Universität Zürich[UZH]

Master's thesis
presented to the Faculty of Arts and Social Sciences
of the University of Zurich
for the degree of
Master of Arts

Experiments on the Pronunciation Lexicon for Swiss German ASR

Author: Lorenz Nagele
Student ID: 10-126-324

Examiner: Prof. Dr. Martin Volk

Supervisor: Dr. Tanja Samardžić

Institute of Computational Linguistics

Submission date: 31.12.2020

Abstract

This thesis deals with the role of the pronunciation lexicon (PL) within automatic speech recognition (ASR) for Swiss German. On the basis of a Swiss German dictionary, I construct a PL and use it for ASR within the Kaldi Speech Recognition Toolkit. I extend the PL by using a data-driven grapheme-to-phoneme conversion and evaluate its performance when being used in the ASR system. Further experiments analyse the impact of the PL being used in either of the two phases of the ASR system: during the training of the acoustic model and in the decoding process. To test this, I make use of the two PLs created in this work.

Zusammenfassung

In dieser Arbeit befasse ich mich mit der Rolle, die das Aussprachelexikon in der automatischen Spracherkennung für Schweizerdeutsche Dialekte einnimmt. Unter Verwendung eines bestehenden Lexikons, das für Schweizerdeutsche Dialekte entworfen wurde, stelle ich ein Aussprachelexikon, das ich in einem Spracherkenner verwenden kann, zusammen. Um die Spracherkennungssysteme zu modellieren, verwende ich das «Kaldi Speech Recognition Toolkit». Später erweitere ich das Aussprachelexikon mit einem datengetriebenen Ansatz, genannt Graphem-Phonem-Übersetzung. Die Qualität des entstandenen Aussprachelexikons beurteile ich anhand der Resultate des Spracherkenners. In weiterführenden Versuchen untersuche ich, welche Auswirkungen das Sprachlexikon auf die einzelnen Phasen des Spracherkenners hat: auf die akustische Modellierung und auf die Dekodierung. Dazu verwende ich die beiden Aussprachelexika, die ich im Rahmen dieser Arbeit entworfen habe.

Acknowledgement

In the first place, I would like to thank my supervisor Dr. Tanja Samardžić for her guidance, advice and motivation. She not only supported me in my work, but also taught me a lot on a human level. Her presence and readiness to support are much appreciated.

Also, I want to thank Iuliia Nigmatulina and Tannon Kew for helping me to get started with the topic and for laying a great basis to work on.

Furthermore, I would like to thank Dr. Martin Volk and Dr. Manfred Klenner for the quick handling when it was needed.

Last but not least, I would like to thank my friends and family who were there for me all the time, when needing warm hospitality, motivational talks or relaxing walks during that time. It was a tough year for everyone. On to better years!

Contents

List of Figures

List of Tables

List of Acronyms

ASCII	American Standard Code for Information Interchange
CER	Character Error Rate
CMVN	Cepstral Mean and Variance Normalisation
CSV	Comma-Separated Values
CTC	Connectionist Temporal Classification
EXB	EXMARaLDA Basic Transcription
FST	Finite-State Transducer
G2P	Grapheme-to-Phoneme
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IPA	International Phonetic Alphabet
IVR	Interactive Voice Response
LDA	Linear Discriminative Analysis
LM	Language Model
LMWT	Language Model Weight
LTS	Letter-to-Sound
LVCSR	Large-Vocabulary Continuous Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficient
MITLM	MIT Language Modelling Toolkit
MLLT	Maximum Likelihood Linear Transform
MMI	Maximum Mutual Information
MT	Machine Translation
NLP	Natural Language Processing
NN	Neural Network
OOV	Out-of-Vocabulary
P2G	Phoneme-to-Grapheme
PL	Pronunciation Lexicon
PM	Pronunciation Model

RNN	Recurrent Neural Network
SAMPA	Speech Assessment Methods Phonetic Alphabet
seq2seq	Sequence-to-Sequence
STTS	Stuttgart-Tübingen-Tagset
TDNN	Time-Delay Neural Network
URPP	University Research Priority Programs by the University of Zurich
WAV	Waveform Audio File Format (WAVE)
WER	Word Error Rate
WFST	Weighted Finite-State Transducer
WIP	Word Insertion Penalty
WSJ	Wall Street Journal
X-SAMPA	Extended SAMPA
XML	eXtensible Markup Language

1 Introduction

The task of Automatic Speech Recognition (ASR) is to generate the transcription of what is being said in an acoustic speech signal. In recent years, there have been enormous developments in the field of Natural Language Processing (NLP) due to more powerful computing and the existence of more data. The field of ASR is not exempt from these developments and there are reports on systems that are near to perfection when it comes to recognising the correct transcription [Han et al., 2020]; [Synnaeve et al., 2020]. However, systems performing that well are only available for languages that experience a lot of research and therefore have more data available, like English.

In the case of languages that do not have vast amounts of data available, training an ASR system for these so-called low-resource languages can be much more difficult. If such a language does not even have a standardised writing system and is set in a multi-dialectal scenario, like Swiss German, building a speech recogniser turns out to be even more challenging.

This is the exact scenario that I am dealing with in this thesis. I investigate the methods of training an ASR system for Swiss German. Within the operation of such a system, I want to particularly focus on one of the system's components, namely the Pronunciation Lexicon (PL). The PL basically represents a list of words mapped to how they are pronounced. This helps the ASR system better recognise the speech signals and generate the correct words depicted in them.

1.1 Motivation

Being a native Swiss German speaker of a dialect that is spoken by only a few thousand people, I find it very interesting trying to build a speech recogniser for these low-resource languages. When looking at the daily use of ASR systems in the applications such as the speech assistant integrated in mobile phones or switching the channel by speaking to the remote control, it is striking that (almost) nobody uses these systems at all within the Swiss German regions. The main problems are,

on the one hand, that Swiss people do not want to speak in another language for which these systems are built and, on the other hand, that if there are applications recognising Swiss German, their quality is very poor and one does not get acquainted with using them. This is the main reason for me to investigate in how we could improve the automatic recognition for Swiss German dialects.

A lot of work has already been done on the topic of ASR for Swiss German within the Institute of Computational Linguistics at the University of Zurich. Most recently, the Master's theses by Nigmatulina [2020] and Kew [2020] examined two aspects of the ASR system when being used for Swiss German, namely the Acoustic Model (AM) and the Language Model (LM). In my work, which is mostly based on the findings from their theses, I want to focus on the third part which is the PL.

While acquiring the background knowledge for the thesis, I stumbled upon the paper by Jouvet et al. [2012]. It made me realise that the PL can be put into practice within two different phases of the ASR system. Once during the training of the AM and later on, when decoding new acoustic speech signals. This made me want to investigate more on the influence of the PL used in the two phases.

1.2 Research Questions

In this work, I focus on the role of the PL within ASR for Swiss German. Having an existing dictionary as a basis to build a PL for the use in an ASR system, I want to investigate methods to extend the PL to cover more words. At the same time, I want to know how well the baseline PL performs compared to the extended PL. The research questions is formulated as follows:

RQ1: Does a PL extended by data-driven methods improve the performance of the ASR system compared to using a baseline PL in terms of output quality evaluated by an evaluation metric?

Furthermore, I want to examine the impact of the PL being used in the two phases of the ASR system. The phases in which the PL can be used are the training of the acoustic model and the decoding process. This leads to the second research question:

RQ2: How does the PL impact the two phases in the ASR system in which the PL can be used the performance of the ASR system in terms of output quality evaluated by an evaluation metric?

1.3 Thesis Structure

The thesis is structured as follows. In Chapter 2, I describe the background knowledge that I acquired during the work and which is needed to perform the experiments. This includes the general operation of an ASR system, a more in-depth look at the PL and its role in the system and the replication of the baseline system.

In Chapter 3, I present the materials used for the experiments. This comprises the data resources and the tools to process them. The application of the materials is then described in Chapter 4. I explain the methods used to prepare the PL, extend it by a data-driven approach and show how the PL is used in different phases of the ASR system. The results from these experiments are then presented and discussed in Chapter 5. Finally, I give a conclusion to the present work in Chapter 6.

The code I worked and the PLs created in this work can be found on GitHub.[1]

[1] `https://github.com/twipsii/MA-Kaldi-PL`

2 Background

The following chapter deals with the background knowledge that was acquired as a basis for performing the experiments in this work. This includes an introduction to the field of ASR and what the general architecture of a typical ASR system is composed of (Section 2.1), and a more detailed look at the pronunciation lexicon, its role in the ASR system and how it is built (Section 2.2). Moreover, I describe how I replicated the baseline system elaborated in previous work, including a detailed view of its pipeline (Section 2.3).

2.1 Automatic Speech Recognition

The goal of ASR is to build a system that maps an acoustic signal representing spoken language to the corresponding sequence of words. With the increasing digitisation in the recent years, more and more human-computer interaction is happening by using our voice. ASR can be applied in a wide range of applications. One of these are Interactive Voice Response (IVR) systems used, for example, in telecommunications where customers can interact with a company's bot system in a dialogue design to perform various services (e.g. block credit card in banking). Other applications are digital dictation systems which display the recognised words as they are spoken (e.g. used by doctors for medical documentation). Including further technologies, ASR is also used in speech translation where the recognised text is machine translated into another language and possibly even further processed with speech synthesis to reproduce the utterance in the target language.

Given the increased interest and demand in speech applications, there have also been considerable advancements in the performance of ASR systems. Recent papers reported word error rates (WER, see Section 2.1.2) of less than 5% on the widely used LibriSpeech[1] benchmark test set [Han et al., 2020]; [Synnaeve et al., 2020].

[1] LibriSpeech is a large-scale and freely available ASR corpus containing approximately 1000 hours of read English speech [Panayotov et al., 2015]. The data is derived from read audio books from the LibriVox project. Its official test set is commonly used as a benchmark for the evaluation of speech recognition tasks.

This was achieved on English, a language for which there is more training data available than for any other language.

The performance of an ASR system is highly dependent on the language. This is firstly due to the amount of training data that is available. For less researched languages there is consequently less data. Secondly, languages with a greater lexical variety and/or more complex morphology are considered harder to process. The selection of training data and the resulting performance of an ASR system is not only dependent on the language, but also on the aims of the corresponding speech recognition task. One of these dimensions of variation is the size of the vocabulary. It is easier if the number of words that are needed to recognise is smaller, e.g. digit tasks where sequences of digits are recognised have a very limited vocabulary (zero to nine). On the other hand, tasks like transcribing conversations or given speeches need a large vocabulary with 20,000 to 60,000 words and are therefore much harder to handle [Jurafsky and Martin, 2009]. To better represent how the words from the vocabulary are actually expressed, they can be enriched with a pronunciation (sequence of phones). This is defined in the pronunciation lexicon (PL). More information on the PL can be found in Section 2.2.

A second dimension of variation is the type of the speech, i.e. how fluent, natural or conversational the speech is. We are differentiating between *isolated word* recognition, where each word is uttered individually and surrounded by some sort of pause, and *continuous speech* recognition, where words fluently follow each other and have to be segmented. The words in the latter are thus much harder to recognise.

Another dimension of variation is the accent (or dialect). If the language to be recognised is in a standard dialect and the system was trained on this particular data, it is much easier to recognise the speech. However, if there are multiple dialects as in Swiss German, this can make the task more difficult.

Given these dimensions of variation in speech tasks, we are talking about Large-Vocabulary Continuous Speech Recognition (LVCSR) in this work, where we have a large vocabulary (between 20,000 and 60,000 words) and the speech is continuous in a way that words are run together naturally. In the following sections, I want to describe the general architecture of an ASR system built for LVCSR and give an overview of how the written output is generated.

2.1.1 Architecture

The goal of speech recognition is to generate the string of words (output) that is depicted by the acoustic speech signal (input). In a conventional ASR system based on the Hidden Markov Model (HMM), this task can be represented by the *noisy channel model*. In this model, the acoustic waveform can be seen as a "noisy" or "corrupted" version of its text transcription, which was transmitted through this noisy channel. The "noise" introduced by this channel makes it hard to recognise the true transcription. Thus, the goal is to build a model that figures out how the transcription was modified and then recovers the true transcription. Knowing how the noisy channel distorts the source, the correct source sentence for a waveform can be found by searching through each sentence. For any new acoustic speech signal, running all possible sentences through the noisy channel model reveals which one best matches the input signal [Jurafsky and Martin, 2009].

In Figure 1, you can see a simplified version of the general architecture of an ASR system within the noisy channel logic, from processing the input speech signal to generating the output transcription. Given an acoustic speech signal as an input, the model tries to find the most probable transcription as an output.

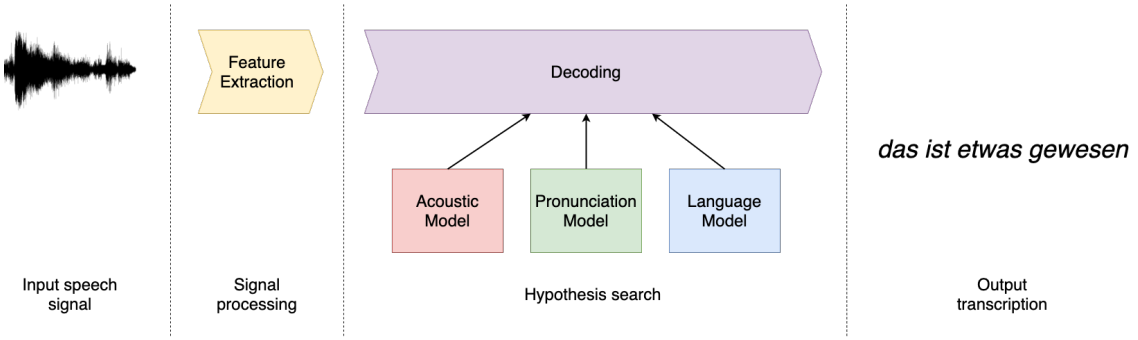

Figure 1: Simplified architecture of a conventional ASR system from the input speech signal to the output transcription (adapted from Kew [2020])

In a more formal way, an acoustic input signal X is sliced up into overlapping frames x_i (usually with a duration of 25ms shifted by 10ms each time) from which the feature vector can be extracted:

$$X = x_1, x_2, x_3, ..., x_t \tag{2.1}$$

where t indicates the number of frames the acoustic signal is divided into. Each index represents a time interval. In the same way, a sentence W is composed of a

sequence of words w_j:

$$W = w_1, w_2, w_3, ..., w_n \tag{2.2}$$

where n represents the number of words in the sentence.

Therefore, the task of ASR can also be formulated as finding the most likely sentence W^* out of all sentences in the language $\mathcal{L}$ given some acoustic input X. The probabilistic representation of this looks as follows:

$$W^* = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(W|X) \tag{2.3}$$

However, for a generative approach like HMMs it is easier to model the observation sequence $P(X|W)$ than $P(W|X)$ directly [Gales and Young, 2007]. To take this into account, we can make use of Bayes' rule to transform Eq. 2.3 into:

$$W^* = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \frac{P(X|W)P(W)}{P(X)} \tag{2.4}$$

The probability of the acoustic observation sequence $P(X)$ is the same for all possible candidate sentences given an input utterance. Thus, its computation can be omitted from the equation which leaves us with:

$$W^* = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(X|W)P(W) \tag{2.5}$$

As we can see, the most probable sentence W^* given some acoustic observation sequence X can be computed by multiplying the two probabilities $P(X|W)$ and $P(W)$ for each sentence and selecting the sentence for which this product is the greatest. The factors in Eq. 2.5 can be computed more easily than in the inital Eq. 2.3. $P(W)$, the prior probability of the sentence (or word sequence) itself, is estimated by the language model (LM). $P(X|W)$, the observation likelihood, is computed by the acoustic model (AM) and the pronunciation model (PM). In the decoding phase, the maximisation of these two factors is used to find the most probable sentence W^*.

2.1.2 Evaluation

In order to efficiently compare the performance of different ASR systems, there are evaluation metrics to automatically rate the system. The standard evaluation metric within ASR is the word error rate (WER). The WER is based on how much the output generated by the ASR system differs from the correct reference transcription. Given the output transcription, the WER computes the minimum number of words needed to be substituted, inserted and deleted to get to the same sequence of words as the reference transcription Jurafsky and Martin [2009]. The formula of the WER is defined as:

$$\text{WER} = 100 * \frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Total Words in Reference Transcription}} \tag{2.6}$$

Another commonly used evaluation metric used within ASR is the character error rate (CER). It works the same way as the WER, but instead of computing the number of word changes, it does the same with characters. The CER is usually lower than the WER.

2.2 Pronunciation Lexicon

The pronunciation lexicon (PL), also known as (pronunciation) dictionary, is a list of words including a pronunciation for each word expressed as a sequence of phones. Within speech technologies, it is used for both, speech recognition and speech synthesis. Table 1 shows an example of a PL with three entries as it is used in an ASR system. Each line represents a word and its pronunciation. Based on the system used to train the ASR system, the PL can be stored in a simple text file where the words and pronunciations are separated by a whitespace.

Word	**Pronunciation in ARPAbet**
dictionary	d ih k sh ah n eh r iy
pronunciation	p r ow n ah n s iy ey sh ah n
speech	s p iy ch

Table 1: Pronunciation lexicon consisting of three words and their pronunciations which are transcribed in ARPAbet format. The examples are taken from the CMUdict [Lenzo, 2007].

In the following, I describe the role of a PL within the architecture of an ASR system

and have a closer look at end-to-end systems, which get by without the use of a PL. Furthermore, I take a closer look at how PLs are constructed and what needs to be taken into consideration before doing so. Finally, I present a method how the PL can be automatically extended and clarify why this can be beneficial.

2.2.1 Role in the ASR Architecture

The general architecture of a conventional ASR system was already presented in Section 2.1.1. The PM represents one of the three models and is thus a key component of the ASR system. Let's recap the equation of the noisy channel model described in Section 2.1.1. It represents how the most probable sequence of words is found using the observation likelihood and the prior probability.

$$W^* = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(X|W)P(W) \tag{2.7}$$

We have seen that the prior probability $P(W)$ in Eq. 2.7 is computed by the LM, and that the observation likelihood $P(X|W)$ is calculated by a combination of the AM and PM. In a system based on HMMs, words are represented as a sequence of HMM states Q. Using the HMM states, we can reformulate the equation:

$$\begin{aligned} W^* &= \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(X|Q,W)P(Q,W) \\ &\simeq \underset{W \in \mathcal{L}}{\operatorname{argmax}} \sum_{Q} P(X|Q)P(Q|W)P(W) \\ &\simeq \underset{W \in \mathcal{L}}{\operatorname{argmax}} \max_{Q} P(X|Q)P(Q|W)P(W) \end{aligned} \tag{2.8}$$

The representation of the LM is not affected by the phone states and it still depicts the probability of the word sequence $P(W)$. The AM, $P(X|Q)$, is now expressed as the probability of the acoustic signal X given the phone states Q. Finally, the PM described by $P(Q|W)$, now depicts the probability of the phone states Q given the words W.

As you can see, the PM, as one the three models, plays a decisive role in the architecture of a conventional ASR system. However, recent modelling techniques try get by without relying on a PL at all. Such systems, that directly map the acoustic signal to the transcription, are called end-to-end systems.

2.2.1.1 End-to-End Systems

When looking at the benchmark results of ASR systems[2], it quickly becomes clear that the best performing systems rely on end-to-end approaches. In end-to-end models, an input sequence X is directly mapped to the output sequence W. It therefore directly estimates $P(W|X)$ as seen in Eq. 2.3. This is achieved by combining the AM, PM and LM into a single NN. This way, the model removes the need for a separate PL to map from the phone sequences to the words.

There are various techniques used for end-to-end modelling. One of them is *Connectionist Temporal Classification* (CTC). CTC is a kind of sequence modelling technique without requiring frame-level alignments, i.e. it does not need to match each input frame to an output token. The probability of the output sequence is computed by summing over all possible alignments using the CTC loss function. The model is typically applied to recurrent NN (RNN) systems as used in the Deep Speech system [Hannun et al., 2014].

Another approach for end-to-end modelling is the use of an *encoder-decoder* architecture which can also be based on RNNs. In this setup, the encoder transforms a sequence of acoustic input vectors into a sequence of feature representations. The decoder then generates the output sequence consisting of words. This way, the dependence on a PL and LM is removed. The approach of the RNN-based encoder-decoder for ASR was first discussed in Lu et al. [2015].

As mentioned before, end-to-end models can achieve great results in ASR. However, they tend to need a lot of training data to do so. This can be troublesome for low-resource languages like Swiss German. Within such a setting, using the conventional ASR approach based on HMMs, including an elaborate and accurate PL, might be the better option.

2.2.2 Construction of a Pronunciation Lexicon

While AMs and LMs are automatically built from large speech and text corpora using data-driven processes, PLs are typically manually constructed by professional linguists or phoneticians. This process is therefore time-consuming and expensive.

In order to build a reliable and consistent PL, some parameters need to be agreed on upfront. On the one hand, this concerns the phonetic notation that is used to represent the pronunciations. Based on the goal of the ASR system, it makes

[2] `https://paperswithcode.com/task/speech-recognition`

sense to use one option rather than the other. On the other hand, words may have multiple pronunciations. Therefore, it has to be decided whether these pronunciation variants should be included in the PL, and if so, whether a limit is to be set on how many variants can be covered. Having in mind that the process of building a PL is elaborate, it may make sense to only construct part of the PL and generate the remaining pronunciations using automatic methods. All these considerations are further discussed hereinafter.

2.2.2.1 Phonetic Representations

Each word within the PL is expressed by a sequence phones. There are different phonetic transcriptions that can be used to represent these phones. The phonetic transcriptions vary in the number of phones that can be expressed and the character set that is used to represent the phones. When constructing a PL, the phonetic representation must be selected in advance. It has to be consistent throughout the PL in order not to introduce more uncertainty to the mapping of the word and phone sequence.

In the following, I present four selected phonetic representations that can potentially be used in a PL for the pronunciations. Without going into too much detail, I describe their origin, show in what settings they are used and compare their phone sets.

Table 2 shows a comparison of the first three phonetic representations. A few selected English words are displayed including the corresponding pronunciation in IPA, SAMPA and ARPAbet. The consonants or vowels to be examined are marked in bold. While the IPA uses "special" characters not present in the ASCII encoding, the SAMPA and ARPAbet symbols only consist of printable ASCII characters. As a matter of fact, the ARPAbet only uses Roman letters to represent the pronunciations. Also, the ARPAbet is case-insensitive, i.e. it can also be written in uppercase letters without changing the meaning, or rather the pronunciation in this case.

IPA

The International Phonetic Alphabet (IPA) [International Phonetic Association, 1999] is a standard in phonetic representations and was developed back in 1888 by the International Phonetic Association. Its goal is to provide the phonetic notation to transcribe the sounds of all human languages.

The set of symbols consists of 107 letters representing consonants and vowels. Based on how narrow the transcription needs to be, the letters can be modified with 31

Word	IPA	SAMPA	ARPAbet
		Consonants	
buy	**b**aɪ	**b** a I	**b** ay
see	**s**i	**s** i	**s** iy
thick	**θ**ɪk	**T** I k	**th** ih k
cherry	**tʃ**eri	**t S** e r i	**ch** eh r iy
ri**ng**	rɪ**ŋ**	r I **N**	r ih **ng**
		Vowels	
c**a**t	k**æ**t	k { t	k **ae** t
c**i**ty	s**ɪ**t**i**	s **I** t **i**	s **ih** t **iy**
b**u**tter	b**ʌ**tə	b **V** t @	b **ah** t er
fl**ow**er	fl**aʊ**ə	f l **a U** @	f l **aw** er

Table 2: Comparison of selected English words and their pronunciations in the three different phonetic representations of IPA, SAMPA and ARPAbet. The consonants or vowels in focus are highlighted in bold.

diacritics and 19 additional signs that indicate suprasegmental qualities such as length, intonation or stress. These are organised in the famous official IPA chart.[3]

Due to its large inventory of symbols, the IPA also uses special characters which require a rich encoding standard. These are not convenient for computational representations of pronunciations as used in PLs for ASR. Therefore, subsets of the IPA are used as a basis for phonetic representations that are specifically designed for such tasks. Two of those alphabets are the SAMPA and the ARPAbet.

SAMPA

The Speech Assessment Methods Phonetic Alphabet (SAMPA) [Wells, 1997] can be seen as a subset of the IPA and represents the phones using the 7-bit printable ASCII characters which are case-sensitive. ASCII is a character encoding standard based on the English alphabet.

SAMPA is widely used within speech technologies as a phonetic representation. Its first version was intended to include the phoneme inventories for English, German, French, Italian, Dutch and Danish. This was later extended to cover other European and major non-European languages. The extended version, X-SAMPA, tries to

[3] `https://www.internationalphoneticassociation.org/content/ipa-chart`

represent every symbol of the IPA with ASCII characters, including the diacritics.

The PL that I use as a basis for my experiments on Swiss German, the *Swiss German Dictionary*, consists of symbols from a modified version of SAMPA (see Section 3.1.2). The version using a reduced phone set only consists of 59 symbols of which each represents a distinct phoneme.

A commonly used PL for ASR, *CELEX* [Baayen et al., 1995], also uses SAMPA to represent the pronunciations. It is known as the most richly annotated dictionary [Jurafsky and Martin, 2009]. There are versions for English, German and Dutch. The English version contains pronunciations for over 160,000 word forms.

ARPAbet

The ARPAbet [Shoup, 1980] is another phonetic alphabet that is based on the IPA and uses ASCII symbols. It was specifically designed for American English. The individual phones are represented by up to three letters. There is a total of 50 phones that are covered in the ARPAbet.

Two commonly used PLs that are based on the ARPAbet are the *CMU Pronouncing Dictionary* [Lenzo, 2007], also known as CMUdict, and the *PRONLEX* [Kingsbury et al., 1994]. The CMUdict is a PL designed for North American English and contains over 134,000 words and their pronunciations. Its phone set consists of 39 phones derived from the ARPAbet. The PRONLEX has pronunciations for over 90,000 words and is particularly advantageous because of the many proper names (over 20,000) that it includes.

Dieth Orthography

The ArchiMob corpus (see Section 3.1.1), which I use as the training data for my experiments, is transcribed according to Dieth's orthography [Dieth, 1986]. Even though I do not use the Dieth transcription as the reference in the data for the ASR system, the normalised version that I use was generated based on the Dieth transcriptions.

Eugen Dieth was a Swiss linguist, phonetician and dialectologist who published guidelines on the orthography of Swiss German dialects. These guidelines aim to reflect the phonological characteristics and the individuality of the various Swiss dialects. Dieth's orthography can thus be characterised as a system of phonetic transcription.

Dieth's orthography is mainly used by professional linguists and phoneticians that want to transcribe spoken Swiss German. Although the transcriptions are often inconsistent due to the interpretation of the transcriber, they serve as a useful phonetic

representation for processing Swiss German in ASR systems.

In Table 3 you can see a list of a few selected German words with their notation in Dieth orthography of Zurich dialect and their corresponding SAMPA representation. Based on the training data and the desired output of a Swiss German ASR system, the PL contains the list of words either in Dieth orthography or the standard German equivalent, and their phonetic representations. For the PL in my experiments I use the standard German word, or in some cases a normalised version of the Swiss German word (e.g. *schaffen* instead of *arbeiten*), and its SAMPA representation.

Word	**Dieth Orthography**	**SAMPA**
Stolz ('pride')	schtolz	S t o l ts
Krieg ('war')	chrieg	x r i @ k
gespielt ('played')	gschpilt	k S p i l t
etwas ('something')	öppis	2 p i s
Esszimmer ('dining room')	ässzimmer	{ s s ts i m m @ r
Öffentlichkeit ('public')	öffentlichkäit	2 f f @ n t l l x k x a l t
arbeiten ('work')	schaffe	S a f f @

Table 3: Comparison of selected German words in Dieth orthography of Zurich dialect and its SAMPA representation. The English translation is shown in brackets.

2.2.2.2 Pronunciation Variability

When looking at the words in a PL, most of them have a single pronunciation. However, it is possible that the same word is listed more than once because it has multiple pronunciations. This can have several causes. One of them are homographs, which are words that have the same written form but have different meanings and in some cases are pronounced differently. An example for this is the English verb *read* and its past forms (/ɹiːd/ vs. /ɹɛd/) or the standard German *liebe* ('love') which can either be the noun or the verb in Swiss German (/liəbə/ vs. /liəbi/).

Other words that commonly have more than one pronunciation are frequently used function words like articles, conjunctions or particles. The Switchboard corpus, a commonly used resource of English conversational speech, contains 12 different pronunciations for *because* and hundreds for *the* [Jurafsky and Martin, 2009]. The pronunciation variation can also be caused by the context, particularly by the following word. For example in English, when the word after *the* begins with a consonant,

the is usually pronounced as /ðə/. Otherwise, when the word after *the* begins with a vowel, the pronunciation changes to /ði/.

A third reason for having multiple pronunciation variants of words in a PL is the use in multi-dialectal ASR. Since the same word might be pronounced differently in various dialects of the language, the PL can be enriched to account for the pronunciation variants. The downside of having multiple pronunciations, however, is the higher flexibility added to the model. Different words are likely to share some of the pronunciation variants, which increases the number of potential ambiguities and can then lead to erroneously decoded words in the output of the ASR system. This was also shown in Hain [2002], where having only one, but carefully constructed, pronunciation per word led to a slightly more accurate system than having multiple variants (average of 1.18 pronunciation variants per word).

Multiple pronunciations can also cause problems in Viterbi decoding. Since the Viterbi decoder tries to find the best string of phones rather than the best word string, it biases against words with many pronunciation variants. The more pronunciations variants a word has, the more likely it is to be chosen. These words have therefore a higher probability compared to other words. Instead of using multiple pronunciation variants, a context-dependent acoustic model with triphones can be enough to model phonological phenomena like vowel reduction [Jurafsky et al., 2001].

The average number of pronunciation variants per word in state-of-the-art ASR systems seems to range from 1 to 2.5 [Jurafsky and Martin, 2009]. The PL that I use for my experiments only has a few words with multiple pronunciations. The dictionary which the PL is based on, includes pronunciations for six Swiss German dialects (see Section 3.1.2). Using all of those pronunciations in the PL would result in too much confusion when deciding what word to choose in the output. Therefore, I only use the pronunciations from the Zurich dialect. For the 10,968 different word forms in my PL, there are 11,044 pronunciations. This results in an average of 1.007 pronunciations per word.

Another problem in pronunciation modelling for ASR is the incomplete coverage of words. There are a lot of words that are not covered by the PL. In order to deal with these unseen words, there are ways to automatically augment the PL with pronunciations of those words. Popular techniques to extend the PL are discussed in the next section.

2.2.3 Extending the Pronunciation Lexicon

Words that the ASR system is not familiar with, or rather which it did not see during the training, are called out-of-vocabulary (OOV) words. Common sources for OOV words are named entities (e.g. personal names, geographical places or brands and products) or the extended vocabulary of languages that are morphologically richer than English. This larger vocabulary includes, among other things, compound words in German and the various word forms in highly inflected languages like Finnish or Arabic. OOV words can have a high impact on the quality of the ASR system. Each OOV word is said to result in 1.5 to 2 extra errors due to the loss of contextual information. Therefore, it is important to minimise the OOV rate.

The OOV rate is the percentage of words in the testing data that is not contained in the PL. Training the acoustic model requires pronunciations for all words in the training data in order to construct a HMM for each training utterance. In order to decrease the OOV rate, the PL can be extended with automatic methods. This is particularly beneficial in my Swiss German setting with a rather small PL that consists of roughly 10,000 word forms.

The most popular way to extend the PL is to train a model on the existing words and pronunciations and predict the pronunciations for new words based on this model. This method is further described in the next section.

2.2.3.1 Grapheme-to-Phoneme Conversion

The process of converting a sequence of letters (graphemes) into a sequence of phones is called grapheme-to-phoneme (G2P) conversion. Therefore, the task of a G2P algorithm is to turn a letter string like *dictionary* into its phonetic representation [`d ih k sh ah n eh r iy`]. This can be achieved by manually writing letter-to-sound (LTS) rules. The earliest such rules were written in the Chomsky-Halle rewrite rule format [Chomsky and Halle, 1968]. The LTS rules are applied in order, beginning with the most specific to later default rules that only apply if the context for the earlier rules is not applicable. LTS rules are defined in the following format:

$$A \longrightarrow \text{B} \;\; / \;\; [\text{pre-context} \;\; \_ \;\; \text{post-context}] \tag{2.9}$$

Jurafsky and Martin [2009] present a simple pair of such rules for pronouncing the letter c in English:

$$\begin{aligned} c &\longrightarrow \text{[k]} \;/\; \_\ \{\text{a,o}\} \quad \text{; context-dependent} \\ c &\longrightarrow \text{[s]} \qquad\qquad\quad \text{; context-independent} \end{aligned} \tag{2.10}$$

The rules in Eq. 2.10 define that if the letter c is followed by the letter a or o, it is pronounced as [k]. Otherwise, it is pronounced as [s]. Actual rules are much more complicated and thus expensive to create. A more recent way is to use automatic methods based on machine learning. These methods do not require any hand-written rules and rely on already transcribed data.

The probabilistic approach of G2P conversion was first formalised by Lucassen and Mercer [1984]. Given the letter sequence L, the most probable phone sequence P^* out of all phone sequences is found:

$$P^* = \underset{P}{\operatorname{argmax}}\, P(P|L) \tag{2.11}$$

Statistical approaches like the semi-automatic method of Black et al. [1998] try to find the LTS alignment by computing probabilities for each letter-phone pair from the training set. This needs to be done for each word in the training set. One letter can align to multiple phones or to no phones at all. The approach is semi-automatic because it relies on a pre-defined list of allowable phones that each letter can align to.

In order to find the phone sequence for new words, a machine learning classifier is trained on the aligned training set. For each letter of the word, the classifier generates the most probable phone. The accuracy can be improved by also considering the context. In the case of G2P conversion, this is a fixed number of previous and following letters of the letter the phone has to be predicted of.

Since the G2P problem consists of converting one sequence into another, similar techniques are used for the conversion as in machine translation, which is also a sequence-to-sequence (seq2seq) task. Instead of translating words from one language into another, the task of G2P is to translate from a sequence of letters into a sequence of phones.

Therefore, more recent NN models tend to achieve better results in finding the corresponding phone sequence of new words. The toolkit I used to extend the PL

(see Section 3.2.2) is based on the Transformer model [Vaswani et al., 2017]. Unlike RNNs, the Transformer does not require the sequential data to be processed in order (e.g. in G2P, letters in a word from left to right). Instead it relies entirely on an attention mechanism, which represents the correspondence (or relevance) of each item of the input sequence at each timestep, which is needed to generate the output. This way it can draw global dependencies between the input and output.

2.3 Replicating the Baseline System

This section shows the steps needed to replicate the baseline model which is used as a foundation to perform further experiments on the pronunciation lexicon in this work. The model to be replicated was worked out and presented in Nigmatulina [2020]. I am going to explain the individual steps in the pipeline of the baseline system. This includes the data preparation, feature extraction, training of the acoustic models, graph compilation from the architecture's components, and decoding on a new test set. Before showing the results of the baseline system, I want to highlight the data that I used as the input and describe where it originates from.

2.3.1 Description of the Pipeline

The framework I use to train the ASR system is the Kaldi Speech Recognition Toolkit, further referred to as *Kaldi*. Complete *recipes* (pipelines) are made available by the community or researchers to be used in Kaldi. See Section 3.2.1 to learn more about Kaldi.

The pipeline that I am working with is based on the *wsj* recipe which was initially created for speech recognition trained on the Wall Street Journal (WSJ) corpus. The model was adapted for the ArchiMob corpus of Swiss German (see 3.1.1) and later on further improved for experiments on acoustic modelling in Iuliia Nigmatulina's Master's thesis [Nigmatulina, 2020]. In a second Master's thesis by Kew [2020], Tannon Kew investigated language modelling techniques for the same pipeline. In this first step, I am trying to replicate the best performing model with the best performing resources from those two theses, and explain the pipeline and the underlying processes.

The pipeline consists of two main bash scripts which represent a) preliminary Kaldi data preparation steps, feature extraction and the training of the acoustic model (`run.sh`), and b) the decoding step given a new set of audio files to be transcribed

(`compile_decode.sh`). This includes the graph compilation from the architecture's components and evaluation of the given set.

2.3.1.1 run.sh: Data Preparation

The pipeline to train the ASR system in Kaldi starts with the script `run.sh`. It integrates the steps for data preparation, feature extraction and the training of acoustic models. The arguments that the script takes include the training file in CSV format, the folder containing the audio files, the pronunciation lexicon, the transcription type and a folder to save the outputs to.

archimob/prep_archimob_training_files.sh
First, the data preparation steps are performed which are needed for the Kaldi recipe to train the acoustic models. Therefore, within `run.sh`, the script `archimob/prep_archimob_training_files.sh` is called. It collects and prepares the data described below and stores it in the output folder under `initial_data`.

From the CSV file, the transcriptions together with the corresponding audio file names are extracted. Based on the transcription type, this is the transcription in Dieth orthography or the normalised version of it.[4] Also, the utterances belonging to at least one of the invalid categories (`anonymity`, `speech_in_speech`, `missing_audio` and `no_relevant_speech`) are filtered out in this step.

Secondary files are then created by calling the Python script `archimob/create_secondary_files.py`. This includes mappings of the utterance IDs and their corresponding locations of the audio files (`wav.scp`), mappings of the speaker IDs and the utterance IDs (`spk2utt`) and vice-versa, mappings from utterance IDs to the corresponding speaker IDs (`utt2spk`).

There is an option to create a simple pronunciation lexicon at this step. It is generated by a concatenation of the word's graphemes (one-to-one grapheme to phone mapping) and some grapheme clusters that represent a single phone (e.g. *ch* or *tsch*) from the transcriptions. However, this is not needed for my experiments, since I am working with a pre-defined pronunciation lexicon.

In further steps, the non-silence phones are extracted from the lexicon and also files for the list of non-speech symbols like silence phones, optional silence and extra questions (for tree building) are created. The transcriptions include non-speech events such as `<SPOKEN_NOISE>` for language disfluencies (e.g. stutters) and `<SIL_WORD>` for pauses in the recordings to distinguish these sounds from speech

[4]For the experiments in this work, the transcription type is always the normalised version.

units.

utils/prepare_lang.sh
In the second data preparation step, the script `utils/prepare_lang.sh` stores various data files in the `initial_data` folder used for training and decoding. This - among others - includes mappings between phonemes and corresponding states, between words and the phoneme representations, and the pronunciation lexicon together with a second lexicon showing the probabilities of each pronunciation variant. These probabilities are not relevant in our case since there is not more than one variant per word in the lexicon for Zurich dialect. Additionally, the resources in the folder mentioned above are checked for validity with various Perl scripts.

2.3.1.2 run.sh: Feature Extraction

In this step, the MFCC features are extracted from the audio files with a sample frequency of 8 kHz (`steps/make_mfcc.sh`). The resulting feature files contain matrices in Kaldi format, where the dimension is equal to the length of the audio file in 10ms intervals.[5] Furthermore, statistics for Cepstral Mean and Variance Normalisation (CMVN) are computed on a per-speaker basis (`steps/compute_cmvn_stats.sh`).

2.3.1.3 run.sh: Acoustic Model Training

Now that all the data is prepared and the features are extracted, the main part of the `run.sh` script consists of training the acoustic model(s). A total of six AMs are trained subsequently and each time, make use of the alignments between the phonemes and the acoustic signal of the respective predecessor model (`steps/align_si.sh`). There are four Gaussian Mixture models (GMM), one trained on monophones (`mono`) and three on triphones (`tri`, `tri_lda` and `tri_mmi`), and two neural network (NN) based models (`nnet2` and `nnet_disc`):

mono GMM with context-independent monophone training (e.g. /o/)

tri GMM with context-dependent triphone training (e.g. /f-o+r/)

tri_lda triphone GMM with additional LDA and MLLT feature transformation

tri_mmi triphone GMM with LDA feature transformation and discriminative boosted MMI training

nnet2 neural network model with p-norm activation

[5]The audio signal is typically divided into frames of 25ms shifted by 10ms each time.

nnet_disc discriminative neural network model

The first model (`mono`) is trained from a flat start, whereas the following models build up on the phoneme-signal alignments of the previous model. The alignments of the `tri_mmi` model also serve as a basis for the final model (`ivector`) which yielded the best results in Nigmatulina [2020]. It is based on a time-delay neural network (TDNN), a feed-forward neural network which is used to model long-term temporal dependencies. To better model these longer sequences, the time-delay design allows the network to discover acoustic-phonetic features and temporal relationships between them [Waibel et al., 1989].

Additionally, the data for the TDNN model is augmented with a technique called *data perturbation*. It operates on the audio-level and produces copies of the original acoustic data through modifications on the acoustic signal [Ko et al., 2015]. Two of the data perturbation types are used for this model. In *speed perturbation*, the data is reproduced by changing the speed of the audio with factors of 0.9, 1.0 and 1.1, generating three versions of the original signal. The data is further modified by applying *volume perturbation* which adds more variability to the audio volume (factor ranges from 0.125 to 2 of the original signal). This allows for a better generalisation and also helps when using *iVectors*.

The concept of iVectors was originally introduced in the task of speaker recognition / verification to extract information about the speaker or acoustic environment. However, this technique also proved useful within ASR for speaker and accent adaptation [Karafiát et al., 2011]. The approach of iVectors is able to reduce large-dimensional sequential input data to a low-dimensional fixed length feature vector without losing too much relevant information.

ivector TDNN model with data perturbation and iVector features

All acoustic model trainings are initiated by executing the corresponding Shell script which can be controlled via the `run.sh` script. Having trained the acoustic models, the next step is the compilation of the graph from the system's components followed by the decoding of the development set and evaluation of the model's performance.

2.3.1.4 compile_decode.sh: Graph Compilation

The compilation of the graph is performed in the second main script `compile_decode.sh`. Based on the acoustic model used for the decoding, the script is named differently:

compile_decode_gmm.sh Used to decode with the four GMM based models `mono`, `tri`, `tri_lda` and `tri_mmi`

compile_decode_nnet.sh Used to decode with the NN based models `nnet2`, `nnet_disc`

uzh/decode_tdnn.sh Used to decode with the TDNN model `ivector`

In the decoding step, the system's components - acoustic model, pronunciation lexicon and language model - are compiled to a fully expanded decoding graph by combining them.

First, the contents of the lexicon are pre-processed and stored in a temporary folder. The lexicon used for the training of the acoustic model does not have to be the same as the one used for the decoding. In this setup, however, it is the same lexicon used for both.

Then, the FST representation of the input LM is created. Together with the AM and the PL, the final decoding graph is compiled (`utils/mkgraph.sh`).

2.3.1.5 compile_decode.sh: Decoding and Evaluation

Before decoding the development set, the same data preparation steps are performed as in Sections 2.3.1.1 and 2.3.1.2: pre-processing of the CSV file containing the transcriptions and corresponding utterance IDs, creation of secondary files and extraction of MFCC and CMVN features. For the TDNN model, the iVectors are additionally extracted from the development set.

The decoding graph compiled in the last step is then used to generate the hypothesis transcriptions of the development set. Based on the language model weight (ranges from 7 to 17 in steps of 1) and the word insertion penalty (0.0, 0.5 and 1.0) defined as parameters of the script, the decoding is done for all possible combinations. The best (lowest) WER and CER, together with the corresponding scoring options (values for LMWT and WIP) are stored in separate files. These values can be used as input parameters when decoding new audio files.

2.3.2 Input Data

Before looking at the results, I want to present the data that I used as the inputs for the system. These resources were elaborated in the Master's theses by Nigmatulina [2020] and Kew [2020] who worked with the same data set and system. Therefore,

some of the files are already in a pre-processed format. The following list shows the files that have been pre-processed already and where they originated from:

- **Audio files, transcriptions and alignment**: The original data from the ArchiMob corpus consisted of 34 video recordings of interviews with a duration of approximately 1-2h each (see Section 3.1.1). To get more suitable data for the alignment during acoustic modelling, the long recordings are divided into smaller chunks. Therefore, the audio is extracted from the videos as WAV format (`archimob/extract_audio.sh`). Then, all the information relevant for ASR from the original transcriptions (in EXMERaLDA [EXB] format) are compiled in a single CSV file (`archimob/process_exmaralda_xml.py`). The much shorter speech utterances with a duration of 4-8s each are derived from the timestamp information in the EXB files together with the corresponding extracted WAV file chunks.

- **Data split into training, development and testing set**: The data sets for training the speech recogniser, fine-tuning the model parameters and evaluating the system's performance are already available as separate CSV files. The distribution of these sets was prepared by Samardžić et al. [2016] for the evaluation of part-of-speech tagging performance on the ArchiMob corpus. In order to compare the results with the models from the previous Master's theses, I use the same data sets. Table 4 shows the distribution of speech utterances in each data set, including the separation in total number of utterances and the number of which are valid and therefore used for the experiments.

Data Set	Total	Valid
Training	78,303	67,693
Development	2,266	1,710
Testing	1,870	1,486
Total	82,439	70,889

Table 4: Distribution of the speech utterance counts per data set. Only utterances from the category *Valid* are used in the experiments.

The data sets in CSV format mentioned above are needed as the input for Kaldi, which expects one utterance per line. Each utterance has to be defined with a unique ID that represents the name of the corresponding audio file. There is a total of nine columns in the CSV files:

utt_id unique utterance ID that consists of the speaker ID and the utterance ID which indicate the corresponding audio file

transcription human annotated dialectal transcription following the Dieth orthography

normalized automatically generated normalised transcription

speaker_id speaker ID which is unique for each speaker

audio_id name of the original audio file

anonymity whether or not the utterance contains anonymised personal information (marked by the first letter followed by three asterisks, e.g. *herr e****)

speech_in_speech whether or not the utterance overlaps with another utterance according to the timestamps in the EXB file

missing_audio whether or not the utterance is missing the corresponding audio file

no_relevant_speech whether or not the utterance is missing the transcription, i.e. is empty

Utterances that have at least one of the last four columns with binary values set to "1" are excluded from further processing in Kaldi. The distribution of valid utterances is also shown in Table 4.

An example of a few selected utterances from the CSV file of the training set is shown in Figure 2.

utt_id	transcription	normalized	speaker_id	audio_id	anonymity	speech_in_speech	missing_audio	no_relevant_speech
1008_B-1008-1244	gottseidank isch es fertig	gottseidank ist es fertig	1008_B	d1008-	0	0	0	0
1008_B-1008-1362	as si a de front schtönd	dass sie an der front stehen	1008_B	d1008-	0	0	0	0
1008_B-1008-1393	i der erschte woche	in der ersten wochen	1008_B	d1008-	0	1	1	0
1008_B-1008-1513	hed aber au e name gchaa dä isch ine namentlech	hat aber auch einen namen gehabt dann ist ihnen namentlich	1008_B	d1008-T1425	0	0	0	0
1240_R-1240-0357	üsi grenze guet vertäidiget	unsere grenze gut verteidigt	1240_R	d1240-T349	0	0	0	0
1270_M-1270-0009	nur für schpezielli fäl gsii	nur für spezielle fälle gewesen	1270_M	d1270-T5	0	0	0	0
1270_M-1270-0023	bim <SPOKEN_NOISE> am rand vo walisele gäge dübedorff	beim <SPOKEN_NOISE> am rand von wallisellen gegen dübendorf	1270_M	d1270-T24	0	0	0	0

Figure 2: Example of a few selected utterances in the CSV file which is used as an input for Kaldi

- **Language model**: The LM used for my experiments was elaborated by Kew [2020] in his Master's thesis. It is a statistical 5-gram LM with interpolated Kneser-Ney smoothing [Chen and Goodman, 1999] built with the open-source toolkit MITLM [Hsu and Glass, 2008]. The data that it was trained with

consists of the 76k text utterances from the training set of the ArchiMob corpus and additional out-of-domain data to get a total of 80k utterances. The LM comes as a single file in ARPA format and its size is approximately 55MB (see Section 3.1.3).

- **Pronunciation lexicon**: The PL used for the baseline model consists of the Zurich Swiss German pronunciations from the *Swiss German Dictionary* (see Section 3.1.2). The Zurich dialect is also the most represented dialect in the ArchiMob corpus.[6] In experiments with different pronunciation lexicon variants, this setting showed the best results in Kew [2020].

2.3.3 Results

Using the input data and methods described above I train a total of seven models and evaluate their outputs. The time it takes to train the acoustic models is different for each one of them. An overview of the models, the training duration and their performance on the development set is shown in Table 5.

Acoustic Model	Training Time (in s)	WER (in %)	CER (in %)
mono	1,010	77.58	53.66
tri	555	58.91	37.19
tri_lda	626	56.66	34.96
tri_mmi	6,676	57.78	42.91
nnet2	45,111	48.72	29.54
nnet_disc	9,294	50.02	34.07
ivector	56,262	36.53	21.71

Table 5: Overview of the ASR models trained and decoded with the baseline pronunciation lexicon on the development set.

The results on the development set using the best scoring parameters are reported by the WER and CER. The scores of the models are getting better each time by learning from the previous alignments and by using a more advanced learning technique. There is a noticeable difference from using context-independent monophones (`mono`) to train the AM compared to using context-dependent triphones (`tri`). The WER improves from 77.58% to 58.91%.

[6] Approximately 28% of ArchiMob's data accounts for Swiss German from Zurich, followed by Lucerne (14%), Bern and Basel (12% each).

There are prominent improvements in the scores from the GMM-based trainings to the NN models. From the best performing GMM (`tri_lda`) to its counterpart in the NN models (`nnet2`), the WER decreases from 56.66% to 48.72%. The same phenomenon can be observed when going from the NN models to the TDNN model using iVectors (`ivector`), which yielded the best result over all models with a WER of 36.53%.

One thing that I find striking is that the scores for the models using discriminative training (`tri_mmi` and `nnet_disc`) are slightly worse than those of the corresponding previously trained models that they rely on. This is something to keep an eye on in my experiments on the PL.

3 Materials

The performance of ASR systems is - among others - dependent on the amount and quality of the language data. This applies to the data of all three model components within an ASR system. The training of the AM requires transcribed speech data, meaning audio files annotated with the corresponding transcription. On the other hand, the LM is trained with a corpus containing written language data whose task it is to express typical word sequences within the language. Finally, the PM is dependent on a representative pronunciation lexicon. The quality of the data is even more important for low-resource language like Swiss German since there are not vast amounts of resources available as e.g. in English. But it is not only the data that is important, but also the system in which it is used. Together with the training data, the system's parameters and training methods define the quality of the resulting model. In this chapter, I want to describe the linguistic data (Section 3.1) and tools (Section 3.2) that I used for the experiments during the project.

3.1 Data

On the data side, I particularly want to highlight the *ArchiMob Corpus of Spoken Swiss German* used for the training and testing of the model, and the *Swiss German Dictionary* which was used as a basis for the pronunciation lexicon. Additionally, the *Language Model* elaborated in previous work is also described in this section.

3.1.1 ArchiMob Corpus of Spoken Swiss German

The ArchiMob Corpus of Spoken Swiss German [Samardžić et al., 2016] is the main resource where the data for the training of the ASR systems in my experiments comes from. Its first version was released in 2016 and it was the first multi-dialectal corpus of transcribed spoken language for Swiss German. The intention was to represent continuous speech, so the words appear as they are actually used in texts, as opposed to existing sources of Swiss German data, which consist of isolated word types. The

corpus consists of approximately 70 hours of manually transcribed spoken Swiss German. It is available under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License and the annotated transcriptions can be downloaded in XML format.

The data for the corpus originates from a project by the *Archimob* (Archives de la mobilisation) association[1]. From 1999 to 2001, the association conducted a total of 555 interviews with contemporary witnesses of the Second World War in Switzerland, making it the largest oral history project ever conducted in Switzerland. The individual recordings are between 1h and 2h long. The interviewed persons are from all linguistic regions of Switzerland and represent both genders, different social backgrounds and different political views. 300 of the 555 video recordings are in Swiss German, of which 43 were selected to be included and transcribed in the ArchiMob corpus. This selection was done on the basis of the linguistic representativeness of the speakers (speakers with no language contact are considered the most representative) and the sound quality of the recordings.

3.1.1.1 Transcription of the ArchiMob Corpus

The transcription of the documents was done in four phases by five transcribers. The variation of transcription guidelines across the phases and the use of different tools led to inconsistencies in the transcribed texts. Despite the aims to correct these issues, the transcriptions are not perfect and still contain inconsistencies which is not beneficial in terms of training an ASR system on this data. In the first phase (2006–2012), 16 documents were transcribed without the use of a specific tool. During the second phase (2011–2013), 7 documents were transcribed using the tool FOLKER [Schmidt and Schütte, 2010]. For the third phase (2015), the transcription tool was changed to EXMARaLDA [Schmidt, 2012] and 11 documents were transcribed. The same tool was also used for the fourth and final phase (2016–2017), in which the last 9 documents were transcribed. These final documents were released in a second version of the ArchiMob corpus in 2019 [Scherrer et al., 2019a]; [Scherrer et al., 2019b].

Having the transcriptions, they were divided into utterances with a length of 4-8 seconds. These smaller units can be analysed more easily and serve as more comprehensive training data for an ASR system. The utterances are additionally annotated with a speaker ID. The alignment between the transcribed utterance and the sound source was done by the transcribers directly within the tools FOLKER and

[1]More information at: `http://www.archimob.ch`

EXMARALDA. Since no such tool was used for the transcriptions in the first phase, the alignment had to be done in retrospect. This was approached by automatically aligning the transcriptions with the sound source at the level of words using the online tool WebMAUS [Kisler et al., 2012]. The resulting alignment was then joined into larger units and imported into EXMARaLDA to perform manual corrections.

Since there is no orthographic standard for Swiss German, the texts were transcribed according to the rules proposed by Dieth [1986]. Eugen Dieth's aim to represent a phonetic transcription of Swiss German dialects (see more about Dieth's orthopraghy in Section 2.2.2.1). There are different ways of how Dieth's system can be implemented depending on how close the phonetic qualities are reflected. This implementation changed over the transcription phases, so that more fine-grained phonetic distinctions were made in the first phase compared to the later phases (e.g. distinction between open and closed vowels; phase 1: *èèr* vs. phase 3: *er* [engl. "he"]). A manual inspection of the transcriptions showed that there were more inconsistencies, not only between the transcription phases, but also the interpretation of the rules by the different transcribers and even within a single text transcribed by the same expert. This - among other reasons - led to the decision to add a separate annotation layer with normalised word tokens, which is what I use as the training data of the ASR systems in this work.

3.1.1.2 Normalisation of the ArchiMob Corpus

To understand the normalised version of the transcription, it is important to note that it cannot be regarded as a translation into standard German. This also represents a discrepancy to the PL that I use as a baseline for my experiments. The PL only contains standard German words (see Section 3.1.2). The normalisation is a word-by-word annotation of lexical identity whose exact forms can be seen as arbitrary. A real translation into standard German would require the Swiss German texts to apply changes in syntax and to completely replace lexical items. Instead, variations of a specific word are combined in a normalised version. These variations are caused by two phenomena. First, words may be pronounced differently due to the dialectal variation of the different regions. Second, words may be written differently due to intra-speaker variation and inconsistencies related to the transcribers.

Table 6 shows examples of the Swiss German transcription and its normalised version from the ArchiMob corpus. In these examples, a real translation into standard German would be formulated differently and in some cases use different words, e.g. the normalised *schaffen* instead of the standard German *arbeiten*. There are also

discrepancies between word boundaries due to cliticisation in Swiss German, e.g. *innere* combines the preposition *in* and the indefinite article. In the normalised version, this is represented by combining the two standard German words with an underscore: *in_einer*.

Transcription (in Dieth orthography)	**Normalisation**
was sind daas für schtelle gsii wo si ghaa händ	was sind das für stellen gewesen wo sie gehabt haben
innere famili gsii	in_einer familie gewesen
und de het er amig gschpaart het sich gsait me het sich nie es möschtli	und den hat er allweg gespart hat sich gesagt man hat sich nie ein möstlein
haig er mit dene und d italjäner hend òü fräüd ghaa und die haiged gfäschtet	habe er mit denen und die italiener haben auch freude gehabt und die haben gefestet
im april sibezgi da han ich nümme chöne schaffe	im april siebzig da habe ich nicht können schaffen
isch das lang gange	ist das lange gegangen

Table 6: Examples of utterances from the ArchiMob corpus where the normalisation of the transcription does not depict a translation into standard German.

For the normalisation of the transcriptions, a semi-automatic approach was used. Again, the output of this approach is not perfectly reliable. In some cases, this resulted in incorrectly normalised texts, which introduces inconsistencies in the training of the ASR system. First, a small set consisting of six documents was normalised manually by three annotators. This was done according to a guideline in order to preserve consistency. In a second step, an automatic normalisation tool was trained on these documents. The use of a combination of memory-based word-by-word translation, character-level machine translation (MT) and language modelling yielded an average accuracy of 77.28%. The procedure of the automatic normalisation is further described in Samardžić et al. [2015]. This was later improved to 90.46% by using training the character-based MT system on pairs of segments instead of pairs of words [Scherrer and Ljubešić, 2016]. Having trained the automatic normalisation system, the documents were pre-processed in small sets in a bootstrapping approach. This way, the errors in the normalised documents could be corrected manually in a more efficient way, which was then used to improve the system for the next round.

In further experiments, the task of automatic part-of-speech (POS) tagging was tested on the corpus. A subset of 1742 utterances was tagged manually according

to the Stuttgart-Tübingen-Tagset (STTS). This is the same subset that I use as the test set for my experiments. Also achieved through a bootstrapping approach, the best accuracy score of the POS tagger was increased from 73.09% to 90.09% after four rounds.

3.1.2 Swiss German Dictionary

The main focus of my thesis is the pronunciation lexicon (PL). It consists of a list of words aligned with a representation of how the word is pronounced. This phonetic representation is expressed as a sequence of phones. See Section 2.2 to learn more about the role of the PL in an ASR system, what options there are to construct a PL and how it can be extended to account for words that are not yet listed in it.

The PL I use as a basis for my experiments is the Swiss German Dictionary [Schmidt et al., 2020]. It was compiled by a collaboration of the University of Zurich's Research Priority Programm (URPP) Language and Space Lab and Swisscom, the largest telecommunications provider in Switzerland. This dictionary is also available upon request under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

The dictionary contains a total of 11,238 standard German words. Compared to large-scale dictionaries[2] usually used in ASR, this is a rather small number of words. Each word is annotated with its pronunciations from six variants of Swiss German, namely Zurich, Basel, Bern, St.Gallen, Stans and Visp. Figure 3 shows the location of the regions in the map of Switzerland.

The pronunciations are represented by a modified version of SAMPA (see Section 2.2.2.1). Additionally, Swiss German spontaneous writings are paired with the standard German word and its Swiss German pronunciations. These should represent how the words may be used in written communication, e.g. in text messages, by native speakers. There are two versions of the dictionary consisting of two differently sized sets of SAMPA characters. The extended set uses 137 phones and allows for a detailed representation of the diverse pronunciation in Swiss German. The reduced set only uses 59 phones and is intended to be used for ASR due to its easier computation. This is also the version that I used as the baseline for my experiments. The phone reduction is mainly attained by splitting up SAMPA-characters that express a combination of two phonemes. This includes - among others - diphthongs, /AI/ to /A I/, or affricates, /ks/ to /k s/.

[2]Three commonly used PLs for English ASR systems are CELEX, CMUdict and PRONLEX. These contain pronunciations for 160,000, 134,000 and 90,000 wordforms respectively.

Figure 3: Six Swiss German variants that were chosen to be included in the Swiss German Dictionary

3.1.2.1 Construction of the Swiss German Dictionary

The regional language variation in Switzerland is continuous, meaning the Swiss German variants first had to be discretised before deciding what dialects should be used in the dictionary. To identify the different varieties, Schmidt et al. [2020] used a dialectometric analysis, which is described in more detail in Scherrer and Stöckle [2016]. It is based on lexical, phonological and morphological data of Swiss German. They defined a set of ten linguistic varieties, which was then reduced to the six variants mentioned above due to time restraints and substantial overlap between some of the variants. The chosen variants are represented by major cities in that region and can be seen in Figure 3.

The manual annotation was done by native speakers of the Zurich and Visp variants. For these two dialects, they mainly used their knowledge as native speakers. For the other four dialects and for further analysis, they consulted dialect specific grammars and lexica, or listened to recordings from the ArchiMob corpus [Samardžić et al., 2016]. For each standard German word, they manually annotated its phonetic representation in the six dialects in SAMPA notation. The fact that no native speakers were used to annotate the pronunciations for the regions of Basel, Bern, St.Gallen and Stans may have led to more errors in the phonetic representations of these dialects. In terms of using them in an ASR system, they could be less beneficial than the Zurich and Visp variants.

In order to complement the annotation for the Swiss German spontaneous writing,

Schmidt et al. [2020] compiled a larger subset of 9000 of the 11,238 standard German words. These subsets were manually annotated by native speakers for the variants of Zurich and Visp. They did this by only looking at the standard German word and without consulting the SAMPA representations created before. This way, biases concerning the phonetics as well as the meaning of the word could be avoided. For the other four Swiss German variants, smaller subsets of 600 words each were annotated manually by the same annotators who are not native speakers of that dialect. For these dialects, the annotators relied on the corresponding SAMPA representations created before. The Swiss German writings for the remaining standard German words were then generated by using automatic methods.

The intention of this automatic process was to generate the Swiss German writing from the phonetic SAMPA representation that was previously manually annotated. This task can be viewed as a sequence-to-sequence (seq2seq) problem, where the input is a sequence of phonemes and the output is a sequence of graphemes. For this phoneme-to-grapheme (P2G) task, they used a Transformer-based model. This decision was based on the premise that such models have shown to perform well for seq2seq tasks.

3.1.2.2 Benefits of P2G and G2P Conversion

In order to evaluate the quality of the P2G model, a test set consisting of 1000 words was looked at by the annotators for the varieties Zurich and Visp. The test set for the other varieties only contained 200 words each. For each word, the P2G model predicted five candidates of Swiss German writings which were then rated either as correct or incorrect. The first candidates were correct in 94.6% of the cases for Visp dialect, and 87.3% for Zurich, respectively. The results were lower for the other dialects since their SAMPA notations included characters that the model was not familiar with since it was trained only on the SAMPA representations of the Zurich and Visp dialects. After that discovery, they added a set of 600 manually annotated Swiss German writings for these dialects to improve the model. Unfortunately, no further results are reported after the addition of the manual annotations.

More interesting, particularly for the needs of an ASR system, is the vice-versa conversion, namely Grapheme-to-Phoneme (G2P). The quality of an ASR system is highly dependent on the quality of the PL (see Section 2.2). The same architecture could be used to train the G2P model to extend the PL with words not yet included. The only difference is the change of the input and output sequences. Having the trained model, Schmidt et al. [2020] were able to generate the phonetic represen-

tation for a given word that was not yet integrated in the dictionary. This can be very useful and efficient for OOV words. The results from 2,412 testing samples of Zurich and Visp dialect were then compared with those of a joint-sequence G2P model named *Sequitur* [Bisani and Ney, 2008]. The model by Schmidt et al. [2020] outperforms the predictions from the Sequitur model.[3]

3.1.3 Language Model

The language model (LM) computes the probability of a given sequence of words in a language. It is a vital part of an ASR system and, together with the AM and PL, is used to find the best possible output transcription for a given acoustic source.

The LM used for my experiments was elaborated by Kew [2020] in his Master's thesis. It is a statistical 5-gram LM and was trained with interpolated Kneser-Ney smoothing [Chen and Goodman, 1999]. The open-source toolkit MITLM [Hsu and Glass, 2008] was used to train the model. Results showed that higher order N-gram LMs performed better when modelling normalised Swiss German. In his experiments, he compared 3-gram with 5-gram LMs within different settings.

The training data for the LM consists of the 76k normalised text utterances from the training set of the ArchiMob corpus, and additional out-of-domain data to get a total of 80k utterances. In experiments, where the training data was gradually increased with 10k out-of-domain data, the language model with a total of 80k utterances yielded the lowest perplexity scores. The performance of LMs is usually measured by the perplexity score [Goodman, 2001]. It indicates the amount by which the LM reduces the uncertainty about the next word. The LM is a single file and is encoded in the widely used ARPA format. It has a file size of approximately 55MB.

3.2 Tools

In this section, I describe the *Kaldi Speech Recognition Toolkit* used to train, decode and evaluate the ASR model. In addition, the grapheme-to-phoneme toolkit *g2p-seq2seq* is introduced. This was used to extend the initial pronunciation lexicon.

[3] Out of the 2,412 samples, of which 1,294 are from Zurich dialect and 825 from Visp, their model predicts 978 annotations correctly, compared to 795 by the Sequitur model.

3.2.1 Kaldi Speech Recognition Toolkit

The Kaldi Speech Recognition Toolkit [Povey et al., 2011] is the framework used to train the speech recognition models in this work. It is licensed under the Apache License v2.0. This allows the code to be publicly available and permits to make modifications and re-releases (also for commercial use). Its idea was originally developed in a workshop by the Johns Hopkins University in 2009. Two years later, after subsequent meetings and further development, the code was released to the public. A general-purpose speech toolkit with an open license was created.

The goal of the Kaldi project is to make complete *recipes* (pipelines) for building speech recognition systems available. These recipes rely on the standard ASR architecture, consisting of an acoustic model, pronunciation lexicon and language model, as opposed to end-to-end architectures, where the acoustic signal is directly mapped to the output text and does not need these intermediate models to build the whole.

Kaldi is mostly written in `C++`. The individual steps in a Kaldi recipe are executed with Bash scripts from the command line. It also integrates some data processing scripts written in Perl and Python. For the compilation of the architecture's components and the decoding phase, Kaldi relies on Finite-State Transducers (FSTs) [Mohri et al., 2008]. Therefore, it makes use of OpenFst[4], which is an open-source library for Weighted FST (WFST) operations, such as constructing, combining, optimising and searching WFSTs.

The Kaldi framework has code and scripts for many state-of-the-art acoustic modelling techniques, including different feature and model-space transformations like Linear Discriminant Analysis (LDA), Maximum Mutual Information (MMI), Maximum Likelihood Linear Transform (MLLT), but also speaker-adaptive transformations like feature-space Maximum Likelihood Linear Regression (fMLLR) and iVectors. The trainings can be performed with Gaussian Mixture Models (GMM), Subspace GMM (SGMM) and neural network based models which can be trained on top of the GMM models.

3.2.2 Sequence-to-Sequence G2P Toolkit

Building a quality PL for an ASR system is expensive and time-consuming. Additionally, it is almost impossible to feature all words and the corresponding pronunciations that appear in spoken utterances. This leaves the system with OOV words

[4] `http://www.openfst.org/twiki/bin/view/FST/WebHome`

which can be a problem. One way to tackle this problem is to extend the PL with automatic processes such as G2P conversion. It is the process of converting a sequence of letters into a sequence of phones. More information about G2P conversion can be found in Section 2.2.3.1.

The toolkit I used to automatically generate the pronunciations for the words not yet depicted in the baseline PL is *g2p-seq2seq*[5]. This G2P conversion tool is built with Python and is based on Transformers. The Transformer architecture relies entirely on the attention mechanism to model the dependencies from the data [Vaswani et al., 2017]. The toolkit Tensor2Tensor[6] is used to implement the Transformer model which is based on the TensorFlow[7] framework. g2p-seq2seq is licensed under the Apache License v2.0 and is therefore freely available.

[5] `https://github.com/cmusphinx/g2p-seq2seq`

[6] `https://ai.googleblog.com/2017/06/accelerating-deep-learning-research.html`

[7] `https://www.tensorflow.org/tutorials`

4 Methods

This chapter describes the methods that I used in the experiments on the PL and what steps I needed to take. In Section 4.1, the preparation of the PL is discussed. This includes the steps of getting from the original format to the file used in the ASR system. It represents the PL I used for the baseline system. Section 4.2 then deals with the extension of the PL, which is done by G2P conversion. Finally, in Section 4.3, I explain how I set up the experiments for the comparison of using the baseline and extended PL within two phases of the ASR system, the training of the AM and the decoding phase.

4.1 Preparation of the Pronunciation Lexicon

Before using a PL in an ASR system, it needs to be in a specific format that the system can work with. In Kaldi, this is a text file consisting of words and their pronunciations represented as a sequence of phones. Therefore, each line in the text file contains a word and the phones of its pronunciation, all separated by a blank space. This looks as follows:

```
<word1> <phone1> <phone2> ...
<word2> <phone1> <phone2> ...
```

Here are examples from the final PL that I used. It shows the standard German word and the phones of the Zurich Swiss German pronunciation represented in SAMPA format. I build this format by converting the entries in the Swiss German Dictionary (see Section 3.1.2).

```
dorfplatz d o r f p l a ts
gegen g { g @
mittagstisch m i t a k s t I S
```

The original dictionary is separated into six CSV files. This separation is caused, on the one hand, by the origin of the data and, on the other hand, by different phases of the collaboration between the URPP Language and Space Lab and Swisscom. Each CSV file contains a list of standard German words, the corresponding pronunciations of the six dialects in an adapted SAMPA format, and (for the majority of the entries) samples of the Swiss German writing. The number of words in each file ranges from 960 to 2846 and adds up to a total of 11,207 words.

Each one of the six CSV files comes in two variants, one using a big set of 137 phonemes and one using a smaller set of 59 phonemes to represent the pronunciations (see Section 3.1.2). I only use the files with the reduced phoneme set to build the PL. It is more beneficial for the ASR system when there is less variation to choose from in the phoneme set. This makes it easier to compute.

In Table 7, we can see some entries from the Swiss German Dictionary before applying any processing. Due to lack of space it only shows the pronunciation for the Zurich dialect.

Standard German Word	Zurich SAMPA	Other Dialects	Swiss German Writing
kälte	x e l t i	...	
Sonnenschein	s u n @ S i:	...	{'sunneschi', 'sunneschii'}
von dem_LXA	f o m	...	vom
abbrechen_MRA	a p p r { x @	...	ZH (Zürich-Seebach)_abbräche
an den	a _ d E	...	{'a', 'an'}
e-banking	i: b E N k i N	...	
abends	a m _ a b I k	...	

Table 7: Examples from the Swiss German Dictionary showing the standard German word, the pronunciation in Zurich dialect and the Swiss German writing for each entry.

Merging the files and format normalisation

I start by combining the contents of the six CSV files to have all 11,207 entries in one large file. The Zurich dialect is the most commonly used within the ArchiMob corpus, representing 28% of the data. Therefore, and in order not to have multiple pronunciation variants per word, I only keep the pronunciations of the Zurich dialect. I proceed to remove the unneeded columns for the five other dialects and also the column for the Swiss German writing, which is not of interest for the PL.

Although the files have the extension CSV, the columns are not separated by a comma, but use the tab as a delimiter instead. To make the subsequent normalisa-

Standard German Word	Zurich SAMPA
kälte	x e l t i
sonnenschein	s u n @ S i:
von_dem	f o m
abbrechen	a p p r { x @
an_den	a d E
ebanking	i: b E N k i N
abends	a m a b I k

Table 8: The same entries from Table 7 after being processed for the PL. Changes include lowercasing of words, addition of underscores in words, removal of hyphens in words and removal of underscores in pronunciations.

tion easier to apply, I replace the tabs by a comma. I can then apply the following rules to normalise the format:

- **Normalise whitespace characters**: There are different variants of whitespaces (e.g. non-breaking space) in the document. They need to be normalised and are therefore converted to a standard space.
- **Replace multiple spaces with one**: Some entries contain two or more spaces between the individual phones in the pronunciation. This can lead to errors in the ASR system and they are replaced with one space.
- **Remove spaces before and after delimiter**: There are only two columns left in the list. However, some entries contain a space before or after the comma delimiter. In the PL, this would lead to a different word form or pronunciation, based on the placement in regards to the delimiter. These additional spaces need to be removed.

Word normalisation, manual checks and finalisation

After having normalised the format of the file and removed additional whitespaces, there are some adjustments on the word forms and pronunciations left to do.

- **Remove meta information from words**: Some word forms are followed by a character sequence _LXA or _MRA[1]. These do not add any value to the PL and can be removed.

[1]Unfortunately, the documentation of the Swiss German Dictionary does not mention what these character sequences stand for.

- **Remove underscores from words**: There are still some words left that contain an underscore. This would add noise to the PL these words could not be recognised by the ASR system. Since there are not many and some of them consist of one letter only, I check them manually and remove the underscore or the whole entry based on my finding.
- **Remove hyphen from words**: The same goes for hyphens within words, which mostly originate from suffixes (e.g. *e-banking*). I check these entries manually and decide whether the hyphen or the whole entry can be removed.
- **Combine multi-word entries**: In the standard German writings, there are multi-word entries separated by a space. These are pronounced as one word in Swiss German (e.g. standard German *von dem* vs. Swiss German pronunciation [`f o m`]). Multi-word entries in the PL cannot be realised as separate words or the additional words would be interpreted as phones by the ASR system. Therefore, I replace the space by an underscore, which is also, to some extent, how it is realised in the normalised ArchiMob transcriptions.
- **Remove underscores from pronunciations**: The opposite case, a standard German word being realised as multiple words in the Swiss German pronunciation, is represented with an underscore in the pronunciation (e.g. standard German *abends* vs. Swiss German pronunciation [`a m _ a b I k`]). I remove these underscores since I do not see any advantages for the PL.
- **Lowercase words**: Finally, the reference transcriptions that I use as training data for Kaldi are written in lowercase and this needs to be reflected in the PL too. Therefore, all word forms are converted to lowercase.

I then sort the resulting list of words alphabetically, replace the comma delimiter with a space and save the contents as a text file (`.txt`). The PL can now be used as an input in Kaldi. Out of the 11,207 entries in the original Swiss German Dictionary, only a few entries have been removed and the final PL is left with 11,180 entries. Some example entries after the preparation of the PL can be seen in Table 8.

4.2 Lexicon Extension with G2P

The normalised transcriptions from the ArchiMob corpus used to train the ASR system consist of 29,228 different word forms. Only 7,692 of these words are also present in the PL and have a corresponding pronunciation. This corresponds to a coverage of roughly 26%. In order to increase the coverage of the words occurring

in the training data, I extend the PL by using G2P conversion.

The G2P conversion is done by first training a G2P system with the available words and pronunciations in the PL. Having trained a G2P model, I can then predict the remaining OOV words from the training data not present in the PL. For the training, I use the freely available g2p-seq2seq toolkit (see Section 3.2.2), which is based on the Transformer architecture. The toolkit can be installed using Python and the commands to start a training are performed using the command line.

The data used for the training of the G2P model needs to be in the same format as the PL for Kaldi, i.e. a text file with a word and its pronunciation represented as a phone sequence on each line. This also means that I do not need to change the format and can use the prepared PL directly. The training data, if not defined otherwise, is automatically split into training (85%), development (5%) and testing (10%) set.

As for the hyperparameters of the model, I do not change anything and use the default parameters. The following parameters are defined by default:

`--max_epochs` (Default: 0) The value of this parameter represents the maximum number of training epochs. The default value "0" means that the training stops when no improvement is observed from one epoch to the next. This method is also called validation-based early stopping.

`--num_layers` (Default: 3) This defines the number of layers in the model. By stacking the layers, the model can learn to focus on different combinations of attention from its attention heads. This potentially boosts the prediction quality, however, it also slows down the training.

`--size` (Default: 256) This value defines the size of each model layer.

`--filter_size` (Default: 512) The size of the filter layer in a convolutional layer is defined by this value.

`--num_heads` (Default: 4) This value represents the number of heads in the multi-attention mechanism. The self-attention process is called a *head*. Each head produces an output vector that gets concatenated into a single vector. This way, each head might learn something different, which improves the quality of the predictions.

The trained model can then be used to predict pronunciations for new words. This is done for all words in the training data that are not covered in the PL. The result is an extended PL. It combines the pronunciations from the baseline PL with those

automatically generated by the G2P model. The number of words in the PL has thus increased from 11,207 to 34,830.

The extended PL can now be used to train and evaluate the same AMs in Kaldi as described in Section 2.3.1.3. I train the seven AMs with the same settings but use the extended PL instead. The evaluation is also done on the same development set, so the results can be compared coherently.

4.3 Differences of using the PL in Acoustic Modelling and Decoding

There are two phases within an ASR system where we can make use of the PL, during the training of the AM and in the decoding process. Similar to the work by Jouvet et al. [2012], I want to analyse the impact of using a PL in the corresponding phase. They did this evaluation based on PLs with multiple pronunciation variants generated by G2P converters. The main focus of their analysis was on the different G2P systems. Instead, I want to evaluate the impact on the performance of the ASR system when using two different PLs in the corresponding phases.

To do this, I rely on the two PLs that I compiled in Sections 4.1 and 4.2. The difference between the two PLs is given by the size and, as a result of that, by the coverage of the vocabulary. Thereby, the PL that I created from the entries in the Swiss German Dictionary serves as the *Baseline PL*. The PL that I increased by G2P conversion trained on the word-pronunciation mappings in the baseline PL is further referred to as the *Extended PL*.

The goal is to use the baseline PL in the training of the AM and the extended PL in the decoding process. The results of this scenario are then compared to those of the vice-versa setup, where the extended PL is used to train the AM and the baseline PL to decode, respectively. In a further analysis, this can be compared to the results of using the same PL in both phases, which is the standard setup.

Preparation of the PL in Kaldi

Within the Kaldi pipeline, the PL can be defined separately for both, the AM training (`run.sh`) and the decoding (`compile_decode.sh`). In the data preparation step of the AM training (see Section 2.3.1.1), the PL given as an argument is stored under the folder `initial_data/ling`. It is further processed by the file `utils/prepare_lang.sh` and stored under the folder `data/lang` where the WFST graph for the lexicon is created.

During the data preparation phase before decoding, the PL is stored and processed in another location. The same script (`utils/prepare_lang.sh`) is used to prepare the PL and create the graph in a given output path for the evaluation (in my case `eval/[AM_TYPE]/[PL_TYPE]`) in the folder `tmp/lang`.

Decoding with separate PL

Having trained the AMs for both PLs already in the previous experiments, I can use the other PL to decode the development set. Except from the PL, I do not change any of the parameters, i.e. LM and the set to be decoded are the same. For both scenarios, the decoding and evaluation is done with six different AMs: `mono`, `tri`, `tri_lda`, `tri_mmi`, `nnet2` and `nnet_discriminative`.[2]

[2]Previous experiments were done with seven different AMs. Using two different PLs was not realised for the `ivector` model. The decoding process for the TDNN model using iVectors is slightly different compared to the other AMs and would need further adjustments in the decoding script (`decode_tdnn.sh`).

5 Results and Discussion

In this chapter, I present the results from the experiments performed in Chapter 4 and discuss their outcomes. In Section 5.1, I have a look at the output of the G2P conversion, which is used to extend the PL. Furthermore, I reveal how well the resulting PL performs when being used in the ASR model and compare the scores to those of the baseline PL. In Section 5.2, I show the results of using the two PLs in two different stages of the ASR architecture, in the training of the AM and in the decoding process.

As for the evaluation of the ASR system's performance, I use the same development set as described and presented in Section 2.3.2 in Table 4. This makes the resulting scores more coherent and comparable. To represent the quality of the model, I use the WER and CER scores as an evaluation metric (see Section 2.1.2).

5.1 Extended Pronunciation Lexicon

In order to cover more words from the training data, I extended the PL by training a G2P model on the entries in the baseline PL (see Section 4.2). The model was then used to predict the pronunciations for the words from the training data missing in the PL. The resulting word-pronunciation mappings were combined with those from the baseline PL to form an extended PL. Having this extended PL, I can use it as an input for the ASR system and see how the results compare to models using the baseline PL.

In the following, I show the results of the G2P conversion and have a look at some example pronunciations generated by the G2P model. Also, I present the results from the extended PL in action being used in Kaldi.

5.1.1 Output of the G2P Conversion

Looking at the results from the G2P conversion on a randomly held-out test set in Table 9, the number of correctly predicted pronunciations and the accuracy are not very promising. Only 618 out of the 1,120 words in the test set got the correct phone sequence. This results in an accuracy of a bit more than 55%, which is similar to the findings by Nigmatulina et al. [2020] who used the same data as a basis.

Words in Test Set	Correct	Incorrect	Accuracy
1,120	618	502	55.18%

Table 9: Results from the G2P model evaluated on a held-out test set. It shows the number of words in the test set, how many pronunciations the model predicted correctly and incorrectly, and the accuracy of the model.

Although the results may seem poor, the increased number of words in the extended PL can still be beneficial for use in the ASR system. Table 10 shows some examples from the extended PL of which the pronunciation was generated by the G2P model.

Standard German Word	SAMPA generated by G2P Model
alpkäse ('alpine cheese')	a l p k x E: s
ferienmädchen ('holiday girl')	f e: r i @ m aI t l i
rhetorisch ('rhetorical')	r e t o: r I S
detailgeschäft ('retail shop')	d e t aI j I k S { f t
tagesthema ('topic of the day')	t a g e S t e m a
jazzmusik ('rhetorical')	j a ts m u s I k
rheinau ('Rheinau')	r i: j u:
rheinzone ('Rhine zone')	r i: ts o: n t s o: n

Table 10: Example of words of which the phone sequences have been generated by the G2P model.

The first three pronunciations seem to be alright. The model correctly predicted the phone sequences for the compound words *alpkäse* and *ferienmädchen*. The diminutive suffix in *ferienmädchen*, which changes from standard German *-chen* to Swiss German *-li*, can be sen well. However, for both compound words, the individual words, which it is made up from, are in the baseline PL already. For the third word, *rhetorisch*, there are no derived words in the baseline PL and its pronunciation was predicted correctly.

However, the remaining words in Table 10 show some flaws in the predicted pronunciations. The first three words represent compound words again. Although both parts making up the word *detail-geschäft* are in the baseline PL, the model adds the erroneous phone sequence [`j I`] in between. This is probably due to other words including *detail*, which show this phone sequence in their pronunciation: *detaillierte* - [`d e t aI j I @ r t i`], *detaillierten* - [`d e t aI j I @ r t @`]. In both cases, the phone sequence is correct and is caused by the letters *li*.

The word *tages-thema* is an interesting case. The compound is built by adding a Fugen-'s' to the first word.[1] Since the second word starts with the letter *t* and the combination of *st* usually is pronounced as [`S t`], the model interpreted it as such. However, in this compounding situation, the two letter *s* and *t* are pronounced separately and the phone sequence should therefore be [`t a g e s t e m a`].

Within *jazzmusik* the fragment *jazz* represents a word coming from the English language and should be pronounced as such. Foreign words can be a problem for the G2P conversion, especially when the training data for the G2P model is rather sparse and does not include many of them. This is certainly the case for the Swiss German Dictionary, from which the words in the baseline PL originate.

The next word represents the name of a municipality next to the river Rhine. From the training data in the baseline PL, I cannot derive how the model generated the phone sequence [`j u:`]. This error reveals another major issue for G2P conversion, named entities. In many cases, the pronunciation of named entities does not follow the general pronunciation rules of the corresponding language. This makes it particularly hard for the G2P model to learn and predict pronunciations for named entities.

Last but not least, an example of a word where the G2P model produced a 'gibberish' pronunciation. For the word *rheinzone*, the model simply repeated the phone sequence [`ts o: n`] at the end. This is a common phenomenon also found in machine translation and other seq2seq tasks. The repetition might come from the type of search used on the decoder side.[2] Holtzman et al. [2019] propose a workaround to mitigate this issue within the task of text generation.

After having analysed erroneous pronunciations generated by the G2P model, let's

[1] The Fugen-'s' is a grammatical feature of certain German compounds. The letter 's' is added between the noun stems building the compound. Historically, it marks the genitive case of the first noun. But it also often occurs after nouns which do not actually take an 's' in their genitive cases.

[2] Type of search here means the sampling of items in a sequence when decoding. In the context of G2P conversion, the items are phones.

see how this impacts the coverage of the words in the data sets and the performance of the ASR system using the extended PL.

5.1.2 Extended Pronunciation Lexicon in Action

Through the pronunciation predictions of the G2P model on the words in the training data, the number of words in the extended PL increased from 11,207 to 34,830. This also has an impact on the coverage of the words in the data sets. Table 11 shows a comparison of the coverage of the baseline PL and the extended PL on the different data sets.

Set	Baseline PL			Extended PL		
	Unique words	Words in PL	Coverage	Unique words	Words in PL	Coverage
Train	29,228	10,971	**37.54%**	29,228	29,221	**99.98%**
Dev	2,546	1,587	**62.33%**	2,546	2,101	**82.52%**
Test	2,265	1,484	**65.52%**	2,265	1,919	**84.72%**

Table 11: Coverage of the words in the corresponding sets based on the baseline PL and extended PL.

The coverage by the extended PL is higher for all data sets compared to the baseline PL. The highest increase can be observed in the training set, which only had a coverage of 37.54%. Although the resulting PL was extended with the pronunciation from the words in the training set, the extended PL does not cover all words (99.98%). A few words (7 out of 29,228) are not included because the G2P system uses some pre-processing rules which led to a mismatch between the graphemes in the training data and those predicted by the system.

The increase of the training set's coverage is potentially beneficial for the training of the AM. This way, almost all HMM state sequences of the words can be mapped to the corresponding phone sequences. But also the sets to be decoded show a much higher coverage. For the development set, the coverage increased from 62.33% to 82.52%. This corresponds to an increase of 32% in the number of words to be recognised by the PL. A similar increase can be identified for the test set.

For all data splits, the coverage could be increased massively. But what impact does this have on the performance of the trained AMs within Kaldi? Let's have a look at the results and how they compare to those of the system using the baseline PL. In Table 12, we can see the comparison of the WERs achieved on the development set by the baseline PL and the extended PL using the different AMs.

PL	WER (in %)						
	mono	tri	tri_lda	tri_mmi	nnet2	nnet_disc	ivector
Baseline PL	77.58	58.91	56.66	57.78	48.72	50.02	36.53
Extended PL	76.09	55.94	52.57	49.42	43.97	43.14	**31.86**

Table 12: Comparison of the WER (in %) on the development set using the baseline PL and the extended PL with different AMs.

Table 12 clearly shows that the WERs improved for all AMs using the extended PL. This improvement can be observed even better in Figure 4 where the red line of the extended PL is always below the blue line of the baseline PL.

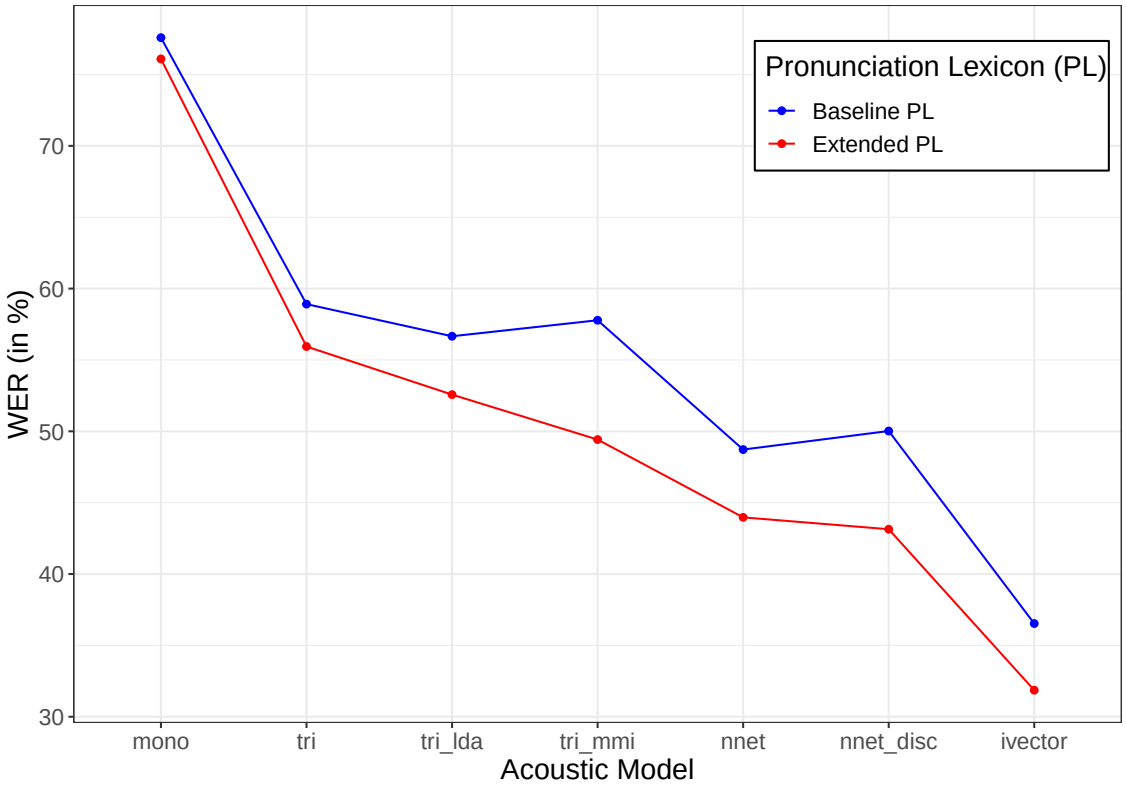

Figure 4: Comparison of the WER (in %) on the development set using the baseline PL and the extended PL.

Within the four GMM based AMs (`mono`, `tri`, `tri_lda` and `tri_mmi`), the performance is slightly getting better and it diverges more and more from that of the baseline PL. The jump from the GMM based to the NN based AMs (`nnet2` and `nnet_disc`) is larger for the baseline PL, although the results for the extended PL are still way better. Finally, there is again a major improvement of the WER by

using the TDNN model with iVectors (`ivector`). The best WER achieved with the extended PL outperforms that of the baseline PL by a relative difference of almost 13% and is at 31.86%. Similar improvements from the the `nnet_disc` to the `ivector` AM are reported by Nigmatulina et al. [2020].

As I already highlighted in Section 2.3.3, the results of the baseline PL show a slight decline in WER for the discriminative AMs (`tri_mmi` and `nnet_disc`) compared to the previously trained model which they rely on. This is not the case when using the extended PL. The WERs improve from each model subsequently. This suggests that discriminatively trained AMs require a PL with a high coverage to outperform the non-discriminative counterpart. If this behaviour originates from training the AM or from the decoding phase is analysed in the next section.

5.2 Impact of PL in Acoustic Modelling and Decoding

The PL can be used within two phases of the ASR system in Kaldi, during the training of the AM and in the decoding process. With the two PLs (baseline PL and extended PL) compiled in this work, I perform experiments to see how the PLs impact the performance of the ASR system when being used in one of these phases. Each PL is used in one of the two phases at a time, whereas the other PL is used in the other phase.

The choice of the PL not only impacts the performance of the resulting models, but also the time it takes to decode a set of new acoustic signals. In Table 13, we can see the effect on the time used to decode when using the baseline or the extended PL.[3] As for the set to be decoded, I use the development set again.

PL used for Decoding	**Decoding Time (in s)**						**Total Time (in s)**
	`mono`	`tri`	`tri_lda`	`tri_mmi`	`nnet2`	`nnet_disc`	
Baseline PL	953	811	711	821	684	734	**4,714**
Extended PL	1,147	1,030	948	1,131	960	1,059	**6,275**
Ratio	**1.20**	**1.27**	**1.33**	**1.38**	**1.40**	**1.44**	**1.34**

Table 13: Comparison of the decoding time needed when using the baseline PL or the extended PL. The bottom row shows the ratio between the time needed by the extended PL and the baseline PL. The decoding was done on the development set.

[3]The decoding time reported in Table 13 includes both, the compilation of the WFST graph and the decoding of the new set.

The time used to decode the development set with each AM is reported in seconds. When using the baseline PL, the decoding process takes approximately 11 to 16 minutes, which results in an average of 13 minutes per AM. Using the extended PL, however, the decoding process takes considerably more time. The durations to decode the same development set with the extended PL range from 16 to 19 minutes, resulting in an average of 17.5 minutes. The last row in Table 13 indicates the ratio between the decoding times of the two PLs. On average, it takes 1.34 times more time to decode the same set of acoustic signals with the extended PL than with the baseline PL. This can be attributed to the size of the extended PL, which contains significantly more word-pronunciation entries.

This increased time used to decode suggests, that in scenarios, where the time used to get the output transcription plays an important role, it might make sense to rely on a smaller PL. However, this is accompanied by a loss of quality as seen in the last section. Next, I present the results on the performance of the ASR system in context of using different PLs for AM training and decoding.

As for the evaluation of the performance, I decoded the development set for each mixed PL setting. Table 14 reports the resulting WERs for both scenarios using each AM training method.

PL Setting		**WER (in %)**					
AM Training	**Decoding**	`mono`	`tri`	`tri_lda`	`tri_mmi`	`nnet2`	`nnet_disc`
Baseline PL	Extended PL	**76.66**	**56.88**	**53.87**	56.31	**46.13**	48.49
Extended PL	Baseline PL	77.14	58.57	55.80	**52.70**	47.78	**46.73**

Table 14: Comparison of the ASR performance using two different settings: 1. baseline PL used in AM training and extended PL used for decoding, and vice-versa, 2. extended PL used in AM training and baseline PL used for decoding.

The best values per AM type are marked in bold in Table 14. For four of the six AM types, using the baseline PL for the AM training and the extended PL for the decoding outperforms the vice-versa setting. This is also visualised in Figure 5.

The best WER over both settings is also achieved by using the extended PL in the decoding phase. The `nne2` model using this setting yields a WER of 46.13%. Compared to the results of using the extended PL in both phases, the scores by both mixed settings are worse. However, they outperform the models using the baseline PL in both phases. From this we can deduce that using the extended PL in either of the phases has a positive effect on the performance of the ASR system.

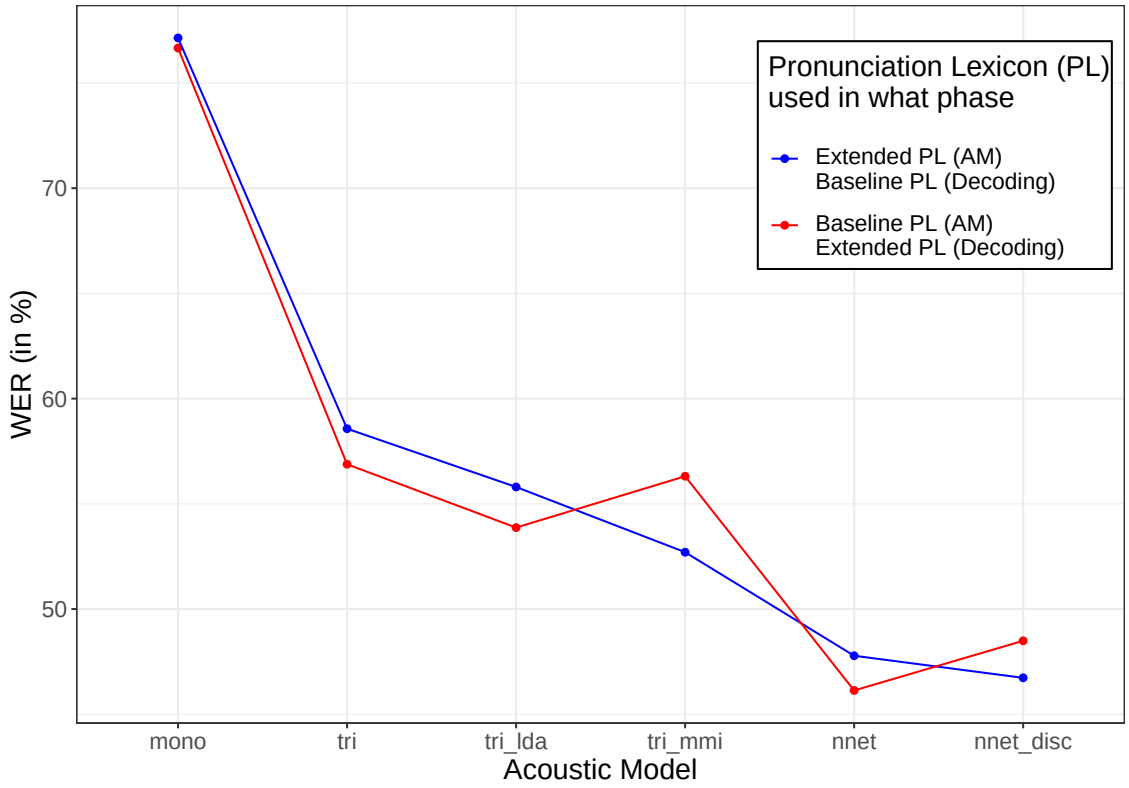

Figure 5: Comparison of the ASR performance using two different settings: 1. baseline PL used in AM training and extended PL used for decoding, and vice-versa, 2. extended PL used in AM training and baseline PL used for decoding.

In the setting with the baseline PL used for the training of the AM, we can see the decline in the discriminative models (`tri_mmi` and `nnet_disc`) again. This suggests that for the discriminative models, the PL chosen for the AM training has more impact on the performance of the ASR system. Then again, for the the other four models (`mono`, `tri`, `tri_lda` and `nnet2`) it seems to be the case that the PL used in the decoding phase has a higher impact on the ASR model's performance.

5.3 Discussion

At first, the results of the G2P conversion with achieving an accuracy of roughly 55% did not look promising. However, the resulting extended PL was able to cover much more of the words occurring in the different data sets used for training the ASR system and decoding. As we have seen in Section 2.2.3, coverage is an impor-

tant topic when speaking about the use of PLs in ASR. OOV words can lead to more errors in the output transcriptions due to the lack of context. Therefore it is important, to cover as many words as possible in the PL. G2P conversion can be an efficient and beneficial method to achieve more coverage.

In terms of output quality, using the extended PL as an input for the ASR system led to significantly better results than using the baseline PL. For all trained AMs, the WERs of using the extended PL outperformed those of using the baseline PL. The best WER achieved on the development set was 31.86% using the TDNN model with iVectors. Going back to my first research question, an extended PL indeed improves the performance of the ASR system compared to using a baseline PL in terms of output quality. The WERs yielded by ASR systems using the extended PL clearly show this improvement.

When looking at the impact of the phase in which the PL is used, it is hard define where it has a higher impact. Jouvet et al. [2012] concluded that the impact of the PL is more critical when used in the decoding process and that the PL in the training process is more tolerant to pronunciation errors. However, this was based on G2P generated pronunciation variants, whereas in my case, I did not work with multiple pronunciation variants but only one per word. Looking at the results of my experiments, the impact of the phase in which the PL is used is dependent on the acoustic modelling technique. For discriminative models (`tri_mmi` and `nnet_disc`), the impact of the PL used in the AM training was higher. On the other side, the PL used in the decoding phase had a higher impact for the the other four models (`mono`, `tri`, `tri_lda` and `nnet2`). This decline in WER when using the baseline PL for discriminative AMs was quite unexpected. Using the extended PL with a significantly higher coverage seems to mitigate that decline and lets the ASR model achieve much better WERs.

Relating these findings to my second research question, using the PL in the two different phases has a recognisable impact on the outcome of the ASR system. The phase which is impacted more seems to depend on the acoustic modelling technique. In terms of the output quality, the discriminative models used in my experiments show better results when the extended PL is used for the AM training. The other models prefer the extended PL being used in the decoding phase.

5.4 Future Work

For future projects, it would be interesting to investigate more on the impact of the PL on discriminative acoustic modelling. A possible research question could address the coverage needed in the PL to achieve that turning point where the performance gets better.

Within Kaldi, there is an interesting option to give probabilities to pronunciation variants. This way, the pronunciation variability could be used as an advantage without introducing too much confusion to the ASR system by using multiple pronunciation variants. Particularly in the context of multi-dialectal ASR, this could interesting to examine.

Generally, making more use of the other regional variants of Swiss German in the PL could be beneficial for the performance. I only used the pronunciations for the Zurich dialect, which is the most common dialect represented in the ArchiMob corpus, but only makes up for 28% of all data in the corpus. There is a lot of potential for multi-dialectal Swiss German ASR.

As for the training of the G2P model, I only tried out one toolkit and used the default values. A possible research topic could approach the impact of the G2P training hyperparameters on the final performance of the ASR system using the resulting extended PL. There are various G2P toolkits available which make use of different modelling techniques. They could be leveraged for the lexicon extension and the resulting PLs could be evaluated on how they perform in action.

6 Conclusion

In this work, I learnt how to build different ASR systems for Swiss German using the Kaldi Speech Recognition Toolkit. Within that conventional ASR architecture consisting of Acoustic Model, Language Model and Pronunciation Lexicon, my main focus was on the latter. Based on the findings of the recent Master's theses about the AM and LM by Nigmatulina [2020] and Kew [2020], I reproduced the ASR system using their best performing settings, which then served as my baseline for further experiments.

I used the word-pronunciation mappings from the Swiss German Dictionary to prepare the baseline PL. This required some pre-processing before it could be used in the ASR system. Using a data-driven G2P approach, I trained a Transformer based conversion model on the entries of the PL I constructed before. This extended PL covered more of the words occurring in the sets used for decoding and thus, the ASR models using this PL achieved significantly better scores compared to those using the baseline PL. The best WER achieved on the development set is 31.86% using a TDNN model with iVectors. This marks an improvement on the WER of 13% compared to the same model using the baseline PL.

Using the baseline PL or the extended PL for decoding not only has an effect on the performance of the ASR system, but also on the time it takes to decode the new acoustic speech signals. The extended PL contains about three times as many entries as the baseline PL and is therefore bigger in size. On average, the models using the extended PL take 1.34 times as long as those using the baseline PL for the same set to decode.

In further experiments, I investigated the impact of the PL when being used in the two different phases of the ASR system, during the training of the AM and in the decoding process. To do this, I used evaluated two mixed settings, where the baseline PL is used for AM training and the extended PL for decoding, and the vice-versa setup. Results show that both settings have a noticeable effect on the ASR system's performance in terms of WER. The type of the acoustic modelling technique seems to determine what phase has a bigger impact on the outcome. The discriminative

models used in my experiments show better WERs when the extended PL is used for the AM training, whereas the other models prefer the extended PL being used in the decoding phase. Future research can be done to investigate this behaviour more thoroughly, e.g. on the basis of the PL's coverage.

References

Baayen, R. H., Piepenbrock, R., and Gulikers, L. (1995). CELEX2 LDC96L14. Linguistic Data Consortium, Philadelphia, PA, USA.

Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Black, A. W., Lenzo, K., and Pagel, V. (1998). Issues in Building General Letter to Sound Rules. In *Third ESCA/COCOSDA Workshop on Speech Synthesis 1998*, pages 77–80, Jenolan Caves House, Blue Mountains, NSW, Australia.

Chen, S. F. and Goodman, J. (1999). An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, 13(4):359–394.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper and Row, New York, USA.

Dieth, E. (1986). *Schwyzertütschi Dialäktschrift: Dieth-Schreibung*, volume 1. Sauerländer.

Gales, M. and Young, S. (2007). The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends Signal Processing*, 1(3):195—304.

Goodman, J. (2001). A Bit of Progress in Language Modeling. *Computer Speech & Language*, 15(4):403–434.

Hain, T. (2002). Implicit pronunciation modelling in ASR. In *ISCA Pronunciation Modeling Workshop*, pages 129–134, Aspen Lodge, Estes Park, Colorado, USA.

Han, W., Zhang, Z., Zhang, Y., Yu, J., Chiu, C.-C., Qin, J., Gulati, A., Pang, R., and Wu, Y. (2020). ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context.

Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. *CoRR*.

Holtzman, A., Buys, J., Forbes, M., and Choi, Y. (2019). The Curious Case of Neural Text Degeneration. *CoRR*.

Hsu, B.-J. P. and Glass, J. R. (2008). Iterative Language Model Estimation: Efficient Data Structure & Algorithms. In *In Proceedings of the Ninth Annual Conference of the International Speech Communication Association*, pages 841–844, Brisbane, Australia.

International Phonetic Association (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, Cambridge, UK.

Jouvet, D., Fohr, D., and Illina, I. (2012). Evaluating grapheme-to-phoneme converters in automatic speech recognition context. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4821–4824, Kyoto, Japan. IEEE.

Jurafsky, D., Bell, A., Gregory, M., and Raymond, W. D. (2001). The effect of language model probability on pronunciation reduction. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 801–804, Salt Lake City, UT, USA.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2nd edition.

Karafiát, M., Burget, L., Matějka, P., Glembek, O., and Černocký, J. (2011). ivector-based discriminative adaptation for automatic speech recognition. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 152–157, Waikoloa, HI, USA.

Kew, T. (2020). Language Representation and Modelling for Swiss German ASR. Master's thesis, University of Zurich.

Kingsbury, P., Strassel, S., McLemore, C., and MacIntyre, R. (1994). CALLHOME American English Lexicon (PRONLEX) LDC97L20. Linguistic Data Consortium, Philadelphia, PA, USA.

Kisler, T., Schiel, F., and Sloetjes, H. (2012). Signal processing via web services: The use case WebMAUS. In *Proceedings of Digital Humanities Conference 2012*, pages 30–34, Hamburg, Germany.

Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2015). Audio Augmentation for Speech Recognition. In *INTERSPEECH 2015 - 16th Annual Conference of the International Speech Communication Association*, pages 3586–3589.

Lenzo, K. (2007). The CMU Pronouncing Dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

Lu, L., Zhang, X., Cho, K., and Renals, S. (2015). A Study of the Recurrent Neural Network Encoder-Decoder for Large Vocabulary Speech Recognition. In *INTERSPEECH 2015*, pages 3249–3253, Dresden, Germany.

Lucassen, J. and Mercer, R. (1984). An information theoretic approach to the automatic determination of phonemic baseforms. In *ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 9, pages 304–307, San Diego, CA, USA.

Mohri, M., Pereira, F., and Riley, M. (2008). Speech Recognition with Weighted Finite-State Transducers. In Rabiner, L. and Juang, F., editors, *Handbook on speech processing and speech communication*, pages 559–584. Springer-Verlag, Heidelberg, Germany.

Nigmatulina, I. (2020). Acoustic modelling for Swiss German ASR. Master's thesis, University of Zurich.

Nigmatulina, I., Kew, T., and Samardzic, T. (2020). ASR for Non-standardised Languages with Dialectal Variation: the case of Swiss German. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 15–24, Barcelona, Spain. International Committee on Computational Linguistics (ICCL).

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, Brisbane, QLD, Australia.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Samardžić, T., Scherrer, Y., and Glaser, E. (2015). Normalising orthographic and dialectal variants for the automatic processing of Swiss German. In *Proceedings*

of the 7th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, Poznan, Poland.

Samardžić, T., Scherrer, Y., and Glaser, E. (2016). ArchiMob - A Corpus of Spoken Swiss German. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia.

Scherrer, Y. and Ljubešić, N. (2016). Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS)*, Bochum, Germany.

Scherrer, Y., Samardžić, T., and Glaser, E. (2019a). ArchiMob: ein multidialektales Korpus schweizerdeutscher Spontansprache. *Linguistik Online*, 98(5):425–454.

Scherrer, Y., Samardžić, T., and Glaser, E. (2019b). Digitising Swiss German: how to process and study a polycentric spoken language. *Language Resources and Evaluation*, pages 1–35.

Scherrer, Y. and Stöckle, P. (2016). A quantitative approach to Swiss German – Dialectometric analyses and comparisons of linguistic levels. *Dialectologia et Geolinguistica*, 24(1):92–125.

Schmidt, L., Linder, L., Djambazovska, S., Lazaridis, A., Samardžić, T., and Musat, C. (2020). A Swiss German Dictionary: Variation in Speech and Writing. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, France.

Schmidt, T. (2012). EXMARaLDA and the FOLK tools – two toolsets for transcribing and annotating spoken language. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 236–240, Istanbul, Turkey. European Language Resources Association (ELRA).

Schmidt, T. and Schütte, W. (2010). FOLKER: An Annotation Tool for Efficient Transcription of Natural, Multi-party Interaction. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Shoup, J. E. (1980). Phonological aspects of speech recognition. *Trends in Speech Recognition*, pages 125–138.

Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. (2020). End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Wells, J. C. (1997). Sampa computer readable phonetic alphabet. In Gibbon, D., Moore, R., and Winski, R., editors, *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter. Part IV, section B, Berlin, Germany and New York, USA.

7 Tables

Vowels	Diphthongs	Consonants	Affricates
2	aI	N	pf
2:	aU	R	ts
9	ou	S	tS
9:		b	
@		d	
E		d	
E:		f	
I		g	
O		h	
U		j	
Y		k	
a		l	
a:		m	
e		n	
e:		p	
i		r	
i:		s	
o		t	
o:		v	
u		w	
u:		x	
y			
y:			
{			
{:			

Table 15: Set of phonemes used in the pronunciation lexicon using a modified version of SAMPA for the phonetical representation.