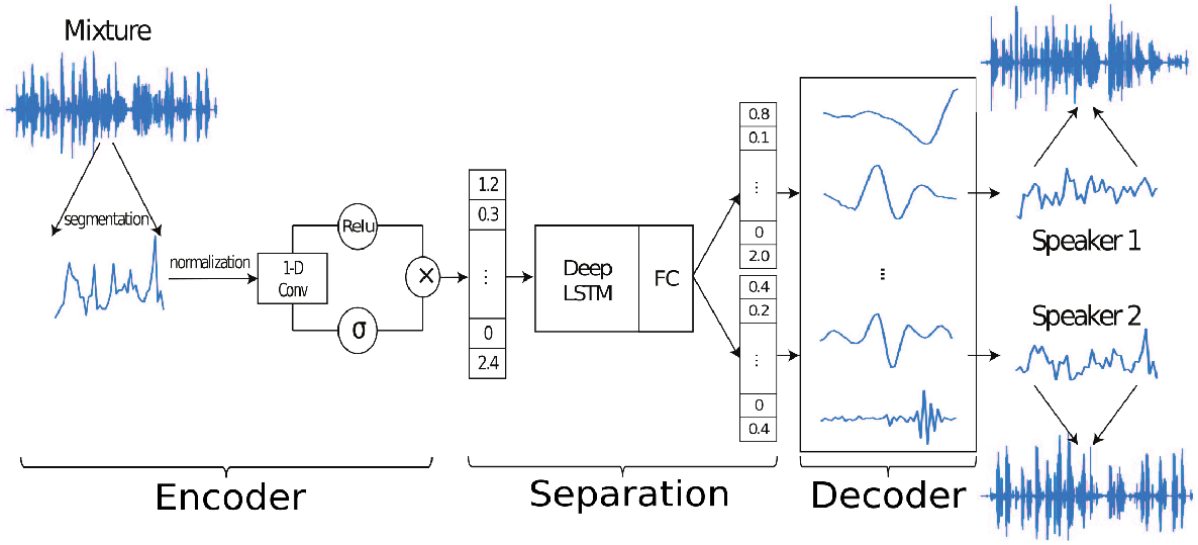


Abbildung 3.3.: Architektur des TasNet (aus [13]).

Autoencoder

Das Signal wird mit einem Convolutional Autoencoder modelliert. Für die Abschätzung des Gewichtevektors w_k für ein Segment k von x wird ein Gated[2] Convolutional 1D-Layer (mit L Eingangs- und N Ausgangskanälen) als Encoder verwendet:

$$w_k = \underbrace{ReLU(x_k * U)}_{\text{convolution}} \odot \underbrace{\sigma(x_k * V)}_{\text{gate}} \tag{3.7}$$

wobei $\odot$ für die elementweise Multiplikation steht. Die Matrizen $U \in \mathbb{R}^{N\times L}$ und $V \in \mathbb{R}^{N\times L}$ bestehen jeweils aus N Vektoren (Basissignale) der Segmentlänge L.

Mit dem Maskierungsvektor $m_{i,k}$ für ein Segment k eines Ursprungssignales s_i kann der Decoder dessen Gewichtematrix $d_{i,k}$ berechnen:

$$d_{i,k} = w_k \odot m_{i,k} \tag{3.8}$$

[2]Zusätzliche parallele Faltung mit der Sigmoid-Aktivierungsfunktion. Für weitere Informationen siehe Dauphin et. al [27].

und damit das gesuchte Segment des Ursprungssignals über die Multiplikation mit der Basissignal-Matrix B:

$$s_{i,k} = d_{i,k}B. \tag{3.9}$$

Separation

Für die Ermittlung der Sprecher-Masken wird ein deep LSTM-Netzwerk verwendet mit einem anschliessenden Fully-connected Layer mit Softmax-Aktivierungsfunktion. Der Input des Netzwerks ist die Sequenz von k Gewichtevektoren der Segmente und der Output sind die k Maskierungsvektoren $m_1, ..., m_k \in \mathbb{R}^{1xN}$.

Das Training-Ziel ist die Minimierung der SI-SDR mit der PIT-Loss-Function (siehe Kapitel 3.3.2). Abbildung 3.3 gibt einen Überblick über den gesamten TasNet-Aufbau.

3.3.2. Loss Function: Permutation Invariant Training (PIT)

Frühere Ansätze trainierten die Modelle über die Minimierung des Mean Square Errors (MSE) bezüglich der Masken oder direkt auf den Signalen [28]. Da die Modelle jedoch mehrere Output-Layer haben (einen pro Ursprungssignal), stellt sich das Problem, dass nicht klar ist, welcher Output $\hat{s_i}$ zu welchem Ursprungssignal s_i gehört. Als Lösung für dieses Permutations-Problem hat sich das Permutation Invariant Training (PIT) [25] durchgesetzt: Es wird paarweise der MSE zwischen jedem $|\hat{s_i}|$ und $|s_i|$ berechnet. Bei jeder Permutation werden die MSE addiert und die Zuweisung mit dem niedrigsten totalen MSE wird für die Loss Function übernommen. Es wird also gleichzeitig eine Fehler-Abschätzung und eine Label-Zuweisung durchgeführt. Auf Abbildung 3.4 ist PIT am Beispiel eines Modells mit zwei Ursprungssignalen zu sehen.

PIT wird für jedes Segment x_k eines Inputsignals x durchgeführt. Dabei muss damit gerechnet werden, dass die Label-Zuweisung zwischen den einzelnen Segmenten unterschiedlich ist. Deshalb verwenden SOTA-Modelle in der Regel die leichte Abwandlung Utterance-level PIT (uPIT) [29] als Trainings-Kriterium. Bei uPIT werden die paarweisen MSEs für *alle* Segmente eines Inputsignales ("Utterance") berechnet. Es wird dieselbe *optimale* Permutation für alle Segmente des Inputsignales als Zuweisung für die Loss Function übernommen.

Bei PIT-Training mit SI-SDR wird die Minimierung des MSE durch die Maximierung der SI-SDR ersetzt. Als Loss Function ergibt sich:

$$\mathcal{L} = -SI\text{-}SDR(s^*, \hat{s}) \tag{3.10}$$

mit s^* als Permutation mit der maximalen SI-SDR.

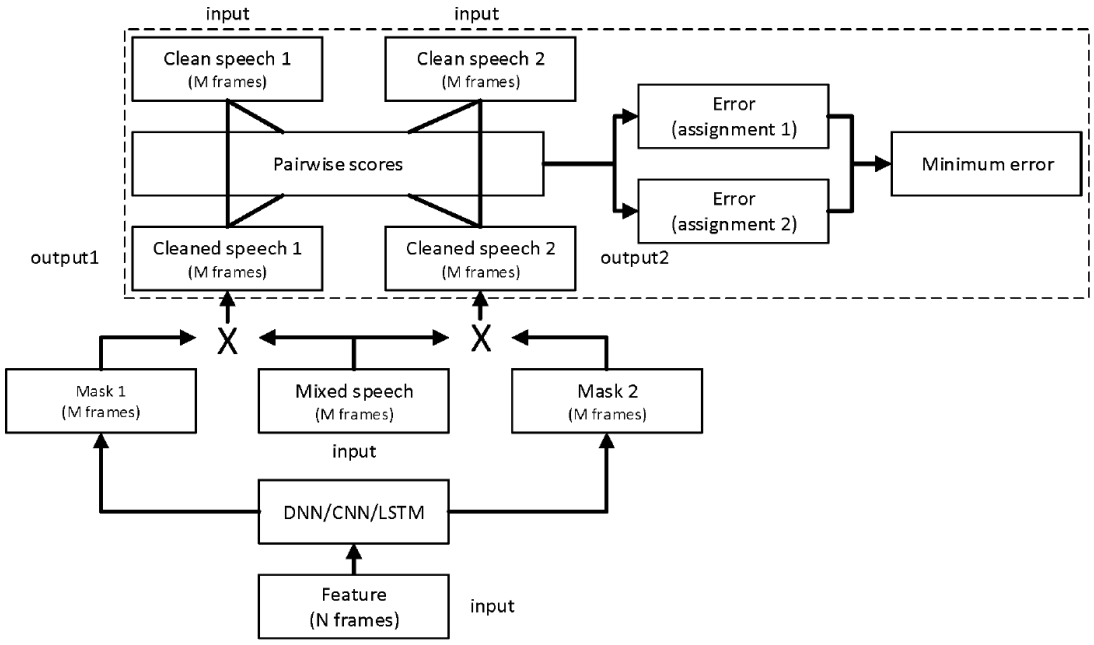


Abbildung 3.4.: PIT-Training am Beispiel eines Modells mit zwei Ursprungssignalen (aus [25]).

3.3.3. Dual-Path RNN (DPRNN)

Der nächste Meilenstein nach dem TasNet war das Dual-Path RNN (DPRNN) [30]. Ein Problem des TasNets und seinen Nachfolgern [31, 32] ist, dass das Lernen von Langzeitabhängigkeiten bei langen Inputsequenzen schwierig ist. Aus diesem Grund wird beim DPRNN die in N Basissignale encodete Inputsequenz in S sich überschneidende Abschnitte ("Chunks") der Länge K segmentiert. Die Schrittweite H entscheidet über die Ausprägung der Überschneidung. Diese Chunks werden für ein Inputsignal in einem 3D-Tensor $T \in \mathbb{R}^{N \times K \times S}$ gebündelt zuerst durch ein *Intra-Chunk* RNN und anschliessend durch ein *Inter-Chunk* RNN gereicht. Das Intra-Chunk RNN verarbeitet die Chunks analog zu den früheren Architekturen einzeln entlang der Zeit-Dimension K. Das Inter-Chunk RNN hingegen verarbeitet das gesamte Inputsignal über alle Chunks gemeinsam entlang der Dimension S. Beim Intra-Chunk RNN werden also *lokale* und beim Inter-Chunk RNN *globale* Abhängigkeiten modelliert.

Ein DPRNN-Block mit den zwei RNNs kann für eine grössere Tiefe des Netzes mehrfach hintereinander geschaltet werden. Abbildung 3.5 zeigt einen solchen Block: Die Richtungen der Striche bei den RNN-Inputs visualisieren die Anwendung der RNNs entlang unterschiedlicher Dimensionen des Tensors.

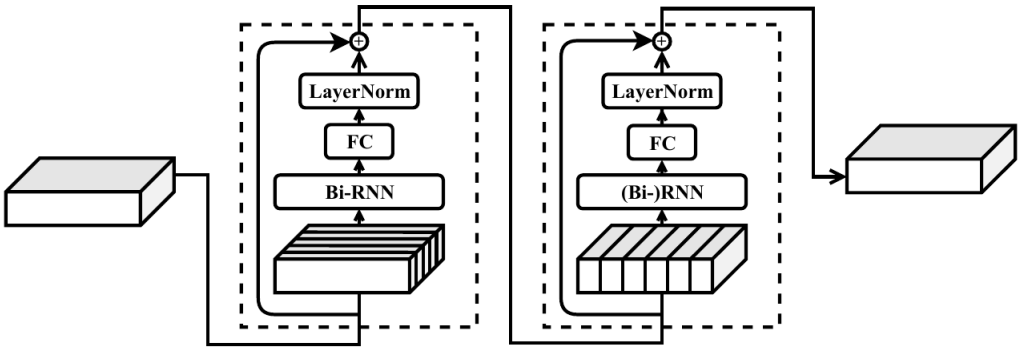


Abbildung 3.5.: DPRNN-Block, bestehend aus dem *Intra-Chunk* RNN (links) und dem *Inter-Chunk* RNN (rechts) (aus [30]).

3.4. State of the Art

3.4.1. Transformer-basierte Modelle

Die folgenden SOTA-Modelle basieren direkt auf der mit DPRNN eingeführten Dual-Path-Architektur. Beim Dual-Path Transformer[3] Network (DPTNet) [33] wurden die RNNs durch sogenannte "Improved Transformers" ersetzt. Bei dieser Art von Transformern wird ein RNN in das Feedforward-Netz eingebettet. Analog zum DPRNN besteht jeder DPTNet aus einem *Inter-Transformer* und einem *Intra-Transformer*. Wie auf Tabelle 3.1 ersichtlich ist, führt das bereits zu besseren Resultaten auf der Standard Benchmark für zwei Sprecher WSJ0-2mix. Durch die Verwendung von RNNs können allerdings die Performance-Vorteile durch Parallelisierung der Transformer nicht genutzt werden. Aus diesem Grund schlägt SepFormer [26] ein RNN-freies Dual-Path-Netzwerk aus Transformern vor. Dies schlägt sich in Verbesserungen bei der Berechnungszeit und dem Memory-Verbrauch nieder. Zudem ist es mit einer SI-SDRi von 22.3 zurzeit das beste Modell auf der WSJ0-2mix Benchmark.

3.4.2. Transformer-freie Modelle

Es erzielen auch Transformer-freie Weiterentwicklungen der Dual-Path Architekturen SOTA-Ergebnisse: Bei Sandglassset [34] wird die Granularität der Inputsequenz bei jedem Dual-Path-Block vergröbert (d.h. grössere Chunks) und ab der Hälfte dann wieder verfeinert. Dies verleiht dem Model die namensgebende Sandglas-förmige Struktur. Weiter erwähnenswert ist das Gated DualPathRNN [35], welches im Vergleich zu den anderen Modellen auch auf Datensätzen mit mehr als drei Sprechern gut funktioniert. Doch es

[3]Der Begriff *Transformer* bezeichnet im Folgenden stets nur den Encoder der ursprünglichen Transformer-Architektur aus [8].

erzielen auch Architekturen gute Resultate: Bei Wavesplit [36] werden mittels Clustering SOTA-Werte erreicht.

Modell	SI-SDRi	Jahr
SepFormer	22.3	2020
Wavesplit	22.2	2020
Sandglasset	21.0	2021
DPTNet	20.2	2020
Gated DualPathRNN	20.1	2020
DPRNN	18.8	2019
Conv-TasNet	15.3	2018
TasNet	10.8	2017

Tabelle 3.1.: WSJ0-2mix Benchmark (Zahlen übernommen von [37]).

3.5. Speech Separation für ASR

In jüngster Zeit gab es vermehrt Bestrebungen, Speech Separation bezüglich des realistischen Anwendungsfalls der ASR zu trainieren und zu evaluieren. Gemäss Chen et. al korrelieren die Evaluationsmetriken aus der Signalverarbeitung (SDR und SI-SDR) nur schwach mit der WER aus der ASR [38]. Sie schlagen auf der Grundlage von [39] den neuen Task *Continous Speech Separation* (CSS) auf *kontinuierlichen* Aufnahmen vor. Der dazugehörige Datensatz LibriCSS enthält im Vergleich zur klassischen Speech Separation sehr lange Sequenzen mit einer unbekannten Anzahl an Sprechern, die sich nur *teilweise* überlappen. Das Ziel ist eine möglichst tiefe WER. Aufgrund der äusserst jungen Geschichte von CSS und der damit verbundenen kleinen Anzahl an Modellen fällt es schwer, den SOTA zu charakterisieren. Bereits gute Ansätze zeigen [40] und [41].

Fernab von CSS gibt es weitere Versuche einer Synthese von Speech Separation und ASR. Der von Wang et. al vorgeschlagene VoiceFilter [42] trainiert zusätzlich einen Speaker Encoder mit einer separaten Referenzaufnahme eines Zielsprechers. Mit Hilfe des Outputs dieses Encoders wird dann auf dem gemischten Inputsignal der Zielsprecher von den restlichen Sprechern und Rauschen separiert. Ist man an den separierten Aufnahmen aller Sprecher interessiert, kann das Prozedere für jeden Sprecher wiederholt werden. Auch hier erfolgte die Evaluation anhand der WER.

4. Methoden

4.1. Ausgangslage

Das Ziel der Experimente dieser Arbeit ist, einen ersten Eindruck zu gewinnen, ob sich der SOTA der Speech Separation bereits in ASR-Systemen einsetzten liesse und welche Faktoren dabei eine Rolle spielen. Anhand erster praktischer Versuche zeigte sich jedoch rasch, dass die Separier-Leistung von Speech Separation-Modellen auf realen Aufnahmen schlechter zu sein scheint als auf den Standard-Datensätzen (vgl. Kapitel 3.2.2), bei denen die Sprecher virtuell übereinandergelegt werden. Gründe könnten die schlechtere Mikrofon-Qualität oder erwünschte Nachhall-Effekte sein, wobei die Separation bei virtuell hinzugefügtem Nachhallen und Lärm nach wie vor ziemlich gut funktioniert [43]. Es musste für die Experimente also eine Lösung gefunden werden, welche dem realistischen Anwendungsfall näher kommt als die virtuelle Überschneidung. Im Folgenden werden die beiden verwendeten Varianten beschrieben.

4.2. Varianten

4.2.1. Variante 1: Abspielen und erneut aufnehmen

Ausgangpunkt für diese Variante sind Originalaufnahmen von den Sprechern ohne Überschneidung. Diese werden gleichzeitig auf identischen Lausprechern abgespielt und von einem Mikrofon erneut aufgenommen. So können auch Aufnahmen aus existierenden Datensätzen verwendet werden. Im Verlauf der Experimente hat sich jedoch gezeigt, dass beim Abspielen und erneuten Aufnehmen wohl spektrale Charakteristiken der Stimmen verloren gehen. Für das menschliche Gehör sind die separierten Signale zwar gut verständlich, doch die ASR scheint erhebliche Mühe bei der Transkription zu haben, wie sich bei den Resultaten der Experimente zeigen wird. Aus diesem Grund wurden die Experimente bei dieser Variante als Vergleich jeweils auch noch virtuell übereinandergelegt. Die Lautsprecher-Variante soll dann den Worst-Case bezüglich Audioqualität in einer realistischen Anwendung abdecken und die virtuelle Variante den Best-Case. Im Normalfall ist bei praktischen Beispielen eine Qualität irgendwo dazwischen zu erwarten.

Die Evaluations-Metrik ist die WER zwischen den ASR-Transkripten des Signals vor dem Abspielen und Wiederaufnehmen und den ASR-Transkripten der separierten Signale. So

kann der Anteil an der WER des verwendeten ASR-Systems aufgrund seiner Imperfektion auf ein Minimum reduziert werden. Es wird nur der Fehler gemessen, der durch die Separation entstanden ist.

4.2.2. Variante 2: Direkt aufnehmen

Eine andere Möglichkeit besteht darin, direkt neue Audiodateien mit Überschneidungen aufzunehmen. Dies entspricht dem realistischen Anwendungsfall, doch der damit verbundene Aufwand, würde den Rahmen dieser Arbeit sprengen. Schliesslich muss jede Audiodatei manuell gesprochen und aufgezeichnet werden. Zudem fehlen dann die getrennten Originalaufnahmen der Sprecher, da nur die überschnittene Version existiert, die erst durch Speech Separation wieder getrennt werden kann.

Bei dieser Variante ist die Evaluations-Metrik die WER zwischen den gesprochenen Texten und den entsprechenden ASR-Transkripten der separierten Signale. Im Vergleich zu Variante 1 fliesst hier auch der Fehler, der bei der ASR entsteht, in die WER ein.

4.3. Setup

Modelle

Das verwendete ASR-System für die Transkription ist die Google Speech-to-Text API [44]. Für die Separation wurde das führende SOTA-Modell SepFormer verwendet, welches im Open Source Python-Toolkit [45] SpeechTrain enthalten ist. Davon wurden auf WSJ0-2mix und WHAMR! vortrainierte Modelle verwendet, jeweils in der 8kHz *min* Variante.

Hardware

Für die Variante 1 “abspielen und erneut aufnehmen” wurden zwei Freisprechanlagen des Typs Jabra Speak 510 als Lautsprecher verwendet. Das Mikrofon für das Wiederaufnehmen war vom Typ Studio Projects LSM. In Anhang B.2 ist ein Bild des Aufbaus zu finden.

Für die Experimente mit eigenen Aufnahmen, wurden je nach Fall andere Mikrofone verwendet. Waren gerichtete Mikrophone erforderlich, so wurden Steckmikrofone des Typs Sony ECM-CS3 verwendet. Als Raummikrofon wurde wiederum das Jabra Speak 510 verwendet. Bei den Beschrieben der Experimente in Kapitel 5 wird die Mikrofonart jeweils angegeben.

Raum

Die Experimente wurden in einem circa 15 Quadratmeter grossen möblierten Raum durchgeführt. Es waren keine schalldämmenden Elemente verbaut.

4.4. Toolkit

Für die Ausführung und Dokumentation der Experimente wurde im Rahmen dieser Arbeit das generische *Speech Separation Toolkit* geschrieben. Eine ausführliche Dokumentation befindet sich in Anhang A. Mit dem Toolkit können automatisiert Speech Separation-Experimente auf dem GPU-Cluster der ZHAW durchgeführt werden. Es bietet Funktionen für die physische (Lautsprecher) und virtuelle Überschneidung einer Menge von Audiodateien an. Diese werden anschliessend auf dem Cluster separiert und je nach Bedarf können ASR-Transkripte und WER-Werte abgefragt werden. Die Resultate eines vollendeten Experiments werden zusammengefasst und inklusive Input-Daten an einem gemeinsamen Ort abgespeichert, damit sie auch zu einem späteren Zeitpunkt nachvollziehbar sind (für die Resultate dieser Arbeit siehe Anhang B.1). Abbildung 4.1 zeigt schematisch den Ablauf. Es werden Speech Separation-Modelle der Open Source-Toolkits SpeechBrain [45] und Asteroid [46] unterstützt. Letzteres enthält Implementationen der meisten SOTA-Modelle (mit Ausnahme von SepFormer).

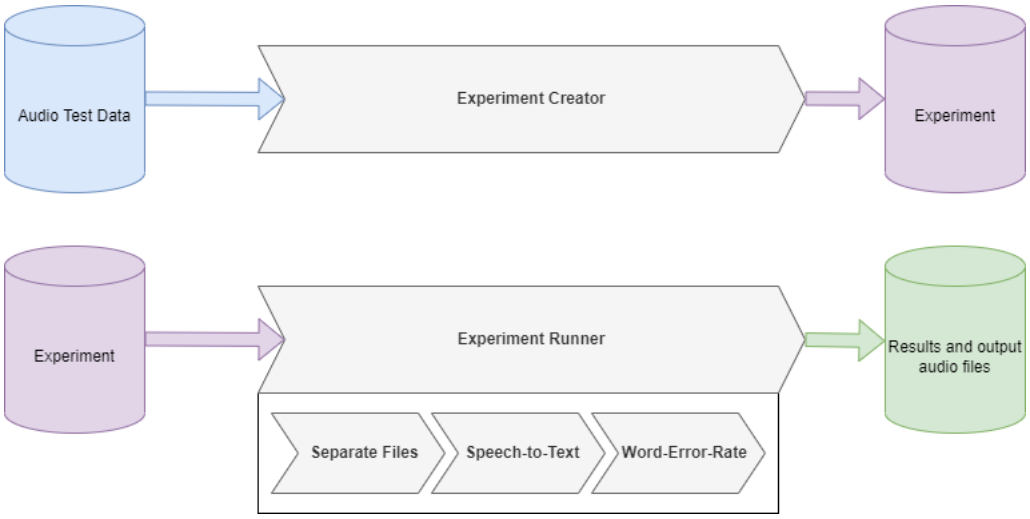


Abbildung 4.1.: Schematischer Ablauf eines Exeperiments mit dem *Speech Separation Toolkit*.

5. Resultate

Es wurden nur Experimente mit zwei Sprechern durchgeführt. Dass in einem realen Szenario mehr als zwei Sprecher gleichzeitig reden, scheint eher unwahrscheinlich. Weiter erfolgt an dieser Stelle nochmals der Hinweis, dass die in diesem Kapitel vorgestellten Experimente *keinen* wissenschaftlichen Anspruch erheben. Dafür ist die Datenmenge zu knapp und die Aufnahmequalität unzureichend. Es soll vielmehr ein erster Eindruck über den möglichen Nutzen von Speech Separation in der ASR vermittelt werden, auf dessen Basis Folgeexperimente oder praktische Umsetzungen durchgeführt werden können.

5.1. Experiment 1: LibriSpeech

Inhalt

Dies ist das zentrale Experiment der vorliegenden Arbeit mit der grössten Datenmenge. Hier geht es um einen allgemeinen Eindruck über die Einsetzbarkeit von Speech Separation in der ASR. In den anderen Experimenten werden dann einzelne Aspekte untersucht.

Setup

Das Experiment erfolgte nach Variante 1 "abspielen und erneut aufnehmen" (siehe Kapitel 4.2.1). Es wurden zufällig 400 Aufnahmen des Korpus LibriSpeech [23] gewählt, woraus 200 Paare gebildet wurden. Die Sprecher überschneiden sich meistens nicht durchgehend, da die Originalaufnahmen unterschiedlich lang sind.

Resultat

	Lautsprecher		**Virtuell**	
WER [%]	*Mittelwert*	*Median*	*Mittelwert*	*Median*
SepFormer WSJ0-2mix	69.1	60.4	31.4	23.9
SepFormer WHAMR!	65.1	52.9	40.2	34.8

Tabelle 5.1.: Resultate des Experiments 1 "LibriSpeech", n=200.

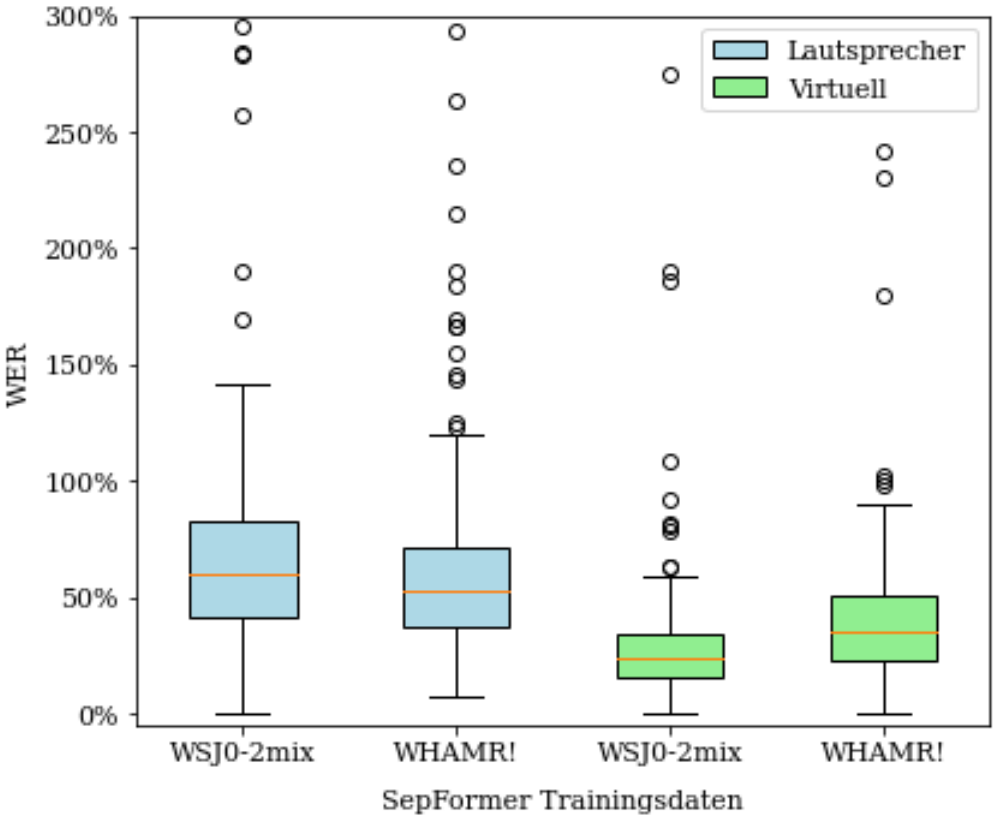


Abbildung 5.1.: Boxplots des Experiments 1 “LibriSpeech”, n=200.

Fazit

Die Werte der Lautsprecher-Variante müssen wie bereits in Kapitel 4.2.1 beschrieben mit Vorsicht betrachtet werden. Sie müssen als Praxisfälle mit unterdurchschnittlicher Audioqualität interpretiert werden. Der Medianwert von rund 24% beim Best-Case verspricht einen potentiellen Nutzen von Speech Separation in realen ASR-Systemen, wobei eine grosse Variation in der Qualität der Resultate in Kauf genommen werden muss. Die wichtigste Bedingung für einen erfolgreichen Einsatz in der Praxis scheint eine gute Audioqualität zu sein, wie die grosse Differenz zwischen den beiden Varianten zeigt. Wie zu erwarten war, funktionieren WHAMR!-Trainingsdaten besser bei der Lautsprecher-Variante, da dort Rausch- und Nachhalleffekte auftreten und WJS0-2mix-Trainingsdaten besser bei der virtuellen Variante. Bemerkenswert ist die grosse Streuung bei der Variante mit den Lautsprechern (siehe blaue Boxen auf Abbildung 5.1). Man bedenke zudem, dass Wörter bei der WER auch bei gleichem Wortstamm mit unterschiedlicher Endung als Fehler gewertet werden, obwohl die Bedeutung so erhalten bleibt.

Des Weiteren fallen die teilweise sehr hohen Ausreisser auf (bis zu 545%). Hier zeigt sich eine weitere Schwäche der WER als Metrik: Funktioniert eine Separation von zwei *unterschiedlich langen* Aufnahmen schlecht, so resultiert dies beim kürzeren Sprecher in

unterschiedlich langen ASR-Transkripten, was eine übermässig hohe WER zur Folge hat. Bei unterschiedlich langen Aufnahmen mit guter Separation besteht diese Problematik trivialerweise nicht. Allerdings zeigt sich auch, dass sich die WERs im Vergleich zu Tabelle 5.1 entgegen der Erwartung nicht gross unterscheiden, wenn nur diejenigen Beispiele berücksichtigt werden, bei welchen sich die Länge der ASR-Transkripte der Originalaufnahmen der beiden Sprecher um maximal 10 Buchstaben[1] unterscheidet (siehe folgende Tabelle 5.2). Dies betrifft 26 von 200 Sprecherpaare und der höchste WER-Wert beträgt in diesem Fall 104%.

	Lautsprecher		**Virtuell**	
WER [%]	*Mittelwert**	*Median**	*Mittelwert**	*Median**
SepFormer WSJ0-2mix	55.9	55.5	29.7	23.3
SepFormer WHAMR!	52.2	50.9	40.8	37.9

Tabelle 5.2.: Resultate des Experiments 1 “LibriSpeech”, n=26. *Nur die Aufnahmen, bei welchen sich die Länge der ASR-Transkripte der Originalaufnahmen um maximal 10 Buchstaben unterscheidet.

5.2. Experiment 2: Überschneidungsdauer

Inhalt

Bei Experiment 2 soll der Unterschied zwischen sich teilweise überschneidenden und sich komplett überschneidenden Sprechern untersucht werden.

Setup

Das Experiment erfolgte nach Variante 1 “abspielen und erneut aufnehmen” (siehe Kapitel 4.2.1). Es wurden analog zu Experiment 1 Sprechersequenzen von LibriSpeech verwendet. Jedoch ist die Länge der Sequenzen dieses Mal nicht willkürlich: Es wurden 50 Sprecherpaare von je 5 Sekunden Länge gebildet und 50 Sprecherpaare, bei welchen ein Sprecher ebenfalls 5 Sekunden spricht und der andere 14 Sekunden. Die Toleranz liegt bei ±0.5 Sekunden.

[1] Es wurden bewusst Buchstaben statt Wörter verwendet, da die verzerrende Ursache in der unterschiedlichen Länge der Aufnahme liegt.

Resultat

	Lautsprecher		Virtuell	
WER [%]	*Mittelwert*	*Median*	*Mittelwert*	*Median*
Komplette Überschneidung				
SepFormer WSJ0-2mix	68.4	68.0	29.1	25.6
SepFormer WHAMR!	60.4	59.0	41.8	35.5
Teilweise Überschneidung				
SepFormer WSJ0-2mix	93.6	69.2	34.4	29.9
SepFormer WHAMR!	76.8	66.5	43.8	36.3

Tabelle 5.3.: Resultate des Experiments 2 “Überscheidungsdauer”, n=100.

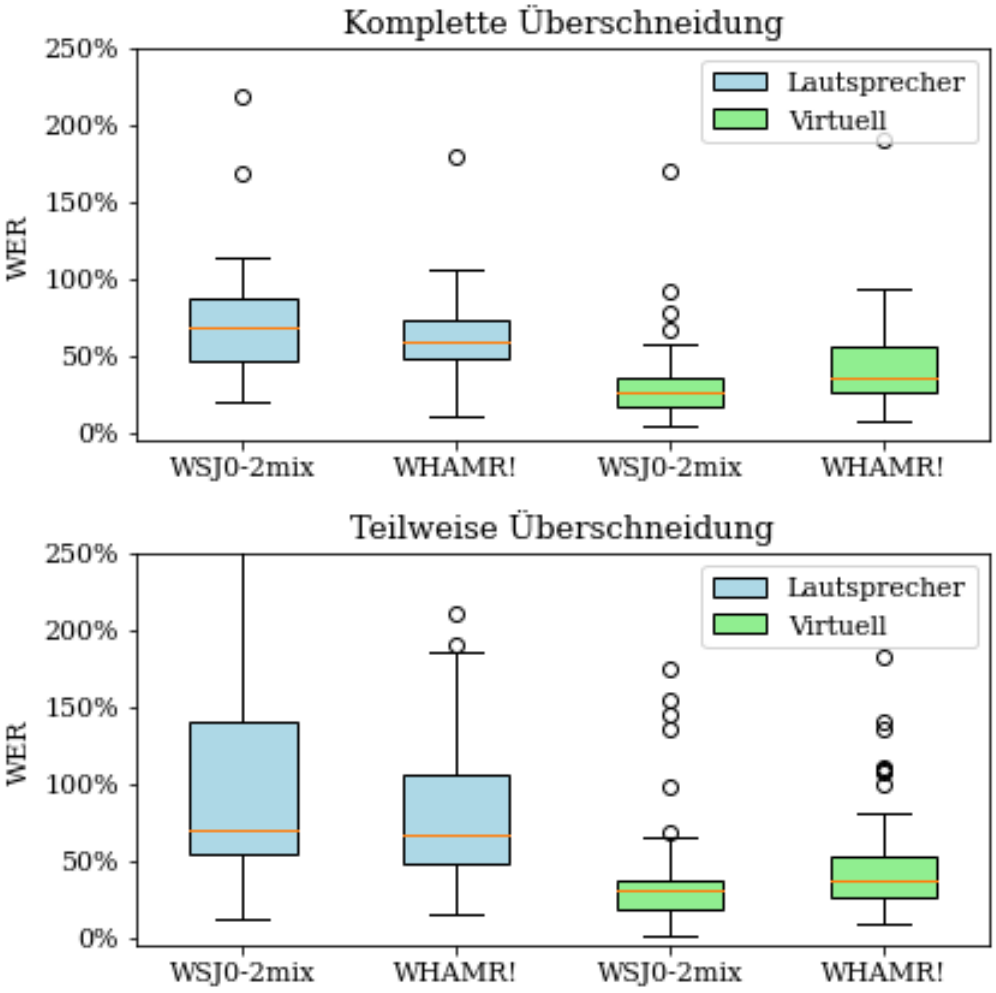


Abbildung 5.2.: Boxplots des Experiments 2 “Überscheidungsdauer”, n=100. Der obere Teil zeigt die komplette Überschneidung und der untere die teilweise Überschneidung.

Fazit

Das Resultat scheint stark von der Audioqualität abzuhängen: Bei der virtuellen Überschneidung unterscheiden sich die beiden untersuchten Varianten nicht gross, doch bei der Lautsprecher-Variante zeichnet sich eine deutliche Verschlechterung der teilweisen Überschneidung ab. Es muss allerdings berücksichtigt werden, dass bei schlechter Separation auch hier das Problem der übermässig hohen WER aufgrund unterschiedlicher Längen der Transkripte auftritt (wie bei Experiment 1). Deshalb erfolgt bei der folgenden Tabelle 5.4 eine weitere Auswertung der teilweisen Überschneidung, wobei dieses Mal nur die WER des längeren Sprechers einfliesst. Der Vergleich zwischen den beiden Tabellen zeigt, dass die Separation bei unterschiedlich langen Sprechersequenzen für die längere Sequenz nicht schlechter funktioniert als bei einer kompletten Überschneidung. Die Ursache für die höheren WERs bei der teilweisen Überschneidung in Tabelle 5.3 scheint also nicht eine schlechtere Separationsleistung zu sein, sondern lediglich die oben beschriebene WER-Problematik bei der kürzeren Sequenz.

	Lautsprecher		**Virtuell**	
WER [%]	*Mittelwert**	*Median**	*Mittelwert**	*Median**
Teilweise Überschneidung				
SepFormer WSJ0-2mix	42.5	40.0	26.5	25.0
SepFormer WHAMR!	46.1	47.4	33.3	30.3

Tabelle 5.4.: Resultate der teilweisen Überschneidung des Experiments 2 “Überscheidungsdauer”, n=100. *Nur die WER des längeren Sprechers.

5.3. Experiment 3: Anzahl Mikrofone

Inhalt

Bei diesem Experiment soll ermittelt werden, ob es einen Unterschied macht, wenn für die Sprecher separate Mikrofone anstatt ein gemeinsames Raummikrofon verwendet werden.

Setup

Das Experiment erfolgte nach Variante 2 “direkt aufnehmen” (siehe Kapitel 4.2.2). Pro Sprecherpaar wurden je 9 Aufnahmen mit einem Raummikrofon und 9 Aufnahmen mit zwei separaten gerichteten (Steck-)Mikrofonen durchgeführt. Es gibt zwei Sprecherpaare: Eines aus zwei männlichen Sprechern und eines aus einem männlichen und einem weiblichen Sprecher. Die Länge der Aufnahmen liegt zwischen 10 und 15 Sekunden.

Resultat

	1 Mikrofon		2 Mikrofone	
WER [%]	*Mittelwert*	*Median*	*Mittelwert*	*Median*
SepFormer WSJ0-2mix	36.6	23.9	28.0	26.2
SepFormer WHAMR!	41.0	30.5	31.4	30.4

Tabelle 5.5.: Resultate des Experiments 3 “Anzahl Mikrofone”, n=18. Vorsicht: Die WER muss hier anders interpretiert werden als bei den Experimenten 1-2 und 4.

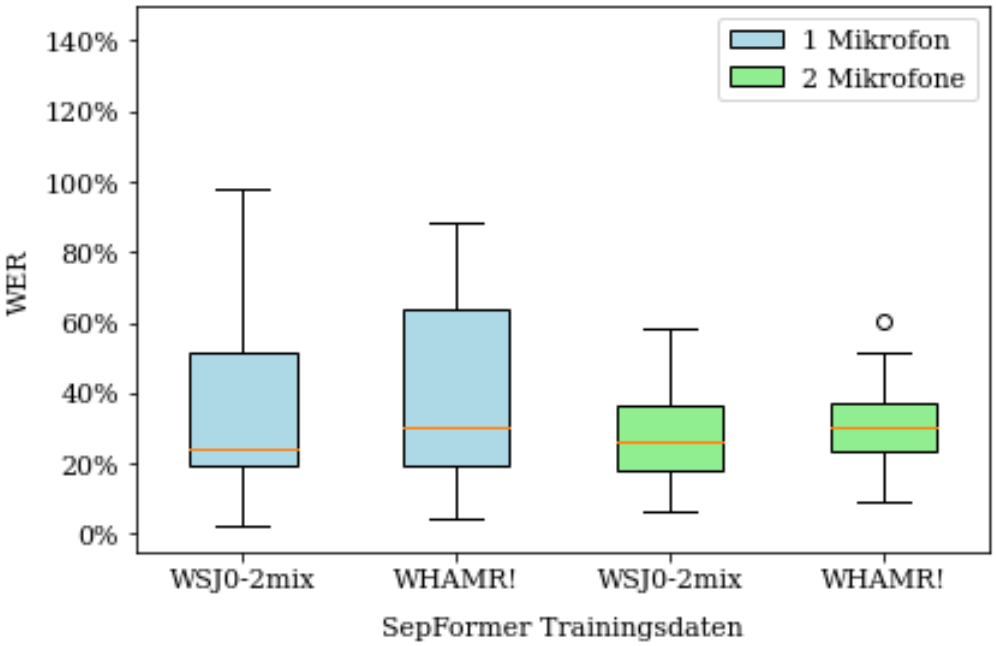


Abbildung 5.3.: Boxplots des Experiments 3 “Anzahl Mikrofone”, n=18.

Fazit

Bei diesem Experiment muss die WER anders interpretiert werden als bei den übrigen Experimenten, weil das Referenz-Transkript kein ASR-Transkript ist. Es fliesst also auch der Anteil am Fehler ein, den Google Speech-to-Text ohne Überschneidung machen würde.

Die Ergebnisse können so interpretiert werden, dass sich die Anzahl Mikrofone nicht in der Separationsleistung bemerkbar macht. Die leicht tieferen WERs bei der Variante mit zwei Mikrofonen sind wohl eher auf die bessere Aufnahmequalität zurückzuführen, da sich die Mikrofone näher bei den Personen befinden. Bemerkenswert ist ferner, dass die auf WSJ0-2mix trainierten Modelle bei beiden Varianten besser zu funktionieren scheinen, obwohl es sich um reale Aufnahmen ohne Studioqualität handelt.

5.4. Experiment 4: Ähnliche Stimmen

Inhalt

Bei diesem Experiment soll ein Eindruck gewonnen werden, ob ähnliche Stimmen für einen Einsatz in der ASR ausreichend gut separiert werden können.

Setup

Das Experiment erfolgte nach Variante 1 "abspielen und erneut aufnehmen" (siehe Kapitel 4.2.1). Es wurden 20 verschiedene Aufnahmen desselben Sprechers mit einem gerichteten Mikrofon aufgezeichnet. Die Hälfte davon wurde jeweils mit der Stimmenverzerrer-Software Vovsoft Voice Changer [47] verzerrt. Die Verzerrung wurde einmal stärker und einmal schwächer durchgeführt (die genauen Einstellungen sind in Anhang B.2 aufgeführt). Der andere Sprecher ist jeweils die unverzerrte Stimme. So sollen unterschiedlich ähnliche Stimmen (im Vergleich zur unverzerrten Stimme) simuliert werden. Von Interesse ist nur die WERs der unverzerrten Stimme.

Resultat

	Lautsprecher		**Virtuell**	
WER [%]	*Mittelwert*	*Median*	*Mittelwert*	*Median*
Stärkere Verzerrung				
SepFormer WSJ0-2mix	50.2	46.9	29.6	33.5
SepFormer WHAMR!	44.9	48.1	17.0	13.5
Schwächere Verzerrung				
SepFormer WSJ0-2mix	46.8	46.9	31.9	37.1
SepFormer WHAMR!	31.6	32.9	9.9	8.3

Tabelle 5.6.: Resultate des Experiments 4 "Ähnliche Stimmen" (nur WER der unverzerrten Stimme), n=10.

Fazit

Das Resultat lässt keine Rückschlüsse auf die gestellte Frage zu: Die Sequenz mit der leicht verzerrten Stimme lässt sich sogar besser separieren. Vermutlich eignet sich die Stimmverzerrung nicht zur Simulation ähnlicher Stimmen. Besser wäre wohl ein Experiment mittels Clustering auf einer Teilmenge eines grossen Korpus wie LibriSpeech.

Das Experiment sei trotz des negativen Ausgangs in dieser Arbeit aufgeführt, da sich ein interessanter Nebeneffekt beobachten lässt: Unabhängig des Grades an Verzerrung scheint das auf WHAMR! trainierte Modell die normale Stimme deutlich besser von der

verzerrten separieren zu können als jenes auf WSJ0-2mix. Offenbar werden Anteile des verzerrten Stimmspektrums vom Modell als Lärm klassifiziert und herausgefiltert.

6. Diskussion und Ausblick

Die Aufgaben der Projektarbeit waren zweierlei. Zum einen sollte ein Überblick über den aktuellen Stand der Technik der Speech Separation gegeben werden. Dies konnte mittels einer umfassenden Literaturrecherche umgesetzt werden. Da es zurzeit keine aktuellen Übersichtspaper gibt, kann dieser Teil der lesenden Person den Einstieg ins Gebiet erleichtern. Zum anderen sollte in dieser Arbeit anhand erster Experimente mit dem Modell SepFormer ein Eindruck vermittelt werden, ob sich ein Einsatz von Speech Separation in ASR-Systemen wie Interscriber lohnen könnte, um überlappende Sprecher zu separieren und anschliessend zu transkribieren. Obwohl die Experimente angesichts fehlender Zeitressourcen *keinen* Anspruch auf Wissenschaftlichkeit erheben, können interessante Beobachtungen gemacht werden. Moderne Speech Separation-Methoden scheinen bereits genügend ausgereift zu sein für einen Einsatz in ASR-Systemen und so auch im konkreten Anwendungsfall Interscriber. Die durchschnittliche WER der Transkripte von circa 30% ist ausreichend tief, dass wenigstens partiell verständlicher Text entsteht. Von besonders grosser Wichtigkeit für ein zufriedenstellendes Resultat scheint eine gute Aufnahmequalität zu sein. Weiter hat sich angedeutet, dass es keinen Unterschied zu machen scheint, ob ein gemeinsames Mikrofon oder separate Mikrofone für jeden Sprecher verwendet werden. Bei unterschiedlich langen Sprechsequenzen scheint die Separation für die längere Sequenz nach wie vor gut zu funktionieren, doch bei der kürzeren Sequenz können in gewissen Fällen übermässig hohe WERs resultieren.

Diese Arbeit sollte vor allem auch den Startpunkt für weitere Experimente in diesem Bereich und eine eventuelle Umsetzung in Interscriber setzen. Die Experimente könnten mit einer grösseren Datenmenge und einem besseren Setup wiederholt werden, um aussagekräftigere Ergebnisse zu erhalten. Weiter könnte der Ansatz von [48] aufgegriffen werden, Speech Separation-Modelle direkt über eine Minimierung der WER als Loss Funciton zu trainieren. Sicherlich lohnt es sich auch, die Entwicklung des neu entstehenden Gebiets der Continous Speech Separation zu beobachten, in welchem realistischere Szenarien als bei der herkömmlichen Speech Separation betrachtet werden.

Ferner bestehen Parallelen zwischen Speech Separation und dem Gebiet des *Speech Enhancement*, bei welchem es um die Verbesserung der Audioqualität einer Aufnahme geht. Möglicherweise liessen sich leicht modifizierte oder anders trainierte Speech Separation-Systeme auch zu diesem Zwecke verwenden. Denkbar wäre beispielsweise ein Modell ähnlich wie VoiceFilter [42], bei welchem der Input auch eine Aufnahme des Hintergrundrauschens eines Raums als Referenz-Aufnahme enthält. Im Unterschied zu VoiceFilter würde dann nicht ein Sprecher, sondern die störenden Geräusche separiert werden, welche im Anschluss zu subtrahieren wären.

Literaturverzeichnis

[1] Interscriber. *Automatische Transkriptionen von Interviews.* [Online]. URL: https://interscriber.com/de/ [Stand: 21.12.2021].

[2] E. C. Cherry, „Some experiments on the recognition of speech, with one and with two ears", *The Journal of the acoustical society of America*, Bd. 25, Nr. 5, S. 975–979, 1953.

[3] D. Wang und J. Chen, „Supervised speech separation based on deep learning: An overview", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Bd. 26, Nr. 10, S. 1702–1726, 2018.

[4] I. Goodfellow, Y. Bengio und A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[5] S. Hochreiter und J. Schmidhuber, „Long short-term memory", *Neural computation*, Bd. 9, Nr. 8, S. 1735–1780, 1997.

[6] F. A. Gers, J. Schmidhuber und F. Cummins, „Learning to forget: Continual prediction with LSTM", *Neural computation*, Bd. 12, Nr. 10, S. 2451–2471, 2000.

[7] H. Zheng, J. Yuan und L. Chen, „Short-term load forecasting using EMD-LSTM neural networks with a Xgboost algorithm for feature importance evaluation", *Energies*, Bd. 10, Nr. 8, S. 1168, 2017.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser und I. Polosukhin, „Attention is all you need", in *Advances in neural information processing systems*, 2017, S. 5998–6008.

[9] M. Forsberg, „Why is speech recognition difficult", *Chalmers University of Technology*, 2003.

[10] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, „State-of-the-art speech recognition with sequence-to-sequence models", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, S. 4774–4778.

[11] D. Wang, X. Wang und S. Lv, „An overview of end-to-end automatic speech recognition", *Symmetry*, Bd. 11, Nr. 8, S. 1018, 2019.

[12] P.-S. Huang, M. Kim, M. Hasegawa-Johnson und P. Smaragdis, „Deep learning for monaural speech separation", in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, S. 1562–1566.

[13] Y. Luo und N. Mesgarani, „Tasnet: time-domain audio separation network for real-time, single-channel speech separation", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, S. 696–700.

[14] E. Vincent, R. Gribonval und C. Fevotte, „Performance measurement in blind audio source separation", *IEEE Transactions on Audio, Speech, and Language Processing*, Bd. 14, Nr. 4, S. 1462–1469, 2006.

[15] J. Le Roux, S. Wisdom, H. Erdogan und J. R. Hershey, „SDR–half-baked or well done?" in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, S. 626–630.

[16] M. Siemering, „Real-time speech separation with deep attractor networks on an embedded system", Master's thesis, Universität Hamburg, 2020.

[17] E. Vincent, C. Févotte und R. Gribonval. *bss_eval*. [Online]. URL: https://gitlab.inria.fr/bass-db/bss_eval [Stand: 18.12.2021].

[18] J. S. Garofolo, D. Graff, D. Paul und D. Pallett, „CSR-I (WSJ0) Complete LDC93S6A. Web Download." 1993.

[19] J. R. Hershey, Z. Chen, J. Le Roux und S. Watanabe, „Deep clustering: Discriminative embeddings for segmentation and separation", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, S. 31–35.

[20] G. Wichern, J. Antognini, M. Flynn, L. R. Zhu, E. McQuinn, D. Crow, E. Manilow und J. L. Roux, „WHAM!: Extending speech separation to noisy environments", *arXiv preprint arXiv:1907.01160*, 2019.

[21] M. Maciejewski, G. Wichern, E. McQuinn und J. Le Roux, „WHAMR!: Noisy and reverberant single-channel speech separation", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, S. 696–700.

[22] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge und E. Vincent, „LibriMix: An Open-Source Dataset for Generalizable Speech Separation", 2020.

[23] V. Panayotov, G. Chen, D. Povey und S. Khudanpur, „Librispeech: An ASR corpus based on public domain audio books", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, S. 5206–5210.

[24] Z. Chen, Y. Luo und N. Mesgarani, „Deep attractor network for single-microphone speaker separation", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, S. 246–250.

[25] D. Yu, M. Kolbæk, Z.-H. Tan und J. Jensen, „Permutation invariant training of deep models for speaker-independent multi-talker speech separation", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, S. 241–245.

[26] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi und J. Zhong, „Attention is all you need in speech separation", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, S. 21–25.

[27] Y. N. Dauphin, A. Fan, M. Auli und D. Grangier, „Language modeling with gated convolutional networks", in *International conference on machine learning*. PMLR, 2017, S. 933–941.

[28] Y. Wang, A. Narayanan und D. Wang, „On training targets for supervised speech separation", *IEEE/ACM transactions on audio, speech, and language processing*, Bd. 22, Nr. 12, S. 1849–1858, 2014.

[29] M. Kolbæk, D. Yu, Z.-H. Tan und J. Jensen, „Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Bd. 25, Nr. 10, S. 1901–1913, 2017.

[30] Y. Luo, Z. Chen und T. Yoshioka, „Dual-path RNN: efficient long sequence modeling for time-domain single-channel speech separation", 2020.

[31] Y. Luo und N. Mesgarani, „Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation", *IEEE/ACM transactions on audio, speech, and language processing*, Bd. 27, Nr. 8, S. 1256–1266, 2019.

[32] Y. Luo und N. Mesgarani, „Real-time single-channel dereverberation and separation with time-domain audio separation network", in *Proc. Interspeech 2018*, 2018, S. 342–346.

[33] J. Chen, Q. Mao und D. Liu, „Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation", 2020.

[34] M. W. Y. Lam, J. Wang, D. Su und D. Yu, „Sandglasset: A light multi-granularity self-attentive network for time-domain speech separation", 2021.

[35] E. Nachmani, Y. Adi und L. Wolf, „Voice separation with an unknown number of multiple speakers", 2020.

[36] N. Zeghidour und D. Grangier, „Wavesplit: End-to-end speech separation by speaker clustering", 2020.

[37] Papers With Code. *WSJ0-2mix benchmark (speech separation)*. [Online]. URL: https://paperswithcode.com/sota/speech-separation-on-wsj0-2mix [Stand: 19.12.2021].

[38] Z. Chen, T. Yoshioka, L. Lu, T. Zhou, Z. Meng, Y. Luo, J. Wu, X. Xiao und J. Li, „Continuous speech separation: dataset and analysis", 2020.

[39] T. Yoshioka, H. Erdogan, Z. Chen und F. Alleva, „Multi-microphone neural speech separation for far-field multi-talker speech recognition", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, S. 5739–5743.

[40] S. Chen, Y. Wu, Z. Chen, J. Wu, J. Li, T. Yoshioka, C. Wang, S. Liu und M. Zhou, „Continuous speech separation with conformer", in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, S. 5749–5753.

[41] Z. Zhang, T. Yoshioka, N. Kanda, Z. Chen, X. Wang, D. Wang und S. E. Eskimez, „All-neural beamformer for continuous speech separation", 2021.

[42] Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. Hershey, R. A. Saurous, R. J. Weiss, Y. Jia und I. L. Moreno, „VoiceFilter: Targeted voice separation by speaker-conditioned spectrogram masking", 2019.

[43] Hugging Face. *speechbrain/sepformer-whamr*. [Online]. URL: https://huggingface.co/speechbrain/sepformer-whamr [Stand: 19.12.2021].

[44] Google. *Cloud Speech-to-Text*. [Online]. URL: https://cloud.google.com/speech-to-text [Stand: 19.12.2021].

[45] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori und Y. Bengio, „SpeechBrain: A general-purpose speech toolkit", 2021, arXiv:2106.04624.

[46] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas, D. Ditter, A. Frank, A. Deleforge und E. Vincent, „Asteroid: the PyTorch-based audio source separation toolkit for researchers", in *Proc. Interspeech*, 2020.

[47] Vovsoft. *Voice Changer*. [Online]. URL: https://vovsoft.com/software/voice-changer/ [Stand: 20.12.2021].

[48] J. Seo, „Minimum Word Error Rate Training for Speech Separation", Master's thesis, University of Stavanger, Norway, 2019.

Abbildungsverzeichnis

Abkürzungsverzeichnis

ASR	Automatic Speech Recognition
CNN	Convolutional Neural Network
CSS	Continous Speech Separation
DNN	Deep Neural Network
DPRNN	Dual-Path RNN
DPTNet	Dual-Path Transformer Network
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MSE	Mean Square Error
PIT	Permutation Invariant Training
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SDR	Source-to-Distortion Ratio
SI-SDR	Scale-invariant Source-to-Distortion Ratio
SI-SDRi	Scale-invariant Source-to-Distortion Ratio Improvement
SOTA	State of the Art
TasNet	Time-Domain Audio Separation Network
uPIT	Utterance-level Permutation Invariant Training
WER	Word Error Rate

A. Dokumentation Toolkit

URL zum Repository: `https://github.zhaw.ch/islermic/speech-separation/`

Table of Contents

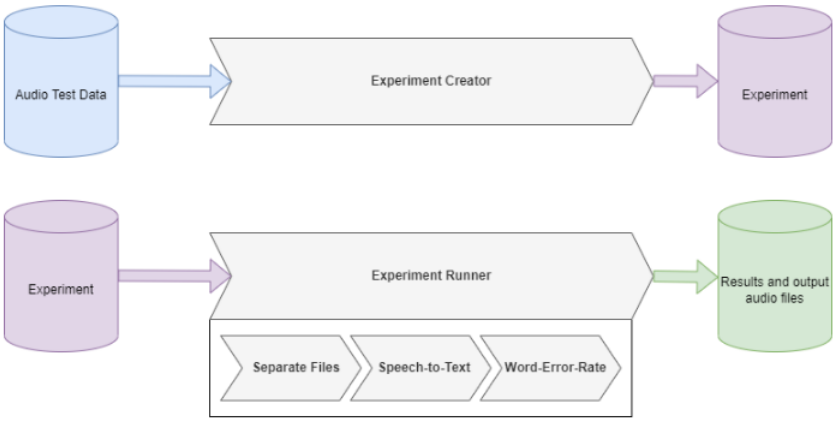
About the tool

The main purpose of this tool is to help with the following tasks:

- Creating datasets for speech separation analysis
- Evaluate different speech separation models
- Execute speech separation for datasets and rate them with a score

The tool is split into the following modules:

- Experiment Creator
- Experiment Runner
  - Separating files
  - Speech-to-Text
  - Word-Error-Rate



Language:

- Python

Getting Started

1. Clone the repository into a local folder
2. Install the following packages into your python environment on your local computer:
   - console-menu
   - paramiko
   - scp
   - numpy
   - google-cloud-speech
   - google-cloud-storage
   - google-auth
   - sounddevice
   - soundfile (windows only/ linux not required)
   - pydub

   To do so, you can run the following command:

   `pip install console-menu paramiko scp numpy google-cloud-speech google-cloud-storage google-auth sounddevice soundfile pydub`
3. Pull the *tensorflow-21.04-tf1-py3* singularity image on the GPU-Cluster into the folder *nvcc-img/*

   The image can be downloaded on to the NVIDIA NGC Page
4. Install the following packages into the python environment of the tensorflow image on the GPU-Cluster:
   - asteroid
   - torchaudio
   - speechbrain

   To do so, you can run the following command:

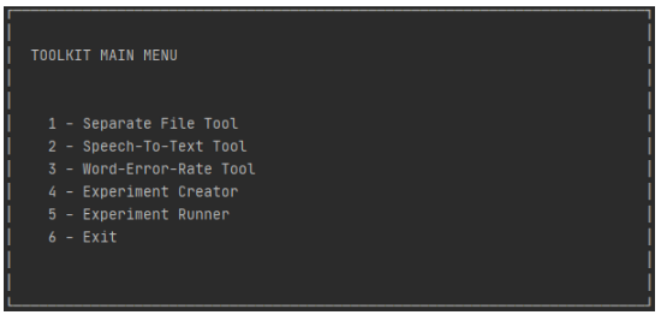
   `pip install asteroid torchaudio speechbrain`
5. Create an account for Google Speech to Text. Create a service account and get a JSON-Credentials file for that account.

   After that set the variable *googleAuthPath* in the project-file *SpeechToText.py* equal to the path of your credential file.
6. (*Optional but recommended*) Download the data here. Copy the files provided in the ZIP-Archive into the subfolder *SSH_Executor* of the repository root.

   It contains data to easily get started with this toolkit.

   Important if you skip this step: Create the subdirectories *output*, *input*, *experiments* and *scratches* in the directory *SSH_Executor*
7. Open a terminal in the folder *SSH_Executor* and run the tool using this command: `py SpeechSeparationToolkit.py`

   The main menu will show up like this:



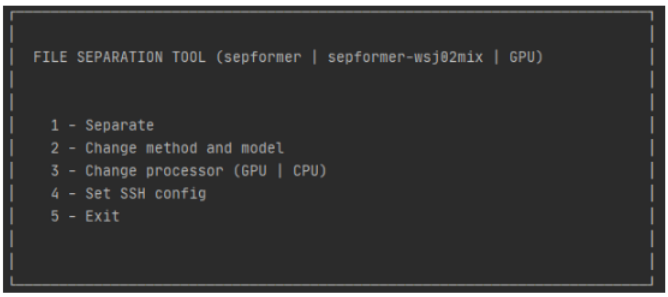
Usage

In the main menu you will be able to navigate into different subtools.

1. Separate File Tool

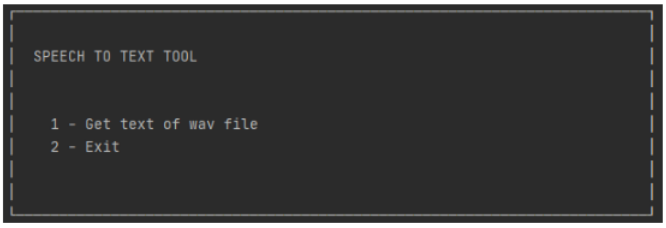
In this subtool you can configure and run a separation of an input file (wav-file in the folder *input*).

- You can specify the method and model (e.g. sepformer wsj02mix).
- You can also specify if it should be executed on the CPU or on the GPU.
- You can configure the SSH-Credentials, so they dont have to be entered every time.



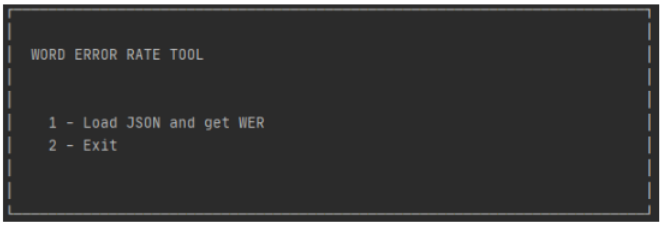
2. Speech-To-Text Tool

In this subtool you are able to get the transcript of an input file (wav-file in the folder *input*)



3. Word-Error-Rate Tool

In this subtool you can get the Word-Error-Rate of texts stored in a json input file (json-file in the folder *input*)

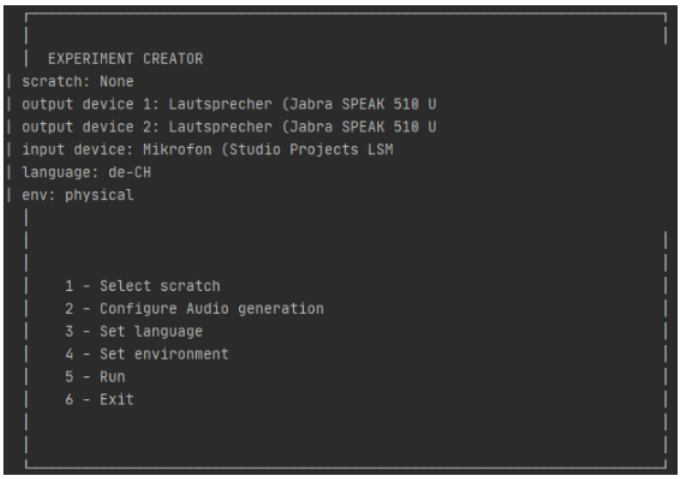


4. Experiment Creator

The experiment creator will create an experiment from a *scratch*. The experiment can then be runned with the experiment runner. There are two types of environments available:

- physical: Play the audio files of the speakers on the defined output devices and record it with the specified input device
- virtual: The audio files will be mixed programmatically

Supported languages: - German - English



5. Experiment Runner

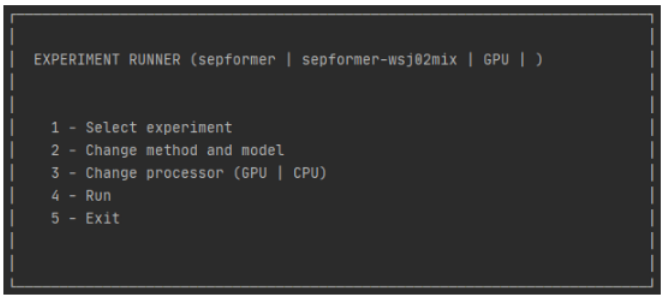
In the experiment runner you can select an experiment and run it with the model you want.

It will then execute the following steps for each dataset in the experiment:

- speech separation followed by a transcription (Google Text-To-Speech).
- get the Word-Error-Rate of the estimate and the source.

The results will be appended to a *results.json* file.

The *results.json* file and all the audio files which are important for the separation will then be stored in the *output* folder.



Examples

Some of the followig instructions require the files from step 6 in the section Getting Started. They are marked with a + in brackets.

Separating a file

1. Copy the wav file which you want to separate into the folder *input*
2. Start the toolkit and navigate into the file separation tool by choosing option 1
3. Configure your desired method and model for the separation
4. (*Optional but recommended*) Set the SSH config
5. Choose option *1 separate* to continue
6. Select the file which you copied into the *input* folder in step 1
7. The separation will be executed and the outputs will be available in the *output* folder

Creating an experiment from a scratch (+)

1. Start the toolkit and navigate into the *experiment creator* by choosing option 4
2. Select a scratch
3. (*Only required for physical environment*) Configure the audio generation by choosing option 2
   - select an input device (microphone)
   - select output device 1 (loudspeaker 1)
   - select output device 2 (loudspeaker 2)
4. Change the environment.
   - virtual => programmatically mix the audios
   - vhysical => play the speakers on loudspeakers and record it with a microphone
5. Choose run and enter a name for the experiment. When finished you can find the created experiment in the folder *experiments*

Running an experiment (+)

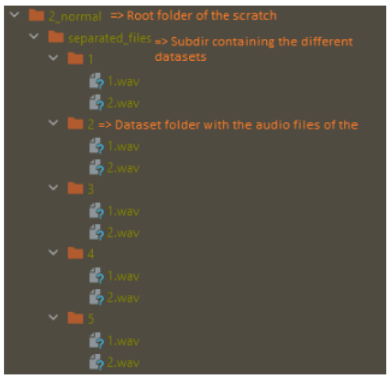
1. Start the toolkit and navigate into the *experiment runner* by choosing option 5
2. Select the experiment you want to run
3. Configure your desired method and model for the separation
4. Run the experiment. When finished you can find the results and output files in the folder *output*

Definition scratch and experiment

Scratch

A scratch is a folder containing datasets with audio files of different speakers. The scratch can be used in the *experiment creator* to create an experiment.

The directory structure is explained in the following image:



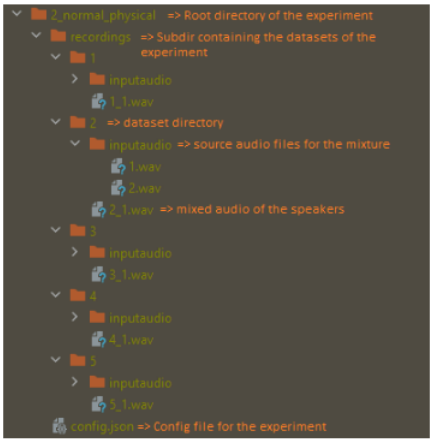
Experiment

An experiment contains datasets with a mixed audio (2 speakers overlapping each other) and a JSON config file for the parameters.

The language is specified in the config file as property "language". Options: *de-CH*, *de-DE*, *en-US*, *en-GB*

- If the dataset is of type "audio_files", it also contains the input audios of which the mixed audio has been created.
- If the dataset is of type "texts", the source transcript of the speakers is given as an array in the property "texts".

The directory structure is explained in the following image:



The following image shows an example config file with type "audio_files":

```
{
  "language": "en-US",
  "experiments": [
    {
      "id": 1,
      "filePath": "recordings/1/",
      "type": "audio_files",
      "rawInputAudioFiles": {
        "1": "recordings/1/inputaudio/1.wav",
        "2": "recordings/1/inputaudio/2.wav"
      }
    }
  ]
}
```

The following image shows an example config file with type "texts":

```
{
  "language": "de-CH",
  "experiments": [
    {
      "id": 1,
      "type": "texts",
      "filePath": "recordings/one_mic/",
      "texts": {
        "1": "Er hörte leise Schritte hinter sich.",
        "2": "Gerade jetzt, wo er das Ding seines Lebens gedreht hatte."
      }
    }
  ]
}
```

B. Experimente

B.1. Daten

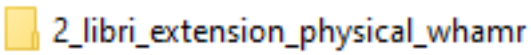
Die Daten der Experimente sind unter folgender Url als ZIP-Archiv erhältlich:

`https://zhaw.sharepoint.com/:u:/s/PA-Speech-Separation/EU2pyb7U69FEsYLlJQ2kDUUB1Rr_ncQ6DuiwmOff4mi_9w?e=pytXCY`

Die Experimente sind in Ordnern mit folgender Bezeichnung zusammengefasst:

`<Experimentnummer>_<Name[1]>_[physical, virtual]_<Trainingsmodell>`

Beispielweise steht das nachfolgende Beispiel



für:

- Experiment 2 “Überschneidungsdauer”
- extension, d. h. teilweise Überschneidung
- physical, d. h. Lautsprecher-Variante
- WHAMR!-Trainingsdaten

In jedem Ordner befindet sich auf der obersten Ebene eine JSON-Datei mit der Bezeichnung *results.json*, welche die Trankripte und WERs aller Aufnahmen des Experiments enthält. Die Input- und die separierten Ouput-Aufnahmen sind in den entsprechenden Unterordnern zu finden.

[1] Kann auch “_” enthalten.

B.2. Ergänzende Materialien

Abbildung B.1.: Aufbau für die Experimente der Variante 1 “abspielen und erneut aufnehmen”. Im Hintergrund sind die beiden Freisprechanlagen von Jabra zu sehen, die als Lautsprecher dienen.

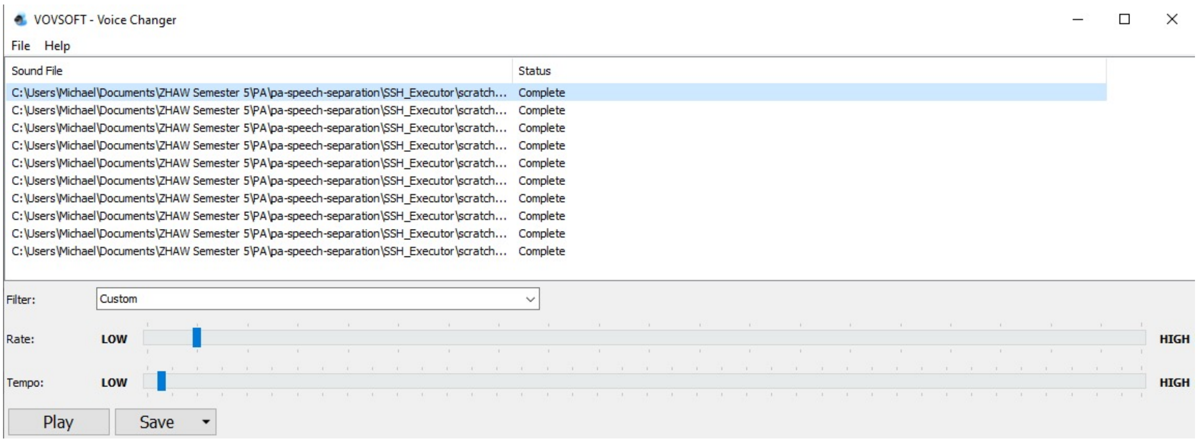


Abbildung B.2.: Einstellungen der Software Vovsoft Voice Changer bei der “stärkeren” Verzerrung.

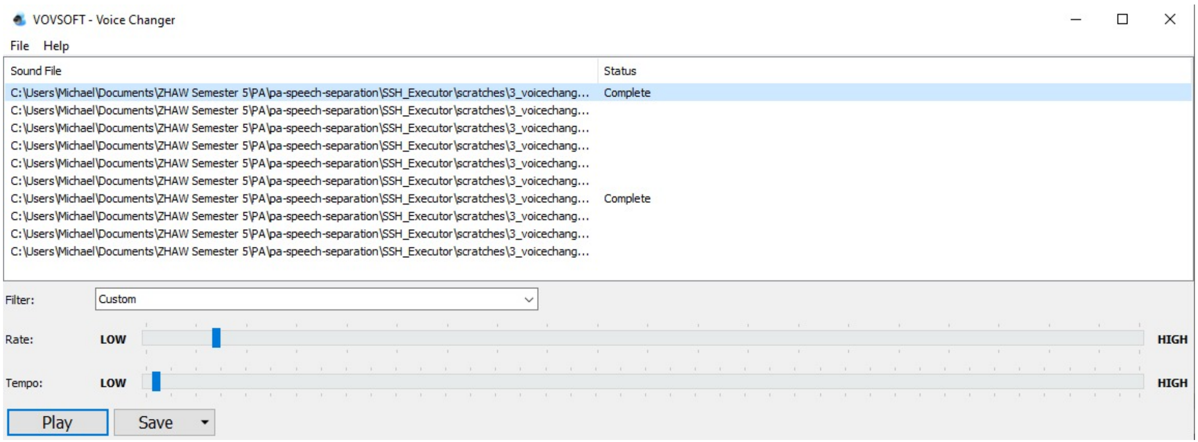


Abbildung B.3.: Einstellungen der Software Vovsoft Voice Changer bei der “schwächeren” Verzerrung.

C. Aufgabenstellung

zhaw School of Engineering — [Online Administration Praktische Arbeiten] — [Dept. T Admin Tools]

zurück — Logout

Projektarbeit 2021 - HS: PA21_ciel_09

Allgemeines:

Titel:	Speech Separation: Generate Separate Audio Tracks of Parallel Speakers [Audio, Machine Learning]
Anzahl Studierende:	2
Durchführung in Englisch möglich:	Ja, die Arbeit kann vollständig in Englisch durchgeführt werden und ist auch für Incomings geeignet.

Betreuer:

HauptbetreuerIn: Mark Cieliebak, ciel

Zugeteilte Studenten:

Diese Arbeit ist vereinbart mit:
- Michael Isler, islermic (IT)
- Marco Zala, zalamar1 (IT)

Fachgebiet:

AI	Artificial Intelligence
DA	Datenanalyse
ML	Machine Learning
SOW	Software

Studiengänge:

ET	Elektrotechnik
IT	Informatik
WI	Wirtschaftsingenieurwesen

Zuordnung der Arbeit :

InIT Institut für angewandte Informationstechnologie

Infrastruktur:

benötigt keinen zugeteilten Arbeitsplatz an der ZHAW

Interne Partner :

Es wurde kein interner Partner definiert!

Industriepartner:

Es wurden keine Industriepartner definiert!

Beschreibung:

Speech-to-Text systems by Google, IBM etc. can generate a text from an audio recording of a dialogue (e.g. a interview or meeting). These systems work well for single speakers, or as long as each speaker speaks individually. It becomes challenging as soon as several speakers speak at the same time; then current automatic systems are overstrained and don't deliver meaningful output.

In this case the user has to transcribe the parallel speech manually. This is a tedious and time-consuming process, as it is also difficult for humans to keep the different parallel speakers apart. Therefore, in this thesis a system for "*Speech Separation*" will be implemented, which separates the audio signals of the individual speakers. This would make it possible to play each speaker individually or the whole sound without a specific speaker.

Goal of this Thesis

There exist already several papers and algorithms for Speech Separation that can be used as a basis. These shouldbe evaluated and compared in the course of this thesis. If there are even open-source implementations or commercial products, these should also be included in the comparison.

Ultimate goal of the project is to implement an API that allows to separate the audio signal of, say, 2-5 parallel speakers automatically.

Voraussetzungen:

This is a very challenging project which is at the boundary of current research. It requires high ambition, strong technical skills and the willingness to read and digest scientific papers.

Implementation will be probably in Python.

If you are interested in this very interesting topic, please get in touch. We will then meet and discuss the details. Email: ciel@zhaw.ch, Phone: 058 934 72 39.

zurück — Logout