Speech-to-Text Translation von Schweizerdeutsch auf Hochdeutsch

ZHAW School of Engineering

Bachelorarbeit

Studiengang: IT17tb_WIN

Betreuer: Prof. Dr. Mark Cieliebak

Assistent: Jan Milan Deriu

Verfasser: Bogumila Dubel
Dialogplatz 7
8400 Winterthur

Abgabedatum: 11.06.2021

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit 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 Bachelorarbeit 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:	**Name Studierende:**
11.06.2021	Bogumila Dubel

Zusammenfassung

Als Schweizerdeutsch bezeichnet man die Gesamtheit aller in der Deutschschweiz gesprochenen Dialekte. Dabei handelt es sich nicht um eine offizielle Sprache mit einheitlicher Grammatik und Schreibweise. Jeder Dialekt ist spezifisch und hat einen eigenen Wortschatz. Diese Aspekte stellen eine grosse Herausforderung für die Speech Translation und Speech Recognition Systeme dar. Diese Arbeit beschäftigt sich mit Speech-to-Text Translation von Schweizerdeutsch auf Hochdeutsch und arbeitet ein solches System aus. Die entwickelten Lösungen basieren auf dem FAIRSEQ S2T Toolkit und der Transformer-Architektur. Des Weiteren werden systematisch Massnahmen zur Optimierung der System-Performance untersucht. Diese basieren auf der Erweiterung der Trainingsdaten, dem Einbezug von vortrainierten ASR Encodern in den Trainingsprozess und dem Durchführen eines Ensembles. Das Ergebnis dieser Arbeit ist ein System, welches einen Score von 58.50 BLEU Punkten auf dem Swiss Parliament Korpus und von 39.48 BLEU Punkten auf dem «Clickworker»-Testset erreicht. Das System hat dazu beigetragen, dass das ZHAW-Team den zweiten Rang beim SwissText Shared Task «Swiss German Speech to Standard German Text» erreicht hat.

Abstract

Swiss German refers to all dialects spoken in the German-speaking part of Switzerland. It is not an official language with uniform grammar and spelling. Each dialect is specific and has its own vocabulary. These aspects pose a great challenge for Speech Translation and Speech Recognition systems. This thesis deals with Speech-to-Text Translation from Swiss German to High German and elaborates such a system. The developed solutions are based on the FAIRSEQ S2T toolkit and on the Transformer architecture. Furthermore, measures to optimize the system performance are systematically investigated. These are based on the extension of the training data, the inclusion of a pre-trained ASR encoder in the training process and the execution of an ensemble. The result of this work is a system that achieves a score of 58.50 BLEU points on the Swiss Parliament corpus and 39.48 BLEU points on the "Clickworker" test set. This result contributed to the ZHAW team taking second place on the Shared Task "Swiss German Speech to Standard German Text" at SwissText.

Vorwort

Speech Translation ist ein spannendes und dynamisches Gebiet, auf dem zurzeit sehr intensiv geforscht wird. Das Bearbeiten dieser Aufgabe hat mir ermöglicht, mich mit einem sehr interessanten, aber auch herausfordernden Thema zu befassen. Für die Unterstützung und die vielen wertvollen Inputs möchte ich mich bei meinem Betreuer Prof. Mark Cieliebak und seinem Assistenten Jan Milan Deriu herzlich bedanken.

Rückblickend finde ich den spontanen Themawechsel gut und es hat mir Spass gemacht diese spannende Aufgabe zu bearbeiten.

Zudem möchte ich mich beim ZHAW-Shared-Task-Team für die angenehme Zusammenarbeit bedanken.

Inhaltsverzeichnis

1. Einleitung

Die Forschung auf dem Gebiet von Speech Translation (ST) hat in den letzten Jahren grosse Fortschritte gemacht. Die immer leistungsstärkeren Rechner, beinahe unlimitierter Speicherplatz als auch exponentiell wachsende Datenmenge haben einen grossen Beitrag zu dieser Entwicklung geleistet. Je präziser die ST-Systeme werden, umso grössere Bedeutung gewinnen sie im marktwirtschaftlichen Kontext. Inzwischen wurden zahlreiche kommerzielle Produkte für ST entwickelt, wie Vasco Translator[1], Pocketalk[2], Speech Translation von Microsoft Azure[3] oder Mobile-Apps[4]. Die Auswahl ist mittlerweile sehr gross. Es kann jedoch beobachtet werden, dass die Qualität der Übersetzungen sehr stark von der Menge und der Qualität der Trainingsdaten abhängt. ST-Systeme für Sprachen wie Englisch oder Spanisch, welche in vielen Ländern gesprochen werden und zu denen viele Trainingsdaten verfügbar sind, erreichen bereits eine sehr gute Übersetzungspräzision. Demgegenüber sind die Sprachen im Nachteil, welche nur von einem kleinen Teil der Weltbevölkerung gesprochen werden und entsprechend über wenig Trainingsdaten verfügen. Schweizerdeutsch gehört leider auch zu den benachteiligten Sprachen. Die zahlreichen Dialekte in der Deutschschweiz stellen eine zusätzliche Herausforderung in Bezug auf ST dar.

Diese Arbeit beschäftigt sich mit Speech-to-Text Translation (STT Translation) von Schweizerdeutsch auf Hochdeutsch. Es wird untersucht, wie gut ein STT-System auf andere Dialekte und neuen Wortschatz generalisieren kann, nachdem es mehrheitlich mit Daten eines Dialektes und mit eingeschränktem Vokabular trainiert wurde. Im zweiten Schritt werden Experimente durchgeführt, mit dem Ziel diese Generalisierung zu verbessern.

1.1. Ausgangslage

Speech Translation ist eine komplexe Aufgabe, welche mit unterschiedlichen Architekturen und Ansätzen gelöst werden kann. Je nach Wahl der Architektur werden mehrere Komponenten zur Generierung der Übersetzungen zusammengesetzt. Diese Komplexität bietet einen grossen Spielraum für Experimente und als Konsequenz ist eine grosse Vielfalt an Literatur

[1] https://vasco-translator.com/en/

[2] https://www.pocketalk.com/de/

[3] https://azure.microsoft.com/en-us/services/cognitive-services/speech-translation/

[4] https://play.google.com/store/apps/collection/cluster?clp=ogosCBEqAg-gIMiQKHmNvbS5oYXdzb2Z0Lm1vYmlsZS5zcGVlY2h0cmFucxABGAM%3D:S:ANO1ljJ_auQ&gsr=Ci-iCiwIESoCCAgy-JAoeY29tLmhhd3NvZnQubW9iaWxlLnNwZWVjaHRyYW5zEAEYAw%3D%3D:S:ANO1ljJfmt0&hl=gsw&gl=US

zu diesem Thema entstanden. Das Themenspektrum erstreckt sich von Analyse und Ausarbeitung von verschiedenen Architekturen, über Vorstellung von ST-Korpora bis hin zu diversen Experimenten mit Pseudo-Labeling, Self-Training, synthetische Datengenerierung usw.

Sperber und Pauli [1] geben einen sehr guten Einstieg in das Thema, da alle wichtigen Aspekte wie Übersicht über die Architekturarten, Modellierungstechniken, Trainingsstrategien, Applikationsarten mitsamt ihrer chronologischen Entwicklung zusammengefasst werden.

ST ist eine verhältnismässig junge Disziplin. Das erste ST-System JANUS [2] wurde vor 30 Jahren entwickelt, im Jahr 1991. JANUS basiert auf einem Cascaded-Ansatz, mit welchem bis vor 5 Jahren alle ST-Systeme umgesetzt wurden. Bei diesem Ansatz wird die ST-Aufgabe durch voneinander unabhängig trainierten Automatic Speech Recognition (ASR) und Machine Translation (MT) Komponenten gelöst. Im Jahr 2016 hat Bérard et al. [3] das erste End-to-End Speech Translation System (E2E-ST-System) entwickelt. Jenes basiert auf der Sequence-to-Sequence (Seq2Seq) Architektur, welche zwei Jahre früher von Sutskever et al [4] entwickelt wurde. Im E2E-Ansatz verschmelzen die Komponenten von ASR und MT zu einem Modell, welches allein die komplexe ST-Aufgabe lernt. Im Jahr 2017 wurde die Transformer Architektur von Vaswani et al [5] eingeführt, welche momentan der State-of-the-Art auf dem Gebiet von ST ist. Die Transformers sind eine Weiterentwicklung von Seq2Seq.

Mit dem Einführen von E2E-ST-Systemen ist die Nachfrage nach entsprechenden Trainingsdatensätzen entstanden. Früher hat man Datensätze entweder für ASR oder MT aufgebaut. Ein ASR-Datenansatz besteht aus Audiodateien in der Ausgangssprache mit den entsprechenden Transkriptionen in Textform in der gleichen Sprache. Ein MT-Datensatz besteht aus Texten in der Ausgangssprache mit den entsprechenden Translationen in Textform für die Zielsprache. Bei einem ST-Datensatz erübrigt sich die Zwischenrepräsentation in Form von der Transkription in der Ausgangssprache. Stattdessen werden für die Audiodateien in der Ausgangssprache direkte Translationen in der Zielsprache benötigt. CoVoST [6] ist momentan der grösste Korpus für E2E-ST und bietet insgesamt 2880 Stunden für Englisch und andere 21 Fremdsprachen an. Es gibt jedoch sehr viele Sprachen, für welche keine oder sehr wenige Trainingsdaten für ST vorhanden sind. Diese Sprachen werden Low Resource Langauges genannt. Im Gegensatz dazu werden Sprachen, welche über viele Trainingsdaten verfügen als High Resource Langauges bezeichnet. Wang et al. [7] beschäftigt sich mit der Frage, ob Low Resource Langauges, während dem Training, von Daten für High Resource Langauges profitieren können. Da der Aufbau eines ST-Datensatzes mit grossem Aufwand verbunden ist, werden Methoden zum synthetischen Erstellen von Trainingsdaten untersucht, Jia et al. [8]. Das Pseudo Labeling ist ein weiterer Ansatz, bei dem das Modell sich selbst Translationen für die Audiodateien generiert. Baevski et al [9] haben mit wav2vec eine ASR-Architektur entwickelt, welche Training mit ungelabelten Daten[5] ermöglicht. Das wav2vec Mo-

[5] In diesem Kontext Audiodaten, welche über keine alignierten Transkriptionen respektive Translationen verfügen.

dell ist aktuell State-of-the-Art auf dem Gebiet von Speech Recognition. Wu et al. [10] integrierten wav2vec in ein ST-Training. Pino et al. [11] trainierten ein ST-System mit einer grossen Menge (30'000 Stunden) an ungelabelten englischen Audiodateien. Die Liste der Veröffentlichungen auf dem ST Gebiet ist noch viel länger. Ein paar weitere Arbeiten werden im darauffolgenden Kapitel 2 vorgestellt.

Für die Evaluation der ST-Systeme wird in der Regel die BLEU Metrik [12] verwendet. TER [13] und METEOR [14] sind weiteren Metriken, welche jedoch weniger populär sind.

In der deutschsprachigen Schweiz wurden bereits ein paar Publikationen auf dem Gebiet von ST und ASR veröffentlicht. Garner et al. [15] hat ein ASR- und ST-System für den Walliserdeutsch Dialekt entwickelt. Die recapp IT AG[6] hat in Zusammenarbeit mit der ETH ein kommerzielles Produkt «Swiss Voice Assistent» ausgearbeitet. «Swiss Voice Assistent» kann hochdeutsche Translationen für schweizerdeutsche Sprache erstellen. Am GermEval 2020 Task 4 [16] wurden 3 Lösungen für ein System, welches die schweizerdeutsche Sprache in einen hochdeutschen Text übersetzt, vorgestellt. Die verfügbaren Korpora für Schweizerdeutsch sind: Swiss Parliament [17], SwissDial [18] oder ArchiMob [19], welche im Kapitel 4 ausführlich vorgestellt werden.

1.2. Zielsetzung

Das Ziel der Arbeit ist ein STT Translation System zu entwickeln, welches die schweizerdeutsche Sprache unabhängig vom Dialekt in einen hochdeutschen Text übersetzt.

Die Herausforderung bei dieser Aufgabenstellung besteht darin, mit der Vielfalt der schweizerdeutschen Dialekte umzugehen, welche mit unterschiedlichen Aussprachen und Wortschatz verbunden sind. Da Trainingsdaten nicht für alle Dialekte im gleichen Ausmass verfügbar sind, muss das Modell die Fähigkeit entwickeln, Übersetzungen für neue, ihm unbekannte Dialekte herzuleiten. Im Rahmen dieser Arbeit soll systematisch untersucht werden, mit welchen Mitteln diese Dialekt-Generalisierung erreicht werden kann. Mögliche Vorgehensweise ist der Einbezug von unterschiedlichen Korpora oder vortrainierten Speech Recognition Komponenten in den Trainingsprozess. Das Einarbeiten in ein STT Translation Toolkit eigener Wahl (z.B. FAIRSEQ S2T oder ESPNet ST) als auch das Beschaffen und Aufbereiten der Korpora sind Teil dieser Arbeit.

Parallel zur Bachelorarbeit findet der Shared Task «Swiss German Speech to Standard German Text»[7] an der SwissText statt. Ein Ergebnis der Arbeit kann die Teilnahme an diesem Wettbewerb sein. Des Weiteren ist das Messen der eingesetzten Optimierungsmassnahmen

[6] https://recapp.ch/

[7] https://www.swisstext.org/task-3-swiss-german-speech-to-standard-german-text/

auf dem offiziellen Test Set ein guter Indikator für die Modell-Performance bezüglich unterschiedlicher schweizerdeutscher Dialekte. Die Aufgabe von Shared Task wird ausführlich im Kapitel 3 vorgestellt.

In Kapitel 2 werden die Entwicklung und die wichtigen Meilensteine auf dem ST-Gebiet eingeführt. Kapitel 4 gibt eine Übersicht über die ST-Korpora für den Schweizerdeutsch und erläutert ihre Eigenschaften. In Kapitel 5 wird das in dieser Arbeit verwendete ST-Toolkit FAIRSEQ S2T mit Alternativen verglichen. Zudem wird eine kurze Einführung zu diesem Framework gegeben. Kapitel 6 zeigt den Trainingsprozess und das eigentliche Vorgehen bei den einzelnen Teilaufgaben des Trainings auf. In Kapitel 7 werden die Resultate, welche hinsichtlich des Shared Tasks ausgearbeitet wurden, als auch ihre Interpretation zusammengeführt. Kapitel 8 stellt weitere Ergebnisse vor, welche unabhängig vom Shared Task entstanden sind. Abschliessend wird in Kapitel 9 kritisch auf das Ergebnis dieser Arbeit zurückgeblickt. Zudem wird ein Ausblick in die Zukunft bezüglich möglicher weiterer Fragestellungen gegeben.

Der Bericht ist auf Fachexperten ausgerichtet, welche über Grundlagenwissen in den Bereichen Machine Learning, Software Entwicklung und Datenanalyse verfügen. Das nötige Fachwissen auf dem Gebiet von Speech Translation wird im nächsten Kapitel eingeführt.

2. Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen auf dem ST-Gebiet vorgestellt. Im ersten Abschnitt wird der Begriff ST und die wichtigsten Ausprägungen der ST-Applikationen eingeführt. In den Abschnitten 2.2 und 2.3 werden die ST-Architekturen und die relevanten Beispiele erläutert. Anschliessend werden im Abschnitt 2.4 die Evaluations-Metriken für das Messen der Performance von ST- und ASR-Systemen vorgestellt.

2.1. Speech Translation

Speech Translation ist ein Vorgang, in dem ein Audio-Input[8] in der Ausgangssprache (Source Language) in einen Text respektive in einen Audio-Output in der Zielsprache (Target Language) übersetzt wird. Liegt die Übersetzung in Textform vor, spricht man von einer Speech-to-Text (STT) Translation. Wird aus der Übersetzung zusätzlich synthetisch Sprache generiert, bezeichnet man den Vorgang als Speech-to-Speech (STS) Translation.

Grundsätzlich wird zwischen folgenden Anwendungen unterschieden [1]:

1. Batch Mode
2. Consecutive Mode
3. Simultaneous Mode

Diese bestimmen, in welcher Form das Audio-Input vom ST-System eingelesen, bearbeitet und ausgegeben wird. Beim Batch-Mode ist der Audio-Input eine in sich abgeschlossene Einheit, in der Regel eine Audiodatei. Das ST-System nimmt diese entgegen und gibt den Output in Form einer Übersetzung zurück, erst nachdem die vollständige Datei verarbeitet wurde. Die Dauer der Verarbeitung ist sekundär, da das Modell nicht primär auf schnelle Antwortzeiten ausgerichtet ist. Beispiele für diese Art der Applikation sind z.B. Filmuntertitel [20] oder Film Dubbing [21]. Der Consecutive Mode wird bei Echtzeit-Applikation angewendet, welche eine Kommunikation in zwei Richtungen ermöglichen und eine gewisse Latenz zulassen. Hier ist der Input ähnlich wie im Batch-Mode eine in sich abgeschlossene Einheit. Der Unterschied ist jedoch, dass die Antwortzeiten eine grosse Rolle spielen. Ein Bespiel für eine Anwendung ist ein Übersetzungs-App auf einem Mobiltelefon [22]. Beim Simultaneous Mode handelt es sich um eine Echtzeit-Übersetzung. Dabei ist der Input ein Audio-Stream. Dies ist der anspruchsvollste Ansatz, weil die Applikation auf sehr schnelle Übersetzungszeiten angewiesen ist und konstant auf die aus dem Input-Stream kommenden Informationen reagieren muss.

Die Aufgabe von ST kann mit zwei Ansätzen gelöst werden, welche in den darauffolgenden Kapiteln 2.2 und 2.3 erläutert werden.

[8] Audio-Input kann eine Audiodatei oder Audio-Stream sein.

In dieser Bachelorarbeit werden STT Systeme behandelt, welche für eine Applikation im Batch oder Consecutive Mode implementiert werden können. Alle STT-Systeme wurden mit der State-of-the-Art Transformer Architektur trainiert.

2.2. Cascaded Architektur

Die Cascaded Architektur ist ein Ansatz, der die ST-Aufgabe mithilfe von mehreren Komponenten, welche unabhängig voneinander trainiert oder erstellt werden, löst. STT wird aus der ASR- und MT-Komponente zusammengesetzt. ASR ist für das Transkribieren des Audio-Inputs in der Ausgangssprache verantwortlich. MT übersetzt diese Transkription in die Zielsprache in einer Textform. Bei STS wird das System nach der MT-Komponente um eine Speech Synthesis Komponente erweitert, welche aus den Textübersetzungen synthetische Sprache generiert. Abbildung 1 visualisiert schematisch die beiden Konzepte.

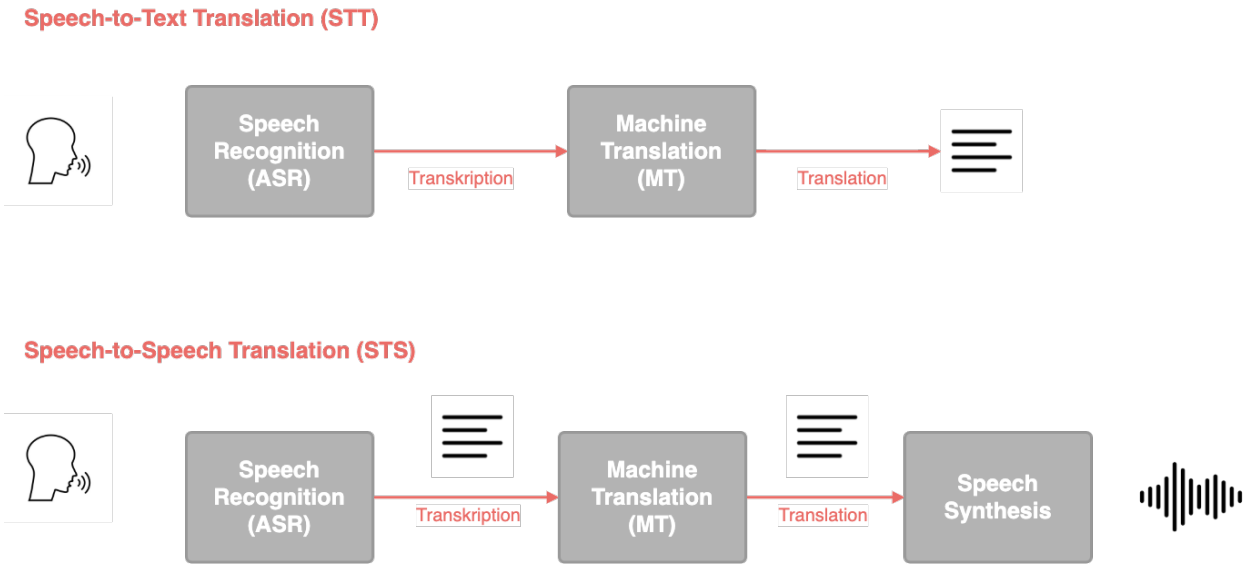

Abbildung 1: Konzept Cascaded Architektur

Der erste Ansatz für die Lösung der ST-Aufgabe ist das JANUS System und wurde von Waibel et al. [2] vorgestellt. JANUS basiert auf einem Cascaded Ansatz und ist ein STS-System, welches in der Lage ist, die englische Sprache ins Japanische und Deutsche zu übersetzen. Beim Betrachten von Abbildung 1 ist offensichtlich, dass bei diesem Ansatz die Präzision der Endkomponente von der Präzision der vorherigen abhängt. Das ist auch der grösste Nachteil des Systems, welches die von der ASR-Komponente verursachten Fehler an die restlichen Komponenten weiterleitet. Dies kann die Präzision der Übersetzung negativ beeinflussen [1]. Dieses Phänomen wird als Error Propagation bezeichnet. Als Reaktion auf diese Problematik folgten weitere wissenschaftliche Beiträge, welche Ansätze zur Abschwächung von Error Propagation untersuchten. Vidal [23] hat einen Ansatz auf Basis von Finit-State Models ausgearbeitet, um die Komplexität der ST-Aufgabe zu reduzieren und eine bessere Integration der ASR- und MT-Komponente zu erreichen. Tsetkov et al. [24] stellt eine Technik zur Verbesserung von Robustheit der MT-Komponente vor, welche auf der Erweiterung des MT-Modells

um verrauschte und fehlerhafte Daten basiert. Eine gute Übersicht von den weiteren Veröffentlichungen auf diesem Gebiet stellt Sperber und Paulik in [1] zusammen.

2.2.1. ASR-Komponente

Die Aufgabe von ASR innerhalb eines STT-Translation Systems ist ein Audio-Input in eine textuelle Repräsentation, eine Transkription, zu bringen und diese als Input an die MT-Komponente weiterzuleiten. Ähnlich wie ST ist ASR eine komplexe Aufgabe und kann mit unterschiedlichen Architekturen und Ansätzen gelöst werden. Heutzutage wird ein ASR System tendenziell mit einer E2E-Architektur trainiert. In einem E2E-Ansatz verschmelzen die einzelnen Komponenten in einem neuronalen Netz, die jeweiligen Aufgaben werden jedoch trotzdem implizit gelernt. In diesem Abschnitt wird die traditionelle ASR-Architektur vorgestellt, da an ihrem Beispiel die Zuständigkeiten jeder Komponente klar ersichtlich sind. Sie besteht aus den folgenden Komponenten: Audio Processing, Acoustic Model, Language Model und Decoder. Abbildung 2 zeigt eine schematische Darstellung der traditionellen ASR Architektur. Unterstehend werden die einzelnen Komponenten und ihre Aufgaben beschrieben.

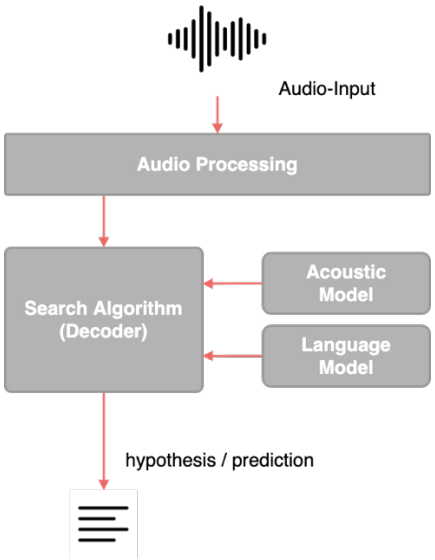

Abbildung 2: Piktogramm Traditionelle ASR Architektur [25]

Audio Processing

Innerhalb von Audio Processing wird der Audio-Input prozessiert und dessen Features extrahiert (Features Extraction). Das Ziel ist den Audio-Input in eine kompakte Vektorform zu bringen, welche die Audiomerkmale (Audio Features) repräsentiert und von einem Rechner effizient verarbeitet werden kann [25]. Dieser Vorgang ist nötig, damit das Modell in der Lage ist die Informationen aus den verschiedenen Audio-Inputs miteinander zu vergleichen und daraus während des Lernprozesses Wissen zu gewinnen. Bei Audio Processing wird im ersten Schritt eine Speech Analysis durchgeführt, welche das Rohsignal in Frames einer definierten Länge aufteilt. Die typische Frames-Grösse variiert zwischen 20 und 40 Millisekunden, wobei

sich die Frames zu 50% (+/- 10%)[9] überlappen. Anschliessend werden Audiomerkmale extrahiert und in einem Vektor gespeichert. Wie und welche Audiomerkmale in diesem Vektor festgehalten werden, hängt davon ab, mit welcher Repräsentation gearbeitet wird. Hierfür gibt es eine grosse Vielfalt an Möglichkeiten. Für das Training von E2E-Systemen haben sich die Mel-Scaled Filterbanken durchgesetzt. Früher waren Mel-Frequency Cepestral Coefficients (MFCC) die erste Wahl, da diese für das Training von Gaussian Mixture Models-Hidden Markov Models (GMMs-HMMs) gut geeignet sind[9]. GMMs-HMMs Modelle waren eine lange Zeit State-of-the-Art auf dem ASR-Gebiet.

Acoustic Model

Das Acoustic Model stellt einen Zusammenhang zwischen den Audiomerkmalen und einer vordefinierten akustischen Spracheinheit, in der Regel einem Phonem, her. Ein Phonem ist als ein sprachabhängiges Lautsegment zu verstehen, z.B. *three (θriː)* besteht aus den Phonemen '*θ*', '*r*', '*iː*'. Die Spracheinheit kann auch auf dem Word- oder Sub-Word-Level definiert werden. Die Auswahl der Granularität hat jedoch einen grossen Einfluss auf die Rechen-Performance. Früher wurde das statistische Hidden-Markov-Modell (HMM) am häufigsten eingesetzt [25]. Mit dem E2E-Ansatz hat sich das HMM-Acoustic Modell erübrigt und diese Aufgabe wird von einem Encoder übernommen (siehe Kapitel 2.3.1). Das statistische Modell hat für diese Arbeit wenig Relevanz, weshalb nicht näher auf das Funktionsprinzip eingegangen wird.

Language Model

Das Language Model hat die Aufgabe den Phonemen, welche vom Acoustic Model vorgeschlagen werden, einen Kontext zu geben. Dies ist notwendig, weil es in der Aussprache viele ähnlich klingenden Lauten gibt. Ohne Kontext würden jene beim Dekodieren möglicherweise zu einer Aneinanderreihung von zufälligen Buchstaben oder Wörtern führen. Ein Mensch ist in der Lage mit seinem Wissen Phonemen richtig einzuordnen und einen zusammenhängenden Satz aus dem Gehörten zu bilden. Das Model muss diese Beziehungen erst lernen. Deshalb ist die Aufgabe vom Language Model die Regeln zu definieren, welche Wort- respektive Spracheinheit-Sequenzen in der Zielsprache zulässig sind. Diese können z.B. durch die grammatikalische Regel definiert werden, welche normalerweise von einem Sprachexperten zur Verfügung gestellt werden müssen. Das ist mit einem grossen Aufwand verbunden. Aus diesem Grund wird oft ein statistisches Modell trainiert, welches diese Zusammenhänge aus dem Trainingskorpus lernt, indem es die Statistiken zu jedem Wort- respektive Spracheinheit-Paar im Trainingsdaten schätzt [25]. Somit wird die Auftretenswahrscheinlichkeit für jede mögliche N-Gram-Einheiten innerhalb vom Trainingsdaten festgelegt. Je nach Grösse des Vokabulars und der N-Gram-Länge können Millionen von Kombinationen entstehen. In der Regel werden Bigram (2-Gram) oder Trigram (3-Gram) Modelle gebildet [25]. Bezüglich der Granularität des Vokabulars arbeitete man früher meistens auf dem Word-Level. In neuen

[9] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Architekturen werden öfters Language Modelle auf dem Character Level [26] oder Bytes Level [27] gebildet.

Decoder

Der Decoder kombiniert das Wissen aus dem Acoustic Model und dem Language Model, um aus den prozessierten Rohaudiodaten die am wahrscheinlichste Prediction vorherzusagen. Die Prediction ist der Transkription gleichbedeutend. Untenstehend ist die Formel, welche eine optimale Lösung für das Problem findet, abgebildet (1):

$$\check{W} = \underset{W \in L}{argmax}\, P(O|W)\, P(W) \qquad (1)$$

$\check{W}$ ist die Prediction. $P(O|W)$ wird von Acoustic Model berechnet. Es ist die Wahrscheinlichkeit für das Auftreten eines Phonems O gegeben eine a priori Wahrscheinlichkeit einer Wordsequenz W. $P(W)$ ist die Auftretenswahrscheinlichkeit für eine Wordsequenz W, welche zur Menge der Wordsequenzen aus der Language L gehört. $P(W)$ wird vom Language Model berechnet [25]. Das Auswerten aller möglichen Wahrscheinlichkeiten für $\check{W}$ bei einem Vokabular von der Grösse 75'000[10] führt zu Millionen von Zuständen. Das Finden der besten Lösung endet in einem nicht berechenbaren Problem. Aus diesem Grund wird in der Praxis mit approximativen Suchtechniken gearbeitet, welche im grösstenteils auf dem Viterbi Algorithmus [28] basieren.

2.2.2. MT Komponente

Die Aufgabe von Machine Translation ist eine Übersetzung für die von der ASR-Komponente erstellte Transkription zu generieren. Der erste automatische Übersetzer wurde im Jahr 1954 entwickelt, welcher in der Lage war, einen russischen Text ins Englische zu übersetzen. Seitdem wurden verschiedene Ansätze ausgearbeitet, welche wie folgt unterteilt werden [29]:

- Regelbasiert (RBMT)
- Corpus Based (CBMT)
- Hybrid

Der Regelbasierte Ansatz stützt auf den grammatikalischen Regeln und dem Vokabular, welche von einem Sprachexperten erstellt werden müssen. Das Aufbereiten dieser Informationen ist mit einem sehr aufwendigen Prozess verbunden. Zudem müssen die Regel und das Vokabular ständig aktualisiert werden, da die Sprache ein lebendiges Konstrukt ist. Ein Beispiel für ein RBMT wurde von Sumita et al. [30] vorgestellt. Als Reaktion auf die Problematik von RBMT wurde der Corpus Based Ansatz (CBMT), auch Data Driven MT genannt, entwickelt. Beim CMBT lernt das Modell implizit aus den Beispielen in den bilingualen Datensätzen die Translationen zu generieren. Als Referenz wird das Beispiel von Imamura [31] genannt. Der

[10] 75'000 ist die geschätzte Grösse der deutschen Standardsprache (https://de.wikipedia.org/wiki/Wortschatz).

Hybrid Ansatz kombiniert die beiden Konzepte und ist auf Sprachen mit wenig Korpora-Ressourcen ausgerichtet. Die Idee dabei ist die Übersetzungsregel automatisch aus wenigen alignierten Daten herzuleiten. Probst et al. [32] hat ein Hybrides MT-Modell vorgestellt.

Die hier beschriebenen Ansätze wurden heutzutage von den im kommenden Abschnitt beschrieben E2E-Ansätzen abgelöst.

2.3. End-to-end (E2E) Architektur

Bei einem E2E-Ansatz kann eine komplexe Aufgabe innerhalb von einem Modell erlernt werden. Der grosse Vorteil dessen ist, dass auf das unabhängige Trainieren der Zwischenkomponenten wie dem Acoustic Model und/oder dem Decoder verzichtet werden kann. Des Weiteren entfällt die Problematik des Transferierens der Fehler von einer auf die andere Komponente, was als Error Propagation vorgestellt wurde. Die Aufgabe von E2E-ST ist ein Audio-Input in der Ausgangssprache in einen Text in der Zielsprache zu übersetzen. Dies erfordert, dass der Audio-Input direkt in eine Translation übertragen wird. Um ein solches System erfolgreich zu trainieren, wird eine grosse Datenmenge benötigt, was leider für viele Sprach-Kombinationen nicht verfügbar ist. Wang et al. [6] stellt einen CoVoST-Korpus vor, welcher auf den von Mozilla gesammelten Open Source Common Voice Daten[11] basiert. Dieser Korpus besteht aus insgesamt 2880 Stunden an Audiodaten, welche Übersetzungen für 21 Sprachen ins Englische und vom Englischen in 15 Sprachen anbieten. Dies ist zurzeit der grösste Datensatz für ST. Für die europäischen Sprachen ist Europarl ST [33] der grösste ST-Korpus, welcher Audiodaten mit entsprechenden Translationen für 9 Sprachen mit insgesamt 72 Sprachrichtungen zur Verfügung stellt. Die Kollektion enthält insgesamt 1642 Stunden an Audiodaten nur für die Trainsets, dazu kommen noch die Dev- und Testsets mit jeweils 3 bis 6 Stunden an Audiodaten. Dies ergibt im Durchschnitt ca. 23 Stunden pro Sprachkombination (1642/71 ~ 23h), wobei die Daten nicht auf alle Sprachen gleichverteilt sind. Auf jeden Fall reichen 23 Stunden an Audiodaten nicht aus, um ein gutes E2E-ST-Modell zu trainieren. Des Weiteren ist das Aufstellen eines solchen Korpus eine sehr zeitintensive Aufgabe, weshalb in der Wissenschaft Alternativlösungen gesucht werden, um das Problem des Datenmangels umzugehen. Data Augmentation ist ein Ansatz, in dem die bestehenden Trainingsdaten mit verschieden Techniken verarbeitet werden, um anschliessend die Trainingsdaten mit diesen veränderten Daten zu erweitern. Park et al. stellt in [34] eine Methode für Data Augmentation vor. Jia et al. beweist in [8], dass das Verwenden eines vortrainierten MT-Modells, die ST-Performance verbessern kann. Des Weiteren wird in der gleichen Veröffentlichung das Einbeziehen in den Trainingsprozess von synthetisch erstellten Daten (weakly supervised data) untersucht. Das FAIRSEQ ST2 Toolkit (siehe Kapitel 5) stellt eine Schnittstelle zur Verfügung, welche das Trainieren eines E2E-ST-Modells mit einem vortrainierten ASR Encoder ermöglicht. Die Resultate dieses Ansatzes werden ausführlich im Kapitel 7.3 beschrieben.

[11] https://commonvoice.mozilla.org/en/datasets

Der E2E-Ansatz kann mit den folgenden Architekturen gelöst werden:

- Sequence-to-Sequence
- Transfomer

welche in den zwei darauffolgenden Kapiteln eingeführt werden.

2.3.1. Sequence-to-Sequence

Die Sequence-to-Sequence (Seq2Seq) Architektur wurde zum ersten Mal von Sutskever et al. [4] als Ansatz zum Lösen der MT-Aufgabe vorgestellt. Die Hauptidee des Konzepts wird mit seinem Namen selbst ausgedrückt: Transformation einer gegebenen Sequenz in eine neue Sequenz[12]. Die grösste Herausforderung dieses Ansatzes liegt darin, dass die Länge der Eingangssequenz variabel und die Länge der Zielsequenz nicht im Vorfeld bekannt ist [4]. Als Konsequenz kann die Architektur des Modells nicht auf einen Inputvektor mit fixer Grösse konditioniert werden, stattdessen muss sie flexibel auf unterschiedliche Vektorgrössen reagieren können. Des Weiteren müssen die Zwischenzustände der Eingangssequenz gemerkt werden, da alle voneinander abhängen. Im Fall der Translation wird der Input nicht als eine Einheit, sondern als eine Sequenz von Worten (Token) variabler Länge betrachtet. Diese wird dann in eine Tokensequenz in der Zielsprache mit einer unbekannten Länge umgewandelt. Die von Sutskever et al. [4] vorgeschlagene Architektur basiert auf zwei Multi-Layer Long Short-Term Memory Netzen (LSTM)[13]. Das erste Netz ist der Encoder, das zweite der Decoder.

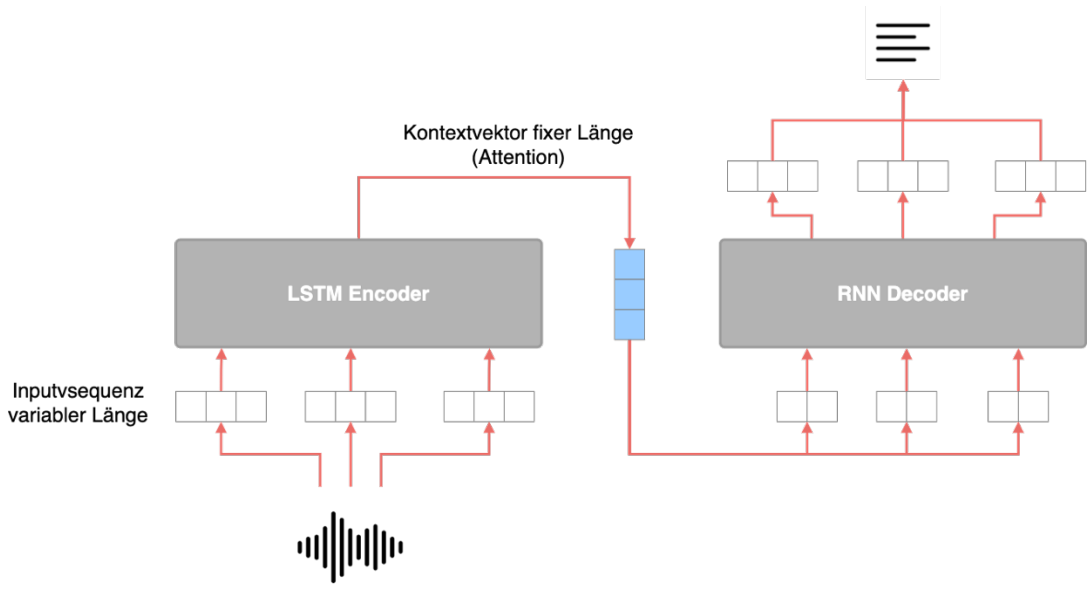

Abbildung 3 Konzept Seq2Seq Architektur

[12] https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

[13] LSTM-Architektur wird am Ende des Abschnitts erläutert.

Die Aufgabe des Encoders ist dabei die Eingangssequenz auf einen Vektor mit definierter Dimensionalität abzubilden. Dieser Vektor wird Kontextvektor genannt. Der Kontextvektor wird dann an dem Decoder weitergeleitet, wessen Aufgabe ist diesen in die Zielsequenz zu dekodieren. Der Decoder ist ein Recurrent Neural Network (RNN). Sein Aufbau ist auf die Länge des Kontextvektors konditioniert. In dem Kontextvektor wird die enkodierte Übersetzung für den Token und zusätzliche Kontext-Informationen gespeichert. Die Kontext-Informationen beschreiben den Kontext des gegebenen Tokens in der Tokensequenz. Sie helfen dem Decoder bei Mehrdeutigkeiten einer Token-Übersetzung, den Kontext besser zu verstehen und die passende sprachliche Repräsentation zu wählen[12]. Dieser Mechanismus wird Attention genannt und wurde von Graves [35] eingeführt. Abbildung 3 visualisiert schematisch den beschriebenen Prozess.

Das erste E2E-ST-System vom Jahr 2016 basiert auf der Seq2Seq Architektur und wurde von Bérard et al [3] entwickelt. Zwei Jahre später entwickelten dieselben Autoren eine Folgearbeit in [36]. Weiss et al. vertieft diesen Ansatz in [37].

Recurrent Neural Network (RNN)

Ein RNN ist ein neuronales Netz, welches Verbindungen nicht nur zum nächsten Layer, sondern auch zum eignen und dem vorherigen zulässt. Dies erlaubt es dem Modell auf die früheren Zustände des Inputs zuzugreifen, was das Bearbeiten von Sequenzen möglich macht. Die RNN-Modelle sind schwierig zu trainieren, da innerhalb des Neuronalen Netzes Zyklen entstehen können. Des Weiteren müssen die Trainingsdaten sequenziell bearbeitet werden, was eine Parallelisierung ausschliesst. Abbildung 4 zeigt schematisch die RNN-Architektur.

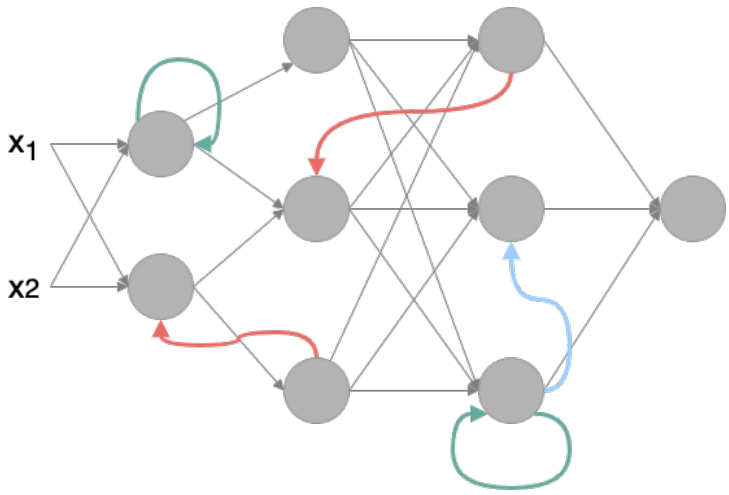

Abbildung 4: Konzept RNN-Architektur

Long Short-Term Memory (LSTM) Netze

LSTM-Netze sind eine Weiterentwicklung von RNN und wurden von Hochreiter und Schmidhuber [38] vorgestellt. LSTM sind so ausgebildet, dass sie im Gegenteil zu einem RNN-Modell über ein Gedächtnis verfügen, wodurch sie nicht mehr auf die langanhaltenden Abhängigkeiten angewiesen sind. Dieses erlaubt den LSTM-Netzen zu lernen, welche Informationen aus

den vergangenen Zuständen für das Verarbeiten des aktuellen Zustands wichtig sind und welche vergessen werden können[14]. In Abbildung 5 ist eine LSTM-Zelle einer RNN-Zelle gegenübergestellt. RNN-Zelle verfügt nur über einen Layer, LSTM hingegen über vier. Die zusätzlichen Layers sind für das Zwischenspeichern und Verarbeiten der Informationen aus den früheren Zuständen verantwortlich.

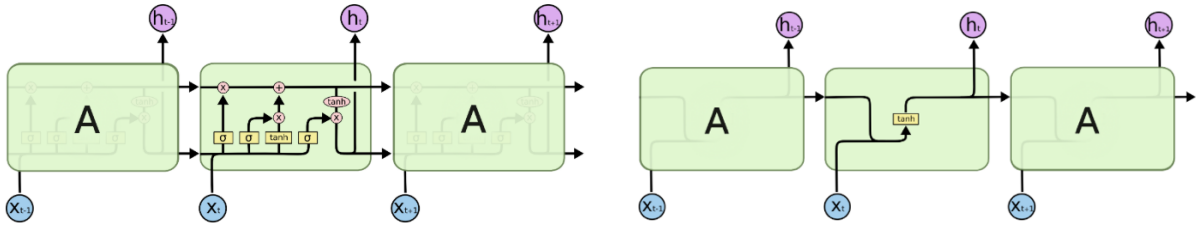

Abbildung 5: Eine LSTM-Zelle (links) und eine RNN-Zelle (rechts)

2.3.2. Transformer (self-attention)

Die Transformer Architektur wurde von Vaswani et al. [39] eingeführt und basiert ähnlich wie Seq2Seq auf dem Encoder-Decoder Ansatz. Der Unterschied ist jedoch, dass der Decoder nicht mehr als ein RNN modelliert wird, was dank der Einführung von Multi-Headed Self-Attention ermöglicht wurde. Mit dem Ausschliessen von RNN reagiert man auf die folgende Problematik:

- In der RNN Architektur hängt jeder versteckter Zustand von der Ausgabe des vorherigen versteckten Zustands ab. Diese sequenzielle Natur schliesst parallele Bearbeitung eines Trainingsbeispiels aus. Bei Trainingsdaten, welche viele Beispiele mit längeren Sequenzen enthalten, kann es zu Arbeitsspeicherproblemen führen, was das Training verlangsamt respektive mehrere Arbeitsspeicher-Ressourcen erfordert. [39]

Das Konzept von Multi-Head-Attention erlaubt es das Berechnen des Kontextvektors, was mit der Attention-Funktion erfolgt, zu parallelisieren. In RNN wurde der Vorgang pro Trainingsbeispiel sequenziell durchgeführt. Die Multi-Head-Attention verteilt das Berechnen der Attention-Funktion auf mehrere Layers und fügt anschliessend alle Teilergebnisse in einem Kontextvektor zusammen. Abbildung 6 zeigt schematisch diesen Vorgang. Dabei Q (Queries) ist eine Vektorrepräsentation für die Übersetzung eines Tokens innerhalb des Training-Beispiels. V und K sind Vektorrepräsentationen von Values und Keys. Beide geben dem Q den Gesamtkontext innerhalb des Training-Beispiels.

Multi-Head-Attention wird auf die folgenden Layers verteilt:

- «encoder-decoder-attention»
- «self-attention encoder»

[14] https://www.eoda.de/wissen/blog/long-short-term-memory-tutorial/

- «self-attention decoder»

Dies simuliert die RNN-Architektur und erlaubt dem Decoder und Encoder auf jeden Token innerhalb der ganzen Eingabesequenz zuzugreifen. Mit dem Poisitional Encoding wird dem Modell die Information zur Position des Tokens innerhalb der Sequenz übermittelt.

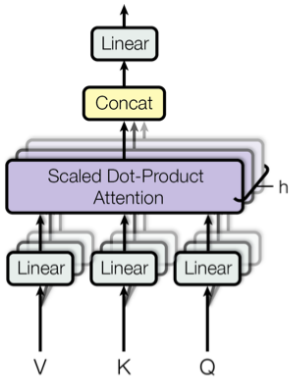

Abbildung 6: Multi Head Attention, Vaswani et al., 2017 in [39]

In dieser Arbeit wird mit einem Transformer Modell von FAIRSEQ S2T gearbeitet, welches im Kapitel 6.4 vorgestellt wird.

2.4. Evaluation

Die Modell-Performance wird gemessen, indem das Modell mittels einem Testset durch eine festgelegte Metrik evaluiert wird. Die meistverbreitete Metrik für die Evaluation der ST-Systeme ist BLEU [12]. ASR-Systeme werden in der Regel mit der WER Metrik ausgewertet. Beide Metriken wurden in dieser Arbeit eingesetzt. In den Abschnitten 2.4.1 und 2.4.2 werden sie eingeführt. Dabei wird die folgende Terminologie verwendet:

Prediction	die von dem Modell generierte Transkription / Translation
Label	die im Testset zur Verfügung stehende hochqualitative Transkription / Translation
n-grams	Anzahl Wörter
Insertion	hinzugefügtes Wort
Deletion	gelöschtes Wort
Substitution	ersetztes Wort

2.4.1. BLEU

BLEU zählt die Übereinstimmung von n-grams in der Prediction mit den n-grams im Label, dabei wird die Reihenfolge von den Wörtern im Label nicht berücksichtigt. Mathematisch wird der BLEU Score in Brevity Penalty und n-gram overlap aufgeteilt, siehe Gleichung (2).

$$BLEU = (\text{Brevity Penalty} * \text{n} - \text{gram overlap}) * 100 \quad (2)$$

Brevity Penalty bestraft die Vorhersagen, welche im Vergleich zum Label zu kurz sind, siehe Gleichung (3).

$$Brevity\ Penalty = min(1, exp(1 - \frac{Pred_{Length}}{Label_{Length}})) \quad (3)$$

n-gram overlap berechnet, wie viele Unigrams, Bigrams, Trigrams und 4-grams aus der Prediction sich mit den entsprechenden n-grams aus dem Label überlappen, siehe Gleichung (4). Die Überlappung wird als Präzision ausgedrückt.

$$\text{n} - \text{gram overlap} = (\prod_{n=1}^{4} precision_n)^{\frac{1}{4}} \quad (4)$$

Die Präzision für n-gram wird, wie in der Gleichung (5) dargestellt, berechnet:

$$precision_n = \frac{\sum_{sentence \in Predi. - Label} \sum_{n \in sentence} min(m^n_{Pred}, m^n_{Label})}{w^n_{total}} \quad (5)$$

Dabei:

m^n_{Pred} ist die Anzahl der n-gram-Überlappungen von der Prediction mit dem Label

m^n_{Label} ist die Anzahl der n-grams im Label

w^n_{total} ist die totale Anzahl der n-grams in der Prediction und wird, wie in der Gleichung (6) aufgezeigt, berechnet.

$$\sum_{sentence' \in Pred. - Label} \sum_{n' \in sentence'} m^{n'}_{Pred} \quad (6)$$

Das Resultat BLEU ergibt eine Zahl zwischen 0 und 100. 100 bedeutet ein perfektes Alignement der Vorhersage mit dem Label. 0 heisst, dass die Vorhersage und das Label kein Alignement aufweisen. Von daher gilt je höher der BLEU Score, umso besser ist die Performance des ST-Systems. In Tabelle 1 wird die Interpretation der BLEU-Scores übersichtlich dargestellt.

BLEU Score	*Interpretation*
< 10	Fast unbrauchbar
10 – 19	Schwer nachvollziehbar
20 – 29	Nachvollziehbar, mit vielen grammatikalischen Fehlern
30 – 39	Verständliche bis gute Translationen
40 – 49	Hochqualitative Translationen
50 -60	Sehr hochqualitative und akkurate Translationen
> 60	oft besser als menschliche Translationen

Tabelle 1: Interpretation der BLEU-Scores[15]

[15] https://cloud.google.com/translate/automl/docs/evaluate#interpretation

2.4.2. Word Error Rate (WER)

Die Word Error Rate (WER) misst, wie gut die Vorhersage mit dem Label übereinstimmt. Für diese Berechnung werden alle Insertions (I), Deletions (D) und Substitutions (S) in der Vorhersage zusammengezählt und durch die Gesamtanzahl der Wörter im Label (N) geteilt, siehe Gleichung (7). Das Resultat ergibt eine Zahl zwischen 0 und 1. 0 bedeutet ein perfektes Alignement der Vorhersage mit dem Label. 1 heisst, dass die Vorhersage und das Label kein Alignement aufweisen. Von daher gilt je tiefer der WER Score, umso besser ist die Performance des ASR-Systems.

$$WER = \frac{I + D + S}{N} \qquad (7)$$

2.4.3. Bemerkung

Die Auswertungsresultate können abhängig von den Testdaten stark variieren. Deshalb muss beim Bewerten des Systems das Testset so ausgewählt werden, dass die untersuchten Fragestellungen sinnvoll beantwortet werden können respektive die angestrebte Lösung erreicht werden kann. Ist das Ziel ein Real-Life-System zu entwickeln, welches trotz vieler Hintergrundgeräusche gute Translationen generiert, müssen im Testset auch solche Beispiele vorhanden sein. Ist dies nicht der Fall und das Testset beinhaltet nur hochqualitative Audioaufnahmen, besteht die Gefahr, dass die Auswertungen unbrauchbar sind und keine Aussage bezüglich der zu lösenden Aufgabe getroffen werden können.

3. Shared Task

In den Kapiteln 3 bis 5 werden die Grundlagen aufgeführt, welche für die praktische Umsetzung dieser Arbeit analysiert und aufbereitet wurden. Als erstes wird in diesem Kapitel die Aufgabenstellung vom Shared Task[16] und das Shared-Task-Team vorgestellt. Im Kapitel 4 werden die Korpora und im Kapitel 5 der FAIRSEQ S2T Toolkit vorgestellt.

3.1. Aufgabe

Das Ziel von Shared Task ist ein System zu entwickeln, welches einen beliebigen Dialekt der schweizerdeutschen Sprache in einen Text in schriftdeutscher Sprache übersetzt. Den Teilnehmern wurden der Swiss Parliament Korpus mit 293 Stunden an gelabelten Audiodaten (siehe Kapitel 4.2) und 1208 Stunden an ungelabelten Audiodaten, hauptsächlich in Zürcher Dialekt, zur Verfügung gestellt. Die entwickelten STT-Systeme sind auf dem Clickworker Testset zu validieren. Das Clickworker Testset besteht aus 13 Stunden Audiodateien, welche eine gute Repräsentation aller in der Deutschschweiz gesprochenen Dialekte darstellen. Diese Dialektverteilung kommt der Realität sehr nah. Für das Testset wurden keine Labels zur Verfügung gestellt.

Der BLEU Score wird mit der NLTK Library[17] und den Default Parametern ausgewertet. Die Übersetzungen, auf welchen die Modellergebnisse gemessen werden, sind normalisiert und enthalten nur die folgenden Zeichen: Kleinbuchstaben inkl. deutsche Umlaute: ö, ä, ü. Die Nummern werden ausgeschrieben. Die Interpunktionen und Sonderzeichen werden entfernt.

Das Team, welches den besten BLEU Score auf dem Clickworker Testset erreicht, gewinnt den Wettbewerb.

3.2. Team und Gesamtsystem

Parallel zu dieser Bachelorarbeit haben sich im Frühlingssemester 2021 zwei weitere Studenten mit verwandten Themen befasst. Um Synergien aus den verschiedenen Beiträgen zu nutzen wurde ein Team unter der Leitung des ZHAW Instituts «Centre for Artificial Intelligence» gebildet. Der eingereichte Beitrag erfolgte im Namen der ZHAW und ist ein Resultat einer Teamarbeit [40].

Neben dem in dieser Bachelorarbeit entwickelten FAIRSEQ-System wurden zwei weitere STT-Systeme (wav2vec 2.0 und Jasper) und ein Post-Processing Verfahren ausgearbeitet. Das wav2vec System basiert auf dem XLSR wav2vec Modell [41], welches mit 56 Tausend Stunden an ungelabelten Daten in 53 Sprachen vortrainiert wurde. Das Jasper System basiert

[16] https://www.swisstext.org/task-3-swiss-german-speech-to-standard-german-text/

[17] https://www.nltk.org/api/nltk.translate.html

auf dem Gewinner-Beitrag des Shared Task 2020 [42]. Beim Post-Processing wurde der Ansatz «spelling correction» analysiert und umgesetzt. Dessen Ziel ist die Fehlerquote in den Outputs aus den drei vorgestellten STT-Systemen zu reduzieren. Auf den Outputs aller Systeme wurden anschliessend zwei Arten von Ensembling (Hybrid Ensembling und Majority Voting) durchgeführt. Das Gesamtsystem wurde in der Veröffentlichung Ulasik et al. [40], welche bis am 4. Juli den Status «In Bearbeitung» trägt, beschrieben.

4. Korpora

Dieses Kapitel erläutert den grundlegenden Aufbau eines Korpus für STT-Translation. Zusätzlich wird eine Übersicht über die bestehenden Korpora für Schweizerdeutsch gegeben sowie deren Eigenschaften detailliert dargestellt.

Ein Korpus für das Trainieren eines E2E STT-Systems besteht aus kurzen Audiodateien in der Ausgangssprache. Zu jeder Audiodatei gibt es eine alignierte Text-Übersetzung für die Zielsprache und zusätzliche Metadaten. Die Metadaten enthalten in der Regel Information zum Sprecher, wie das Alter, das Geschlecht, die Herkunft, der gesprochene Dialekt. Diese Angaben variieren von Korpus zu Korpus, teilweise fehlen sie jedoch vollständig. Die Metadaten sind vor allem für die spätere Analysis der Resultate wichtig. Mithilfe von jenen können Zusammenhänge herausgefunden und Fragestellungen beantwortet werden. Die gewonnen Erkenntnisse können dann genutzt werden, um z.B. die Performance des Modells zu optimieren.

Die Daten in einem Korpus sind normalerweise in zwei Splits unterteilt: Train und Test. Der Train Split besteht aus ca. 80% der Daten und wird für das Trainieren des Modells verwendet. Die restlichen 20% der Daten bilden den Test Split, auf dem die Modell-Performance gemessen wird. In Abbildung 7 sind alle Elemente eines Korpus visuell dargestellt.

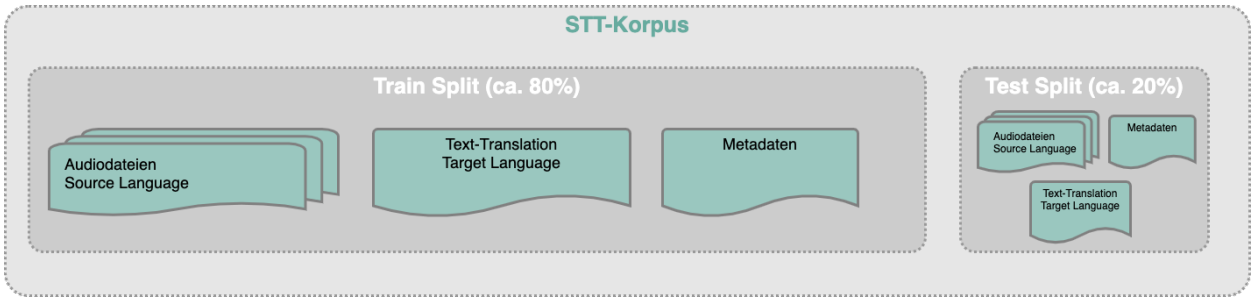

Abbildung 7: Aufbau STT-Korpus

4.1. Übersicht

Das Erstellen eines Korpus für Schweizerdeutsch ist eine herausfordernde Aufgabe, da es sich um keine offizielle Sprache mit einheitlicher Grammatik, Schreibweise und Wortschatz handelt. Unter Schweizerdeutsch wird eine Vielfalt von Dialekten verstanden, welche in der Deutschschweiz gesprochen werden. Es wurden bereits ein paar Korpora für Speech Training verschiedener Art erstellt. Keiner von diesen deckt jedoch gleichmässig alle Variationen der Schweizerdeutschen Dialekte ab. In Tabelle 2 wird eine Übersicht, über die in dieser Arbeit verwendeten und / oder analysierten Korpora gegeben. Der Common Voice DE Korpus wurde in der Tabelle aufgeführt, da dieser Korpus auch innerhalb der Arbeit eingesetzt wird (siehe Kapitel 7.3.1).

Korpus	**Zweck**	**# Stunden**	**Dialekte[18] / Sprachen**
Swiss Parliament	Speech Translation (CH - DE)	293.0h	BE
SwissDial	Speech Synthesis (CH)	26.4h	AG, BE, BS, LU, SG, GR, VS, ZH
ArchiMob	Speech Recognition (CH) Linguistische Forschung	69.0h	ZH, BE, BS, GL, UR, SZ, VS, AG, LU, NW, GR, BL, SH, SG [43]
Radio Rottu Oberwallis	Speech Recognition / Speech Translation (CH - DE)	8.3h	VS
Common Voice DE v4	Speech Recognition (DE)	483.0h	DE

Tabelle 2: Übersicht Korpora

In den kommenden Abschnitten 4.2 bis 4.6 werden die Korpora beschrieben und deren Aufbau erläutert.

4.2. Swiss Parliament

Der Swiss Parliament Korpus wurde von der Fachhochschule Nordwestschweiz als STT-Korpus aufgebaut und basiert auf den Audiodateien und Translationen, welche vom Grossen Rat im Kanton Bern aufbereitet wurden [17]. Die Audiodateien sind in Schweizerdeutsch, hauptsächlich im Berner Dialekt aufgenommen. Die korrespondierenden Translationen sind in Hochdeutsch erfasst. Das Alignement zwischen den gesplitteten Audiodateien und den Translationen wurde automatisch durchgeführt. Der Wert Intersection over Unit (IoU) gibt die Auskunft, wie gut die Audiodatei mit der Translation aligniert ist. IoU kommt aus der Bildverarbeitung und beschreibt den Grad der Überlappung zweier Einheiten. Die Interpretation des IoU-Werts ist in der Abbildung 8 ersichtlich[19]:

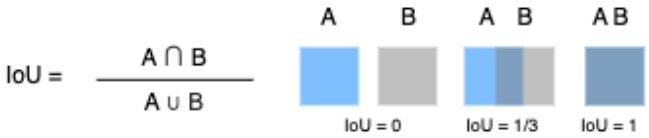

Abbildung 8: IoU

Die Ersteller des Korpus sind gleichzeitig die Organisatoren von Shared Task und haben jenen allen Teilnehmern zur Verfügung gestellt.

Audiodatei:

Das Format der Audiodateien ist FLAC. Die Länge der Audiodateien beträgt zwischen 1 und 15 Sekunden. Alle Dateien sind in einem Ordner abgelegt.

[18] Die Kürzel werden im Anhang «Kantonkürzel» ausgeschrieben.

[19] https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation-model-8b22e2e84686

Translationen / Metadaten:

Die Translationen in Hochdeutsch sind in Test- und Trainset gesplittet. Das Trainset ist in drei Ausführungen aufbereitet:

- All (293h)
- 0.7 IoU (256h)
- 0.9 IoU (176h)

Das Testset besteht aus ca. 6 Stunden Audiodaten.

Zusätzlich werden die folgenden Metadaten gepflegt:

- client_id
- path
- sentence (translation)
- up_votes
- down_votes
- age
- gender
- accent
- iou_estimate

Nicht alle Metadaten sind vollständig. Die Metadaten und Translationen sind in einer CSV-Datei gespeichert. Ein Beispiel ist in Abbildung 9 dargestellt.

client_id	path	sentence	up_votes	down_votes	age	gender	accent	iou_estimate
107	3af2b6f5-8b08-464f-9190-d758f9a75bc3.flac	Er hat, wie es schon die Vorredner gesagt haben, einen sehr guten Eindruck gemacht und verfügt auch einen sehr grossen und breiten Erfahrungsschatz, der uns hier in der bernischen Justiz sehr dienen wird.						0.9271040689329160
224	7e4459c6-ae1c-4cca-a62f-db1248abcaa4.flac	Wir können bis jetzt nicht nachvollziehen, weshalb man vom Gesetz her vorschreibt, dass man eine gemeinsame Buchhaltung führen muss, und weshalb man nicht gerade umgekehrt vorschreibt, dass man den Erfolg aus hauptberuflicher Tätigkeit eben separat ausweist.						0.9413629331977550

Abbildung 9: Sample Swiss Parliament

Fazit:

Der Korpus beinhaltet nur den bernerdeutschen Dialekt und der Wortschatz beschränkt sich auf politische Inhalte. Bezüglich des Ziels der Entwicklung eines Multidialekt Systems ist dies ein Nachteil, weil viele Facetten der Sprache nicht abgebildet werden. Der Korpus wurde von den Autoren als STT-Korpus aufgebaut, wodurch jener für das Training eines STT-Translation Systems gut geeignet ist.

4.3. SwissDial

Der SwissDial Korpus wurde von der Eidgenössische Technische Hochschule (ETH) in Zürich aufgebaut und auf Speech Synthesis Experimenten getestet [18] . Die Audiodateien sind in Schweizerdeutsch, in 8 verschiedenen Dialekten aufgenommen (siehe Tabelle 2). Pro Dialekt

sind im Durchschnitt ca. 3.3 Stunden an Audiodateien mit einer korrespondierenden Transkription in dem gegebenen Dialekt als auch einer hochdeutschen Translation verfügbar. Die Audiodateien wurden in einer Umgebung mit hochqualitativem Audio-Setup aufgenommen. Der Korpus besteht aus sorgfältig ausgesuchten Sätzen, welche einen abwechslungsreichen Wortschatz und ein breites Spektrum an Themen beinhalten. Der gleiche Satz wird in den acht Dialekten vorgelesen, pro Dialekt gibt es einen Sprecher, welcher auch für die Erstellung der Transkriptionen in seinem Dialekt verantwortlich war. Die schweizerdeutschen Transkriptionen sind nicht normalisiert und variieren von Dialekt zu Dialekt.

Audiodatei:

Das Format der Audiodateien ist WAV. Die durchschnittliche Länge der Audiodateien beträgt ca. 4 Sekunden. Die Dateien sind auf 8 Ordner, ein Ordner pro Dialekt, unterteilt.

Translationen / Metadaten:

Alle Translationen sind in einer JSON-Datei enthalten, welche in zwei Ausführungen verfügbar ist:

- sentences_ch_de_numerics.json => Zahlen NICHT ausgeschrieben
- sentences_ch_de_transcribed.json => Zahlen ausgeschrieben

Die Dateien sind nicht in Dev, Train und Test gesplittet. Es sind folgenden Metadaten enthalten:

- ID des Samples
- Hochdeutsche Translation
- Schweizerdeutsche Transkription in acht Dialekten
- Thema des Samples

Ein Beispiel ist in Code Snippet 1 dargestellt.


```
[
    {
        "id": 0,
        "de": "Derzeit ist er in « Parasite », dem Siegerfilm von Cannes, zu sehen.",
        "ch_sg": "Zur Ziit isch er in \"Parasite\", en Siegerfilm vo Cannes, zgseh.",
        "ch_be": "Momentan ischer in \"Parasite\" z gseh, em Siegerfium vo Cannes.",
        "ch_gr": "Derziit isch er in \"Parasite\", am Siegerfilm vu Cannes, z gseh.",
        "ch_zh": "Momentan chamer ihn in \"Parasite\", em Siegerfilm vo Cannes, gseh.",
        "ch_vs": "Derzit ischer in \"Parasite\", dum Siegerfilm va Cannes, zgseh.",
        "ch_bs": "Zur Ziit isch är in \"Parasite\", em Siigerfilm vo Cannes, z gseh.",
        "ch_ag": "Em Moment escher en \"OParasite\", de Siegerfelm fo Cannes z'gseh.",
        "ch_lu": "Momentan escher en \"Parasite\", em Sigerfelm vo Cannes, zgseh.",
        "thema": "kultur"
    }, ….
]
```


Code Snippet 1: Sample SwissDial

Fazit:

Der Korpus ist sehr sorgfältig aufgebaut und bietet eine Vielzahl an Dialekten und einen abwechslungsreichen Wortschatz. Für ein Training von einem STT-Translation System ist der Korpus leider zu klein und muss um zusätzliche Daten erweitert werden.

4.4. ArchiMob

Der ArchiMob Korpus wurde von der Zürcher Universität und der Université de Genève als Korpus für die linguistische Forschung als auch für das Training von ASR-Systemen aufgebaut [19]. Das verwendete Audiomaterial wurde vom Verein ArchiMob[20] in den Jahren 1999-2001 gesammelt und besteht aus Interviews mit den Überlebenden des zweiten Weltkriegs, welche von den Erfahrungen aus dieser Zeit erzählen. Die Transkriptionen sind normalisiert [44] und wurden nachträglich von vier unterschiedlichen Schreibern erfasst. Für den Korpus wurde nur ein Teil der gesamten Kollektion verwendet. Das Alignement zwischen den gesplitteten Audiodateien und Interview-Transkriptionen wurde automatisch mithilfe von mehreren Tools durchgeführt. Die Audiodateien repräsentieren eine Vielzahl an Schweizerdialekten (siehe Tabelle 2) und eine Vielfalt an Sprechern, welche durch die Thematik bedingt im fortgeschrittenen Alter sind. Der Wortschatz bildet dementsprechend nur ein Teil der Sprache ab. Die Audios sind auf Schweizerdeutsch transkribiert. Jedes Wort besitzt eine normalisierte deutsche Entsprechung, welche nicht zwingend dem hochdeutschen Wortschatz entspricht, wie unten anhand eines Beispiels aufgeführt.

Schweizerdeutsche Transkription:

*aber die di sind oben **anegschtanden** und graad **abegfaare** und*

Normalisiert:

*aber die die sind oben **anhingestanden** und gerade **abhingefahren** und*

Audiodatei:

Das Format der Audiodateien ist WAV. Die Länge der Audiodateien beträgt zwischen 4 und 8 Sekunden. Die Dateien sind auf 52 Ordner, ein Ordner pro Interview, unterteilt.

Translationen / Metadaten:

Alle Translationen sind pro Interview in einer XML-Datei gespeichert und sind nicht in Train und Test gesplittet. Folgende Metadaten sind vorhanden:

Pro Audiodatei (tag <u>)

- start
- id
- who

Pro Wort innerhalb der Audiodatei (tag <w>)

- normalised
- id
- text

[20] http://archimob.ch/

Ein Beispiel ist in Code Snippet 2 dargestellt.

```
<u start="media_pointers#d1007-T443" xml:id="d1007-u236" who="person_db#EJos1007">
    <w normalised="so" tag="ADV" xml:id="d1007-u236-w1">son</w>
    <w normalised="ein" tag="ART" xml:id="d1007-u236-w2">es</w>
    <w normalised="hüttlein" tag="NN" xml:id="d1007-u236-w3">hütli</w>
    <w normalised="gehabt" tag="VAPP" xml:id="d1007-u236-w4">ghaa</w>
    <w normalised="ein" tag="ART" xml:id="d1007-u236-w5">es</w>
    <w normalised="kiosk" tag="NN" xml:id="d1007-u236-w6">kiosk</w>
</u>
```

Code Snippet 2: Sample ArchiMob

Zusätzlich wurde eine globale Datei «person_file.xml» erstellt, in der die Informationen zum Sprecher erfasst wurden. Pro Sprecher innerhalb der «person_file.xml» werden die Informationen gespeichert:

- sex
- birth
- occupation
- residence

Fazit:

Der Korpus beinhaltet eine Vielzahl an Dialekten, der Inhalt respektive die Themen der Audiodateien sind jedoch etwa eingeschränkt. Die Transkriptionen eignen sich sehr gut für Training eines ASR-Systems. Die vom Hochdeutschen abweichenden normalisierten Translationen sind nicht optimal für das Training eines STT-Translation Systems. Zudem wurden die Translationen auf der Wortebene durchgeführt, sodass sich in den Translationen teilweise fehlerhafte Satzstrukturen einschleichen.

4.5. Radio Rottu Oberwallis (RRO)

Der RRO Korpus für das Training von ASR-Systemen wurde in Zusammenarbeit mit dem Idiap Research Institut in Martigny als Korpus aufgebaut [15]. Die Audiodateien sind in Walliserdeutsch aufgenommen worden. Die Transkriptionen wurden von zwei Muttersprachlern manuell erfasst. Die Aufnahmen stammen aus dem lokalen Radiosender «Radio Rottu Oberwallis» (PRO) aus Visp. Hochdeutsche Translationen sind nicht vorhanden, es wurden lediglich CSV-Dateien mit einem Mapping von schweizerdeutschen auf hochdeutsche Wörter erstellt. Anbei ein Ausschnitt aus dem Mapping:

Abend, Aabe, Aabe, Aabo, Aabu

Abend, Aabend, Aabend, Aabond, Aabund

ähnlich, äänlich, äänlich, äänlich, äänlich

Audiodatei:

Das Format der Audiodateien ist WAV. Die Länge der einzelnen Audiodateien beträgt ca. 20 Sekunden.

Translationen / Metadaten:

Die Daten sind in Training (6.8h), Development (10min) und Testing (1.3h) gesplittet. Die Transkriptionen sind nicht in einem zentralen Manifest gespeichert. Pro Audiodatei wurde eine separate txt-Datei mit der Transkription erstellt. Das Mapping eines walliserdeutschen Ausdrucks zum hochdeutschen Wort ist auf 5 CSV-Dateien verteilt.

Beispiel:

Walliserdeutsche Transkription (2012-10-01-YW_003.txt):

unner anderem heisst *da zum* ***Bischpill*** *der SVP Nazionalrat Oskar Freysinger nöü Jussuf Freysinger treit en Turban und isch en Flüchtling va Syrie mu will mit der Akzion zum Naadeiche arege*

Das hochdeutsche Mapping für den oberen Satz ist in mehreren CSV-Dateien (t1.csv, t2.csv, …) zu finden. Beispielhaft in Tabelle 3 dargestellt.

File	**Mapping**
t1.csv	unter,**unner**,,,
t2.csv	**anderem**,anderem,anderom,anderum,
t2.csv	**heisst**,heisset,heissot,heissut,
…	…
t2.csv	Beispiel,**Bischpill**,,,

Tabelle 3: PRO Korpus, Mapping Walliserdeutsch - Hochdeutsch

Fazit:

Der Korpus repräsentiert nur den walliserdeutschen Dialekt und hat mit 8.5h an Audiodaten einen sehr kleinen Umfang. Der Aufwand für die Erstellung der fehlenden hochdeutschen Translationen wäre mit grossem Aufwand verbunden. Aus diesem Grund wurde der Korpus in dieser Arbeit nicht für das Training verwendet.

4.6. Common Voice

Der Common Voice Korpus für Hochdeutsch wurde im Rahmen des von Mozilla initiierten Projektes «Common Voice Mozilla»[21] aufgebaut [45]. Der Korpus wird mithilfe von den Freiwilligen online erstellt, Dafür werden vorgegebene Sätze vorgelesen und aufgenommen. Die Audiodaten werden dann von den anderen Benutzern auf ihre Richtigkeit validiert. Auf diese

[21] https://commonvoice.mozilla.org/en

Art hat Mozilla ASR-Korpora für 38 Sprachen erstellt (Stand 11.2019). Thematisch sind die Daten sehr abwechslungsreich und repräsentieren einen breiten Wortschatz.

Audiodatei:

Das Format der Audiodateien ist MP3. Alle Dateien sind in einem Ordner gespeichert.

Translationen / Metadaten:

Die Translationen sind in Test, Train und Dev Set gesplittet. Zudem werden die validierten als auch nicht validierten Daten jeweils in einer TSV-Datei «validated.tsv» respektive «invalidated.tsv» dokumentiert.

Zusätzlich werden die folgenden Metadaten gepflegt:

- client_id
- path
- sentence (translation)
- up_votes
- down_votes
- age
- gender
- accent

Bis auf den IoU-Wert sind es die gleichen Metadaten wie beim Swiss Parliament Korpus (siehe Kapitel 4.2). Ein Beispiel ist in Abbildung 10 dargestellt.

client_id	path	sentence	up_votes	down_votes	age	gender	accent
dfdf96a10ad9874b7e39500942854a67da3590a1099499a845ac38b11c5fe1156860fd806cf2d5e3e56ca1090486ebb6443a4a721613eef5b58ab280ff878dd2	common_voice_sv-SE_19169002.mp3	Von letzteren siedelten sich auch einige im Ort an	2	0	thirties	male	
dfdf96a10ad9874b7e39500942854a67da3590a1099499a845ac38b11c5fe1156860fd806cf2d5e3e56ca1090486ebb6443a4a721613eef5b58ab280ff878dd2	common_voice_sv-SE_19169003.mp3	Der Stadtrat stimmte der Unterstützung derUmbauarbeiten zu	2	0	thirties	male	

Abbildung 10: Sample Common Voice DE

Fazit:

Der Korpus repräsentiert die deutsche Sprache gut und bietet eine Vielzahl an Sprechern und damit unterschiedlichen Aussprachen. Er eignet sich für das Training eines ASR-Systems in Hochdeutsch. Da die Daten nicht im Schweizerdeutsch vorhanden sind, kann der Korpus für das STT-Translation Training nicht verwendet werden.

5. Speech-to-Text Toolkit FAIRSEQ S2T

In diesem Kapitel werden verschiedene Toolkits miteinander vergleichen und die Auswahl von FAIRSEQ S2T [46] begründet. In den weiteren Abschnitten wird eine kurze Einführung in das FAIRSEQ S2T Toolkit und eine Übersicht über seine Funktionen gegeben.

Da es sich bei ST um ein komplexes Problem handelt, ist es sinnvoll die Experimente auf der Basis eines STT-Toolkits aufzubauen und durchzuführen. Es wurden bereits zahlreiche Open Source und kommerzielle Lösungen für diesen Zweck entwickelt. Dabei unterstützt nicht jedes System alle Anwendungen. Bei der Aufgabenstellung für den Shared Task handelt es sich um eine E2E-ST: der Task Korpus besteht aus Audiodateien in Schweizerdeutsch und Translationen auf Hochdeutsch. Es sind keine schweizerdeutschen Transkriptionen vorhanden. Um in einer Vorauswahl die geeigneten Toolkits einzugrenzen, wurden nur die Toolkits weiterverfolgt, welche E2E-ST unterstützen. Tabelle 4 gibt dazu eine Übersicht.

	Toolkit									
	FAIRSEQ S2T [46]	ESPNet-ST [47]	Lingvo [48]	OpenSeq2Seq [49]	RETURNN [50]	SLT.KIT [51]	Tensor2Tensor [5]	OpenNMT [52]	Kaldi [53]	Flashlight (Wav2letter++) [54]
E2E-ST	✓	✓	✓		✓	✓				

Tabelle 4: Übersicht Toolkits für E2E-ST

Im Vergleich zu FAIRSEQ S2T und ESPNet-ST werden die Lingvo, RETURNN und SLT.KIT Toolkits seltener in der Literatur erwähnt. Des Weiteren werden auf den GitHub Repositories von FAIRSEQ S2T[22] und ESPNet-ST[23] täglich neue Tickets eröffnet und neuer Code gemergt. Die restlichen 3 Toolkits sind, gemessen an dem GitHub-Traffic, weniger populär und haben eine deutlich kleinere Community. Aus diesem Grund wurden diese in der weiteren Analyse ausgeschlossen. Im nächsten Schritt wurden die Dokumentationen von FAIRSEQ S2T und ESPNet-ST hinsichtlich ihrer Vollständigkeit und Verständlichkeit überprüft. Besonderes Augenmerk wurde auf das Vorhandensein eines gut dokumentierten Beispiels mit einer nachvollziehbaren Anleitung gelegt. Dabei wurde festgestellt, dass beide Systeme gut dokumentiert sind und über zahlreiche, gut beschriebene Experimente verfügen. Die FAIRSEQ S2T Anwendung erscheint jedoch intuitiver als jene von ESPNet-ST. Zudem ist die Dokumentation von FAIRSEQ S2T sehr strukturiert aufgebaut, sodass die Einarbeitung in FAIRSEQ S2T

[22] https://github.com/pytorch/fairseq/issues

[23] https://github.com/espnet/espnet/issues

schneller erfolgte als jene in ESPNet-ST. Diese Punkte sind jedoch subjektiv und können nicht verallgemeinert werden. Anschliessend wurde vergleichen, wie aktiv die Entwickler Teams auf dem Gebiet der Forschung sind. Diesbezüglich hat das Facebook AI Team, welches FAIRSEQ S2T wartet, einen Vorsprung gegenüber den Entwicklern von ESPNet-ST. In letzter Zeit hat sich Facebook AI auf dem Gebiet von ASR und ST mit vielen interessanten Publikationen bemerkbar gemacht, wobei insbesondere die Veröffentlichung von wav2vec [9] genannt werden muss. Abschliessend ist die Auswahl nicht leichtgefallen, da beide Toolkits viele Vorteile aufweisen. Schlussendlich wurde jedoch FAIRSEQ S2T, aufgrund der gut strukturierten und vollständigen Dokumentation sowie den guten Beispielen und der aktiveren Forschungstätigkeit, bevorzugt.

5.1. Übersicht

Nachfolgend werden die wichtigsten Links bezüglich der Benutzung von FAIRSEQ S2T aufgelistet:

- Online Dokumentation[24]
- Repository mit den Verweisen auf die Beispiele[25]
- Die Einarbeitung in das Toolkit erfolgte anhand des «ST on CoVoST – Beispiels», welches die Anleitung zum Reproduzieren der Experimente aus Wang et al. [6] zur Verfügung stellt[26]

5.2. Funktionalität

In diesem Abschnitt wird eine Übersicht der Funktionen und Schnittstellen von FAIRSEQ S2T gegeben.

FAIRSEQ S2T unterstützt die folgenden Architekturen in unterschiedlichen Ausprägungen:

- Convolutional Neural Networks (CNN)
- Long Short-Term Memory (LSTM)
- Transformer (self-attention) networks

Des Weiteren ist es möglich die Fairseq-Modelle zu erweitern als auch eine eigene Architektur zu registrieren. Für Optimizer, Criterion und Learning Rate Scheduler werden unterschiedliche Implementierungen zur Verfügung gestellt, welche erweitert und angepasst werden können. Die Audiofeatures werden in einen Filterbank Vektor extrahiert, welcher für die weitere Bearbeitung (für das Training) in einer NumPy-Datei gespeichert werden kann. Während dem Training können noch weitere Speech Data Transformationen eingesetzt werden:

[24] https://fairseq.readthedocs.io/en/latest/

[25] https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text

[26] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/docs/covost_example.md

- CMVN (Cepestral Mean and Variance Normalisation)
- Speed Perturbation [55]
- SpecAugment [34]

Diese Massnahmen helfen das Overfitting des Modells abzuschwächen und seine Robustheit zu stärken. Die oben aufgezählten Implementierungen für die Data Transformationen haben eine offene Schnittstelle, sodass diese bearbeitet und angepasst werden können. Für das Text-Pre-Processing werden unterschiedliche Tokenizer integriert:

- Moses[27]
- SentencePiece [26]
- Subword-nmt[28]
- Byte-level BPE (Byte-Pair Encoding) [56]
- Bytes [27]

Tabelle 5 gibt eine Übersicht welche Metriken und Libraries für die Validierung der Systeme verwendet werden.

	Metrik	**Library**
ST	BLEU	sacreblue[29]
ASR	WER	editdistance[30]

Tabelle 5: Übersicht Validierungsmetriken FAIRSEQ S2T

Die in diesem Kapitel aufgeführten Informationen können zum grössten Teil in Wang et al., 2020 [46] nachgelesen werden. Es wurden hier nur die wichtigsten Funktionen und Schnittstellen zusammengefasst, welche auch im Rahmen dieser Arbeit angewendet werden.

[27] https://github.com/moses-smt/mosesdecoder

[28] https://github.com/rsennrich/subword-nmt

[29] https://pypi.org/project/sacrebleu/

[30] https://pypi.org/project/editdistance/0.3.1/

6. Vorgehen und Methodik

Im nachfolgenden Kapitel wird das Vorgehen und der Trainingsprozess beschrieben. Dabei wird auf die einzelnen Komponenten, Prozesse und die generierten Artefakte eingegangen, welche bei der Durchführung der Experimente ausgeführt respektive erzeugt werden.

6.1. Übersicht Prozess

Der gesamte Trainingsprozess mit allen dazugehörenden Pre-und Post-Processing Schritten ist in Abbildung 11 dargestellt. Man kann der Graphik entnehmen, dass der Prozess in drei Stufen aufgeteilt werden kann:

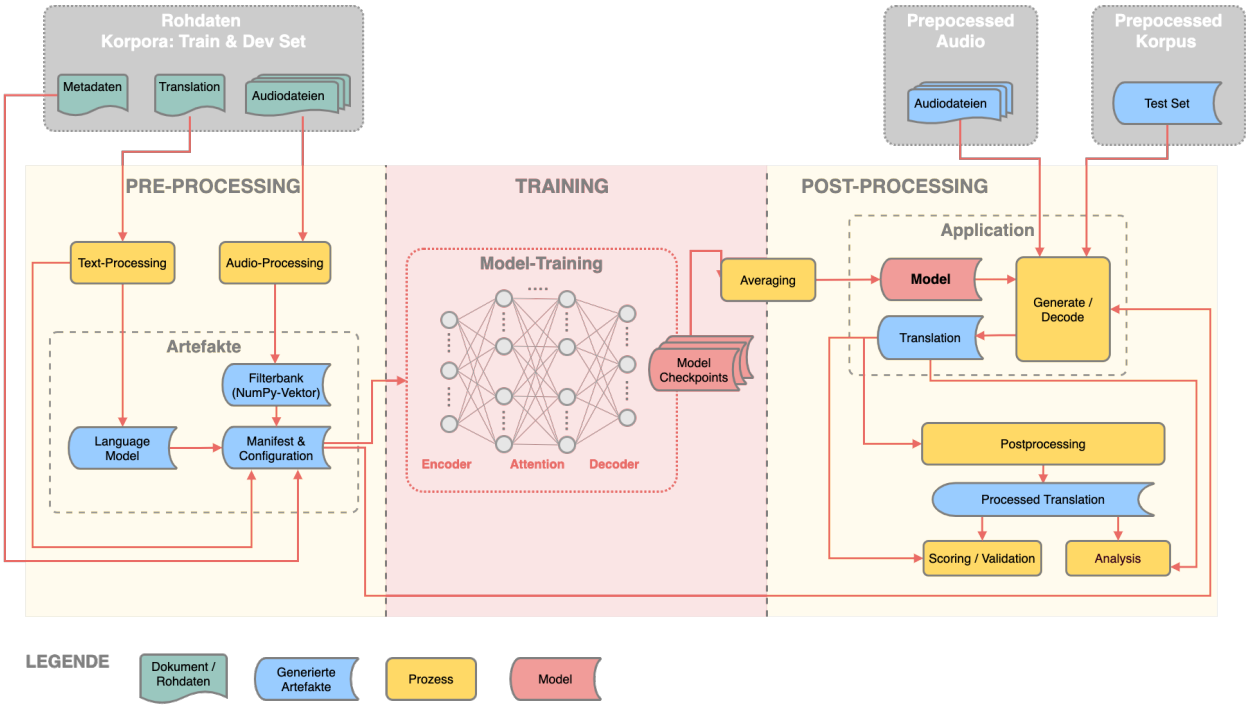

Abbildung 11: Übersicht STT Translation Training Prozess

1. **Pre-Processing**

   Innerhalb des Pre-Processings werden die Rohdaten vorbearbeitet und in eine Form gebracht, welche vom Modell prozessiert werden kann. Die Audiodateien werden in einen Vektor umgewandelt, die Translation-Texte normalisiert. Anschliessend werden alle für das Training notwendige Artefakte, wie das Manifest, die Konfigurationsdatei und das Language Model generiert.

2. **Training**

   Das Training ist der eigentliche Prozess des Lernens, in dem das Modell lernt, die Translationen zu generieren. Alle Experimente in dieser Arbeit wurden auf Modellen durchgeführt, welche auf der Transformer Architektur basieren (siehe Kapitel 6.4).

3. **Post-Processing**

   Im Post-Processing wird als erstes das Modell durchs Averaging (siehe Kapitel 6.5.1) optimiert. Des Weiteren findet in diesem letzten Prozessschritt das Nachbearbeiten der Translationen, die Validation / Scoring sowie Analysis der Translationen statt.

6.2. Rohdaten

In diesem Schritt werden die Rohdaten des Korpus analysiert und auf ihre Eignung für das Training geprüft. Die Bestandteile eines Korpus sind in Abbildung 12 veranschaulicht.

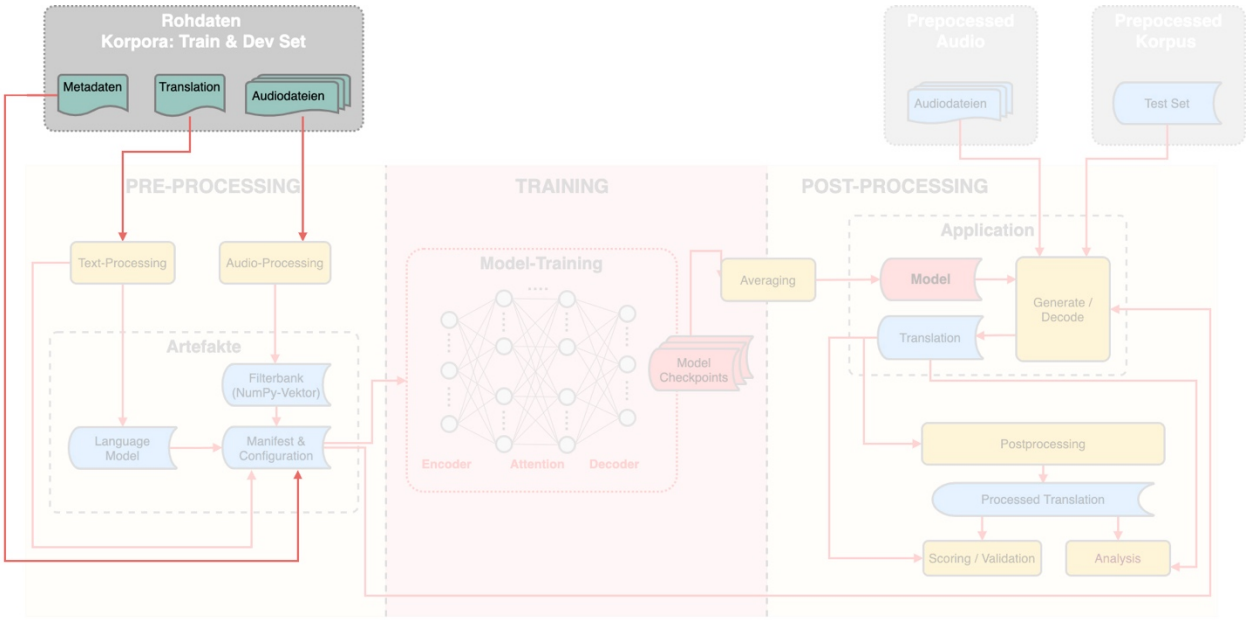

Abbildung 12: Piktogramm Rohdaten

Als erstes wird der Zweck vom Korpus ausfindig gemacht und der Aufbau analysiert, um festzustellen, ob jener für die STT-Translation geeignet ist. Beim Aufbau wird angeschaut, ob die Daten bereits in Training, Dev und Test Set aufgeteilt sind. Falls dies nicht der Fall ist, wird nach einem passenden Kriterium für die Aufteilung gesucht. Als nächstes wird die Struktur angeschaut und die folgenden Fragen beantwortet: Sind die Dateien auf mehrere Ordner verteilt? Gibt es ein oder mehrere Manifeste, in welchen die Translationen und Metadaten für die gesamte Kollektion gespeichert sind? Des Weiteren wird die Länge der einzelnen Audiodateien und das Vorhandensein einer geeigneten Translation in Hochdeutsch verifiziert. Die für die Arbeit verwendeten Korpora werden in Kapitel 4 beschrieben. Das Vorgehen bei Aufteilung der Daten auf Train-, Test- und Devset wird in Kapitel 6.3.4 erläutert.

6.3. Pre-Processing

Das Pre-Processing ist ein sehr wichtiger Prozess, mit welchem die Modell Performance stark beeinflusst werden kann. Aus diesem Grund ist es sehr wichtig diese Aufgabe sorgfältig zu analysieren. Im ersten Schritt werden die Translationen normalisiert und die Audiomerkmale aus den Audiodateien extrahiert. Anschliessend werden das Language Modell, das Manifest und die Konfigurationsdatei erstellt. Alle aufgezählten Artefakte werden als Input für das Modell-Training verwendet. Abbildung 13 visualisiert schematisch diesen Vorgang.

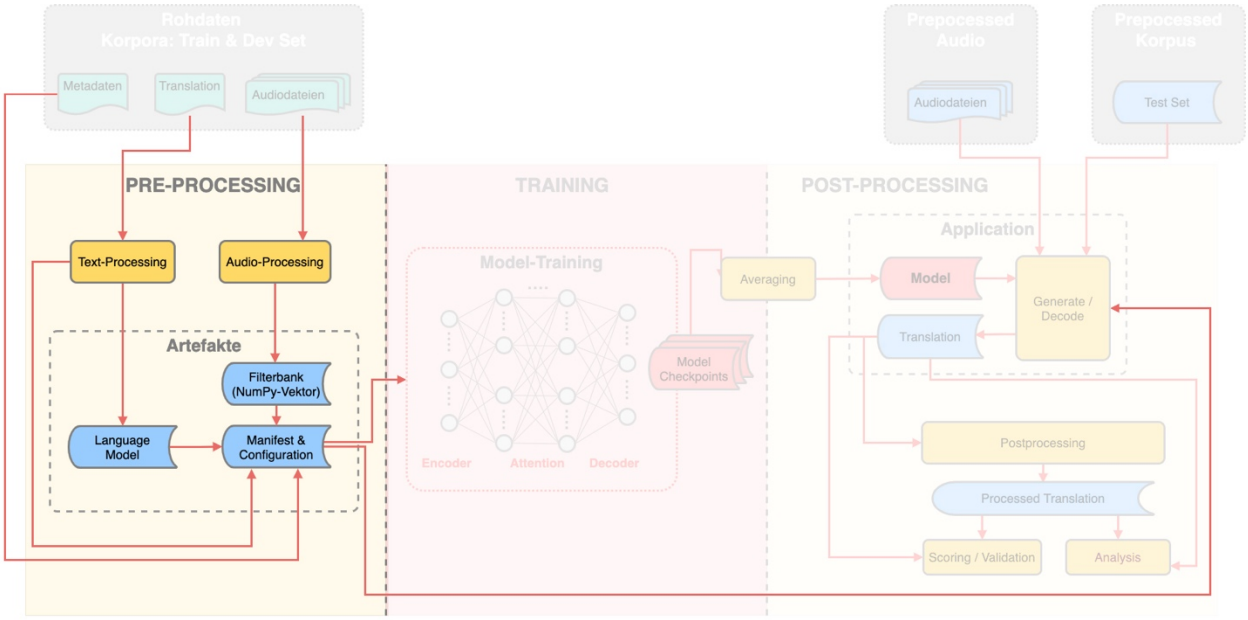

Abbildung 13: Piktogramm Pre-Processing & Artefakte

Der Ansatz für das Pre-Processing muss in Abhängigkeit von dem verwendeten STT-Translation Toolkit und der Architektur gewählt werden. Die Kombination von beiden gibt vor:

- wie das Audio- und Text-Processing durchgeführt werden müssen
- welche Language Modelle unterstütz werden
- wie das Manifest und die Konfigurationsdatei zu strukturieren sind

Das STT-Translation Toolkit wird in Kapitel 5 und das Modell in Kapitel 6.4 beschrieben. In den Abschnitten 6.3.1 bis 6.3.5 werden die einzelnen Prozesse und Artefakte des Pre-Processings erläutert und deren Umsetzung aufgezeigt.

6.3.1. Audio-Processing & Filterbankgenerierung

In dieser Arbeit werden die Merkmale aus den Audiodateien in eine 80-dimensionale Log Mel-Scaled Filterbank mit einer Framegrösse von 25ms und einer Überlappung von 10ms extrahiert. Die Filterbank-Vektoren werden anschliessend pro Audiodatei in einer npy-Datei (NumPy Vektor) gespeichert. Das Prozessieren der Audiodateien erfolgt mit der FAIRSEQ S2T Implementierung, mit der Methode `extract_fbank_features()`[31]. Diese Methode erwartet drei Argumente: die Audio-Wellenfrom (waveform), die Samplingrate (sample rate) und den Pfad der Audiodatei. Die ersten zwei Eigenschaften werden mit Hilfe der tochaudio[32] Library aus der Audiodatei extrahiert.

FAIRSEQ S2T stellt Online Speech Data Transformationen zur Verfügung, welche während dem Training auf die extrahierten Filterbank Speech Features angewendet werden. Für das Training der Modelle werden Cepestral Mean and Variance Normalisation (CMVN) [57] und SpecAugment [34] verwendet.

Das Prozessieren von langen Audiodateien kann die Dauer des Trainings negativ beeinflussen. Aus diesem Grund sortiert FAIRSEQ S2T, vor dem Start des Trainings alle Audiodateien, welche mehr als 3000 Frames lang sind, aus.

6.3.2. Text-Processing

Im Text-Processing werden die Translationen normalisiert. Im ersten Schritt erfolgt die Definition der erlaubten Zeichen. Anschliessend werden Translationen prozessiert und alle nicht erlaubten Zeichen entfernt oder ersetzt. Die Definition der erlaubten Zeichen ist eine Grundsatzentscheidung, da diese Auswahl im Nachhinein den Modell-Output[33] als auch die Modell-Performance bestimmt. Steht die Modell-Performance im Vordergrund wird eine Textnormalisierung empfohlen, welche auf die Klein- Grossschreibung, Interpunktion und alle Sonderzeichen verzichtet. Diese Art der Normalisierung wurde im Shared Task vorgegeben:

ALLOWED_CHARS = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'ä', 'ö', 'ü', ' ', }

Anbei Beispiel eines möglichen Outputs:

ueli studer von der gemeinde köniz war fünfundzwanzig stunden am telefon

Man sieht, dass weder die Eigennamen noch die Substantive grossgeschrieben werden, zudem werden die Zahlen ausgeschrieben. Für ein kommerzielles STT Translation System wird ein Output, welcher die deutschen Schreibregeln beachtet, bevorzugt:

---

[31] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/data_utils.py

[32] https://pytorch.org/audio/stable/index.html

[33] Das Bearbeiten vom Modell Output kann auch nachträglich im Post-Processing erfolgen.

Ueli Studer von der Gemeinde Köniz war 25 Stunden am Telefon.

Trainiert man ein solches Modell, muss die Liste der erlaubten Zeichen um Grossbuchstaben, Interpunktion, Zahlen usw. erweitert werden. Mit dieser Erweiterung steigt die Fehlerwahrscheinlichkeit, welche den Score des Modells negativ beeinflusst.

Diese Arbeit fokussiert sich auf der Shared Task Vorgabe und übernimmt für die Textnormalisierung die oben aufgeführten «ALLOWED_CHARACHTER». Für das Ausschreiben der Zahlen in den Translationen wurde die Library num2words[34] verwendet. Der Link zum verwendeten Script kann in der Fusszeile gefunden werden[35].

6.3.3. Language Model (LM)

Das Language Model bestimmt, welches Vokabular dem Modell zur Verfügung steht. Nur dieses Vokabular wird vom Encoder für das Generieren der Predicitons verwendet. In dieser Arbeit wird mit einem LM auf dem Character Level gearbeitet. Das LM wurde mit einer FAIR-SEQ-S2T Implementierung `gen_vocab()`[36] erstellt, welche auf der Library SentencePiece [26] basiert. Im Verlauf der Arbeit wurden drei LMs erstellt, welche sich durch die Grösse des Vokabulars unterscheiden.

- LM All
- LM NN
- LM N

Das LM All wurde auf nicht normalisierten Translationen erstellt und hat eine Vokabular-Grösse von 155 Zeichen. Das LM NN wurde auf den Translationen, welche nur alle im Shared Task zulässigen Zeichen und Zahlen enthielten, generiert und hat eine Vokabular-Grösse von 79 Zeichen. Das LM N wurde auf den normalisierten Translationen erstellt, sodass die Zahlen im Vokabular nicht inkludiert sind. Jene Vokabular-Grösse beträgt 59 Zeichen. Tabelle 6 zeigt, welche Modelle, mit welchem LM trainiert wurden. Die Übersicht der trainierten Modelle kann in Kapitel 7.1 gefunden werden.

[34] https://pypi.org/project/num2words/

[35] https://github.com/dubelbog/st_ch_de/blob/master/audio_preparation_e2e.py

[36] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/data_utils.py

	STT SYSTEME										ASR ENCODER		
	SP 0.9	SP All	SP & SD	SP & AM	SP + ASR DE	SP & SD + ASR DE	SP & AM + ASR DE	SP & SD + ASR DE (EN)	SP & SD + ASR CH	SP & SD & AM + ASR CH	ASR DE	ASR DE (EN)	ASR CH
LM All (155)	✓	✓											
LM NN (79)			✓		✓								
LM N (59)				✓		✓	✓	✓	✓	✓	✓	✓	✓

Tabelle 6: Übersicht LM

Die Modelle SP 0.9 und SP All, welche mit LM All trainiert wurden, waren die ersten Experimente in dieser Arbeit. Nach der Analyse von LM All wurde festgestellt, dass es viel zu grosses Vokabular im Vergleich zu «erlaubten Zeichen» aufweist. Um das LM zu optimieren und an das zulässige Vokabular anzupassen, wurde ein zweites LM, das LM NN erstellt. Mit dem LM NN wurden die Modelle SP & SD und SP + ASR DE trainiert. Die Predicitons von diesen Modellen wurden im Post-Processing nachbearbeitet, indem die Zahlen mit Hilfe von num2words ausgeschrieben wurden. Zum Einsparen dieses Schrittes wurde das letzte LM N erstellt. Die restlichen Experimente wurden mit dem LM N durchgeführt. Als Konsequenz generieren diese Modelle, ohne zusätzlichen Post-Processing Schritt, die gewünschten normalisierten Translationen.

6.3.4. Manifest

Das Manifest stellt dem Modell während dem Training folgende Informationen zur Verfügung:

- Sample-ID
- Pfad zur Audiofeature-Datei (Filterbank NumPy Vector)
- Translation
- Anzahl der Frames
- Metadaten, welche in Abhängigkeit des verwendeten Korpus variieren

Das FAIRSEQ S2T Modell erwartet eine Manifest-Datei in TSV-Format. In Tabelle 7 werden die Metadaten beschrieben, welche im Manifest zu speichern sind.

Metadata	**Beschrieb**
id	Eindeutige Identifizierung des Samples.
audio	Pfad zur Audiofeature-Datei. In dieser Arbeit der Pfad zur npy-Datei, in welcher der Filterbank Vektor gespeichert ist, siehe Kapitel 6.3.1.
n_frames	Anzahl Frames innerhalb von der Audiodatei. Aus Effizienzgründen werden alle Audiodateien mit mehr als 3000 Frames vor dem Training aussortiert und nicht für das Training verwendet.
tgt_text	Die alignierte Translation.
speaker (optional)	Speaker-ID

Tabelle 7: Aufbau Manifest

Diese Informationen müssen aus jedem Korpus extrahiert und mit dem Manifest-Aufbau gemappt werden. Untenstehend wird dieses Mapping für die einzelnen Korpora erläutert. Die Korpora und dessen Aufbau kann in Kapitel 4 nachgeschlagen werden. Tabelle 8 zeigt übersichtlich, welches Modell mit welchem Korpus trainiert wurde. Die Übersicht der trainierten Modelle kann in Kapitel 7.1 gefunden werden.

	STT SYSTEME										**ASR ENCODER**		
	SP 0.9	SP All	SP & SD	SP & AM	SP + ASR DE	SP & SD + ASR DE	SP & AM + ASR DE	SP & SD + ASR DE (EN)	SP & SD + ASR CH	SP & SD & AM + ASR CH	ASR DE	ASR DE (EN)	ASR CH
Swiss Parliament	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
SwissDial			✓			✓		✓	✓	✓			✓
ArchiMob				✓			✓			✓			✓
Common Voice DE											✓	✓	

Tabelle 8: Übersicht Korpora

Swiss Parliament

Der Swiss Parliament Korpus ist der Shared Task Korpus und wurde für das Training aller STT-Systeme verwendet. Der Aufbau des Korpus kann in Kapitel 4.2 gefunden werden. Tabelle 9 zeigt das Mapping zwischen den Informationen aus dem Swiss Parliament Korpus und dem Manifest.

Metadata	SWISS PARLIAMENT
id	«path»
audio	Absolut Pfad zu Audiofeature in npy-Format
n_frames	Extrahiert aus der Audiodatei mit dem Library torchaudio[37]
tgt_text	«sentence»
speaker	«client_id»

Tabelle 9: Mapping - Swiss Parliament

Für die Validierung wurde der Test Split verwendet.

SwissDial

Der SwissDial Korpus wurde für das Training von mehreren STT-Systemen und dem ASR-System ASR CH verwendet. Der Aufbau des Korpus kann in Kapitel 4.3 gefunden werden. Tabelle 10 zeigt das Mapping zwischen den Informationen aus dem SwissDial Korpus und dem Manifest.

Metadata	SWISSDIAL
id	Dateiname
audio	Absolut Pfad zu Audiofeature in npy-Format
n_frames	Extrahiert aus der Audiodatei mit dem Library torchaudio[37]
tgt_text	STT: Hochdeutsche Translation («de»)
	ASR: Schweizerdeutsche Transkription
speaker	nicht gepflegt

Tabelle 10: Mapping – SwissDial

Der Korpus ist nicht in Test, Train und Dev aufgeteilt, sodass ein Split erstellt werden musste. Beim SwissDial Korpus sind zwei Split-Varianten möglich:

- «Sentence Level»
- «Dialect Level»

Beim «Sentence Level» erfolgt das Split auf der Satzebene, sodass ein Satz, welcher in verschiedenen Dialekten vorgelesen wird, jeweils nur in einem Set vorkommt. So kann sichergestellt werden, dass im Testset keine Sätze vorkommen, welche das Modell schon während dem Training in einem anderen Dialekt gehört hat. Der Nachteil des Splits ist, dass im Testset und Trainset Audiodateien von gleichen Sprechern vorkommen. Eine Alternative zum Umgehen dieses Problems wäre ein Split auf dem «Dialect Level», welches wie folgt aussieht: das Testset besteht aus den vollständigen Daten eines Dialekts, das Trainset beinhaltet die restlichen 7 Dialekte. Der Nachteil bei dieser Lösung ist jedoch, dass das Modell nicht auf den Daten eines Dialekts trainiert wird und ihn somit nicht lernen kann. Um dem Modell eine

[37] https://pytorch.org/audio/stable/index.html

Chance zu geben, alle Dialekte zu «hören», wurde der Split «Sentence Level» für das Training der STT- und ASR-Systeme ausgewählt. In Kapitel 8.1.1 werden jedoch Experimente mit dem Split auf dem «Dialect Level» durchgeführt.

Das «Sentence Level» Testset enthält 2.74 Stunden an Audiodaten. Die Länge der Testsets für «Dialect Level» entspricht der Anzahl Stunden, welche in diesem Dialekt vorhanden sind, siehe Tabelle 26. Die Scripts zum Erstellen der Manifeste für den SwissDial Korpus sind in diesem Folder[38] zu finden, siehe Fusszeile.

ArchiMob

Der ArchiMob Korpus wurde für das Training von mehreren STT-Systemen und dem ASR-System ASR CH verwendet. Der Aufbau des Korpus kann in Kapitel 4.4 gefunden werden. Tabelle 11 zeigt das Mapping zwischen den Informationen aus dem ArchiMob Korpus und dem Manifest.

Metadata	**ARCHIMOB**
id	<Interview-Nummer>/<Audiodateiname>
audio	Absolut Pfad zu Audiofeature in npy-Format
n_frames	Extrahiert aus der Audiodatei mit dem Library torchaudio[37]
tgt_text	STT: «normalised»
	ASR: «text» (Inhalt des Tags <w>)
speaker	«who»

Tabelle 11: Mapping – ArchiMob

Der Korpus ist ähnlich wie bei SwissDial nicht in Test, Train und Dev aufgeteilt, sodass ein Split erstellt werden musste. Die Daten wurden auf dem «Interview-Level» aufgeteilt, indem das Testset aus zwei vollständigen Interviews «1203» und «1261» besteht und das Trainset aus den restlichen Interviews. Somit besteht das Testset aus 3.86 Stunden an Audiodaten. Der Script zum Erstellen der Manifeste für den ArchiMob Korpus sowie die Manifeste sind in diesem Folder[39] zu finden, siehe Fusszeile.

Common Voice v4 DE

Der Common Voice Korpus wurde für das Training von ASR-Systemen ASR DE und ASR DE (EN) verwendet. Der Aufbau des Korpus kann in Kapitel 4.6 gefunden werden. Tabelle 12 zeigt das Mapping zwischen den Informationen aus dem Common Voice Korpus und dem Manifest.

[38] https://github.com/dubelbog/st_ch_de/tree/master/swissdial

[39] https://github.com/dubelbog/st_ch_de/tree/master/archimob

Metadata	COMMON VOICE DE V4
id	«path»
audio	Absolut Pfad zu Audiofeature in npy-Format
n_frames	Extrahiert aus der Audiodatei mit dem Library torchaudio[37]
tgt_text	«sentence»
speaker	«client_id»

Tabelle 12: Mapping - Common Voice DE

Für die Validierung wurde der Test Split verwendet.

Multi- Korpus-Training

Für die Experimente mit mehreren Korpora wurde ein Manifest erstellt, in welchem die Manifeste aus den entsprechenden Korpora zusammengefügt wurden.

6.3.5. Configuration

In der Konfigurationsdatei werden die für das Training relevanten Einstellungen definiert. Alle Modelle in dieser Arbeit wurden mit der gleichen Konfigurationsdatei trainiert. Das verwendete Format ist YAML.

Die Konfigurationsdatei wurde mit der FAIRSEQ S2T Implementierung, mit der Methode `gen_config_yaml()`[40], generiert. Für SpecAugment und Online Speech Data Transformationen wurden die Default Einstellungen übernommen, welche von FAIRSEQ für das Training auf dem CoVoST Korpus verwendet wurden.

[40] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/data_utils.py

6.4. Training

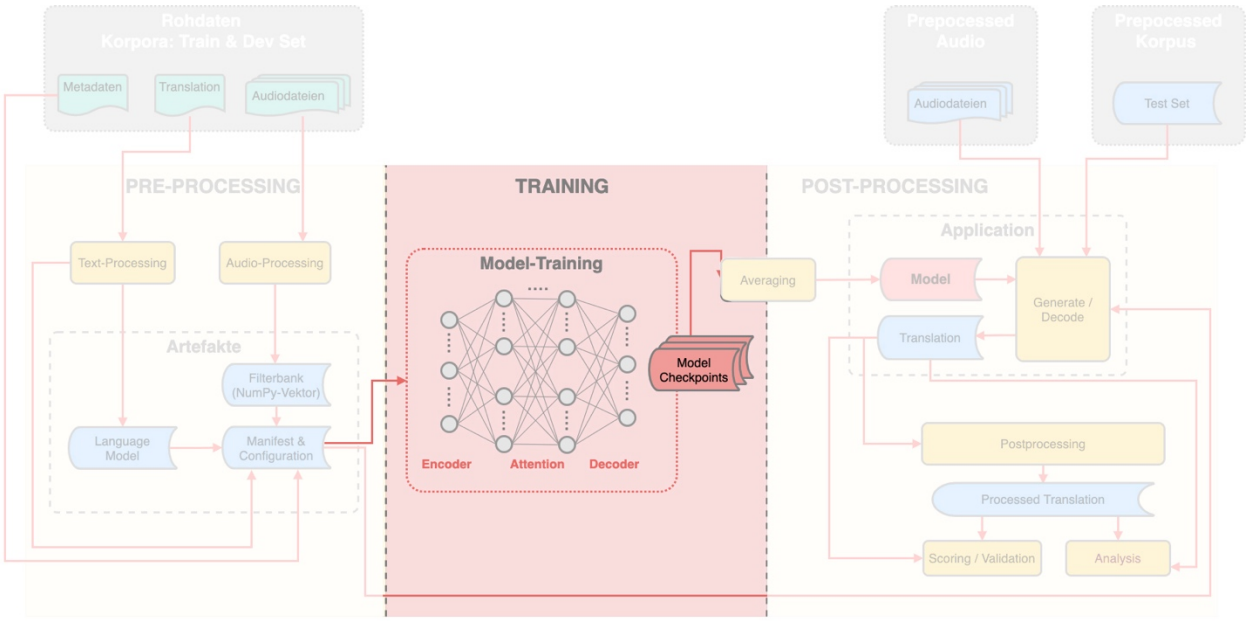

Abbildung 14: Piktogramm Training

Während des Trainings findet der tatsächliche Prozess des Lernens statt. Es werden die im Pre-Processing generierten Artefakte entgegengenommen. Der Output sind die STT-Modell Checkpoints. Jeder Checkpoint bildet für sich ein vollständig ausgebildetes Modell. Diese werden im Averaging Prozess (siehe Kapitel 6.5.1) zu einem optimalen STT-Modell prozessiert. Die Schritte sind schematisch in Abbildung 14 visualisiert.

Alle Experimente wurden mit der Transformer Architektur von FAIRSEQ S2T, mit der Ausprägung Small «s2t_transformer_s» durchgeführt. Diese Modell-Architektur besteht aus 256 Dimensionen, 12 Encoder-Layers, 6 Decoder-Layers, was 27 Millionen Parameters ergibt. Um die Ergebnisse der Experimente miteinander zu vergleichen und die Auswirkung der einzelnen Massnahmen messen zu können, werden alle Systeme mit den gleichen Parameter-Einstellungen trainiert:

- Optimizer: Adam
- Criterion: label smoothed cross entropy
- Label Smoothing: 0.1
- Dropout: 0.1
- Learning Rate Scheduler: inverse square root

Die Definition des FAIRSEQ S2T Modells kann im Script `2t_transformer.py`[41] , siehe Fusszeile, gefunden werden.

6.4.1. ASR-Encoder

FAIRSEQ S2T bietet zwei Möglichkeiten zum Training von ASR-Encodern (Acoustic Model):

1. Online: der Encoder wird während dem Training vom STT-Modell trainiert. Es ist kein pre-training des Encoders notwendig.
2. Training eines ASR-Encoders mit der gleichen Transformer-Architektur, wie oben beschrieben. Anschliessend kann der vortrainierte Encoder in das Training von STT-Modell einbezogen werden. Es ist zu beachten, dass die Architekturen vom Encoder und dem STT-Modell übereinstimmen müssen.

In dieser Arbeit wurden STT-Modelle mit beiden Ansätzen trainiert. Tabelle 13 gibt eine Übersicht, welche STT-Modelle mit welchem Encoder trainiert wurden.

	STT SYSTEME										**ASR ENCODER**		
	SP 0.9	SP All	SP & SD	SP & AM	SP + ASR DE	SP & SD + ASR DE	SP & AM + ASR DE	SP & SD + ASR DE (EN)	SP & SD + ASR CH	SP & SD & AM + ASR CH	ASR DE	ASR DE (EN)	ASR CH
Online	✓	✓	✓	✓									
ASR DE					✓	✓	✓						
ASR DE (EN)								✓					✓
ASR CH									✓	✓			
ASR EN (Librispeech)[40]												✓	

Tabelle 13: Übersicht ASR Encoder

Man kann der Tabelle entnehmen, dass die ASR-Encoders auch mit einem vortrainierten Encoder trainiert wurden. Dieser Ansatz wurde von Wang et al. in [6] in der Einführung des CoVoST Korpus beschrieben. Wang et al. trainierten Encoders für alle Sprachen mit einem Encoder, welcher auf dem englischen Librispeech-Korpus [58] vortrainiert wurde. Dieser Vorgang hilft das Overfitting des Modells abzuschwächen. Dieser Ansatz wurde auch in dieser Arbeit durch das Trainieren des Encoders ASR DE (EN) ausprobiert, siehe Kapitel 7.3.1. Das Modell ASR EN (Librispeech) wurde vom FAIRSEQ S2T Repository[42] heruntergeladen.

[41] https://github.com/pytorch/fairseq/blob/master/fairseq/models/speech_to_text/s2t_transformer.py

[42] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/docs/librispeech_example.md

6.4.2. Checkpoints

Die Checkpoints sind Zwischenstände des Modellzustands nach der n-ten Iteration. Ein STT-Checkpoint kann als STT-Modell ohne zusätzliche Verarbeitung für die Generierung der Translationen verwendet werden. Per Default speichert FAIRSEQ STT ein Checkpoint nach jeder Iteration. Diese Einstellung kann mit dem Parameter `--save-interval` überschrieben werden. Für die meisten Modelle, welche in dieser Arbeit trainiert wurden, wurde das Intervall auf 5 gesetzt, d.h. jede fünfte Iteration wurde ein Checkpoint gespeichert.

6.5. Application

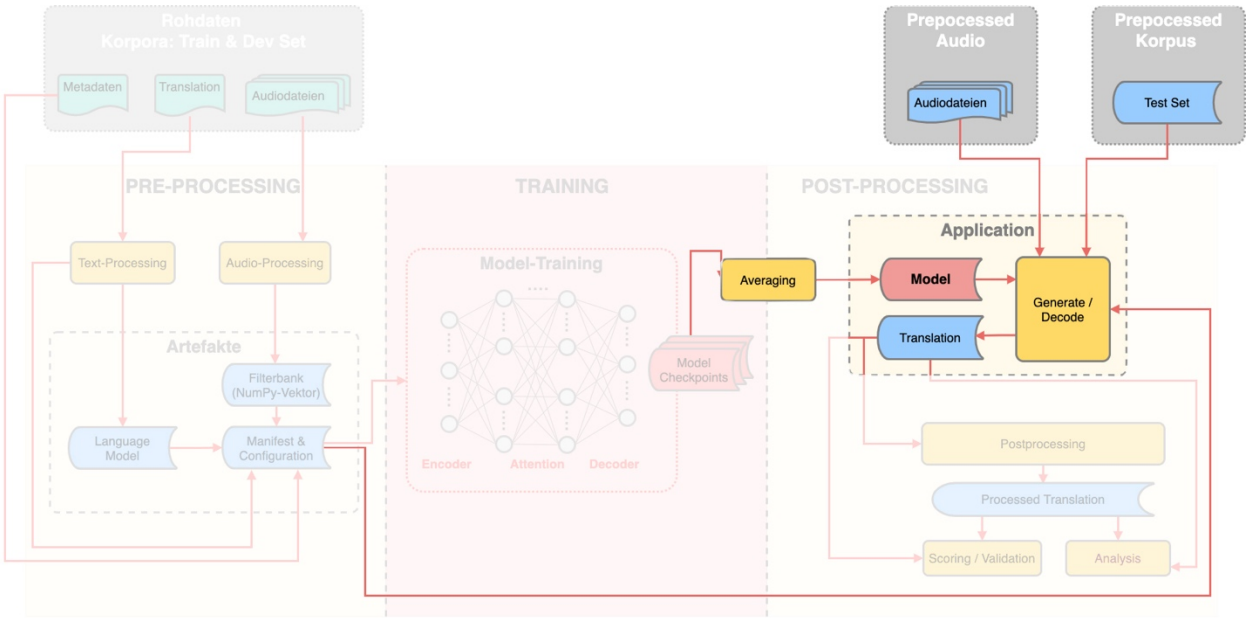

Abbildung 15: Piktogramm Application

In diesem Abschnitt wird die Anwendung des Modells beschrieben. Als erstes wird mit dem Averaging Prozess das optimale Modell aus den gegebenen Checkpoints berechnet. Dieses Modell, die zu übersetzenden vorprozessierten Audiodateien und die im Pre-Processing erstellte Konfigurationsdatei werden dem Prozess Generate übergeben. Dabei können die Audiodateien aus dem Korpus Test Set stammen oder beliebige Dateien sein. Alle Audiodateien müssen im Vorfeld in einen Filterbank-Vektor umgewandelt werden. Der Prozess Generate ist für die Generierung der Translationen zuständig und gibt diese als Output aus. Der oben beschriebene Prozess ist schematisch in Abbildung 15 dargestellt.

6.5.1. Averaging & Modell

Mit dem Averaging wird aus den Checkpoints ein optimales Modell berechnet. Als Input wird eine beliebige Anzahl an Checkpoints übergeben. Innerhalb des Averaging werden die Modellgewichte aller übergebenen Checkpoints gemittelt. Aus diesen gemittelten Gewichten wird ein neues Modell erstellt, welches der Output von Averaging ist. Für diesen Vorgang wird die Implementierung von FAIRSEQ S2T verwendet, welche im Python-Script `average_checkpoints.py`[43] zu finden ist, siehe Fusszeile.

In dieser Arbeit wurden je nach Modell 5 bis 15 Checkpoints fürs Averaging verwendet. Dabei wurden jene mit den besten BLUE-Scores auf dem Swiss Parliament Test Set ausgewählt. Der BLEU-Score von dem gewichteten Modell auf dem Swiss Parliament Test Set wird dadurch im Durchschnitt um ca. 2 BLEU Punkte gegenüber dem besten Score von den übergebenen Checkpoints verbessert.

6.5.2. Generate & Translation

Im Prozess Generate werden die Translationen erstellt. Die als Input übergebenen Artefakte erfüllen die folgenden Aufgaben / Funktionen:

- **Modell**: Die Modellgewichte, welche im Modell gespeichert sind, geben an, wie die Inferenz der Audiofeatures-Vektoren auf die Translation zu erfolgen ist.
- **Konfigurationsdatei**: gibt an, wo das zur Dekodierung benötigte LM zu finden ist und mit welchen Einstellungen die Online Data Transformation durchzuführen ist.
- **Audiodateien als Filterbank**: sind die zu prozessierende Dateien.

In Abhängigkeit vom Input sind folgende zwei Anwendungen möglich:

1. Interaktive Übersetzung einer einzelnen Audiodatei.
2. Übersetzung und Validierung der Audiodateien aus einem Test Set. Dieser Fall wird im Kapitel 6.6.2 beschrieben.

Für die erste Anwendung kann die FAIRSEQ S2T Implementierung aus dem Script `interactive.py`[44] verwendet werden. Die Umwandlung der Audiodatei in einen Audiofeatures-Vektor erfolgt innerhalb des Scripts, sodass im Vorfeld kein Pre-Processing durchgeführt werden muss.

[43] https://github.com/pytorch/fairseq/blob/master/scripts/average_checkpoints.py

[44] https://github.com/pytorch/fairseq/blob/master/fairseq_cli/interactive.py

6.6. Post-Processing

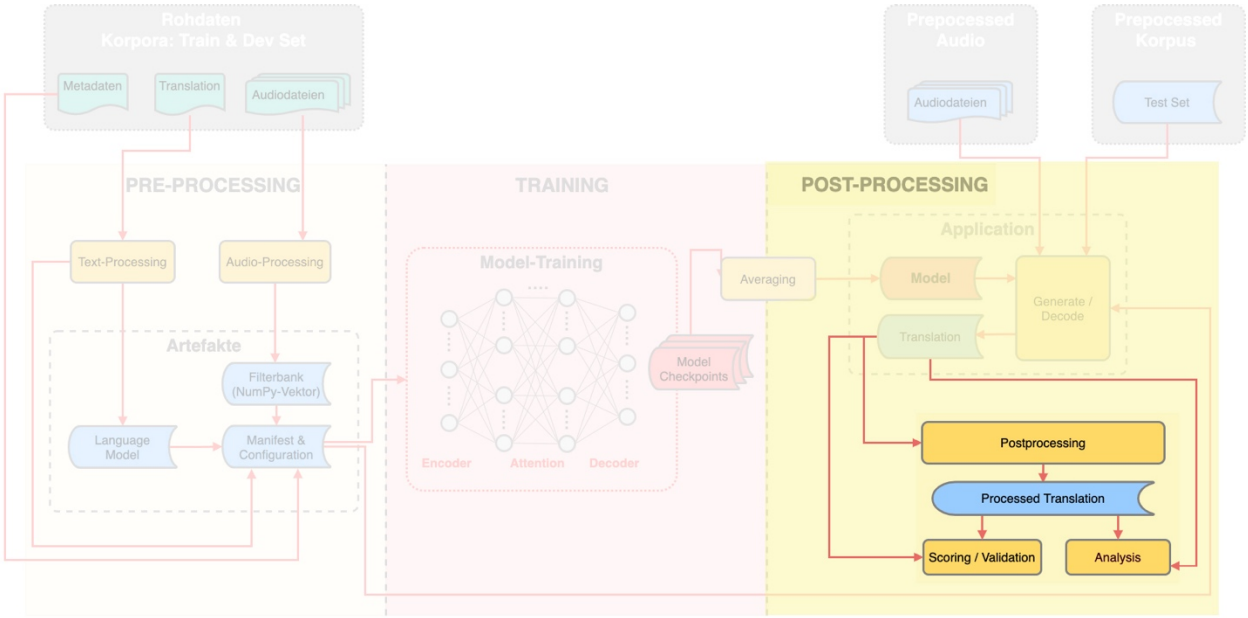

Abbildung 16: Piktogramm Post-Processing

Im Post-Processing werden die generierten Translationen überarbeitet, um z.B. das Ergebnis einer Validierung zu erhöhen oder in einem kommerziellen System die Übersetzungen leserlicher zu gestalten. Die im vorherigen Abschnitt generierten Translationen werden als Input in den Post-Processing Prozess übergeben. Die prozessierten Übersetzungen können weiter analysiert werden oder für das Berechnen der Modell-Performance (Scoring / Validation) verwendet werden. Analysis und Validation kann auch mit den nicht prozessierten Translationen durchgeführt werden. Der hier beschriebene Prozess ist in Abbildung 16 veranschaulicht.

6.6.1. Post-Processing

Diese Arbeit hat sich nur im kleinen Rahmen, in Bezug auf Zahlendarstellung in den Translationen, mit dem Post-Processing befasst. Die Zahlen in den Translationen aus dem Swiss Parliament Korpus wurden nummerisch dargestellt. In Shared Task wurde jedoch vorgegeben, dass die Zahlen in den Translationen ausgeschrieben werden sollen. Für die Translationen, welche von den Modellen SP 0.9, SP All, SP & SD, SP + ASR DE generiert wurden, wurden die Zahlen im Post-Processing mithilfe von dem Library num2words ausgeschrieben (für die Modellbeschreibung siehe das Kapitel 7.1). Bei allen anderen Modellen wurde dieser Schritt vor dem Training, bereits im Pre-Processing, vorgenommen (siehe Kapitel 6.3.2).

6.6.2. Scoring / Validation

In diesem Schritt wird die Modell Performance gemessen. Die STT-Translation Modelle werden mit der BLEU Metrik ausgewertet, die ASR-Modelle mit der WER. Die beiden Metriken sind im Kapitel 2.4 beschrieben. Für die Validation wird die dafür vorgesehen FAIRSEQ S2T Implementierung `generate.py`[45] verwendet. Das Script wurde für die Zwecke der Arbeit wie folgt angepasst: die Ergebnisse der Auswertungen als auch die Translationen wurden für die spätere Analysis zusätzlich in einer Datei ausgeschrieben und gespeichert. Der Link zum Script kann in der Fusszeile gefunden werden[46].

6.6.3. Analysis

Die generierten Translationen werden für die weitere Analysis verwendet. Die Scripts sind in diesem Folder[47] zu finden. Die Plots wurden mithilfe der matplotlib[48] Library generiert. Mit der pandas[49] Library konnten die grossen Dateien effizient prozessiert werden. Auf die Ergebnisse der Analysen wird in dieser Arbeit nicht eingegangen. Sie dienten dazu ein besseres Verständnis von den Modellen zu gewinnen.

6.7. Infrastruktur

Die Experimente wurden auf dem GPU- und Openstack Cluster der ZHAW durchgeführt. In der ersten Hälfte der Arbeit (die ersten 8 Wochen) standen 2 GPUs auf dem GPU Cluster zur Verfügung. Danach für die restlichen 6 Wochen wurden die Ressourcen auf dem GPU-Cluster auf 4 GPUs verdoppelt. Zusätzlich wurden 2 GPUs auf dem Openstack Cluster freigegeben. Da die zugewiesenen Openstack GPUs auf zwei physisch verteilten Server installiert waren, konnten sie nicht gemeinsam für das Training verwendet werden. Die technische Spezifikation der GPU kann den beiden nachfolgenden Abschnitten entnommen werden.

6.7.1. GPU Cluster

Die auf dem GPU Cluster zur Verfügung stehenden GPUs vom Typ Nvidia TITAN Xp haben einen Arbeitsspeicher von 12GB GDDR5X. Weitere Informationen können der Spezifikation [59] entnommen werden.

Der GPU Cluster besteht aus insgesamt 32 GPUs und wird gemeinsam von mehreren Studenten und ZHAW-Mitarbeitern benutzt. Den Benutzern werden mehr GPUs zugewiesen als

---

[45] https://github.com/pytorch/fairseq/blob/master/fairseq_cli/generate.py

[46] https://github.com/dubelbog/st_ch_de/blob/master/fairseq_cli_stchde/generate.py

[47] https://github.com/dubelbog/st_ch_de/tree/master/analysis

[48] https://matplotlib.org/

[49] https://pandas.pydata.org/

physisch vorhanden, ausgehend von der Annahme, dass nicht alle die Infrastruktur gelichzeitig benutzen werden. Dies hatte zur Folge, dass man teilweise bis zu mehreren Stunden auf das Allozieren der Ressourcen für die gestarteten Jobs warten musste. Diese Tatsache hat das Management der zugewiesenen Ressourcen erschwert.

6.7.2. Openstack Cluster

Die auf dem Openstack Cluster zur Verfügung stehenden GPUs vom Typs nVidia Tesla T4 GPU-Card und haben einen Arbeitsspeicher von 16GB GDDR6. Weitere Informationen können der Spezifikation [60] entnommen werden.

Die GPUs auf dem Openstack Cluster werden ausschliesslich einem einzelnen Benutzer zugewiesen. Somit tritt die Problematik mit den langen Wartzeiten nicht auf. Das Management der Ressourcen war entsprechend zuverlässiger als auf dem GPU Cluster.

7. Resultate Shared Task

In diesem Kapitel werden die Experimente beschrieben, welche in Zusammenhang mit dem Shared Task durchgeführt wurden und die erzielten Resultate vorgestellt. Das übergeordnete Ziel war das Ausarbeiten eines STT-Translation Systems mit einer guten Generalisierung über die schweizerdeutschen Dialekte. Der erreichte BLEU Score auf dem Testset von Shared Task «Clickworker» wird als der Indikator dieser Dialektgeneralisierung herangezogen. Ausgehend von dem Training des ersten Modells SP 0.9 wurden stufenweise Optimierungsmassnahmen und Erweiterungen eingeführt. Das Modell SP 0.9 wurde auf den Daten von Swiss Parliament mit dem besten IoU >= 0.9, was 176 Stunden an Audiodaten entspricht, trainiert. Besonderes Augenmerk ist auf die Nachvollziehbarkeit und Messbarkeit der konkreten Massnahmen gelegt worden. Diese werden in drei Gruppen unterteilt:

1. Erweiterung der Korpora (Kapitel 7.2)
2. Einbezug des ASR Encoders (Kapitel 7.3)
3. Ensemble (Kapitel 7.4)

In den Klammern steht jeweils, in welchem Kapitel der Ansatz analysiert wird. Die Übersicht über alle STT-Translation Systeme und ASR Encoders wird im kommenden Abschnitt gegeben. In Kapitel 7.5 wird aufgeführt, welche Massnahmen, die meiste Effektivität auf die Performance-Verbesserung ausübten. Anschliessend wird in Kapitel 7.6 das Endranking von Shared Task präsentiert.

7.1. Übersicht Systeme

Tabelle 14 zeigt die Übersicht über den Trainingssetup aller durchgeführten Experimente. Sie gibt die Auskunft mit welchem Korpus / Korpora die Systeme trainiert wurden. Es wird veranschaulicht, ob und welcher vortrainierter ASR Encoder für das Training einbezogen wurde. Des Weiteren wird aufgezeigt, mit welchem LM und auf welcher GPU-Umgebung trainiert wurde. Die weiterführenden Informationen zu den Korpora können in Kapitel 4 gefunden werden. Die Korpora SwissDial und ArchiMob wurden für das Training von ASR- und STT-Translation Systemen einbezogen. Kapitel 6.3.4 führt auf, welche Transkriptionen respektive Translationen aus den jeweiligen Korpora verwendet wurden. Die ASR Encoders werden ausführlich in Kapitel 7.3.1 vorgestellt. Die LMs werden in Kapitel 6.3.3 beschrieben. Die verwendete GPU-Infrastruktur ist in Kapitel 6.7 erläutert. Es werden keine Zeitangaben zu Trainingsdauer gemacht. Die Systeme wurden auf unterschiedlich vielen und unterschiedlichen GPUs trainiert. Diese zwei Faktoren verursachen, dass die Trainingszeiten der Systeme sehr differenziert ausfallen und nicht vergleichbar sind.

	Korpora					ASR Encoder				LM			Cluster	
	Swiss Parliament 0.9	Swiss Parliament All	SwissDial	ArchiMob	Common Voice DE v4	ASR EN (Librispeech)[50]	ASR DE	ASR DE (EN)	ASR CH	LM All (155)	LM NN (79)	LM N (59)	GPU Cluster	Openstack Cluster
STT-SYSTEME														
SP 0.9	✓									✓			✓	
SP All		✓								✓			✓	
SP & SD		✓	✓								✓		✓	
SP & AM		✓		✓								✓		✓
SP + ASR DE		✓					✓				✓		✓	
SP & SD + ASR DE		✓	✓				✓					✓	✓	
SP & AM + ASR DE		✓		✓			✓					✓		✓
SP & SD + ASR DE (EN)		✓	✓					✓				✓		✓
SP & SD + ASR CH		✓	✓						✓			✓	✓	
SP & SD & AM + ASR CH		✓	✓	✓					✓			✓	✓	
ASR – SYSTEME														
ASR DE					✓							✓		✓
ASR DE (EN)					✓	✓						✓		✓
ASR CH			✓	✓				✓				✓	✓	

Tabelle 14: Übersicht Trainingssetup alle Systeme

In Tabelle 15 werden die BLEU Resultate, gemessen auf den «Clickworker» und Swiss Parliament Testsets, für alle STT-Translation Systeme gegenübergestellt. Alle Systeme wurden auf dem Swiss Parliament Korpus trainiert, was sich in sehr guten BLEU Scores auf seinem Testset widerspiegelt. Im oberen Teil der Tabelle sind die einzelnen STT-Translation Systeme aufgelistet. Im unteren sind die Resultate des Ensembles zu finden, welche ausführlich in Kapitel 7.4 erläutert werden. Das Modell SP All wurde als das Baseline-Modell ausgewählt, mit dem die Resultate der anderen Systeme verglichen werden. Das Modell wurde auf allen Daten von dem Swiss Parliament Korpus trainiert, was 293 Stunden an Audiodaten entspricht. Diese Daten stellen den Baseline-Korpus dar, mit welchem alle vorgestellten Systeme, ausgenommen SP 0.9, trainiert wurden. Das Modell SP&SD&AM+ASR CH hat den besten BLEU Score von 35.33 Punkten auf dem «Clickworker» Testset erreicht und damit um 16.53 Punkte das Baseline Modell überholt[51]. Dieses Modell wurde auf allen schweizerdeutschen Korpora: Swiss Parliament, SwissDial und ArchiMob trainiert. Dies entspricht 388 Stunden Audiodaten

[50] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/docs/librispeech_example.md

[51] Ensemble Resultate werden später analysiert.

in 14 schweizerdeutschen Dialekten, siehe Tabelle 2. Zudem wurde in den Trainingsprozess ein auf Schweizerdeutsch vortrainierter ASR Encoder einbezogen. Der ASR Encoder wurde mit 95 Stunden an schweizerdeutschen Daten vortrainiert (ArchiMob und SwissDial). Die Audiodaten aus den Korpora SwissDial und ArchiMob überschneiden sich somit im Training vom ASR Encoder und dem STT-Translation-Modell.

	Swiss Parliament Test (BLEU)	**Clickworker (BLEU)**
SP 0.9	51.65	14.48 (-4.32)
SP All (Baseline)	54.79	18.80
SP & SD	54.60	23.61 (+4.81)
SP & AM	**55.92**	28.63 (+9.83)
SP + ASR DE	54.37	22.16 (+3.36)
SP & SD + ASR DE	53.24	27.17 (+8.37)
SP & AM + ASR DE	54.77	32.81 (+14.01)
SP & SD + ASR DE (EN)	55.58	33.28 (+14.48)
SP & SD + ASR CH	52.90	31.34 (+12.54)
SP & SD & AM + ASR CH	54.26	**35.33** (+16.53)
ENSEMBLE		
E1	57.76	28.74 (+ 9.94)
E2	54.68	35.80 (+17.00)
E3	**58.50**	**38.27** (+19.47)

Tabelle 15: Übersicht aller STT-Translation Systeme, Resultate in BLEU

Das beste Resultat auf dem Swiss Parliament Testset wurde vom SP&AM mit 55.92 BLEU Punkten erzielt. Dies sind jedoch nur 1.13 Punkte mehr als das Resultat vom Baseline Modell auf dem gleichen Testset. Es ist erstaunlich wenig im Vergleich zu dem Unterschied von 16.53 Punkten, welcher auf dem «Clickworker» Testset erreicht wurde. Das Phänomen kann über die Resultate aller Modelle hinweg beobachtet werden. Alle Systeme performen sehr gut auf dem Swiss Parliament Testset und erzielen im Durschnitt Resultate von 54.21 BLEU Punkten. Auf dem «Clickworker» Testset bewegen sich die Scores zwischen 14.48 (SP 0.9) und 35.33 BLEU Punkten. Dieses Verhalten kann auf die folgenden Faktoren zurückgeführt werden:

- Alle Systeme wurden mit dem Swiss Parliament Daten trainiert und entsprechend haben ähnliches oder sogar gleiches Vokabular, welches im Swiss Parliament Testset vorkommt, bereits gelernt.
- Swiss Parliament Korpus besteht hauptsächlich aus Daten in Bernerdeutsch. «Clickworker» repräsentiert jedoch die reale Dialektverteilung in der Deutschschweiz.
- Der Wortschatz des Swiss Parliament Korpus beschränkt sich auf politische Inhalte, sowie sind umgangssprachliche Ausdrücke fast nicht enthalten. «Clickworker» repräsentiert einen breiten Wortschatz, beinhaltet viele Themen und sprachliche Formen.
- Bei allen betroffenen Modellen stellt der Swiss Parliament Korpus die Mehrheit der Trainingsdaten dar, siehe Tabelle 16.

	Swiss Parliament		SwissDial		ArchiMob	
	h	%	h	%	h	%
SP & SD	293h	91.85%	26h	8.15%	0h	0%
SP & AM	293h	80.94%	0h	0.00%	69h	19.06%
SP & SD & AM	293h	75.51%	26h	6.70%	69h	17.79%

Tabelle 16: Prozentueller Anteil des Korpus zur Korpora-Kombination

Die restlichen Modelle werden in den nachfolgenden Abschnitten 7.2 bis 7.4 analysiert. Sie werden hinsichtlich der eingesetzten Massnahmen gruppiert und miteinander verglichen.

7.2. Einfluss von Korpora

In diesem Abschnitt wird aufgezeigt, inwieweit die Erweiterung der Trainingsdaten die Dialekt-Generalisierung eines STT-Translation Systems verbessern kann. In Tabelle 17 werden die BLEU Resultate für die relevanten Modelle, gemessen auf den «Clickworker» und Swiss Parliament Testsets, dargestellt. Die hier vorgestellten Modelle wurden zusätzlich zu dem Swiss Parliament Korpus mit weiteren Daten und ohne einen vortrainierten Encoder trainiert. Das Baseline Modell ist wie im vorherigen Abschnitt das SP All.

	Swiss Parliament (BLEU)	Clickworker (BLEU)
SP 0.9	51.65	14.48 (- 4.32)
SP All (Baseline)	54.79	18.80
SP & SD	54.60	23.61 (+ 4.81)
SP & AM	**55.92**	**28.63** (+ 9.83)

Tabelle 17: Übersicht STT-Translation Systeme: Erweiterung der Trainingsdaten (BLEU)

Das Modell mit der schlechtesten Performance ist SP 0.9. Das Modell wurde mit gut alignierten Daten aus dem Swiss Parliament Korpus, mit dem IoU grösser gleich als 0.9 und einen Umfang von 176 Stunden, trainiert. Auf dem «Clickworker» Testset erreicht das Modell einen Score von 14.48 BLEU Punkten. Das Resultat ist um 4.32 Punkte tiefer als dieses vom Baseline Modell. Die Trainingsdaten des Baseline Modells bestehen aus zusätzlichen 117 Stunden an Audiodaten mit schlechterem Alignement. Die genaue prozentuelle Zusammensetzung der Daten kann Tabelle 18 entnommen werden.

	SP All		SP 0.9	
	h	%	h	%
$IoU \geq 0.9$	176h	60.06%	176h	100%
$0.9 > IoU \geq 0.7$	80h	27.30%	0h	0%
$IoU \leq 0.7$	37h	12.64%	0h	0%

Tabelle 18: Prozentueller Anteil an Daten mit entsprechenden IoU im Swiss Parliament Trainset All vs 0.9

Daraus lässt sich schliessen, dass die Menge der Daten einen grösseren Einfluss auf die Modell-Performance hat als kleinere Menge an Daten mit guter Qualität. Motiviert durch diese

Erkenntnis wurden zwei weitere Experimente SP&SD und SP&AM durchgeführt. Das Modell SP&SD wurde mit allen Daten aus dem Swiss Parliament und aus dem SwissDial Korpus trainiert. Dabei besteht der SwissDial Korpus aus nur 26 Stunden an Daten. Die Sätze in diesem Korpus werden in 8 Dialekten wiederholt, sodass effektiv nur ca. 3.5 Stunden an neuen Inhalten dazu kommen. Die Bereicherung um die neuen Dialekte ist hingegen bemerkbar. Zudem weisen die Daten eine sehr gute Qualität auf und trotz wenigen Audiodaten doch eine grosse Erweiterung des Wortschatzes, siehe Kapitel 4.3. Dieses Experiment verbessert den Baseline Score um 4.81 BLEU Punkte, womit ein Ergebnis von 23.61 Punkten auf dem «Clickworker» Testset erreicht wurde.

Das Modell SP&AM wurde mit allen Daten aus dem Swiss Parliament und aus dem ArchiMob Korpus trainiert. Der ArchiMob Korpus bringt 13 neue Dialekte und 69 Stunden neuer Inhalte in die Trainingsdaten ein. Dabei eignet sich der ArchiMob Korpus primär für das Training von den ASR Systemen. Die Translationen im ArchiMob sind eine normalisierte Entsprechung eines schweizerdeutschen Ausdrucks, welcher nicht immer einer hochdeutschen Übersetzung entspricht. Zudem wurden diese auf der Wortebene, nicht Satzebene erstellt, was teilweise zu falschen Satzstrukturen führt. Diese hier aufgeführten Probleme, welche auf nicht optimale Datenqualität hinweisen, haben jedoch keinen negativen Einfluss auf die Modell-Performance ausgeübt. Im Gegenteil, das SP&AM Modell hat mit 28.63 BLEU Punkten den Score vom Baseline Modell um 9.83 Punkte auf dem «Clickworker» Testset verbessert.

Das SP&SD erreicht mit 55.92 BLEU Punkten auch das beste Resultat auf dem Swiss Parliament Testset. Es ist um 4.27 Punkte besser als das Resultat des SP 0.9 Modells.

Anschliessend wurden die Translationen, welche von den Modellen SP All, SP&SD und SP&AM generiert wurden, verglichen. Nachfolgend werden 3 Beispiele aufgeführt:

Beispiel 1

Audiodatei:	1eee2128-ccf2-46d4-abb6-19d4bb4b2ed3.flac
Manuelle Übersetzung:	*Die Mumie wird in das britische Museum gebracht*
SP All Vorhersage:	*momen wird dieses prinzip der motion braucht*
SP & SD Vorhersage:	*diese maul wird dieses britische musik gebaut*
SP & AM Vorhersage:	*diese maum haben jeweils displitisch mussion gemacht*

Beispiel 2

Audiodatei:	031eae13-4fce-4174-8ed2-9f82b5ab1490.flac
Manuelle Übersetzung:	*Karabiner Mode*
SP All Vorhersage:	*kommissionspräsident der fiko*
SP & SD Vorhersage:	*ich halte keinen bienenmode*
SP & AM Vorhersage:	*karabiner morgen*

Beispiel 3

Audiodatei:	d7f161f7-9ab9-43fa-8f9a-0d040a0c4bf2.flac
Manuelle Übersetzung:	*Wenig später stirbt sie*
SP All Vorhersage:	*das wenigste der steuer zu sein*
SP & SD Vorhersage:	*wenig städter steht sie*
SP & AM Vorhersage:	*wenigstens später steuert sind*

Beim Betrachten der Sätze fällt es auf, dass das SP All Modell sehr stark auf den Wortschatz aus dem Swiss Parliament Korpus optimiert wurde und schlecht auf die neuen Inhalte generalisiert. Es tendiert dazu die nicht verstanden Inhalte auf das ihm bekannte «politische» Vokabular zu projizieren. In den Translationen der Modelle SP&SD und SP&AM wurde das Phänomen abgeschwächt. Die Predicitons sind zwar immer noch verbesserungsfähig, sie ähneln jedoch zumindest im phonetischen Sinne mehr dem, was in der Audiodatei gesagt wurde.

Anschliessend zeigen die in diesem Abschnitt durchgeführten Experimente, dass hinsichtlich der Optimierung der Modell-Performance die Erweiterung der Trainingsdaten um Daten mit schlechter Qualität einen positiven Einfluss hat. Dies ist vor allem beim Vergleich der Modelle SP 0.9 und SP All ersichtlich, als auch bei den Ergebnissen erzielten vom Modell SP&AM. Das Modell SP&AM erreicht bessere Resultate als das SP&SD, welches mit deutlich besseren qualitativ Daten, jedoch in kleineren Mengen, trainiert wurde.

7.3. Einfluss des ASR Encoder

In diesem Abschnitt wird analysiert, inwieweit der Einbezug von einem vortrainierten ASR Encoder in den Trainingsprozess die Dialekt-Generalisierung eines STT-Translation Systems optimieren kann. Als erstes wird eine Übersicht über die vortrainierten ASR Encoder gegeben. Des Weiteren werden die auf dem «Clickworker» Testset erzielten Resultate der relevanten Modelle verglichen. Anschliessend wird der Einfluss der Korpora Erweiterung dem vom Einbezug eines ASR Encoders gegenübergestellt.

7.3.1. Übersicht ASR Encoders

Wie im Kapitel 6.4.1 erläutert gibt es zwei Ansätze, wie das Acoustic Modell trainiert werden kann. In diesem Abschnitt wird der zweite Ansatz, das Vortrainieren eines ASR Encoders, behandelt. Die ASR Encoders wurden mit der gleichen Architektur trainiert, wie die STT-Translation Systeme. Das Vorgehen folgt dem gleichen Prozess, welcher in Kapitel 6 beschrieben wird. Durch den Einbezug eines vortrainierten ASR Encoders kann sich das Modell während des Trainings stärker auf das Lernen vom Decoding konzentrieren und auf dem Wissen der vortrainierten ASR Komponente beim Encoding basieren. Es wurden insgesamt

drei ASR Systeme trainiert, zwei für Hochdeutsch und eins für Schweizerdeutsch. Die erzielten Resultate werden in Tabelle 19 übersichtlich dargestellt. Die Scores sind ausnahmsweise in WER Metrik angegeben, welche übelicherweise für die Auswertung der ASR Systeme eingesetzt wird.

	Common Voice DE v4 (WER)	SwissDial (WER)	ArchiMob (WER)
ASR DE	24.53		
ASR DE (EN ASR)	**21.67**		
ASR DE (CoVoST) [6]	21.40		
ASR CH		5.20	**30.19**
ASR CH (ArchiMob) [43]			42.38

Tabelle 19: Übersicht ASR Encoders, Resultate in WER

Der ASR DE Encoder wurde mit den hochdeutschen Daten aus dem Common Voice Korpus v4 trainiert und erreicht einen Score von 24.53% WER auf dem Testset des gleichen Korpus. Für das Training vom ASR DE (EN ASR) Encoder wurde ein auf englischen Daten, welche aus dem Librispeech Korpus [58] stammen, vortrainierter ASR Encoder einbezogen. Das Modell ASR EN (Librispeech) wurde vom FAIRSEQ S2T Repository[52] heruntergeladen. Das Verwenden eines vortrainierten Encoders im Training hilft das Overfitting abzuschwächen, wie dies in Kapitel 6.4.1 erläutert wird. Mit diesem Ansatz wurde ein WER Score von 21.67% auf dem Common Voice DE v4 Testset erreicht, womit eine Verbesserung von 2.86% erzielt wurde. Wang et al. [6] erreichten einen Score von 21.40% WER mit dem gleichen Trainingssetup, welches mit einer Differenz von 0.27% sehr nah dem Resultat des ASR DE (EN ASR) kommt. Zuletzt wurde ein ASR Encoder für Schweizerdeutsch auf den Daten aus den SwissDial und ArchiMob Korpora trainiert. Für das Training wurde der ASR DE (EN ASR) einbezogen. Mit diesem Setup konnte ein WER Score von 30.19% auf dem Testset von ArchiMob erzielt werden. Dies bringt eine Verbesserung von 12.19% gegenüber dem ASR System von Iuliia Nigmatulina [43], welches sie im Rahmen Ihrer Masterarbeit im Jahr 2020 trainiert hat.

7.3.2. Resultate Clickworker

In diesem Abschnitt wird der Einfluss der vortrainierten ASR Encodern verglichen. Das SP&SD ist für diesen Vergleich als Baseline Modell ausgewählt worden. Dazu wurden drei Modelle auf den vollständigen Daten aus den Korpora Swiss Parliament und SwissDial, dito das Baseline Modell SP&SD, trainiert. Es wurde jedoch jeweils ein anderer vortrainierter ASR Encoder in den Trainingsprozess einbezogen. Beim Modell SP&SD+ASR DE ist jenes der vortrainierte Hochdeutsche Encoder ASR DE, beim SP&SD+ASR CH der Schweizerdeutsche und beim SP&SD+ASR DE(EN) der um Librispeech Daten erweiterte Hochdeutsche Encoder.

[52] https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/docs/librispeech_example.md

Tabelle 20 zeigt eine Übersicht, welche BLEU Resultate die aufgezählten Modelle erreicht haben.

	Swiss Parliament Test (BLEU)	Clickworker (BLEU)
SP & SD (Baseline)	54.60	23.61
SP & SD + ASR DE	53.24	27.17 (+ 3.56)
SP & SD + ASR DE (EN)	**55.58**	**33.28** (+ 9.67)
SP & SD + ASR CH	52.90	31.34 (+ 7.73)

Tabelle 20: Übersicht Baseline SP&SD vs. Einbezug von ASR Encoders (BLEU)

Wie erwartet, erreichen alle Modelle einen besseren BLEU Score als das Baseline Modell SP&SD. Der Einbezug vom hochdeutschen Encoder im Modell SP&SD+ASR DE hat ein Resultat von 27.17 BLEU Punkte auf dem «Clickworker» Testset erzielt und damit den Score von der Baseline um 3.56 Punkte verbessert.

Die Modell-Performance vom SP&SD+ASR DE(EN) verbessert diese von der Baseline sogar um 9.67 BLEU Punkte, was sich in einem Score von 33.28 BLEU Punkten auf dem «Clickworker» Testset ausdrückt. Vergleicht man die Systeme SP&SD+ASR DE und SP&SD+ASR DE(EN) ist das letzte um 6.11 BLEU Punkte besser. Es ist ein grosser Sprung, wenn man die beiden ASR Encoder vergleicht, durch welche sich die Modelle differenzieren. Das ASR DE(EN) hat einen um 2.86% besseren WER Score als das ASR DE auf dem Testset von Common Voice DE erreicht. Dies kann vereinfacht folgend interpretiert werden, dass die 2.86% WER Punkte eine Verbesserung um 6.11 BLEU Punkten bewirkten. Des Weiteren ist der ASR DE(EN) Encoder durch den Einbezug des vortrainierten englischen Encoders robuster als der ASR DE. Dies bestätigt die Aussage von Wang et al. in [6], dass diese Massnahme das Overfitting eines Modells reduziert. Dieses Modell erreicht auch das beste Resultat von 55.58 BLEU Punkten auf dem Swiss Parliament Testset, was eine Verbesserung der Baseline um 0.98 BLEU Punkte bedeutet.

Zuletzt hat das Modell SP&SD+ASR CH einen Score von 31.34 BLEU Punkten auf dem «Clickworker» Testset erreicht, was einer Verbesserung von 7.73 Punkten gegenüber der Baseline entspricht. Es ist überraschend, dass der Score um knappe zwei Punkte schlechter als das Ergebnis des SP&SD+ASR DE(EN) geworden ist. Eine mögliche Erklärung kann hierfür sein, dass sich die Daten aus dem SwissDial Korpus im Training vom ASR CH Encoder und dem SP&SD+ASR CH überschnitten haben. Als Konsequenz hat der Encoder hinsichtlich der akustischen Datenbereicherung nur aus dem ArchiMob Korpus profitiert. Hierzu würde sich lohnen einen ASR CH Encoder mit anderen Daten zu trainieren als diesen, auf welchen das STT-System trainiert wird. Des Weiteren wäre es sinnvoll, in das Training von ASR CH Encoder anstatt des ASR DE (EN) den auf englischen Librispeech Daten vortrainierten Encoder einzubeziehen.

In Tabelle 21 wurden die Resultate der Modelle SP&AM und SP&AM+ASR DE gegenübergestellt. Wie erwartet hat das Modell SP&AM+ASR DE eine bessere Performance als das SP&AM und erreicht einen Score von 32.81 BLEU Punkten auf dem «Clickworker» Testset.

Gegenüber den vom SP&AM erzielten 28.63 Punkten ist dies eine Verbesserung um 4.18 BLEU Punkte. Diese Differenz entspricht ca. der, welche zwischen den Scores von den oben beschrieben Modellen SP&SD und SP&SD+ASR DE entstanden ist.

	Swiss Parliament Test (BLEU)	Clickworker (BLEU)
SP & AM	**55.92**	28.63
SP & AM + ASR DE	54.77	**32.81**

Tabelle 21: Übersicht Baseline SP&AM vs. Einbezug von ASR DE Encoder (BLEU)

In diesem Abschnitt wurde nachgewiesen, dass die vortrainierten ASR Encoders einen Vorteil für die Modell-Performance bringen und somit dazu beitragen die Dialektgeneralisierung zu verbessern. Überraschenderweise hat der ASR DE (EN) Encoder am meisten zur Verbesserung des Scores beigetragen. Diese Erkenntnis kann somit Inspiration für neue Ideen bei späteren Arbeiten sein. Dieser Ansatz kann vor allem bei der Optimierung des ASR CH Encoders weiterverfolgt werden.

7.3.3. Korpora Erweiterung vs. Einbezug von ASR Encoder

In Tabelle 22 werden die Modelle SP&SD und SP+ASR DE mit deren BLEU Resultaten dargestellt und mit dem Baseline Modell SP All verglichen. Die Erweiterung des Trainingsdaten um den SwissDial Korpus bringt eine Verbesserung von 4.81 BLEU Punkten. Demgegenüber hat der Einbezug von einem auf Hochdeutsch vortrainierten Encoder ASR DE den Score um 3.36 BLEU Punkte verbessert. Aus den zwei Beispielen lässt sich schliessen, dass die Erweiterung der Daten einen grösseren Einfluss auf die Verbesserung der Modell-Performance ausübt als der Einbezug eines vortrainierten Encoders. Die Resultate sind jedoch abhängig vom verwendeten ASR Encoder und der Grösse der erweiterten Daten, sodass keine allgemeine Aussage getroffen werden kann. Nach den Erkenntnissen aus dem vorherigen Abschnitt ist nicht auszuschliessen, dass der Einbezug vom ASR CH oder ASR DE (EN) Encoder in den Trainingsprozess vom Baseline Modell SP ALL besser performen würde als das SP&SD.

	Swiss Parliament Test (BLEU)	Clickworker (BLEU)
SP All (Baseline)	54.79	18.80
SP & SD	54.60	**23.61** (+ 4.81)
SP + ASR DE	54.37	22.16 (+ 3.36)

Tabelle 22: Übersicht STT-Translation Systeme Erweiterung Trainingsdaten vs. Einbezug von ASR Encoder (BLEU)

7.4. Ensemble

In diesem Kapitel wird untersucht, inwieweit ein Ensemble aus mehreren Modellen die BLEU Ergebnisse und damit die Dialektgeneralisierung verbessern kann. Abbildung 17 zeigt schematisch das Prinzip von einem Ensemble. Das Ziel von diesem Prozess ist das Wissen von mehreren Modellen zusammenzubringen und bei der Generierung der Translationen dadurch

die Präzision der Vorhersagen zu optimieren. In dieser Arbeit wird für das Ensemble die Implementierung von FAIRSEQ S2T verwendet[53], siehe Fusszeile.

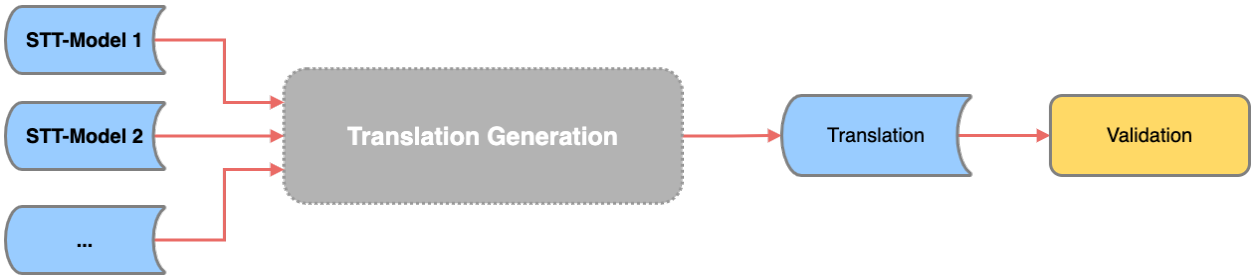

Abbildung 17: Piktogramm Ensemble

Tabelle 23 zeigt eine Übersicht der BLUE Resultate, welche mit dem Ensemble erzielt wurden. Für eine bessere Nachvollziehbarkeit der Fortschritte werden links die für das Ensemble verwendeten Modelle als auch deren Resultate dargestellt. Bei der Auswahl der Modelle für das Ensemble wurde darauf geachtet, dass diese auf unterschiedlichen Setups (Daten und Encoder) trainiert wurden, sodass man von dieser Diversität bei der Generierung der Translationen profitieren kann.

		Modell BLEU		Ensemble Click-worker	Ensemble Swiss Parliament
E1	SP & SD	23.61	27.17 (- 1.57) (SP&SD+ASR DE)	28.74	57.76
	SP + ASR DE	22.16			
E2	SP & SD + ASR CH	31.34	35.33 (- 0.47) (SP&SD&AM+ASR CH)	35.80	54.68
	SP & AM + ASR DE	32.81			
E3	SP & SD + ASR CH	31.34		**38.27**	**58.50**
	SP & AM + ASR DE	32.81			
	SP & AM & SD + ASR CH	35.33			

Tabelle 23: Übersicht Ensemble vs. Einzelmodelle, Resultate in BLEU

Für das erste Ensemble E1 wurden die Modelle SP&SD und SP+ASR DE eingesetzt. Das erste Modell ergänzte das Baseline Modell SP All um weiter Trainingsdaten aus dem Swiss-Dial Korpus. Dem zweiten Modell SP+ASR DE wurden für das Training keine weiteren Daten als Swiss Parliament All hinzugefügt, stattdessen wurde der ASR DE Encoder in den Trainingsprozess miteinbezogen. Mit dieser Konstellation konnte ein Score von 28.74 BLEU erreicht werden. Das Ergebnis verbessert um 1.57 BLEU Punkte die Performance von SP&SD+ASR DE System, welches mit dem Ansatz von den beiden Modellen trainiert wurde. Demzufolge wurde in diesem Beispiel durch das Ensemble ein besseres Ergebnis erzielt als durch das Training eines neuen Systems, welches die Ansätze mehrerer Modelle fusioniert.

[53] https://github.com/pytorch/fairseq/issues/223

Motiviert durch den Erfolg vom ersten Ensemble wurde ein zweites Ensemble E2 durchgeführt. Dieses Mal wurden die Modelle SP&SD+ASR CH und SP&AM+ASR DE eingesetzt. Gleich wie beim ersten Versuch wurden möglichst unterschiedliche Modelle ausgewählt. Das erste wurde mit zusätzlichen Daten aus dem SwissDial Korpus und mit einem auf Schweizerdeutsch vortrainierten Encoder ASR CH trainiert. Das zweite verwendete den ASR DE Encoder und für die Erweiterung der Trainingsdaten den ArchiMob Korpus. Diese Zusammensetzung hat ein Resultat von 35.80 BLEU Punkten erreicht. Vergleichend mit dem Modell SP&SD&AM+ASR CH, welches die Ansätze aus den beiden Modellen in einem Training kombiniert, ist das Ergebnis mit der Differenz von 0.47 BLEU Punkten leicht besser. Damit ist es das zweite Beispiel, in dem nachgewiesen werden kann, dass das Ensemble ein besseres Ergebnis erzielt als das Training eines neuen Systems.

Zuletzt wurde ein drittes Ensemble E3 durchgeführt. Bei diesem Experiment handelt es sich um eine Erweiterung des E2 um das beste Modell SP&SD&AM+ASR CH. Das E3 erreicht ein Ergebnis von 38.27 BLEU Punkten auf dem «Clickworker» Testset. Es ist auch der beste Score in dieser Arbeit. Auch auf dem Swiss Parliament Testset hat das E3 die beste Performance. Mit dem Resultat von 58.50 BLEU Punkten hat es die Baseline (SP All) um 3.71 Punkte verbessert.

7.5. Fazit

Nach dem Durchführen der zahlreichen Experimente, welche in Kapiteln 7.2 bis 7.4 vorgeführt wurden, konnten interessante Schlussfolgerungen gezogen und Erkenntnisse hinsichtlich der Optimierung der Modelle gewonnen werden. Diese drei Massnahmen erwiesen sich als die effektivsten und haben den grössten Beitrag zur Verbesserung der Dialektgeneralisierung geleistet:

1. Menge über die Qualität
2. Training auf einem robustem ASR Encoder
3. Ensemble über das Neutraining

Es konnte nachgewiesen werden, dass eine Erweiterung der Trainingsdaten mit Daten schlechter Qualität, die Modell-Performance steigern kann. Das Verwenden eines vortrainierten Encoders hat ebenso einen positiven Einfluss auf die Modell-Performance. Des Weiteren verbessert das Trainieren eines ASR Encoders auf einem soliden ASR Encoder in einer Fremdsprache die Robustheit des späteren STT-Translation Systems. Anschliessend können mit einem Ensemble bessere Resultate erzielt werden als mit einem neuen Training.

7.6. Auswertung Shared Task

Abschliessend werden in diesem Abschnitt die Endresultate des Shared Tasks bekannt gegeben und kommentiert. Tabelle 24 zeigt eine Übersicht über die offiziellen Resultate. Sie beinhaltet den erreichten Rang, den Systemnamen (falls bekannt sonst den Teilnehmerna-

men) und die Scores, welche auf dem Public- respektive Private-Teil des «Clickworker» Testset erreicht wurden. Der Public Score wurde nach dem Hochladen jeder Submission aktualisiert. Der Private Score wurde erst nach der Deadline veröffentlicht und ist für die Rangierung relevant.

Rang	**Modell / Teilnehmer**	**Public Score**	**Private Score**
1 (1)	Oscar Koller	46.04	46.00
	Baseline	41.11	40.99
2 (2)	**Majority Voting (ZHAW Beitrag)**	**38.70**	**39.39**
3 (2)	FAIRSEQ E3 (ZHAW)	36.83 (38.27)	37.40 (39.48)
4 (2)	Jasper (ZHAW)	32.97	33.98
5 (2)	wav2vec (ZHAW)	30.39	31.28
6 (2)	Post-Processing (ZHAW)	28.74	29.49
7 (3)	DeJa	16.99	17.12

Tabelle 24: Offizielle Resultate Shared Task

Die Beiträge, welche von dem ZHAW-Shared-Task-Team (siehe Kapitel 3.2) eingereicht wurden, sind mit «(ZHAW)» markiert. Zählt man diese zu einem «ZHAW»-Teilnehmer zusammen, wurden insgesamt 3 Beiträge eingereicht, somit hat ZHAW mit dem Majority Voting den zweiten Rang von drei erreicht. Majority Voting wurde aus den Modellen FAIRSEQ E2, Jasper und wav2vec (in der Tabelle grün hinterlegt) generiert, dabei war der Output von FAIRSEQ E2 der Primäre und die von Jasper und wav2vec die Hilfs-Outputs. Der Output aus dem FAIRSEQ E3 wurde aus den zeitlichen Gründen nicht mehr in Majority Voting berücksichtigt. Der Zwischenstand des E3 wurde kurz vor der Deadline eingereicht und erreichte einen Score von 36.83 BLEU Punkten auf dem Public-Teil des «Clickworker» Testsets und 37.40 auf dem Private-Teil. Es ist auch der offizielle Score, mit dem das FAIRSEQ E3 den dritten Rang von 7 einnimmt, wenn man alle ZHAW-Submissions als autonome Teilnehmer berücksichtigt. Nach dem das Training von SP&SD&AM+ASR CH abgeschlossen wurde, wurde erneut ein E3-Ensemle durchgeführt und anschliessend von den Shared-Task-Organisatoren ausgewertet. Die Einträge in Klammern entsprechen den erzielten Resultaten. Der Score hat sich auf dem Public-Teil um 1.44 BLEU Punkte und auf dem Private-Teil um 2.08 verbessert. Somit hat das E3 mit dem Score von 39.48 BLEU Punkten sogar ein besseres Resultat als das Majority Voting erzielt. Da dieses Resultat nach der Submission-Deadline eingereicht wurde, ist es kein Teil der offiziellen Auswertung.

8. Weitere Resultate

In diesem Kapitel wird ein weiteres Experiment beschrieben, welches unabhängig von Shared Task stattgefunden hat. Des Weiteren wird die Entwicklung der BLEU Resultate während dem Training von den Modellen SP&SD+DE und SP&SD+CH analysiert und verglichen.

8.1. Analyse der Dialekte

In diesem Abschnitt wird ein Experiment durchgeführt, welches auf den Daten aus dem Swiss-Dial Korpus basiert. Das Ziel ist herauszufinden, wie gut das SP All Modell auf einen Dialekt, welcher nicht in den Trainingsdaten des Modells beinhaltet ist, generalisieren kann. Der SwissDial Korpus eignet sich hervorragend für das Durchführen dieses Experimentes. Er besteht aus Sätzen, welche in acht unterschiedlichen Dialekten wiederholt werden.

Als erstes wird das Setup für dieses Experiment erläutert. Anschliessend werden die Resultate aus drei verschiedenen Blickwinkeln betrachtet, welche in den Abschnitten 8.1.1 bis 8.1.3 aufgeführt werden.

SETUP

Der SwissDial Korpus wurde auf dem «Dialect Level» in Train- und Testset aufgeteilt. Dieser Vorgang wurde 8-mal pro Dialekt wiederholt, sodass als Resultat 8 verschiedene Trainingskollektionen entstanden sind, welche sich wie folgt zusammensetzen:

- Testset Vollständige Daten aus einem Dialekt
- Trainset Daten aus den restlichen 7 Dialekten

Mit diesen acht Ausprägungen des SwissDial Korpus wurde acht Mal ein Fine Tuning auf dem Baseline Modell SP All durchgeführt. Anschliessend wurden diese acht Systeme auf dem ihm jeweils unbekanntem Dialekt ausgewertet.

8.1.1. Fine Tuning SP All – Dialekt-Generalisierung

Tabelle 25 zeigt die Resultate des oben beschriebenen Fine Tunings auf dem SP All Modell. Dabei gibt das Kanton-Kürzel Auskunft, aus welchem Dialekt das Testset besteht. Im Fall «AG» bedeutet es, dass das SP All Modell auf allen Dialekten bis auf den Aargauerdeutsch fine-tunet wurde und anschliessend auf dem Aargauerdeutsch ausgewertet wurde. Das Ergebnis sagt aus, wie gut das Modell den aargauerdeutschen Dialekt verstehen kann, ohne den Dialekt vorhin gelernt zu haben.

	AG	BE	BS	GR	LU	SG	VS	ZH[54]
SP All Fine Tuning SwissDial Dialekte (BLEU)	92.89	95.53	95.22	97.36	95.22	**97.53**	**89.09**	95.05

Tabelle 25: Dialekt Generalisierung Fine Tuning von SP All auf dem «Dialekt Level» Split

Die Resultate sind sehr gut geworden und der BLEU Score liegt im Durchschnitt bei 94.74 Punkten. Dies entspricht beinah einem perfekten Alignement. Am schlechtesten kann auf das Walliserdeutsch generalisiert werden. Das SP All VS erreicht einen Score von 89.09 BLEU Punkten. Das Resultat für SP All AG für Aargauerdeutsch von 92.89 BLEU Punkten liegt auch unter dem Durchschnitt. Die bestverstandenen Dialekte sind St.Galler-Deutsch und Bündner-Deutsch mit 97.53 respektive 97.36 BLEU Punkten. Bernerdeutsch nimmt nur den Platz 3 mit dem Score von 95.53 BLEU Punkten ein. Dieses Resultat überrascht, weil das Baseline Modell SP All hauptsächlich auf den Bernerdeutsch Daten trainiert wurde. Hier muss erwähnt werden, dass es im SwissDial Korpus nur einen Sprecher pro Dialekt gibt. Das heisst, dass die oben aufgeführten Resultate neben dem Dialekt sich stark auf dem jeweiligen Sprecher beziehen. Abgesehen davon ist die Hauptaussage des Experiments, dass das Lernen neuer Dialekte keine Herausforderung für das Modell darstellt. Somit konnte bewiesen werden, dass das Modell eine hervorragende Dialekt-Generalisierung aufweist.

Das grösste Problem ist für das Modell das Erlernen vom neuen unbekannten Wortschatz, was im kommenden Experiment in Abschnitt 8.1.2 aufgezeigt wird.

8.1.2. Analyse ZH Dialekt

In diesem Abschnitt wird das ZH Split genauer untersucht, da bei der Auswertung im vorherigen Abschnitt auf starke Resultat-Abweichung gestossen wurde. Das Ergebnis erreichte nach der ersten Auswertung von SP All ZH einen Score von 64.42 BLEU Punkten, was ca. 30 Punkte unter dem oben berechneten Durchschnitt von 94.47 BLEU Punkten liegt. Zum Nachvollziehen dieses Resultats wurde das ZH Testset mit den anderen Sets verglichen. Dabei wurde festgestellt, dass das ZH Testset im Vergleich zu anderen Dialekten 1.5 mal so viele Audiodaten, welche 4.55 Stunden entsprechen, beinhaltet. Die anderen Sprachen enthalten im Durchschnitt ca. 3.3 Stunden. Tabelle 26 zeigt eine Übersicht, wie die 26 Stunden Audiodaten aus dem SwissDial Korpus auf die 8 Dialekte verteilt sind.

	AG	BE	BS	GR	LU	SG	VS	ZH
SwissDial Total Audio (h)	2.78	3.41	3.15	2.93	2.54	3.71	3.32	**4.55**

Tabelle 26: Anzahl Stunden pro Dialekt im SwissDial Korpus

[54] Auf «ZH Intersection Split» ausgewertete, siehe Kapitel 8.1.2

Aus dieser Erkenntnis lässt sich schliessen, dass es auf Zürcher-Deutsch viel mehr Sätze vorgelesen werden. Dementsprechend beinhaltet dieses Testset einen viel reicheren Wortschatz als die anderen Dialekt-Sets. Um die Ergebnisse im vorherigen Abschnitt zu konsolidieren, wurde dieses Set in zwei weitere, wie folgt unterteilt:

- ZH Intersection
- ZH Difference

«ZH Intersection» ist die gemeinsame Menge aller Sätze, welche in den Testset VS (Walliserdeutsch) und ZH (Zürcher-Deutsch) vorkommen. Somit wurde sichergestellt, dass die in «ZH Intersection» vorkommenden Sätze einen Wortschatz enthalten, welcher dem Modell bekannt ist. «ZH Difference» beinhaltet die restlichen Sätze und somit mit grosser Wahrscheinlichkeit einen für das Modell unbekannten Wortschatz. Anschliessend wurde das SP All ZH Modell auf den beiden Testsets ausgewertet. Auf dem Testset «ZH Intersection» erreicht das Modell einen Score von 95.05, was ungefähr der Durchschnitt-Performance aller Dialekten entspricht. Auf dem Testset «ZH Difference» erzielt das Modell ein Ergebnis von ernüchternden 4.01 BLEU Punkten. Aus diesem Resultat kann die Schlussfolgerung gezogen werden, dass das Modell massive Probleme mit unbekanntem Wortschatz hat. In einem weiteren Schritt wurden diese zwei Testsets für ZH und ein «ZH All» Testset, welches die gesamte 4.55 Stunden enthält, auf den Modellen SP All (ohne Fine Tuning), SP&AM und SP&AM DE ausgewertet. Alle diese Modelle wurde nicht auf den SwissDial Daten trainiert. Die Resultate sind in Figure 1 dargestellt. Alle drei Modelle erreichen ein ähnliches Resultat auf den drei Testsets. Die starken Schwankungen, welche beim Modell SP All ZH zu beobachten sind, treten nicht auf. Die Resultate auf dem Testset «ZH Difference» liegen bei den Modellen SP All ZH und SP All sehr nah, bei 4.01 und 5.16 BLEU Punkten. Das Modell SP&AM erreicht im Durchschnitt ein Resultat auf den drei ZH Testsets von 8.23 BLEU Punkten, der Durchschnitt vom Modell SP&AM DE liegt bei 10.48 BLEU Punkten. Womit ein leichter Anstieg des BLEU Scores beobachtet werden kann.

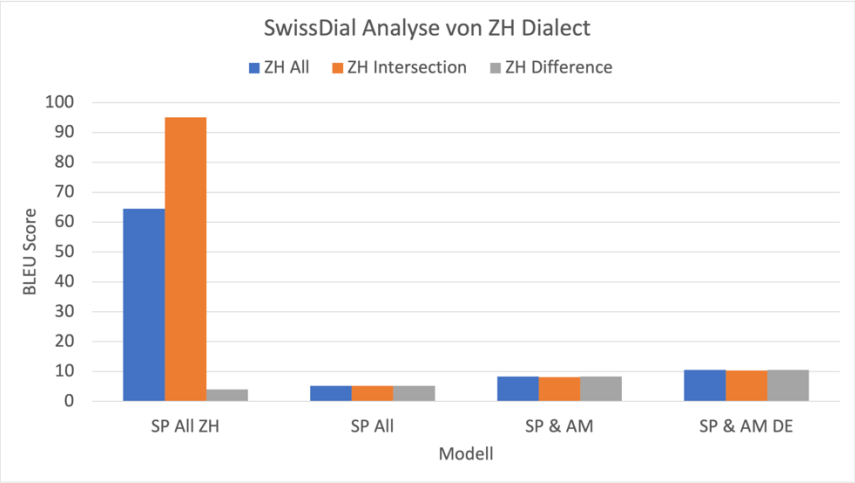

Figure 1: Analyse von ZH Dialect Splits auf verschiedenen Modellen

Anschliessend wird im nächsten Abschnitt die Dialekt-Generalisierung der drei Modelle weiteruntersucht.

8.1.3. Analyse Dialekt Generalisierung

In diesem Abschnitt wird die Dialekt-Generalisierung der Modelle SP All (ohne Fine Tuning), SP&AM und SP&AM DE untersucht, indem die Testsets der 8 Dialekte auf den drei Modellen ausgewertet werden. Für den ZH Dialekt wird der «ZH All» Split verwendet. Figure 2 zeigt die erzielten Resultate.

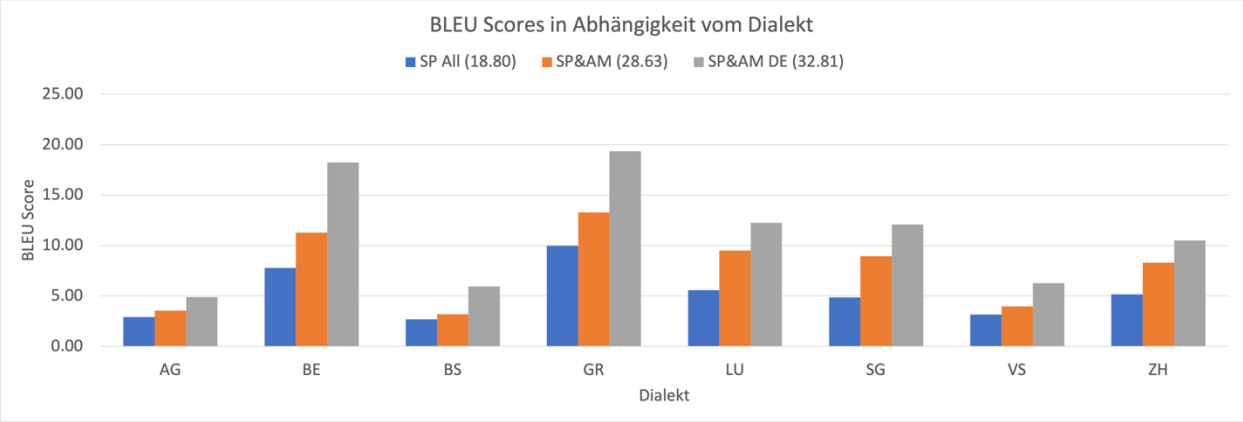

Figure 2: Model Performance in Abhängigkeit von Dialekt

Das Ziel des Experiments ist die folgenden zwei Fragestellungen zu beantworten:

1. Kann die Auswirkung der in Kapitel 7.2 bis 7.4 eingeführten Optimierungsmassnahmen auch auf diesen Testsets bemerkt werden?
2. Stimmen die Ergebnisse dieser Resultate hinsichtlich des Verstehens der Dialekte mit den Resultaten aus Kapitel 8.1.1 überein?

Bezüglich der ersten Fragestellung lautet die Antwort: Ja. In Figure 2 sind in der Legende jeweils in Klammern neben dem Modellnamen seine Resultate auf dem «Clickworker» Testset ersichtlich. Man kann dem Diagramm entnehmen, dass die Resultate auf den Dialekt-Testsets analog den Resultaten auf dem «Clickworker» Testset ansteigen. Dieses Verhalten kann bei allen Dialekten beobachtet werden. Das Baseline Modell SP All erzielt die schlechtesten Resultate, welche im Durchschnitt bei 5.27 BLEU Punkten liegen. Die Erweiterung der Baseline um die Trainingsdaten aus dem ArchiMob Korpus bringt eine leichte Verbesserung. Der Durchschnitt Score liegt bei 7.75 BLEU Punkten. Anschliessend erzielt neben der Datenerweiterung der zusätzliche Einbezug vom ASR Encoder im SP&AM DE einen durchschnittlichen Score von 11.19 BLEU Punkten.

Zu der zweiten Fragestellung. Es können Analogien zwischen den Resultaten in Figure 2 und Tabelle 25 gefunden werden. Sowohl die fine-tunet SP All Modelle als auch die in diesem Abschnitt untersuchten Modell haben die meisten Schwierigkeiten den Aargauer- und Walliserdeutsch zu «verstehen». Alle Modelle können hingegen sehr gut Berner- und Bündner-

Deutsch übersetzen. Diskrepanzen sind bei den Baseler- und St.Galler-Deutsch zu beobachten. Die fine-tunet SP All Modelle können überdurchschnittlich gut St.Galler-Deutsch übersetzen, die Modelle in Figure 2 performen auf diesem Dialekt durchschnittlich. Demgegenüber haben die Modelle aus Figure 2 Schwierigkeiten mit dem Basler-Deutsch. Bei den fine-tunet Modellen treten die Probleme mit Basler-Dialekt nicht auf.

8.1.4. Fazit

In den oberen Kapiteln kann nachgewiesen werden, dass das Baseline Modell eine sehr gute Fähigkeit der Dialektgeneralisierung besitzt. Hingegen stellt unbekannter Wortschatz eine grosse Herausforderung für das Modell dar, was die Analyse vom ZH Testset aufzeigt. Des Weiteren kann bestätigt werden, dass die Optimierungsmassnahmen aus dem Kapitel 7 eine Verbesserung bewirken. Anschliessen wird festgestellt, dass der schwierigste Dialekt Walliserdeutsch und der bestverstandene Bündner-Deutsch ist.

8.2. Analyse BLEU Score Kurve

In diesem Abschnitt wird die Entwicklung der BLEU Resultate während dem Training von den Modellen SP&SD+ASR DE und SP&SD+ASR DE für die ersten 1000 Iterationen verglichen und analysiert. In Figure 3 ist die BLEU Score Kurve für die Korpora Swiss Parliament und SwissDial von dem Modell SP&SD+ASR DE abgebildet. Es ist zu sehen, dass in den ersten Iterationen der SwissDial Korpus schlechtere Ergebnisse als der Swiss Parliament hat. Ab der Iteration 350 wird der BLEU Score vom SwissDial besser und ab dann steigt kontinuierlich bis zum Erreichen von ca. 75 BLEU Punkten. Demgegenüber ist der Verlauf der Swiss Parliament Kurve viel flacher und ab der Iteration 600 verbessert sich kaum merklich.

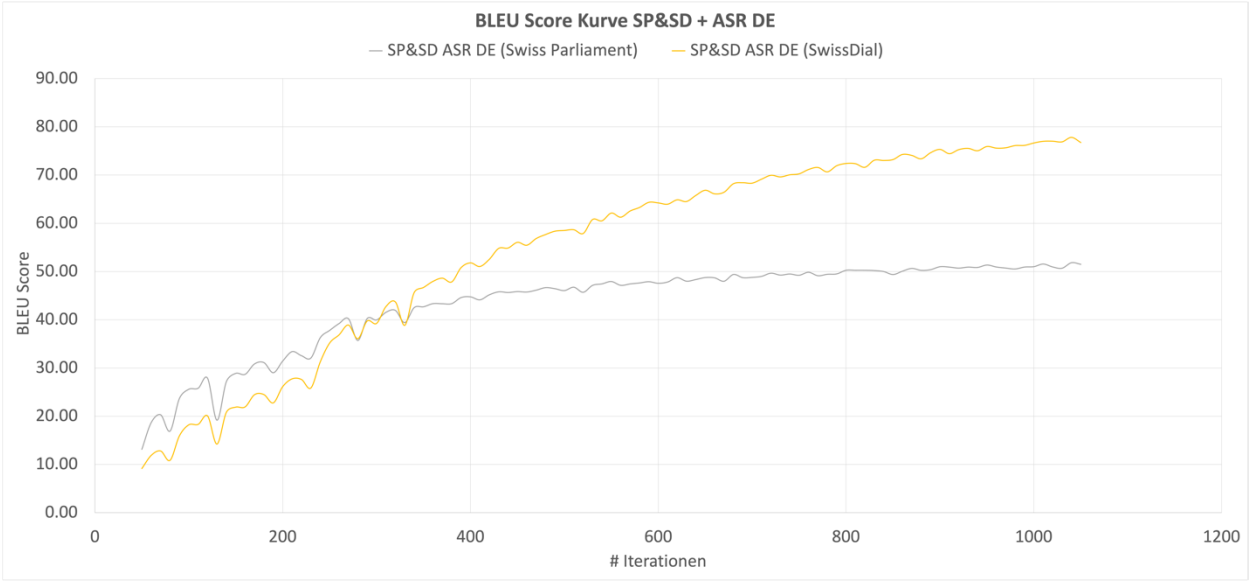

Figure 3: BLEU Score Kurve SP&SD+ASR DE

In Figure 4 ist die BLEU Score Kurve für die Korpora Swiss Parliament und SwissDial von dem Modell SP&SD+ASR CH abgebildet. Beim Training vom SP&SD+ASR CH erreicht der SwissDial von den ersten Iterationen an bessere Resultate als der Swiss Parliament. Diese Entwicklung ist darauf zurückzuführen, dass das Modell mit dem ASR CH vortrainiert wurde. ASR CH wurde unter anderem auf den Daten von dem SwissDial Korpus trainiert, sodass das Modell über ein Vorwissen über den Korpus verfügt. Diese Tatsache widerspiegelt sich in der Entwicklung der BLEU Score Kurve vom SwissDial.

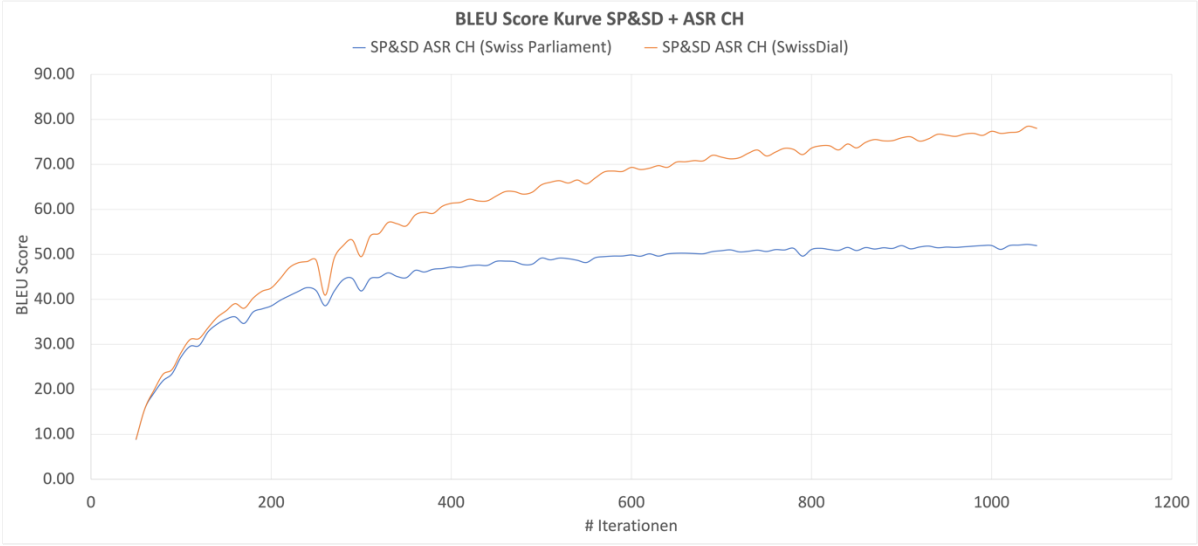

Figure 4: BLEU Score Kurve SP&SD+ASR CH

Zuletzt werden die Kurven der beiden Modelle in einer Graphik dargestellt, siehe Figure 5. Es ist zu sehen, dass die Kurve von dem Swiss Parliament Korpus bei den beiden Modellen einen sehr ähnlichen Verlauf hat. Der Verlauf der SwissDial Kurve unterscheidet sich hingegen stark in den beiden Modellen. Kurz vor der Iteration 1000 treffen sich jedoch die Kurven, sodass beide Modelle einen ähnlichen BLEU Score auf dem SwissDial Korpus erreichen.

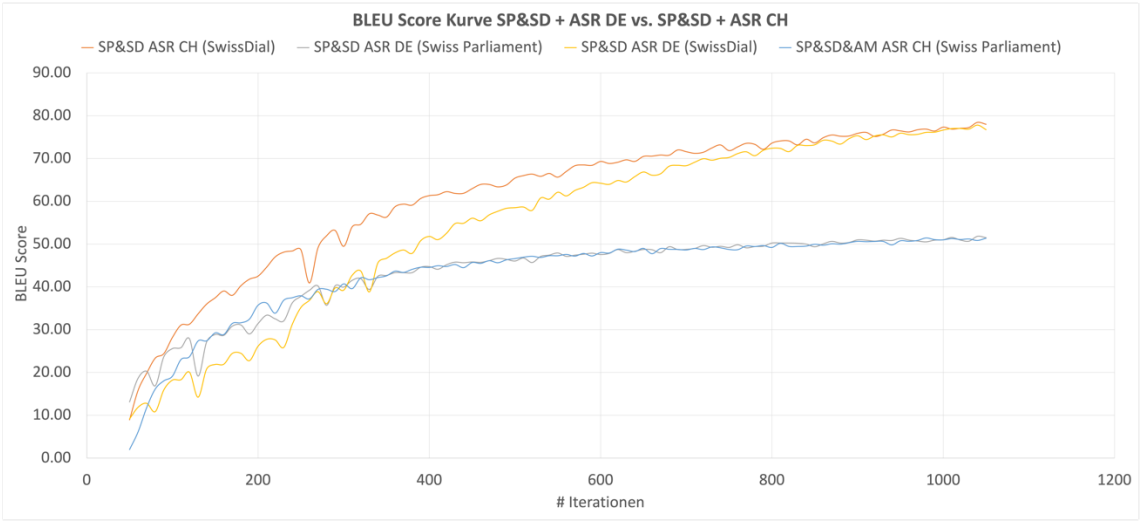

Figure 5: BLEU Score Kurve SP&SD+ASR DE vs. SP&SD+ASR CH

In diesem Abschnitt wurde die Auswirkung von dem vortrainierten ASR Encoder auf den Verlauf auf die BLEU Score Kurve vom SwissDial Korpus analysiert.

9. Diskussion und Ausblick

Im Rahmen dieser Arbeit sind zahlreiche Experimente entstanden. Das Endergebnis ist ein STT Translation System, welches die schweizerdeutsche Sprache unabhängig vom Dialekt in einen hochdeutschen Text übersetzt. Das beste System E3 erreicht einen Score von 39.48 BLEU Punkten auf dem Private-Teil des «Clickworker»-Testsets und einen Score von 58.50 BLEU Punkten auf dem Swiss Parliament Testset. Im Vergleich zum ersten entwickelten System SP 0.9, welches auf dem Public-Teil des «Clickworker»-Testsets einen Score von 14.48 BLUE Punkten erreichte, konnte eine beachtliche Verbesserung von 25.00 BLEU Punkten erzielt werden. In Prozenten ausgedrückt ist der Score um 273% besser geworden. Des Weiteren hat das System als individueller Beitrag den dritten und als ZHAW-Team-Beitrag den zweiten Rang auf dem Shared Task erreicht. Damit wurde das in der Aufgabenstellung formulierte Ziel mehr als erfüllt.

Durch systematische Erweiterung der Systeme konnten Massnahmen identifiziert werden, welche effektiv zur Verbesserung der Modell-Performance führen. Die gewonnen Erkenntnisse können als Inputs und Hilfestellung für die weitere Optimierung eines STT-Translation Systems dienen. Durch die Dialekt-Analyse in Kapitel 8.1 wurden die anspruchsvollsten Dialekte erkannt und die grösste Schwachstelle des Systems gefunden.

In nächsten Schritten würde sich lohnen einen besseren ASR CH Encoder zu trainieren sowie die Trainingsdaten um Audiodateien mit Walliser- und Aargauer-Deutsch zu erweitern. Ein interessantes Experiment wäre das wav2vec Modell als vortrainierten ASR Encoder in den Trainingsprozess einzubeziehen, wie Wu et al. in [10]. Weitere Experimente mit Data Augmentation oder Pseudo-Labeling scheinen auch sehr interessant und erfolgsversprechend zu sein. Vor allem in Anbetracht der Erkenntnis, dass das System mit Daten schlechter Qualität gut umgehen kann. Da es sich bei Speech Translation um ein sehr spannendes und innovatives Gebiet mit einer dynamischen Entwicklung handelt, ist der Spielraum für das Ausprobieren neuer Ideen sehr gross und bietet unzählige Möglichkeiten.

Im linguistischen Sinn wäre es interessant herauszufinden, ob sich die Erkenntnisse in Bezug auf Schweizerdeutsch – Hochdeutsch auf andere ähnliche Sprachkombinationen übertragen lassen. Hierfür bietet sich die spanische Sprache an, mit ihren zahlreichen Ausprägungen und Dialekten auf dem südamerikanischen Kontinent als auch auf der iberischen Halbinsel. Arabische Sprache wäre auch ein guter Kandidat für ein solches Experiment.

10. Verzeichnisse

10.1. Literaturverzeichnis

[1] M. Sperber und M. Paulik, «Speech Translation and the End-to-End Promise: Taking Stock of Where We Are,» 14.04.2020.

[2] A. Waibel, A. N. Jain, A. E. McNair, H. Saito, A. G. Hauptmann und J. Tebelskis, «JANUS: a speech-to-speech translation system using connectionist and symbolic processing strategies,» in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Boston, 1991, pp. 793-796.

[3] A. Berard, O. Pietquin, L. Besacier und C. Servan, «Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation,» in *End-to-end Learning for Speech and Audio Processing Workshop*, NIPS, 2016.

[4] I. Sutskever, O. Vinyals und Q. V. Le, «Sequence to Sequence Learning with Neural Networks,» in *Advances in Neural Information Processing Systems*, 2014, pp. 3104-3112.

[5] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer und J. Uszkoreit, «Tensor2Tensor for neural machine translation,» in *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas*, Boston, Association for Machine Translation in the Americas, 2018, pp. 193-199.

[6] C. Wang, A. Wu und J. Pino, «CoVoST 2 and Massively Multilingual Speech-to-Text Translation,» Facebook AI, 24.10.2020.

[7] C. Wang, J. Pino und J. Gu, «Improving Cross-Lingual Transfer Learning for End-to-End Speech Recognition with Speech Translation,» Facebook AI, 29.10.2020.

[8] Y. Jia, M. Johnson, W. Macherey, R. J. Weiss, Y. Cao, C.-c. Chiu Naveen, A. Stella und L. Yonghui, «Leveraging Weakly Supervised Data to Improve End-to-End Speech-to-Text Translation,» in *International Conference on Acoustics, Speech and Signal Processing*, Brighton, 2019a.

[9] A. Baevski, H. Zhou, A. Mohamed und M. Auli, «wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations».

[10] A. Wu, C. Wang, J. Pino und J. Gu, «Self-Supervised Representations Improve End-to-End Speech Translation,» Facebook AI, 29.10.2020.

[11] J. Pino, Q. Xu, X. Ma, M. J. Dousti und Y. Tang, «Self-Training for End-to-End Speech Translation,» Facebook AI, 09.11.2020.

[12] K. Papineni, S. Roukos, T. Ward und W.-J. Zhu, «BLEU: a Method for Automatic Evaluation of Machine Translation,» Philadelphia, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), 2002, pp. 311-318.

[13] M. Snover, B. Dorr, R. Schwartz, L. Micciulla und J. Makhoul, «A Study of Translation Edit Rate with Targeted Human Annotation».

[14] S. Banerjee und A. Lavie, «METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,» Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization,, Association for Computational Linguistics, 2005, pp. 65-72.

[15] P. N. Garner, D. Imseng und T. Meyer, «Automatic Speech Recognition and Translation of a Swiss German Dialect: Walliserdeutsch,» Martigny, Schweiz, 2014.

[16] M. Plüss, L. Neukom und M. Vogel, «GermEval 2020 Task 4: Low-Resource Speech-to-Text,» Windisch, Switzerland, Institute for Data Science University of Applied Sciences and Arts Northwestern Switzerland, 2020.

[17] M. Plüss, L. Neukom und M. Vogel, «Swiss Parliaments Corpus, an Automatically Aligned Swiss German Speech to Standard German Text Corpus,» Windisch, Switzerland, Institute for Data Science University of Applied Sciences and Arts Northwestern Switzerland, 06.10.2020.

[18] J. Mäder, P. Dogan-Schönberger und T. Hofmann, «SwissDial: Parallel Multidialectal Corpus of Spoken Swiss German,» Zürich, ETH Zürich, Department of Computer Science, 2021.

[19] T. Samardžić, Y. Scherrer und G. Elvira, «ArchiMob - A Corpus of Spoken Swiss German,» Language Resources and Evaluation, Portorož, Slovenia, 2016, pp. 4061-4066.

[20] E. Matusov, P. Wilken und Y. Georgakopoulou, «Customizing Neural Machine Translation for Subtitling,» in *Conference on Machine Translation*, Florence, 2019, pp. 82-93.

[21] A. Saboo und T. Baumann, «Integration of Dubbing Constraints into Machine Translaion,» in *Conference on Machine Translation*, Florence, 2019, pp. 94-101.

[22] R. Hsiao, A. Venugopal, T. Köhler, T. Zhang, P. Charoenpornsawat, A. Zollmann, S. Vogel, A. W. Black, T. Schultz und A. Waibel, «Optimizing components for handheld two-way speech translation for an English-Iraqi Arabic system,» in *Annual Conference of the International Speech Communication Association*, Pittsburgh, 2006, pp. 765-768.

[23] E. Vidal, «Finite-State Speech-to-Speech Translation,» in *International Conference on Acous- tics, Speech and Signal Processing (ICASSP)*, Munich, 1997, pp. 111-114.

[24] Y. Tsvetkov, F. Metze und C. Dyer, «Augmenting translation models with simulated acoustic confusions for improved spoken language translation,» in *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, 2014, pp. 616-625.

[25] S. Karpagavalli und E. Chandra, «A Review on Automatic Speech Recognition Architecture and Approaches,» in *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2016, pp. 393-404.

[26] T. Kudo und J. Richardson, «SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing,» Brussels, Belgium, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2018, p. 66–71.

[27] B. Li, Y. Zhang, T. Sainath, Y. Wu und W. Chan, «Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes,» in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5621-5625.

[28] A. Viterbi, «Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.,» in *IEEE Transactions on Information Theory*, 1967, pp. 260-269.

[29] M. A. Chéragui, «Theoretical Overview of Machine translation,» in *4th International Conference on Web and Information Technologies*, Sidi Bel-Abbes, 2012, pp. 160-169.

[30] S. E, H. Iida und H. Kohyama, «Translating with Examples: A New Approach to Machine Translation,» in *Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*, 1990, pp. 203-212.

[31] I. K, «Automatic Construction of Translation Knowledge for Corpus-based Machine Translation,» 2004.

[32] P. K, E. Peterson, J. Carbonell und L. Levin, «MT for Minority Language Using Elicitation-based Learning of Syntactic Transfer Rules,» Kluwer Academic, 2002, pp. 245-270.

[33] J. Iranzo-Sanchez, J. A. Silvestre-Cerda, J. Jorge, N. Rosello, A. Gimenez, A. Sanchis, J. Civera und A. Juan, «EUROPARL-ST: A MULTILINGUAL CORPUS FOR SPEECH TRANSLATION OF PARLIAMENTARY DEBATES,» 2020.

[34] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk und Q. V. Le., «Specaugment: A simple data augmentation method for automatic speech recognition.,» Interspeech, 2019.

[35] A. Graves, «Generating sequences with recurrent neural networks,» 2013.

[36] A. Berard, L. Besacier, A. C. Kocabiyikoglu und O. Pietquin, «END-TO-END AUTOMATIC SPEECH TRANSLATION OF AUDIOBOOKS,» Grenoble-Alpes, Lille, 12.02.2018.

[37] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu und Z. Chen, «Sequence-to-Sequence Models Can Directly Translate Foreign Speech,» in *Proc. Interspeech*, 2017.

[38] S. Hochreiter und J. Schmidhuber, «Long short-term memory,» in *Neural Computation*, 1997, pp. 1735-1780.

[39] 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, pp. 598-6008.

[40] M. A. Ulasik, M. Hürlimann, B. Dubel, Y. Kaufmann, S. Rudolf, J. Deriu, K. Mlynchyk, H.-P. Hutter und M. Cieliebak, «ZHAW-CAI: Ensemble Method for Swiss German Speech to Standard German Text,» 2021.

[41] A. Conneau, A. Baevski, R. Collobert, A. Mohamed und M. Auli, «UNSUPERVISED CROSS-LINGUAL REPRESENTATION LEARNING FOR SPEECH RECOGNITION,» Facebook AI, 15.12.2020.

[42] M. Büchi, M. A. Ulasik, M. Hürlimann, F. Benites, P. Däniken und M. Cieliebak, «ZHAW-InIT at GermEval 2020 Task 4: Low-Resource Speech-to-Text,» in *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, CEUR-WS, 2020.

[43] I. Nigmatulina, *Acoustic modelling for Swiss German ASR,* Zürich: Philosophische Fakultät der Universität Zürich, 2020.

[44] E. Dieth und C. Schmid-Cadalbert, «Schwyzertütschi dialäktschrift,» Aarau, Sauerländer, 1986.

[45] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers und G. Weber, «Common Voice: A Massively-Multilingual Speech Corpus,» 05.03.2020.

[46] C. Wang, Y. Tang, X. Ma, A. Wu, D. Okhonko und J. Pino, «FAIRSEQ S2T: Fast Speech-to-Text Modeling with FAIRSEQ,» Facebook AI.

[47] H. Inaguma, S. Kiyono, K. Duh, S. Karita, N. Yalta, T. Hayashi und S. Watanabe, «ESPnet-ST: All-in-One Speech Translation Toolkit,» in *In Proceedings of the 58th Annual Meeting of the Association for Computational 37 Linguistics: System Demonstrations*, Association for Computational Linguistics, 30.09.2020, pp. 302-311.

[48] J. Shen, P. Nguyen, Y. Wu, Z. Chen, M. X. Chen, Y. Jia, A. Kannan, T. Sainath, Y. Cao und C.-C. Chiu, «Lingvo: a modular and scalable framework for sequence-to-sequence modeling,» arXiv, 2019.

[49] O. Kuchaiev, B. Ginsburg, I. Gitman, V. Lavrukhin, C. Case und P. Micikevicius, «Openseq2seq: extensible toolkit for distributed and mixed precision training of sequence-to- sequence models,» *In Proceedings of Workshop for NLP Open Source Software,* pp. 41-46, 2018.

[50] A. Zeyer, T. Alkhouli und H. Ney, «RETURNN as a generic flexible neural toolkit with application to translation and speech recognition.,» in *Proceedings of ACL 2018, System Demonstrations*, Melbourne, Association for Computational Linguistics, 2018, pp. 138-133.

[51] T. Zenkel, M. Sperber, J. Niehues, M. Müller, N.-Q. Pham, S. Stüker und A. Waibel, «Open source toolkit for speech to text translation.,» in *The Prague Bulletin of Mathematical Linguistics*, 2018, pp. 125-135.

[52] G. Klein, Y. Kim, Y. Deng, J. Senellart und A. Rush, «OpenNMT: Open-source toolkit for neural machine translation,» in *Proceedings of ACL 2017*, Vancouver, Association for Computational Linguistics, 2017, pp. 67-72.

[53] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian und P. Schwarz, «The kaldi speech recognition toolkit,» in *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society, 2011.

[54] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky und R. Collobert, «wav2letter++: The fastest open-source speech recognition system,» CoRR, 2018.

[55] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer und S. Khudanpur, «A study on data augmentation of reverberant speech for robust speech recognition,» in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, p. 5220–5224.

[56] C. Wang, K. Cho und J. Gu, «Neural machine translation with byte-level subwords,» in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020a, p. 34:9154–9160.

[57] O. Viikki und K. Laurila, «Cepstral domain segmental feature vector normalization for noise robust speech recognition,» in *Speech Communication 25*, Elsevier, 1998.

[58] V. Panayotov, G. Chen, D. Povey und S. Khudanpur, «LIBRISPEECH: AN ASR CORPUS BASED ON PUBLIC DOMAIN AUDIO BOOKS,» in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[59] «Nvidia Titan Xp Spec,» [Online]. Available: https://www.nvidia.com/en-us/titan/titan-xp/. [Zugriff am 30 Mai 2021].

[60] «Nvidia Tesla T4 Spec,» [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf. [Zugriff am 30 Mai 2021].

[61] D. P. Kingma und J. Lei Ba, «Adam: A Method for Stochastic Optimization,» in *ICLR*, arXvi, 2017.

10.2. Glossar

Alignement	Das Abstimmen der Audiodatei mit ihrer Translation / Transkription.
Audio Features	Eigenschaften, welche eine Audiodatei beschreiben.
Baseline	Ein fix definierter Standard.
Cascaded Architektur	Architektur, welche aus mehreren Komponenten besteht
CMVN (Cepestral Mean and Variance Normalization)	CMVN [57] ist eine Technik, welche die Audiofeatures normalisiert, wodurch das Rauschen aus den Rohdaten entfernt wird und die Umgebungseinflüsse wie Aufnahmequalität nicht berücksichtigt werden. Das Ziel dieses Verfahrens ist die Robustheit der ASR- oder ST-Systeme zu verbessern.
Community	Gruppe von Menschen, welche gemeinsame Interessen haben und / oder gemeinsames Ziel verfolgen.
Data Augmentation	Eine Technik zur Erweiterung der Trainingsdaten.
Decoding	In Speech Translation ein Vorgang, in dem eine Translation in Vektorform in die Textform übertragen wird.
Dropout	Regulierungsmethode zum Verringern vom Overfitting. Sie bestimmt prozentuell die Anzahl Verbindungen zwischen Neuronen, welche zu unterbrechen sind.
Error Propagation	Das Weiterreichen eines Fehlers.
Filterbank	Begriff aus dem Speech Processing. Eine Vektorrepräsentation zum Speichern der Audio Features.
Fine Tuning	Das nachträgliche Trainieren eines Modells mit zusätzlichem Trainingsdaten.
Finit-State Models	Endlicher Automat. Ein Automat mit endlicher Anzahl an Zuständen.
GPU	Ein optimiertes Computer-Prozessor
High Resource Language	Sprachen, welche über viele Trainingsdaten verfügen.
Hypothese	Eine vom Modell generierte Translation / Transkription.
Intersection over Union (IoU)	Schnittmenge über die Vereinigungsmenge.
Korpus	Ein Datensatz, welches zum Trainieren von Modellen verwendet wird.
Label	Eine hochqualitative Translation / Transkription für eine Audiodatei
Low Resource Language	Sprachen, welche über wenig Trainingsdaten verfügen.
n-grams	Anzahl von n Elementen
LSTM Netze	Neuronale Netze Architektur mit «Gedächtnis»
Machine Translation	Automatischer Übersetzer
Manifest	Eine Datei mit dem Mapping zwischen Audiodateien und der alignierten Translationen / Transkriptionen sowie weiteren Metadaten.
Mel-Frequency Cepestral Coefficients (MFCCs)	Begriff aus dem Speech Processing. Eine Vektorrepräsentation zum Speichern der Audio Features.
Online Speech Data Transformationen	Eine Technik zur Bearbeitung und Erweiterung der Trainingsdaten während des Modell-Trainings.
Optimizer	Optimizer optimiert die Modellparameter nach jeder Trainingsiteration. Das Ziel dabei ist die Loss Funktion zu minimieren, was zu einer besseren Präzision von den Predicitons führt. Es gibt verschieden Methoden für die Optimierung der Modellparameter. In dieser Arbeit wird der Adam Optimizer [61] eingesetzt.

Overfitting	Ein Verhalten vom Modell, welches stark auf die Trainingsdaten optimiert ist und keine Fähigkeit besitzt auf neue Daten zu generalisieren.
Prediction	Eine vom Modell generierte Translation / Transkription.
RNN	Neuronale Netze Architektur welche Sequenzen bearbeiten kann, indem neuronale Verbindungen in alle Richtungen ermöglicht werden.
Sample rate (Samplingrate)	Abtastung. Der Begriff kommt aus der Signalbearbeitung und beschreibt die Anzahl der Samples pro Sekunde.
Sequence to Sequence Architektur	Eine Architektur, welche aus zwei Neuronalen Netzen, einem Decoder und Encoder besteht. Beide Komponenten tauschen sich einen Kontextvektor, um miteinander zu kommunizieren (Attention).
Simultaneous Speech Translation	Ein Speech Translation System für die Echtzeitübertragung, welches Translationen aus einem Audio-Input-Stream generiert.
Speech Features	Eigenschaften, welche die Sprache beschreiben.
Speech Synthesis	Eine Technik synthetische Sprache zu generieren.
Speech-To-Speech	Bezieht sich auf ein Speech System, welches aus Sprache synthetische Sprache generiert.
Speech-To-Text	Bezieht sich auf ein Speech System, welches aus Sprache einen Text generiert.
State of the Art	Ein Massstab für eine Technik, ein Vorgehen…
Textnormalisierung	Ein Vorgang, in dem ein Text in eine vordefinierte (normalisierte) Form gebracht wird.
Tokenizer	Tokenizer zerlegt einen Text in logisch zusammenhängende Einheiten, welche Tokens genannt werden. Token kann ein Wort, ein Teilwort, aber auch ein Character oder Byte sein.
Trainingsdaten	Daten, welche zum Modelltraining verwendet werden.
ungelabelte Daten	In Speech Translation sind es Audiodateien, welche über keine Translation / Transkription verfügen.
Waveform	Form des zeitlichen Verlaufs einer Schwingung.

10.3. Abbildungsverzeichnis

10.4. Tabellenverzeichnis

10.5. Code Verzeichnis

10.6. Figure Verzeichnis

10.7. Abkürzungsverzeichnis

AM	ArchiMob
ASR	Automatic Speech Recognition
E2E	End-to-End
IoU	Intersection over Unit
LSTM	Long Short-Term Memory
MT	Machine Translation
RNN	Recurrent Neural Network
SD	SwissDial
Seq2Seq	Sequence-to-Sequence
SP	Swiss Parliament
ST	Speech Translation
STS	Speech-to-Speech
STT	Speech-to-Text

Anhang

Kantonkürzel

	Kanton (Dialekt)
AG	Aargau
BE	Bern
BL	Basel-Landschaft
BS	Basel-Stadt
DE	Hochdeutsch
GL	Glarus
GR	Graubünden
LU	Luzern
NW	Nidwalden
SG	St. Gallen
SH	Schaffhausen
SZ	Schwyz
UR	Uri
VS	Wallis
ZH	Zürich

GitHub Repository / OneDrive Ablage

Die im Rahmen dieser Bachelorarbeit erarbeiteten Scripts können in folgendem Repository gefunden werden:

https://github.com/dubelbog/st_ch_de

Die trainierten Systeme sind im folgenden Folder auf dem OneDrive Server zu finden:

https://zhaw-my.sharepoint.com/:f:/g/personal/dubelbog_students_zhaw_ch/ErFnMD-WUXKZGiw2cpqMz2OYBB3pD8rquFOT1uf5M8nkPDg?e=gX0V5H