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

Language representation and modelling for Swiss German ASR

Author: Tannon Kew
Student ID Nr.: 17-717-018

Examiner: Prof. Dr. Martin Volk

Supervisor: Dr. Tanja Samardžić

Department of Computational Linguistics

Submission date: 20.05.2020

Abstract

Automatic Speech Recognition (ASR) describes the task of converting spoken language into text and is an important component for numerous practical applications that aim to encode, understand and respond to spoken language input.

This thesis presents work investigating the use of different textual representations and language models in ASR for low-resource and non-standardised Swiss German. Being a predominantly spoken language with a growing demand for automatic processing, reliable ASR can be seen as a crucial component in any natural language processing pipeline for Swiss German. However, since Swiss German lacks a standardised orthography, such a task raises the question; "if we want to convert Swiss German speech to text, what kind of text should we aim to produce?"

Using the ArchiMob corpus of spoken Swiss German, which provides approximately 70 hours of transcribed continuous speech data, we explore ASR performance for two potential textual representations. The first is a non-normalised spelling that aims to render the true sounds of the language as close as possible and thus contains a high degree of variation among written surface forms. The second of these representations is a semi-automatically normalised spelling that more closely resembles Standard German and thereby reduces the lexical variation in the corpus.

Each of these text types brings with it considerable challenges regarding how we can best represent word-level pronunciations and how to derive reliable statistical language models for this non-standardised language. Thus, using the Kaldi Speech Recognition Toolkit, we explore a number of ASR system setups exploiting different pronunciation lexica and statistical language models in order to determine optimal setups for Swiss German ASR.

Zusammenfassung

Bei der automatischen Spracherkennung (*Automatic Speech Recognition (ASR)*) handelt es sich um die Umwandlung von gesprochener Sprache in Text. Sie ist für zahlreiche praktische Anwendungen relevant, die darauf abzielen, gesprochene Sprache zu kodieren, zu verstehen und auf gesprochene Spracheingabe zu reagieren.

In dieser Arbeit untersuchen wir den Einsatz von verschiedenen Textrepräsentationen und Sprachmodellen in der automatischen Spracherkennung für *low-resource* und nicht standardisiertes Schweizerdeutsch. Als überwiegend gesprochene Sprache mit einem wachsenden Bedarf an automatischer Verarbeitung können zuverlässige Spracherkennungssysteme entscheidende Komponenten in zahlreichen *Natural Language Processing Pipelines* für Schweizerdeutsch darstellen. Da dem Schweizerdeutschen jedoch eine standardisierte Rechtschreibung fehlt, stellt sich bei einer solchen Aufgabe die Frage: "Wenn wir die schweizerdeutsche Sprache in Text umwandeln wollen, welche Art von Textrepräsentation sollten wir anstreben?".

Mit Hilfe des ArchiMob-Korpus, der etwa 70 Stunden transkribierte, kontinuierliche schweizerdeutsche Sprachdaten enthält, untersuchen wir die *ASR*-Leistung für zwei mögliche Textrepräsentationen. Bei der ersten handelt es sich um eine nicht-normalisierte Rechtschreibung, die die wahren Laute der Sprache so genau wie möglich wiederzugeben versucht, weshalb sie ein hohes Maß an Variation zwischen den geschriebenen Oberflächenformen aufweist. Bei der zweiten dieser Repräsentationen handelt es sich um eine halbautomatisch normalisierte Rechtschreibung, die dem Hochdeutschen ähnlicher ist und so die lexikalische Variation im Korpus verringert.

Jede dieser Textarten bringt erhebliche Herausforderungen mit sich, wenn es darum geht, die Aussprache auf Wortebene am besten darzustellen und zuverlässige statistische Sprachmodelle für diese nicht standardisierte Sprache abzuleiten. Unter Verwendung des *Kaldi Speech Recognition Toolkit* untersuchen wir daher eine Reihe von *ASR-System-Setups* auf ihre Nutzung von verschiedenen Aussprache-Lexika und statistischen Sprachmodellen. Ziel der vorliegenden Arbeit ist es, optimale Vorgehensweisen für die automatische Spracherkennung des Schweizerdeutschen zu ermitteln.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Tanja Samardžić for her continuous support, guidance and valuable feedback throughout this project. Our discussions regarding not only the topic of this research, but also balancing life, knowledge and work have been greatly appreciated.

Secondly, I would like to extend my heartfelt gratitude to Prof. Dr. Martin Volk, who, along with the entire team at the Department of Computational Linguistics at the University of Zurich, helped me to rediscover a passion for learning and has provided a constant source of inspiration for all things computational and linguistics throughout my studies at the University of Zurich.

A special thank you goes to Iuliia Nigmatulina, Dr. Thayabaran Kathiresan, and Fran Campillo from Spitch AG, who all provided me with useful insights into ASR and helped me to connect the some of the many dots along the way. Also, thank you to my Swiss-German speaking friends, colleagues and family who provided me with a number of examples of their language that are used in this work.

Additionally, thank you to my fellow students and friends Anastassia and Fabian (FAT32), who always provided laughter, inspiration and many memorable moments during our studies together. Last but not least, I want to thank Thea for her support, patience, guidance and strength. This work would not have been possible without the contributions of all of these people.

Contents

List of Figures

List of Tables

List of Acronyms

AM	Acoustic Model
ASR	Automatic Speech Recognition
BP	Backpropagation
BPTT	Backpropagation Through Time
CD	Context Dependent
CER	Character Error Rate
CH	Confoederatio Helvetica (Switzerland)
CS(R)	Continuous Speech (Recognition)
CSV	Comma-Separated Values
DE	German
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
EM	Expectation Maximisation
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Models
IPA	International Phonetic Alphabet
IWR	Isolated Word Recognition
KN	Kneser-Kney Smoothing
LCC	Leipzig Corpora Collection
LDC	Linguistic Data Consortium
LM	Language Model
LMWT	Language Model Weight
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficients
ML(E)	Maximum Likelihood (Estimation)
NLP	Natural Language Processing
NLU	Natural Language Processing

NN	Neural Network
OOD	Out of Domain
OOV	Out of Vocabulary
PM	Pronunciation Model
PPL	Perplexity
RNN	Recurrent Neural Network
RQ	Research Question
SADS	Syntactic Atlas of Swiss German Dialects
SAMPA	Speech Assessment Methods Phonetic Alphabet
SDS	Swiss German Dialect Atlas
STT	Speech-to-Text
STTD	Speech-to-Text-Dieth
STTN-D	Speech-to-Text-Norm-Dieth
STTN-S	Speech-to-Text-Norm-SAMPA
STTN-Z	Speech-to-Text-Norm-ZRH
SW	Sub-Word
TTR	Type-Token Ratio
URPP	University Research Priority Project: Language and Space
WB	Witten-Bell Smoothing
WER	Word Error Rate
WFSA	Weighted Finite State Acceptor
WFST	Weighted Finite State Transducers
WIP	Word Insertion Penalty
XML	Extensible Markup Language
ZRH	Zurich

1 Introduction

1.1 Motivation

Since the development of the very first computers, human-computer interaction through the use of natural language has been a major goal, giving rise to the field of Natural Language Processing (NLP). Given that natural language is not only extremely complex but also rich in variation – be it syntactic, lexical, phonological or stylistic, among others – NLP faces numerous challenges. One particularly challenging area is Automatic Speech Recognition (ASR), which deals with the crossover between spoken and written language.

The main goal of ASR is to convert a spoken utterance into a textual representation. This text could simply provide a transcript of what was said or it could be used as input for further processing systems, such as information retrieval and search, chat bots, and even machine translation. While such a technology has numerous applications in areas of research, telecommunications and national security, among others, perhaps one of its most lucrative applications is in consumer technology, where voice-operated personal assistants have now become standard features in modern smartphones, laptops, cars and even homes. This development is largely due to ever better and smaller hardware, improved methods in ASR, and, of course, an abundance of appropriate training data. Thus, ASR is a rapidly growing field and is forecast to be one of the most important technologies of the future [Widmer, 2018].

ASR systems such as those deployed in voice-operated devices work well for a few standardised, high-resource languages like English, where state-of-the-art systems are capable of achieving error rates of less than 5% on certain tasks [Chiu et al., 2018]. However, the challenge of handling other languages remains, particularly for dialectal varieties, where error rates commonly exceed 40% [Ali et al., 2017]. In this work, we address this challenge and focus our attention on ASR for Swiss German.

Swiss German is an umbrella term that describes a group of (largely) mutually

intelligible Alemannic dialects spoken by approximately five million people[1] in the northern two-thirds of Switzerland. Despite lacking any form of unifying standardisation, Swiss German enjoys a rather prestigious status and holds significant symbolic value [Watts, 2010].

Traditionally, the language situation throughout the region has been described as an example of medial diglossia [Kolde, 1981], where two distinct language varieties coexist, with one used primarily for spoken communication and the other for written discourse. In German, the words *Mundart* (lit. 'way of the mouth') and *Schriftsprache* (lit. 'script language') help to define this relationship. The former refers to the collective of Swiss German varieties spoken by people in the region and used as the predominant language of everyday life. The latter refers to Standard German, which is used primarily for writing but also spoken in most formal education settings and interactions with non-dialect speakers [Siebenhaar, 2006].[2]

Despite the peaceful coexistence of the two language varieties, the linguistic situation is made more complicated by the fact that native Swiss German speakers often find Standard German unnatural and aloof, making certain situations in which it is enforced uncomfortable [Clematide et al., 2016]. As a consequence, Swiss German is used extensively in many aspects of daily life throughout the region and thus has a strong demand for automatic processing. Since it remains a predominantly spoken language, reliable ASR systems can be seen as essential components of any larger NLP pipelines for Swiss German.

1.2 Task and Challenges

Broadly speaking, ASR aims to convert speech to text. This can be further distinguished into two main subtasks: isolated word recognition (IWR) and continuous speech recognition (CSR). The first aims to identify a standalone spoken word surrounded by an observable silence, whereas the second aims to determine sequences of running words with or without silence [Jurafsky and Martin, 2008, ch. 9]. Nowadays, industrial systems are designed to recognise anywhere between 50,000 and 100,000 unique words in running speech and are thus considered large vocabulary CSR (LVCSR) systems [CallMiner, 2013]. In this work, we adopt the broader term

[1] This figure is estimated based on information provided by the Swiss Federal Statistical Office [2019].

[2] A further, more fine-grained distinction can be made between the Standard German used in Germany and Swiss Standard German as it used in Switzerland, however, in this work we use Standard German to refer to both varieties.

of 'ASR' to refer to the task of converting a spoken utterance of arbitrary length into text.

Conventional ASR systems typically rely on statistical methods and combine probabilistic acoustic models with a handcrafted pronunciation lexicon and a probabilistic language model to convert speech to text [Donaj and Kačič, 2016]. On the one hand, this approach leads to a high level of system complexity, making ASR a challenging task. Yet, on the other hand, it provides a considerable amount of modularity and allows for experimentation through the substitution and adaption of certain components. Luckily, a number of publicly available, open-source toolkits exist, which can be used to develop relatively well-performing speech recognisers.

In this work, we make use of the Kaldi Speech Recognition Toolkit [Povey et al., 2011] and exploit the modular design of conventional ASR systems to focus on one particular aspect, namely, language modelling for Swiss German ASR. Given the long-standing medial diglossia, Swiss German has traditionally only been written in very limited contexts, such as for the purpose of language documentation or a small amount of popular literature. More recently though, with the widespread rise of computer mediated communication, native speakers have increasingly begun to write in Swiss German [Siebenhaar, 2006]. Consequently, different textual representations exist for Swiss German, ranging from the completely spontaneous and highly variant spellings produced by native speakers through modern communication channels to automatically normalised spellings produced using advanced techniques in machine translation. Therefore, our task raises a rather paradoxical question; "if we want to convert Swiss German speech to text, what text should we use?"

We aim to investigate ASR performance given two potential representations, the first of which is semi-standardised orthography used primarily for language documentation, while the other is an automatically normalised orthography that more closely resembles the spellings of Standard German. In doing so, we will address the following research questions:

RQ1: When building an ASR system for a low-resource language with no standardised orthography such as Swiss German, which target textual representation is optimal?

RQ2: Given a particular textual representation, can we improve the language model component of the ASR system and thereby the quality of the system output?

Answering these two research questions is not trivial. For each type of textual representation we face two major challenges. Firstly, we need to be able to effectively model word-level pronunciations with a pre-determined set of phonemes. This is

made difficult by the fact that there is no single standardised pronunciation for Swiss German and that significant phonological variation occurs between different speakers. Secondly, in order to produce accurate transcriptions and reduce ungrammatical errors, we need to effectively model word sequences in the language. This is made challenging due to the high degree of lexical variation and the resulting data sparsity in the limited Swiss German text corpora that is available.

While significant advancements in the area of language modelling have been made recently thanks to approaches based on neural networks, such models are not directly compatible with contemporary ASR system architectures. Thus, we limit the scope of our research to investigating methods which allow themselves to be directly integrated into a system.

1.3 Thesis Structure

The structure of this thesis is as follows: In Chapter 2, we begin with a theoretical introduction to the task of ASR and describe the approach taken in conventional systems. In addition, we provide a discussion on the typical language modelling techniques applied in ASR. Chapter 3 discusses the major challenges facing ASR for Swiss German, pertaining to a large degree of spoken and written variation. In Chapter 4, we present the relevant corpora that we exploit for the purpose of enhancing language models for Swiss German and the various tools used as part of this work.

In order to address RQ1, in Chapter 5, we first introduce our ASR system and conduct exploratory experiments to determine baseline ASR performance given two potential textual representations, each of which gives rise to particular system constraints and challenges. In Chapter 6, we focus on RQ2 and investigate possible improvements in language modelling performance for both textual representations. Following this, in Chapter 7, we examine the impact of incorporating improved language models on the overall performance of our ASR system for Swiss German. Finally, we present our concluding remarks in Chapter 8.

2 Background

In this chapter we first introduce the task of Automatic Speech Recognition (ASR) and then provide an overview of the individual processes and components involved in converting a speech signal to text using the Kaldi ASR toolkit. In Section 2.2, we discuss techniques in language modelling and how these are incorporated into conventional ASR systems.

2.1 Automatic Speech Recognition

The goal of ASR is to convert an acoustic speech signal into a textual representation. Conventional ASR systems do this by applying the noisy channel model. In this context, multiple statistical models representing acoustic and structural properties of language are combined into a single, unified search space which covers all potential output transcriptions. The final challenge is then framed as a matter of efficiently searching this potentially enormous space of possibilities in order to automatically transcribe new spoken utterances. In this section, we describe the theoretical background applied in conventional ASR systems, their fundamental components and the main technologies behind them.

2.1.1 Applying the Noisy Channel Model

The noisy channel model describes an analogy in which given two representations of the same information we assume one to be the 'true' representation and treat the other as a 'noisy' version that has been distorted after being passed through a communications channel of some sort. In the case of ASR, we consider the acoustic input signal to be the 'noisy' version and the corresponding textual representation to be the 'true' form. The logic of the noisy channel suggests that if we are able to construct a model of this channel, we can determine exactly how it modifies a given input signal. Then, for any new acoustic input signal, we can run all possible utterances through our model and select the one which best matches the input signal [Jurafsky and Martin, 2008, ch. 9].

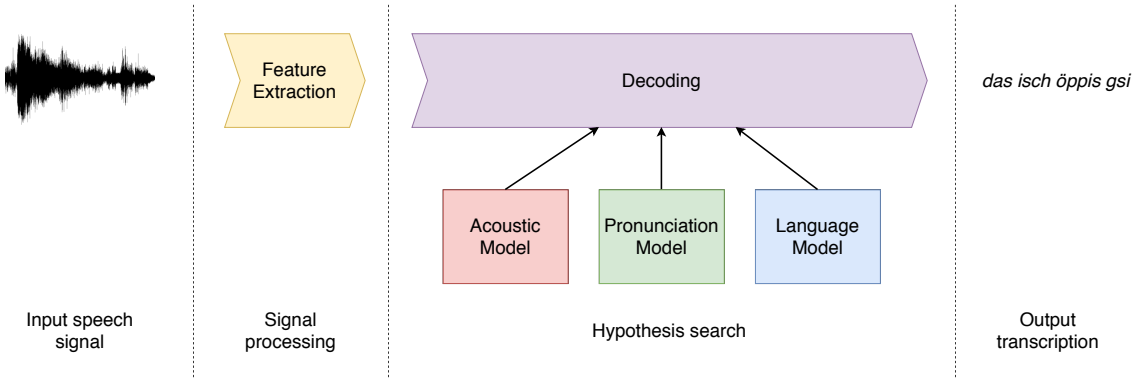

Figure 1: Architecture of a conventional ASR system (adapted from Gales and Young [2007]).

Figure 1 shows the general architecture of an ASR system which applies the logic of the noisy channel to convert speech to text. Given a new speech signal as input, we want to find the most probable text sequence. To do this, we first process the signal to extract a series of fixed-length feature vectors $Y = y_1, \ldots, y_T$, whereby T represents the number of frames in the signal. These features are then 'decoded' in order to produce the best, i.e. the most likely, text sequence given the acoustic features.

The decoding step corresponds to a large-scale hypothesis search and is dependent on three major components that represent acoustic and structural properties of the target language. Firstly, an acoustic model (AM) provides probabilities to determine the speech sounds associated with an acoustic signal. Secondly, a pronunciation model (PM) provides the likelihood of individual words given a sequence of speech sounds. And lastly, a language model (LM) provides the probabilities for potential sequences of words forming an utterance.

The ASR task can be more formally defined as follows. If we consider a spoken utterance as a sequence of words $W = w_1, \ldots, w_n$ in a language $\mathcal{L}$, our aim is to determine the most likely sequence of words $\hat{W}$ given the observed acoustic features Y. Thus,

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(W|Y) \tag{2.1}$$

Here, $P(W|Y)$ corresponds to the posterior probability of a text sequence given only a small amount of information available from the audio signal. Naturally, this

is "difficult to model directly" [Young, 2008, p. 2] and would benefit from additional information regarding the *a priori* likelihood of a given sequence of sounds and/or words in the language. Applying Bayes' Rule allows us to incorporate this valuable information and operationalises the above equation as

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \frac{P(Y|W)P(W)}{P(Y)} \tag{2.2}$$

Since the probability of the observation sequence $P(Y)$ does not change for a particular input utterance, i.e. it is a constant, Eq. 2.2 can be simplified as

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(Y|W)P(W) \tag{2.3}$$

Eq. 2.3 formally describes our basic approach to ASR and denotes two distinct probabilities that can be estimated from large amounts of speech and text data. The first, $P(Y|W)$, is the observation likelihood and is provided by the AM and the PM together. The second, $P(W)$, denotes the prior probability and is calculated by the LM. Finally, maximising these two probabilities allows us to determine the most likely sequence of words in a given utterance, thereby decoding a 'noisy' input signal to find its 'true' representation.

2.1.2 Feature Extraction

A speech signal is a highly complex, non-stationary acoustic signal combining many different frequencies. Typically, in ASR, we start from a digital recording of a speech signal. In order to be processed by a speech recogniser, either for AM training or decoding with a fully developed system, the signal must first be converted into a suitable input representation. This is known as feature extraction.

The most popular acoustic features used in ASR systems are mel frequency cepstral coefficients (MFCC). MFCCs capture the source characteristics of a speech signal according to natural human hearing, allowing us to identify individual speech sounds in the signal [Dave, 2013; Jurafsky and Martin, 2008]. The process of extracting these features is described below.

Given that a speech signal is non-stationary, its characteristics change rapidly over time and thus looking at an entire signal fails to provide any useful information. However, if we focus on a very short segment of the audio signal, say 25 milliseconds, we can consider it to be roughly stationary at that particular point in time.

Windowing describes the process of segmenting an arbitrarily long input signal into many smaller audio frames. Typically, a Hamming window is used to slide across the audio signal and capture audio frames every 10 milliseconds, resulting in 100 frames for each second of audio. The Hamming window reduces the signal to zero at either end of the frame to avoid discontinuities between overlapping frames [Dave, 2013]. Figure 2 depicts how an input signal is split into overlapping frames which are then processed individually to produce a series of MFCC feature vectors to represent the entire speech signal.

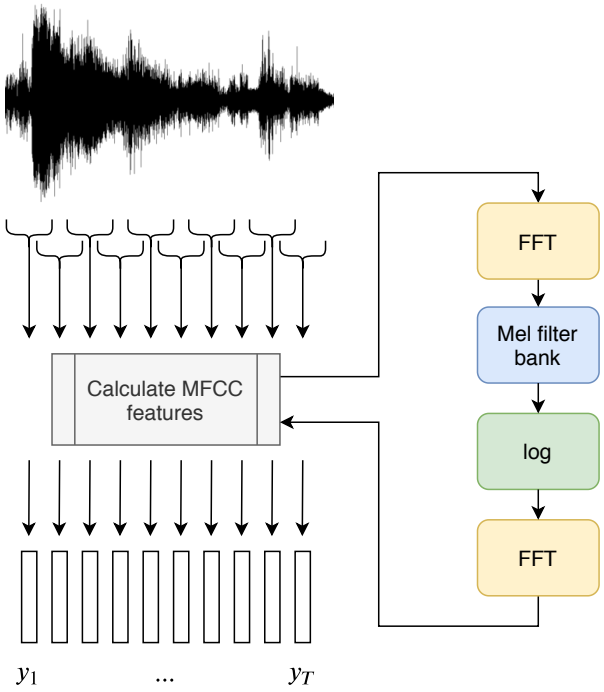

Figure 2: Extracting feature vectors from a digital speech signal. For each 25-millisecond frame, a single feature vector is calculated to provide an input representation suitable for speech recognition (adapted from [Jurafsky and Martin, 2008, ch. 9]).

For each audio frame, we compute the cepstrum of the stationary signal. The cepstrum is defined as the inverse discrete Fourier transform (DFT) of the log magnitude of the DFT of a signal [Taylor, 2009]. Essentially, it comprises a series of transformations that allows us to isolate and parameterise the source characteristics of a stationary signal. As shown on the right-hand side of Figure 2, we compute it by first applying the Fast Fourier Transform (FFT) algorithm, which converts the signal from the time domain into the frequency domain. This representation is known as the spectrum and provides information about the amount of energy for individual frequencies in the signal. In a second step, the resulting DFT is warped using a mel-scale filter bank, which emulates frequency perception according to the

human auditory system. Then the *log* of each audio filter is taken. Finally, a second FFT operation is applied to these interim log values, producing an inverse DFT. The resulting cepstrum[1] contains informative coefficients for each frequency-band filter. The first 12 of these coefficients typically represent information about the physical characteristics of the vocal tract and are thus most valuable for detecting individual speech sounds in ASR [Jurafsky and Martin, 2008, ch. 9].

For each of these cepstral coefficients and an additional energy score feature, delta and double delta features are calculated to account for changes between audio frames [Jurafsky and Martin, 2008, ch. 9]. The resulting 39 values constitute a single feature vector which represents a single audio frame of the input signal. Thus, to represent one minute of speech, we need around 6,000 vectors with 39 dimensions.

2.1.3 Acoustic Model

The AM is the major component of an ASR system that integrates knowledge about acoustics and phonetics in order to identify the distinct speech units, known as 'phones', that make up a word [Yu and Deng, 2014]. As input, the AM takes a sequence of acoustic signal feature vectors, Y. For each $y \in Y$, it calculates a probability distribution over all possible speech units W, representing the likelihood of each speech unit producing the particular feature at that particular time.

Earlier we described the sequence of speech units W as a sequence of words, however, in reality, AM units are more likely to correspond to much smaller sub-word entities, such as phones (e.g. /a/, /k/, /ŋ/, etc.) or even parts of phones. Differences in pronunciation between speakers and variant speech rates result in significant acoustic variability of words in spoken language [Gold et al., 2011]. Attempting to model this word-level variability directly would require an enormous number of training examples for every possible word in the vocabulary, which is infeasible for large vocabulary tasks. Additionally, such an approach, would not allow us to recognise any words not seen at training time and thus significantly limit the capabilities of a system [Singh, 2013]. Modelling spoken language instead using a closed group of sub-word entities, such as phones, allows for a more flexible model that can learn more effectively from the limited examples found in regular data sets and that can generalise, potentially recognising words not seen during training provided that they are encoded in the pronunciation lexicon.

[1] Note that the word 'cepstrum' is also derived from inverting the first four letters of the word 'spectrum', indicating their relation.

2.1.3.1 Hidden Markov Models

Perhaps the biggest breakthrough in the field of ASR came with the application of hidden Markov models (HMMs), which first took place in the 1970's and remains the basis of most modern-day systems [Young, 2008]. HMMs describe a type of generative sequence labelling models that calculate a probability distribution over a set of known but unobserved, and hence 'hidden', labels given an observable input sequence [Jurafsky and Martin, 2008, ch. 6]. In acoustic modelling, this observable input sequence corresponds to the acoustic feature vectors that are extracted from the input signal and the hidden states represent phones, or parts of phones.

Comprising a set of hidden states Q, a HMM makes a transition at each timestep from its current hidden state to one of its connected states and generates an observation associated with that state. Given the compositional and temporal structure of spoken language, a first order HMM lends itself well to this type of modelling task by making two simplifying assumptions [Gales and Young, 2007]. Firstly, the probability of being in a particular state at a particular time is conditioned only on the previous state, which is given as the transition probability a_{ij}. Secondly, the observation generated by being in a particular state is independent of all other states and is given as the emission probability $b_i(o_t)$.

The most basic HMM for used in conventional acoustic modelling consists of a set of states corresponding to a sequence of base phones. This state sequence represents the pronunciation of a given word. Such a model considers phones as context-independent units and is known as a monophone model. Figure 3 depicts a monophone HMM for the Swiss German word *ufe* ('up').

The main components of this HMM can be summarised as follows:

$Q = q_1, ..., q_N$	a set of N states (corresponding to phones)
$A = a_{01}, ..., a_{nn}$	a transition probability matrix A, each a_{ij} representing the probability for each phone taking a self-loop or going to the next one
$B = b_i(y_t)$	a set of emission probabilities, each expressing the probability of an audio feature vector (observation y_t) being generated from the phone state q_i
$Y = y_1, \ldots, y_T$	a sequence of observable acoustic features vectors
q_0, q_{end}	HMM start and end states not linked to any observations, which help with concatenating models together

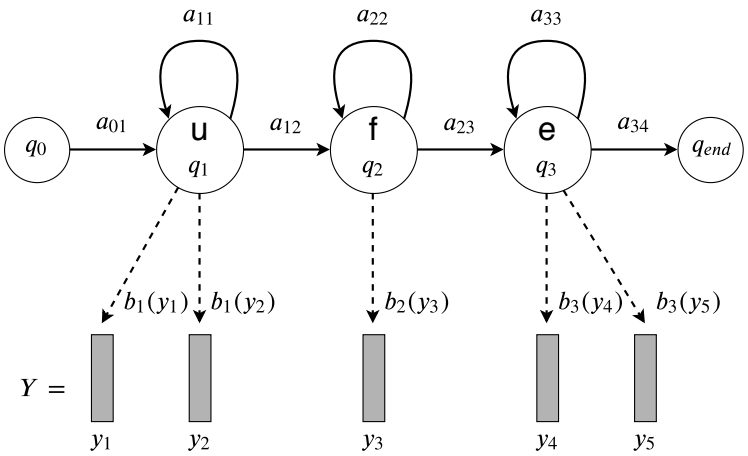

Figure 3: A monophone HMM depicting the context independent phone-level state sequence for the Swiss German word *ufe* ('up') (adapted from Gales and Young [2007]; Rúnarsdóttir [2018]). Note, state transitions can only go in one direction (left to right). Such a model is known as a Bakis HMM.

As previously mentioned, the acoustic feature sequence typically consists of 100 feature vectors for every second of audio. Depending on the rate of speech, a single phone can span a large number of these feature vectors and can exhibit a large amount of spectral variation. Thus, phones are in fact too coarse a unit for modelling directly in large vocabulary continuous speech recognition systems [Singh, 2013]. To compensate for this, phones are typically further segmented into sub-phone entities, such as triphones. Triphones model a phone considering its left and right context and can be derived using rule-based decision trees [Jurafsky and Martin, 2008, ch. 10]. For example, the 'o' in *gross* ('big') is modelled differently than the 'o' in *note* ('score') due to differing neighbouring consonants. Such context-dependent (CD) phone representations allow us to model phones in much more detail and improve the modelling of words by enabling more restrictive probability transitions between states in the HMM.

2.1.3.2 GMM-HMMs and Hybrid DNN-HMMs

An effective method for modelling the frame-based feature vectors of an acoustic speech signal is to use Gaussian mixture models (GMMs). GMMs describe a class of generative data clustering models that combine a number of weighted multivariate Gaussian models in order to represent a complex, non-normal distribution encoded as a vector of random variables [Yu and Deng, 2014]. A GMM is parameterised by a

mean vector $\vec{\mu}$ of length D, whereby D is equal to the length of the acoustic feature vectors, a $D \times D$ covariance matrix Σ and a mixing probability π. Combining these with HMMs allows for modelling the sequential and time-variant nature of spoken language and results in the conventional GMM-HMM acoustic model, in which at each timestep in the HMM, an observation is generated given the current hidden state according to a Gaussian mixture distribution [Yu and Deng, 2014].

Due to the relatively strong performance achieved with GMM-HMMs and the fact that they can be effectively learned from utterance-level transcribed audio examples (see Section 2.1.3.3), they remain the dominant paradigm for many conventional ASR systems. However, more recently, deep neural networks (DNNs) have seen a great deal of attention in ASR due to their classificational strength. This has lead to their exploitation in hybrid HMM-DNN AMs which rely on a neural network to approximate the probability of a particular sub-phone unit given an observed acoustic feature in place of a GMM [Maas et al., 2015].

2.1.3.3 Acoustic Model Training

AM training refers to the process of learning optimum model parameters to effectively describe the set of example instances in the training data. The major advantage of conventional GMM-HMM acoustic models is that they can be trained rather effectively given numerous examples of audio speech data with utterance-level transcriptions through a method known as embedded training.

Embedded training involves first transforming an utterance transcription from a sequence of words into a sequence of phones based on the information provided in the pronunciation model (see Section 2.1.4). Then, the parameters for the GMM-HMMs that represent each individual phone (or sub-phone unit) are learned with a maximum likelihood estimation (MLE) objective. In particular, a version of the Expectation-Maximisation (ME) algorithm for training HMMs, known as the Baum-Welch algorithm, is applied to iteratively estimate the model parameters until it converges.

Given some initial parameter values[2] for a GMM-HMM, the Baum-Welch training algorithm, relies on dynamic programming techniques to efficiently sum over all

[2]In ASR, the initial transition and emission probabilities are typically set such that they are equiprobable (i.e. the transition likelihood of doing a self-loop in a particular state or moving to the next state are 0.5 each and the likelihood of a particular observation being generated from a given state is $\frac{1}{N}$, whereby N is the number of possible emission labels) and the initial parameters of all GMMs are identical based on the gloabal mean and variance of the training data [Jurafsky and Martin, 2008, ch. 9]. This type of initialisation is known as a flat start.

possible segmentations of words and phones by computing the probability of being in a certain state j at time t and generating the observation sequence O. This process describes the expectation step (E-step) of the Baum-Welch algorithm. Then, in a subsequent maximisation step (M-step) the model's parameters are updated to maximise the probability of seeing the ground truth labels with regard to the current parameter estimations. This process is repeated over multiple iterations until convergence, at which point a local optimum is reached and any parameter updates are essentially ineffective.

A further adaption to the Baum-Welch algorithm that speeds up AM training is known as forced Viterbi alignment. While the Baum-Welch algorithm calculates the probabilities for *all* possible paths, the Viterbi algorithm calculates only the *single best*, i.e. the most probable, path. Since the training data provides the true phone sequence for a given acoustic signal, we can force the system to follow this path by setting the HMM transition probabilities accordingly. As a result of explicitly specifying the internal state sequence, the model only needs to optimise the relevant emission probabilities, thereby reducing training time significantly [Jurafsky and Martin, 2008, ch. 9].

After using embedded training to train a monophone GMM-HMM AM, a low-level alignment between the audio signal segments and their corresponding phone labels can be derived automatically. Following this, more advanced AMs can be trained subsequently, benefiting from the phone (or sub-phone) segmentation and alignments attained from the previous model. The purpose of these subsequent models is to improve label classification. While numerous types of models and adaptions have been proposed, popular methods include training CD triphone GMM-HMMs, replacing the MLE objective function with a discriminative objective function, applying feature normalisation steps and exploiting other types of classifier models, such as DNNs.

2.1.4 Pronunciation Model

The pronunciation model provides the system with the knowledge of how words are pronounced. In conventional ASR systems, the pronunciation model, or 'lexicon', comprises a hard-coded list of words mapped to their possible pronunciations, which are represented as a sequence of phone symbols. Typically, most words in the lexicon have a single canonical pronunciation associated with them, however, it is possible for a word to have multiple allowable pronunciations if necessary. For instance, the English word 'the' may be listed with two common variant pronunciations: 'TH AH',

with an unstressed /ə/, and 'TH IY', with a stressed /i/.

During AM training, the lexicon provides the mapping required to transpose our initial word-level utterance transcriptions into a sequence of phones for the purpose of embedded training. During decoding, on the other hand, the lexicon is used to compile the word-level HMMs for all possible words, such as that depicted in Figure 3. In theory, the system is capable of outputting any word that appears in the decoding lexicon, even if it was not seen in the training data [Aggarwal and Dave, 2011].

2.1.5 Language Model

The primary task of a language model (LM) is to determine the probability of a given sequence of words in a language. In ASR, this corresponds to the prior probability $P(W)$ from Eq. 2.3. Given that an input speech signal is typically extremely noisy and potentially ambiguous, the LM is a crucial component in an ASR system [Jurafsky and Martin, 2019, ch. 3]. Its goal is to learn which word sequences are most probable on the basis of examples found in language corpora and to distinguish these from less probable ones in order to reduce erroneous or ill-formed utterance transcriptions. For example, the observed English phone sequence[3]

AY W AA N T AY S K R IY M

corresponds to both the syntactically correct 'I want ice cream' and the syntactically incorrect 'I want I scream'. A good language model should assign a higher probability to the syntactically correct utterance and a lower probability to the incorrect form.

The dominant approach to language modelling in ASR is based on statistical methods that represent text by means of a fixed-size sequence of N words called N-grams [Donaj and Kačič, 2016]. These models are not only quick to train and easy to implement within the conventional ASR framework, but they also perform relatively well and have established strong state-of-the-art benchmarks in many ASR tasks [Goodman, 2001; Schwenk, 2007; Chen et al., 2015]. In recent years, however, neural network language models (NNLMs) have gained significant traction in NLP, particularly in ASR, leading to performance improvements for high-resource languages in a variety of tasks [Chen et al., 2015; Schwenk, 2007; Schwenk and Gauvain, 2004]. Nevertheless, NNLMs pose a number of challenges when it comes

[3]This example uses the pronunciation representation provided in the Carnegie Mellon University (CMU) Pronouncing Dictionary for English, `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

to their integration in existing ASR systems. In Section 2.2 we discuss these two different methods in language modelling and their application in more detail since this is central to our investigation for RQ2.

2.1.6 Decoding

In ASR, decoding describes the task of searching the vast space of possible utterance transcriptions in order to find the *best* hypothesis given an input speech signal. Above, we formally defined our basic approach with Eq. 2.3 which is repeated here,

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(Y|W)P(W) \tag{2.3}$$

According to this equation, we estimate the most probable sequence of words $\hat{W}$ by taking the product of the observation likelihood $P(Y|W)$, as calculated by the AM, and the prior probability $P(W)$, calculated by the LM, for all possible $W \in \mathcal{L}$ and then selecting that which is the highest. However, Eq. 2.3 relies on an incorrect independence assumption. Since the AM estimates the observation likelihood over a sequence of independent acoustic frames corresponding to sub-word units and the LM estimates the prior probability over a sequence of words, combining the two using Eq. 2.3 results in an underestimation of the observation likelihood [Jurafsky and Martin, 2008, ch. 9]. To compensate for this, we add two factors that enable us to re-weight these probabilities.

The first factor is the language model weight (LMWT) which is applied as an exponent on the prior probability and essentially acts to decrease its value in order to balance out the two probabilities. The second factor then accounts for an unwelcome side-effect of the first. By scaling down the LM probability with a LMWT, the decoder will tend towards selecting more, shorter words over fewer, longer words [Jurafsky and Martin, 2008, ch 9.]. Therefore, an additional factor known as the word insertion penalty (WIP) is included to counteract this, resulting in the following equation being applied in decoding[4]:

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(Y|W)P(W)^{LMWT}WIP^{N} \tag{2.4}$$

where N corresponds to the number of words in the hypothesis transcription. Figure

[4]As is common when dealing with multiplication of probabilities, log probabilities are used in practice to avoid numerical underflow.

11 in Appendix B, provides a visualisation showing how these two hyperparameters influence the resulting output evaluation metric.

Given the enormous search space involved in typical LVCSR tasks, efficiently decoding an input speech signal is indeed a non-trivial task. A popular approach to solving this problem, and that which is employed in Kaldi, is based on the weighted finite-state transducer (WFST) framework described by Mohri et al. [2008].

2.1.6.1 Weighted Finite-State Transducers

A finite-state automaton is a computational machine comprised of a finite set of states, including one start state and one or more end states, and transitions between those states. In its most basic form, a finite-state automaton accepts, or recognises, a certain input sequence and is thus often referred to as a finite-state acceptor (FSA) [Mohri et al., 2008]. An input sequence is accepted by an FSA if its elements correspond to the transition labels of any path through the FSA, which begins in the start state and terminates in a valid end state.

Weights may be added to the transitions and/or states, encoding probabilities or costs associated with taking a particular transition or being in a particular state. This results in a weighted FSA (WFSA) that provides a function from an input sequence to a value associated with the final path taken through the automaton [Hannemann, 2013]. A WFST is an extension of a WFSA in which the state transitions have two labels, corresponding to an input and an output symbol. Unlike a WFSA, a WFST can be used to specify a mapping between two levels of representation, e.g. phones and words, as well as a function for deriving the probability or cost associated with taking a particular path through the machine [Mohri et al., 2008].

WFSTs can be used to effectively represent all three major components of an ASR system: HMM AMs, static PMs and N-gram LMs. These individual component-wise WFSTs can then be combined using operations such as composition, determinisation and minimisation in order to provide a single, unified representation for a speech recogniser [Mohri et al., 2008]. Composition describes the operation of combining two transducers T_1 and T_2 such that the resulting transducer has exactly one path $u{:}\,w$ for each pair of paths, the first in T_1 mapping $u{:}\,v$ and the second in T_2 mapping $v{:}\,w$. Therefore, composition is the primary operation for combining two different levels of representation.

Determinisation and minimisation are both optimisation operations that help to

speed up processing of a WFST. A deterministic automaton ensures that no two transitions between states have the same input label, thus reducing redundant transitions which would lead to a larger search space and longer decoding times. Minimisation aims to further reduce the size of a deterministic WFST in such a way that is has the fewest states and the fewest transitions possible [Mohri et al., 2008].

Thus, given the transducers;

H representing a learned HMM structure that maps HMM states to CD sub-phone entities (e.g. triphones),

C representing phonetic context dependency that maps CD sub-phones to phones,

L representing the pronunciation model that maps phones to valid words,

G representing a language model[5] that maps individual words to likely word sequences,

we can compose a large WFST using the following formula:

$$HCLG = min(det(H \circ min(det(C \circ min(det(L \circ G))))))) \tag{2.5}$$

The resulting WFST forms a single unified decoding graph, mapping the entire possible search space for our recogniser [Povey et al., 2012; Renals, 2018]. Given a new utterance corresponding to T audio frames, we construct a new WFSA, U consisting of $T + 1$ states. Each transition label represents a single CD sub-phone HMM state for time t with weights indicating the probability of taking each transition given the acoustic feature at time t [Hui, 2019]. Combining U with our pre-compiled decoding graph $HCLG$ results in a new search space S

$$S \equiv U \circ HCLG \tag{2.6}$$

The decoding task can now be reduced to applying the Viterbi decoding algorithm to find the best path through S, which contains the CD triphones corresponding to HMM states as input labels and words as output labels [Povey et al., 2012]. The result of this decoding step is typically a word lattice with AM and LM probabilities indicating the best paths through the network. The single best hypothesis for the utterance can be extracted from this weighted lattice. Alternatively, it can be processed further with multi-pass decoding techniques, such as re-weighting with more

[5]In the relevant literature the language model is often simply referred to as the 'grammar' and hence abbreviated as G.

advanced LMs (see Section 2.2.2.4). Figure 12 in Appendix B depicts an output word lattice generated by Kaldi.

2.1.7 Evaluating ASR Systems

As previously mentioned, the goal of an ASR system is to output a sequence of words for a given input speech utterance. The standard metric used to evaluate performance of a speech recogniser is word error rate (WER), which indicates the percentage of words incorrectly recognised by the system. WER is formally defined as

$$WER = \frac{S + D + I}{N} \tag{2.7}$$

where S, D and I denote the minimum number of substitution, deletion and insertion operations required to match the hypothesis to the reference transcription[6] consisting of N words. In order to calculate these operations, the hypothesis and reference transcriptions are first aligned using dynamic programming techniques.

While WER is rather straightforward to calculate and provides a reasonable indication of how well our system recognises words in continuous speech, it also has a major drawback. WER assigns equal weighting to every word in the transcription [Jurafsky and Martin, 2008, ch. 9]. As a consequence, the German word *ein* incorrectly recognised as *eine*, (masculine/neuter nominative 'a' vs. feminine nominative 'a') would be penalised the same as if the system outputted the word *Fahrrad* ('bycicle') instead of *Pfarrer* ('priest'). Clearly the latter of these two errors would have a much more significant effect on the semantic interpretation of the transcription than the former. Thus, depending on the intended application of the ASR system, variations of the standard WER metric may be more appropriate.

A simple variation of WER is character error rate (CER), which considers errors at the character level instead of the word level. CER would successfully indicate a larger error for *Fahrrad* – *Pfarrer* (71%) than for *eine* – *ein* (25%), however, when applied to an entire utterance, CER fails to provide us with a good idea of how severe the errors are for the semantic representation of the utterance. Therefore, CER is most suited to evaluating performance for languages like Chinese, where a single character or grapheme can represent an entire concept or word.

One popular adaption to WER is the concept error rate metric, which considers er-

[6]This corresponds to calculating the minimum edit distance between two strings.

rors based on differences between semantic concepts in the reference and the hypothesis transcription [Jurafsky and Martin, 2008, ch. 24]. While such a measure may be more suitable in the context ASR systems for dialectal variants or highly-inflective languages, where a single lexical item may have numerous potential surface forms, an appropriate knowledge base is required to provide information about a concept and its permissible word forms. For most languages, such a resource is difficult and costly to produce. Therefore, given a particular system and recognition task, using standard WER as a basic metric to compare different techniques in language modelling or acoustic modelling is the most informative measure for assessing changes in performance and ultimately the measure we aim to optimise [Mikolov, 2012].

2.2 Advancements in Language Modelling

In this work, we aim to investigate the performance of different LMs on transcribed Swiss German speech data and how these improvements affect the overall performance of an ASR system. Thus, in this section, we provide an overview of language modelling techniques and how they are typically applied in a speech recognition system. First, we discuss traditional count-based N-gram approaches to language modelling (Section 2.2.1) and then introduce more recent techniques that apply neural networks to estimate the probability of a given word sequence (Section 2.2.2). Finally, in Section 2.2.3 we describe how we can evaluate and compare the performance of different LMs.

2.2.1 Statistical Language Models

Statistical LMs consider words as discrete entities and aim to estimate the probability of a word sequence $P(w_1, w_2, \ldots, w_n)$ as the joint probability of seeing each word given all previous words. This can be calculated using the chain rule of probability:

$$\begin{aligned} P(w_1, w_2, ..., w_n) &= P(w_1)P(w_2|w_1)\ldots P(w_n|w_1 w_2 \ldots w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1^{i-1}) \end{aligned} \tag{2.8}$$

However, application of the chain rule suffers a major setback when it comes to long sequences in natural language. Since natural language is highly productive,

new utterances are produced all the time. Even with a huge corpus containing a billion sentences, we will never see sufficient examples of all possible context histories w_1^{i-1} with which we can make reliable probability estimates for the current word w_i [Jurafsky and Martin, 2008, ch. 4]. Therefore, instead of calculating the likelihood of a word conditioned on the entire preceding sequence, we adopt an N-order Markov assumption (see Section 2.1.3.1), which estimates the probability of w_i conditioned on a much shorter context history of $N-1$ words. Thus,

$$P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-N+1}^{i-1}) \tag{2.9}$$

In language modelling, a typical value for N is three or four. Looking at a large corpus, we can count all N-gram co-occurrences and estimate these conditional probabilities using maximum likelihood estimation (MLE) [Jurafsky and Martin, 2019, ch. 3]:

$$P(w_i|w_{i-N+1}^{i-1}) = \frac{C(w_{i-N+1}^{i-1}w_i)}{C(w_{i-N+1}^{i-1})} \tag{2.10}$$

Even still, due to the inherent data sparsity in natural language, it is unrealistic to expect that all possible N-gram sequences will be seen in a training corpus. This results in a complication when we want to apply a statistical LM to new data. For example, if the correct transcription of a test utterance contains a word sequence $w_{i-N+1}^{i-1}w_i$ not seen in training, the maximum likelihood probability of this word sequence will always be 0 and the system will never be able to predict the correct word sequence [Chen and Goodman, 1999]. To account for these inevitable unseen N-grams, we need to apply smoothing.

2.2.1.1 Smoothing

The goal of smoothing is to take a small amount of probability mass from frequently seen N-grams and to redistribute it among unseen events [Goodman, 2001; Mikolov, 2012]. Given that it is crucial in statistical language modelling in order to avoid null probabilities, a number of different smoothing techniques have been proposed. A detailed overview is given in Chen and Goodman [1999].

The most basic smoothing technique is Laplace smoothing, or 'Add+k smoothing'. In this approach, we simply add a constant, k, to all N-grams counts regardless of whether they are novel or not. In order to maintain normalised counts, we also add

the total number of possible words to the denominator, represented as $|V|$.

$$P_{Lap}(w_i|w_{i-N+1}^{i-1}) = \frac{C(w_{i-N+1}^{i-1}w_i) + k}{C(w_{i-N+1}^{i-1}) + k|V|} \tag{2.11}$$

However, simply adding a constant to all N-gram occurrences tends to overestimate rare events, and as a consequence, Laplace smoothing performs poorly for language modelling in practice [Chen and Goodman, 1999; Jurafsky and Martin, 2019, ch. 3]. Some more advanced smoothing techniques can be divided into two main categories: backoff and interpolation.

Backoff techniques, such as Katz Backoff smoothing [Katz, 1987], rely on higher-order N-gram counts if they are available, otherwise, the context history is gradually shortened to consider lower-order N-grams (all the way down to unigrams) until a probability for the current word can be estimated. Thus, the probability for a given word is calculated with a trigram LM as

$$P_{BO}(w_i|w_{i-N+1}^{i-1}) = \begin{cases} \delta(w_i|w_{i-N+1}^{i-1}), & \text{if } C(w_{i-N+1}^{i}) > 0 \\ \alpha(w_{i-N+1}^{i-1})P_{BO}(w_i|w_{i-N+2}^{i-1}) & \text{otherwise.} \end{cases} \tag{2.12}$$

Here, δ denotes a discounted probability for higher-order N-grams, and α is a scaling factor applied to lower-order N-grams. The discounting value ensures that some probability mass is set aside to account for estimating valid probabilities for lower-order N-grams and is typically determined using the Good-Turing backoff algorithm [Jurafsky and Martin, 2019, ch. 3].

Interpolation techniques describe methods which combine the probability estimates for all N-gram counts (e.g. unigram, bigram and trigram), even when higher-order N-grams have been seen in training and thus have a non-zero count. Each probability value is multiplied by a weighting factor, λ, such that $\sum_{k=1}^{n} \lambda_k = 1$. This weighting factor can be optimised for each N-gram according to a held-out data set [Jurafsky and Martin, 2019, ch. 3]. Again considering a trigram model scenario, the probability of a word is formally defined as

$$\hat{P}(w_i|w_{i-N+1}^{i-1}) = \lambda P(w_i|w_{i-2}w_{i-1}) + \lambda P(w_i|w_{i-1}) + \lambda P(w_i) \tag{2.13}$$

The most popular smoothing technique in the context of ASR is modified Kneser-Ney (modKN). ModKN was proposed by Chen and Goodman [1999] as an extension of Kneser-Ney (KN) smoothing, originally introduced by Kneser and Ney [1995].

The original KN smoothing approach is a backoff technique that introduces an element of word co-occurrence. Instead of estimating the unigram probability of a word $P(w)$ based on the sheer number of times w occurs, KN estimates it based on the number of different context histories w completes. This provides us with an idea of how likely w is to occur in a novel N-gram and is known as the continuation probability. Although originally a backoff technique, an interpolated version of KN smoothing has been shown to achieve better performance [Jurafsky and Martin, 2008, ch. 4]. Interpolated KN can be described formally as

$$P_{KN}(w_i|w_{i-N+1}^{i-1}) = \frac{max(C_{KN}(w_{i-N+1}^{i}) - d, 0)}{\Sigma_v C_{KN}(w_{i-N+1}^{i-1}v)} + \lambda(w_{i-N+1}^{i-1})P_{KN}(w_i|w_{i-N+2}^{i-1}) \quad (2.14)$$

where d is an absolute discounting value, $\lambda(w_{i-N+1}^{i-1})$ is the lower-order weighting factor and $P_{KN}(w_i|w_{i-N+2}^{i-1})$ is the lower-order continuation probability. C_{KN} depends on whether we are counting the highest-order N-gram or one of the lower-order N-grams, thus

$$C_{KN}(\bullet) = \begin{cases} Count(\bullet) & \text{for the highest order} \\ ContinuationCount(\bullet) & \text{for lower orders} \end{cases} \quad (2.15)$$

where the *ContinuationCount* is simply the number of unique single word contexts for the random variable $\bullet$ [Jurafsky and Martin, 2019, ch. 3].

ModKN applies the same logic but uses three different discount parameters where the original KN smoothing uses just one. ModKN uses discount values d_1, d_2 and d_3+ for N-grams seen in the training corpus one, two and three or more times, respectively. Chen and Goodman point out that this modification is motivated by the observation that "the ideal average discount for n-grams with one or two counts is substantially different from the ideal average discount for n-grams with higher counts" [1999, p. 370]. The authors also show that it consistently outperforms all other smoothing methods.

A related smoothing technique, which has been compared to modKN (see Rusli [2017]; Hasan et al. [2012]), is Witten-Bell (WB) smoothing, proposed by Witten and Bell [1991]. WB smoothing has its roots in text compression and has been shown to work well when dealing with extremely sparse data [Jayaweera and Dias,

2015]. WB smoothing can be formally defined as

$$P_{WB}(w_i|w_{i-N+1}^{i-1}) = \lambda_{w_{i-N+1}^{i-1}} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1 - \lambda_{w_{i-N+1}^{i-1}}) P_{WB}(w_i|w_{i-N+2}^{i-1}) \quad (2.16)$$

Here, the scaling factor, $\lambda_{w_{i-N+1}^{i-1}}$, indicates the probability of using a particular N-gram model and is estimated by counting the number of unique words that follow a given context history in the training data [MacCartney, 2005; Chen and Goodman, 1999]. Thus, it uses a continuation count, similar to that used in KN smoothing, to ensure that we rely on higher-order N-gram counts if they exist and mix in lower-order N-gram information accordingly. The higher this value, the less we need to rely on lower-order counts.

Both modKN and WB smoothing can be implemented using interpolation and back-off techniques. In Chapter 6 we compare these different smoothing approaches for modelling Swiss German.

2.2.2 Advanced Language Modelling

While smoothing techniques generally aim to deal with the data sparsity issue associated with statistical N-gram LMs, another issue relating to long-distance dependencies in natural language remains [Goodman, 2001]. Because of the Markov assumption, standard N-gram models are only capable of capturing language dependencies spanning the preceding $N-1$ words. For example, in the English utterance, 'it was a hot day so I went swimming', the word 'swimming' can be seen to be dependent on the previous word 'hot'. The distance between these words is four ('day so I went'). A trigram model, however, only considers the previous two words and thus has no indication that 'swimming' may be more likely than, say, 'skiing' in this context.

Given that such dependencies are extremely frequent and can cover arbitrarily long spans, a number of advanced N-gram models have been proposed in an attempt to capture this kind of information. One popular technique is the cache model, which applies the hypothesis that a word is more likely to be repeated by a speaker in an interaction [Goodman, 2001]. Cache models aim to dynamically estimate N-gram probabilities based on what has already been produced and are usually interpolated with regular N-gram models [Mikolov, 2012]. While cache models have been shown to significantly improve LM performance in terms of perplexity (PPL) (see Section 2.2.3), these improvements are typically not reflected when applied

to real-world tasks such as ASR. This is due to the fact that once an erroneous word is selected during decoding, it has a higher chance of being selected again, potentially propagating recognition errors further [Goodman, 2001; Mikolov, 2012]. Indeed the biggest breakthrough in language modelling has been the application of neural networks.

2.2.2.1 Neural Networks

A neural network consists of many "small computing units, each of which takes a vector of input values and produces a single output value" [Jurafsky and Martin, 2019, ch. 7, p. 123]. These small computing units are often referred to as 'neurons' since they reflect our understanding of neurons in the human brain. The neurons are arranged in layers, each of which can be thought of as a vector, whose dimensionality is equal to the number of neurons in the layer. A network consists of a single input layer X, one or more hidden layers $H^{[i]}$, and an output layer Y. Each neuron in the hidden and output layers is associated with a weight vector and a bias value. These weight vectors are combined to form a single weight matrix for each layer.

Figure 4 depicts a very basic feed-forward neural network with an input layer of size two, a hidden layer consisting of three neurons, and an output layer with only a single neuron. Here, the weight matrices W and U are represented as edges connecting nodes in each layer.

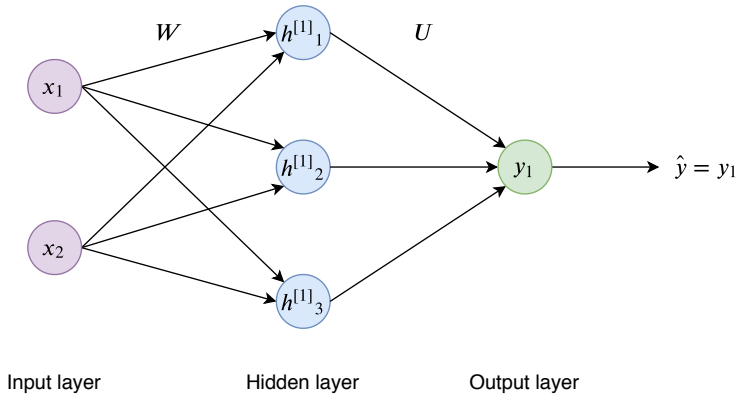

Figure 4: Architecture of a small feed-forward neural network (adapted from Lusetti [2018])

In such a network, information flows along the connections from the input layer to the output layer (in this case, left to right), hence the name 'feed-forward'. A neuron in the hidden layer receives a weighted input value from each actively connected neuron in the previous layer. These input values are then summed and a non-linear

activation function is applied to the result, which then gets passed on as the neuron's output. In this case, the output values of $H^{[1]}$ are passed to a single output neuron y_1, where another non-linear function is applied to produce the final output value $\hat{y}$.[7]

The non-linearity applied to each neuron in the hidden layer enables an NN to construct arbitrarily complex functions. As a result, NNs are capable of modelling extremely complex relationships which can not be modelled by simpler linear models, such as linear regression, thus making them particularly strong classifiers. Popular activation functions include the hyperbolic tangent function (tanh), which squishes an input value between -1.0 and 1.0 following a smooth curve, or the rectified linear activation function (ReLU), which squishes values between 0 and 1 [Goldberg and Hirst, 2017]. The output layer is typically characterised by the non-linear sigmoid function for binomial classification or softmax in the case of multinomial classification, where the output layer consists of multiple neurons. While the sigmoid function is commonly used to transform any real number input into a probability value between 0 and 1, applying softmax to a vector of K real values transforms it into vector indicating a proportional probability distribution, where $\sum_{i=1}^{K} k_i = 1$.

Training a NN refers to the process of finding optimal values for the randomly initialised connection weights and an additional bias value (not depicted in Figure 4), which assists in fitting the model to the data. This is performed by iteratively running labelled training data through the network, and allowing the network to output a value $\hat{y}$, corresponding to the data labels. A loss, or error, function $L(\hat{y}, y)$ provides a measure for how accurately the model predicts the true label for a given data point. The gradient of this loss is calculated with respect to all parameters in the network using the Backpropagation algorithm, which efficiently applies the chain-rule of differentiation through the network (from right to left) [Goldberg and Hirst, 2017]. The resulting gradients indicate the direction in which to update the model's parameters in order to reduce the overall loss, allowing productive updates to be performed according to a specified learning rate η. For efficiency, these calculations are typically performed iteratively over multiple sub-samples, or 'batches', of the training data. Updating the parameters in this fashion is known as stochastic gradient descent.

[7] Such a network may be used to assign binary output values of either 0 or 1 for a simple classification task.

2.2.2.2 Neural Network Language Models

The application of NNs in language modelling was first done by Bengio et al. [2003], who proposed the basic feed-forward NNLM. This model is similar to the N-gram model described above in that the probability of the word w_i is conditioned on the basis of a fixed-length context history w_{i-N+1}^{i-1} (see Eq. 2.9). However, the NNLM differs significantly in regards to the way in which words are represented, allowing for improved generalisation without complex smoothing techniques and reducing the number of free parameters used to effectively model language.

Traditional N-gram models represent words as discrete random variables. This poses a fundamental problem known as the curse of dimensionality, which effectively limits the size of the corpora for training N-gram language models [Jing and Xu, 2019]. Given a vocabulary of size $|V|$, in order to model the joint distribution of an N-gram sequence, there are potentially $|V|^N - 1$ free parameters [Bengio et al., 2003], resulting in models quickly becoming unwieldy as we increase the size of the vocabulary or the context history length. Additionally, by modelling language in this discrete space, it is difficult to achieve reliable generalisation since any change to a random variable in this space can result in an arbitrary change in the resulting N-gram probabilities [Schwenk, 2007]. Thus, the N-gram LM fails to capture an intrinsic understanding of the strong semantic similarity between the utterances 'she fed the dog' and 'he fed a cat'.

In contrast, NNLMs map words to a continuous representation as a dense vector of real numbers which can be efficiently learned by the LM itself. This type of representation has since become known as a word embedding and has proven to be very effective in many NLP applications due to the fact that it is capable of encoding relevant semantic information about a word based on its distribution in large corpora. As a result of this representation, NNLMs are better at generalising to unseen context histories since similar words receive similar feature representations (e.g. 'she' and 'he', 'the' and 'a', 'dog' and 'cat') and the probability function is now a smooth function of these feature values [Bengio et al., 2003]. Thus, a small change in this continuous feature space is reflected by a small change in the output N-gram probabilities.

The basic architecture of the model proposed by Bengio et al. [2003] is depicted in Figure 5. Each input word is initially represented by a one-hot vector, which is a sparse, binary vector whose dimensionality is equal to the size of the vocabulary $|V|$ and contains a one at the index of the particular input word and zeros in all other positions. The one-hot vectors corresponding to the words in the context

history w_{i-N+1}^{i-1} are projected through a shared weight matrix, mapping them to their dense vector representations, which are concatenated together to produce the projection layer – a single vector representing the context history. The projection layer is multiplied by another weight matrix providing input to the hidden layer, where a non-linear activation function is applied to each neuron. The output is then multiplied by a final weight matrix resulting in the output layer vector, whose dimensionality is equal to $|V|$, with node indices corresponding to word indices int he vocabulary. The softmax function is used to ensure that this output layer indicates a valid probability distribution over all vocabulary items w_i.

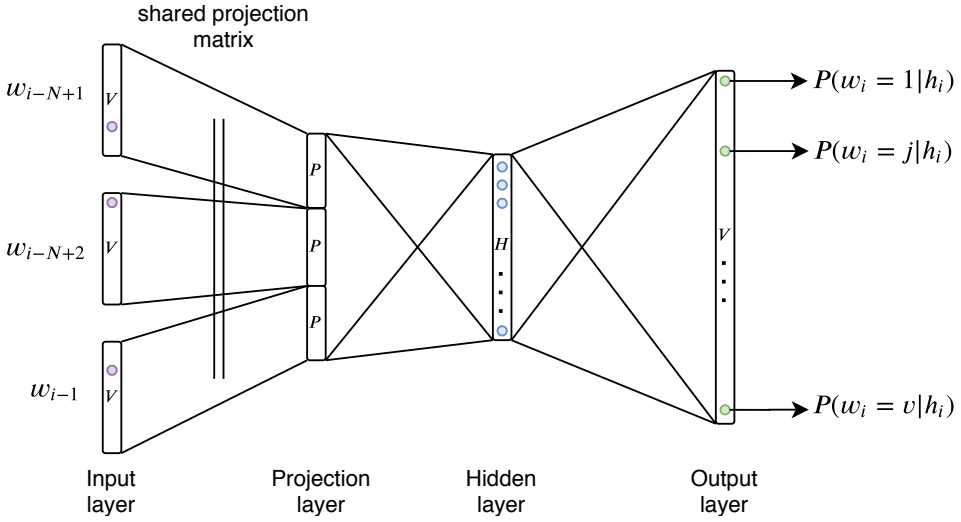

Figure 5: Architecture of a feed-forward neural network language model. Here, V indicates the size of the vocabulary, H is the size of the hidden layer and P is the size of a word's dense vector representation after being projected into a continuous space (adapted from Schwenk [2007]).

2.2.2.3 Recurrent Neural Network Language Models

While the feed-forward NNLM provides a unique solution to attaining better generalisation from a LM, it still relies heavily on a limited context history and thus does not address the problem related to long-distance dependencies in natural language. Recurrent neural network language models (RNNLMs), as introduced by Mikolov et al. [2010], aim to solve this problem. RNNLMs replace the fixed-length context history input w_{i-N+1}^{i-1} with a recurrent connection to the hidden layer from previous timesteps.

Figure 6 depicts the simplest model, or Elman RNNLM, named after the Jeffrey Elman, who first applied this idea to language data in 1990. The input to the

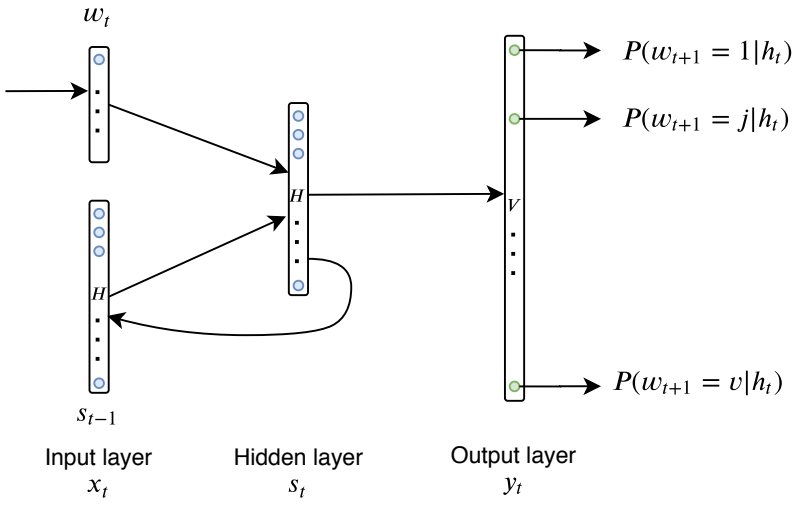

Figure 6: Architecture of a simple recurrent neural network language model as proposed by Mikolov et al. [2010]. Here, H represents the size of the hidden layer and V indicates the size of the vocabulary.

RNNLM is a concatenated vector consisting of the vector $w(t)$, representing the current word, and the output of the context layer from the previous timestep s_{t-1} [Pappas and Meyer, 2012]. After training, the output layer $y(t)$ represents the probability of the current word given the previous word and all the information seen previously, $P(w_{t+1}|w_t, s(t-1))$ [Mikolov et al., 2010].

The training of an RNNLM is similar to that of the standard feed-forward NNLM. However, for RNNLMs, the Backpropagation algorithm used to update parameters in the network is extended to Backpropagation Through Time (BPTT). The main difference between BP and BPTT is that the latter begins by unrolling the network's previous N timesteps into what essentially represents a deep neural network with N hidden layers and their respective weight matrices [Jurafsky and Martin, 2019, ch. 9]. Once unrolled, the parameters of these weight matrices can be updated.

The size of N is limited, in theory, only by the number of context items already seen, i.e. the number of previous timesteps. However, in Elman RNNLMs, large Ns can quickly lead to exploding or vanishing gradients.[7] Exploding gradient describes the problem of the gradients becoming so large that they can no longer be represented by the computer, whereas vanishing gradient describes the opposite scenario, where gradients become so small that changes can no longer be made to the model's

[7]Other neural architectures, such as Long-Short Term Memory (LSTM) and the simpler Gated Recurrent Unit (GRU), have since been developed to provide more control over the flow of information in the network using 'gates', operated by additional trainable parameters.

parameters. Mikolov [2012] proposes using truncated BPTT to deal with this issue, effectively limiting the number of unrolled timesteps, N, to a predefined threshold. He notes that, for word-level RNNLMs, a value of five seems to be sufficient and interestingly, even with this threshold, the model still seems to be capable of capturing information from more than the previous five timesteps.

Thanks to the recurrent connections between hidden layers, RNNLMs offer the ability to model language using unlimited context information that can be retained and cycle through the network indefinitely [Mikolov et al., 2010]. Additionally, RNNLMs simplify the modelling process by reducing the number of hyperparameters that need to be set before training. In a feed-forward NNLM, the context history (N), the size of the projection layer (P) and the hidden layer (H) all need to be selected before training. For RNNLMs, only the latter needs to be chosen in advance [Mikolov et al., 2010].

2.2.2.4 Applying Neural Language Models in ASR

While NNLMs and RNNLMs have proved to be powerful tools for language modelling in a broad range of NLP tasks, finding ways to effectively exploit these more powerful models in conventional ASR systems remains an area of active research (e.g. Lecorvé and Motlícek [2012]; Arisoy et al. [2014]; Beck et al. [2019]). Unlike N-gram LMs, neural models do not store N-gram probabilities in a structured format. Instead, they take a given input and produce a probability distribution over all possible words to predict the next word. As discussed above, contemporary ASR systems are built on the basis of a static decoding graph $HCLG$. Constructing this graph involves repeatedly looking up all context histories on the graph in a table and extending them with the next word according to the LM's probabilities. Initial attempts by Schwenk [2007] to incorporate NNLMs directly during decoding proved unviable due to the time required to calculate predictions for the next word given a particular context history. Moreover, because an "RNNLM theoretically encodes infinite history lengths, it is virtually impossible to compile it to a static decoding graph" [Xu et al., 2018, p. 1].

The most common technique for incorporating these more powerful LMs in ASR systems is multi-pass decoding Xu et al. [2018]. In multi-pass decoding, a simple N-gram LM is used in the first pass to generate an intermediary output of potential hypotheses. Then, in a second pass, a more powerful LM, such as an RNNLM, can be used to rescore or rerank these hypotheses.

The intermediary outputs produced in the first pass can take two forms: a word

lattice or an N-best list. The first of these can be thought of as a directed graph of nodes and arcs. Nodes typically represent points in time and arcs represent possible output words between them. Figure 7 shows a simplified example of a word lattice for the German utterance '*Ist es ein Thema gewesen?*' ('Was it an issue?'). The LM probabilities associated with each word, as calculated by the N-gram LM, are stored in the lattice. These scores can then be partially subtracted and interpolated with new scores assigned by a more powerful LM to update path probabilities.

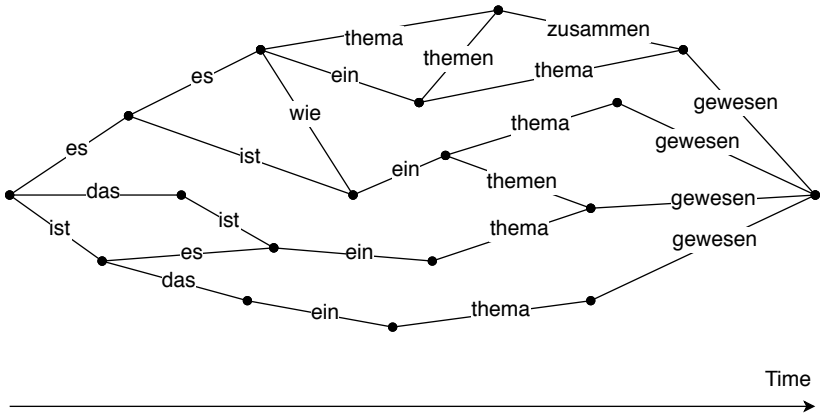

Figure 7: A constructed word lattice for the utterance '*Ist es ein Thema gewesen?*' ('Was it an issue?') (adapted from Gales and Young [2007]). For simplicity AM and LM scores are not shown here. The true word lattice as outputted by the system is depicted in Figure 12 in Appendix B.

The second type of intermediary output, N-best lists, are simply lists containing the N most probable hypothesis transcriptions for a given input utterance. Each hypothesis is listed together with a corresponding AM and LM score for the entire word sequence. The existing LM score can be updated by allowing a more powerful LM to assign a new probability score to the whole utterance. Hypotheses are then reranked according to the new scores and the utterance with the highest is then selected as the 1-best output hypothesis.

Ideally, applying a more powerful LM to either of these intermediary representations will enable the system to reduce the likelihood of any incorrect hypotheses and increase the probability of the correct hypothesis. However, there are two major drawbacks to multi-pass decoding. Firstly, decoding time is increased considerably thanks to the use of an intermediary output for the second decoding pass. Secondly, multi-pass decoding is not guaranteed to help; if the correct hypothesis utterance is not contained within the lattice or in the N-best list after the first pass, it is not possible to produce it in the second pass. Thus, the system is still largely reliant on the N-gram model used to construct the original decoding graph.

2.2.3 Evaluating Language Models

So far we have described a number of techniques introduced to improve the performance of LMs but we have not yet discussed how we can actually measure performance of these models. Recall that the goal of a LM is to assign a probability to a given word sequence. Given a held-out sample of the language in the form of a test set, we can measure the inverse probability of the test set, as calculated by a LM, normalised by the total number of words N:

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} \tag{2.17}$$

This measure is known as perplexity (PPL) and is the most commonly used measure for evaluating and comparing the performance of different LMs [Goodman, 2001]. PPL is an intrinsic evaluation measure which indicates the "amount by which the language model reduces the uncertainty about the next word" [Manning and Schütze, 1999, p. 510].

PPL is equivalent to measuring the cross entropy of the test data given a model but has a number of properties which make it a more attractive measure for reporting LM performance.[8] As discussed by Mikolov [2012], PPL scores are typically whole numbers within the range of 100 to 200, whereas the corresponding entropy values of 6.64 bits and 7.64 bits are more cumbersome. Thus improvements measured in PPL often simply sound better than those reported using entropy. For example, given these numbers, the reduction in PPL is 50% as opposed to 13% if calculated using entropy values. Since PPL can be easily calculated with Eq. 2.17 and it is closely related to entropy, the model that yields the lowest perplexity on a held out test set is closest to the true model which generated the data, and thus the better model.

While PPL provides a reliable measure for comparing different LMs on held-out test data, it fails to indicate how well a given LM performs in a real-world application, such as ASR or statistical machine translation. Therefore, LMs are typically assessed with PPL as an intrinsic measure as well as an extrinsic measure suitable for the application area. Since contemporary ASR systems rely on multiple components, improvements in LM PPL do not necessarily guarantee system improvements according to word error rate (WER). Nevertheless, Klakow and Peters demonstrate

[8] Even though we are describing a particular type of cross entropy, i.e. a measure of difference between two probability distributions, it is usually referred to simply as 'entropy' in the relevant literature usually [Goodman, 2001].

that there is typically a positive correlation between the two measures.

In Chapter 6, we compare various LMs on transcribed Swiss German speech data, reporting PPL in order to assess performance. Following this, in Chapter 7, we integrate the best performing LMs into different speech recognisers and report WER to evaluate their performance in practice.

3 Challenges for Swiss German ASR

Over the last 60 years, extensive research in ASR for major languages like English has lead to high performing systems capable of achieving relatively low word error rates in a wide range of tasks (see for example Makhoul and Schwartz [1995]; Miao et al. [2015]; Hori et al. [2018]). These successes have lead to its application in many consumer products such as voice-activated personal assistants, in-car systems and dictation software, among others. However, adapting current state-of-the-art systems to non-standardised and low-resource languages poses a number of significant challenges. In this chapter, we describe some of the major challenges we face in performing ASR for Swiss German.

3.1 Spoken Variation

Swiss German describes a group of dialectal variants rather than a single, uniform language. A great deal of spoken variation exists across four major levels of linguistic structure: phonology, lexicon, morphology and syntax. Two large-scale projects have attempted to document all four of these levels, resulting in the Swiss German Dialect Atlas (*Sprachatlas der deutschen Schweiz* (SDS)) and the Syntactic Atlas of Swiss German Dialects (SADS) [Bucheli and Glaser, 2002]. Recent work by Scherrer and Kellerhals [2014] has combined these two distinct resources and digitised the large collection of maps showing the spread of linguistic differences across German-speaking Switzerland in order to facilitate dialectal studies.[1]

Figure 8 provides an areal depiction of two types of linguistic variation in Swiss German based on the newly digitised maps. The image on the left shows an interpolated distribution of eight differing realisations of 'ä' as it occurs in the word *spät* ('late'), while the map on the right shows six major lexical differences for the word *auch* ('also'). As can be seen, there is no single clear overlap between these two particular forms of variation, and certainly none that strictly follows cantonal borders. This shows that linguistic change is often gradual and multifaceted across the

[1] `http://dialektkarten.ch/`

landscape. Therefore, it is often difficult to establish distinct boundaries between different Swiss German varieties or even to identify the exact number of varieties that exist [Scherrer in Fahy, 2016].

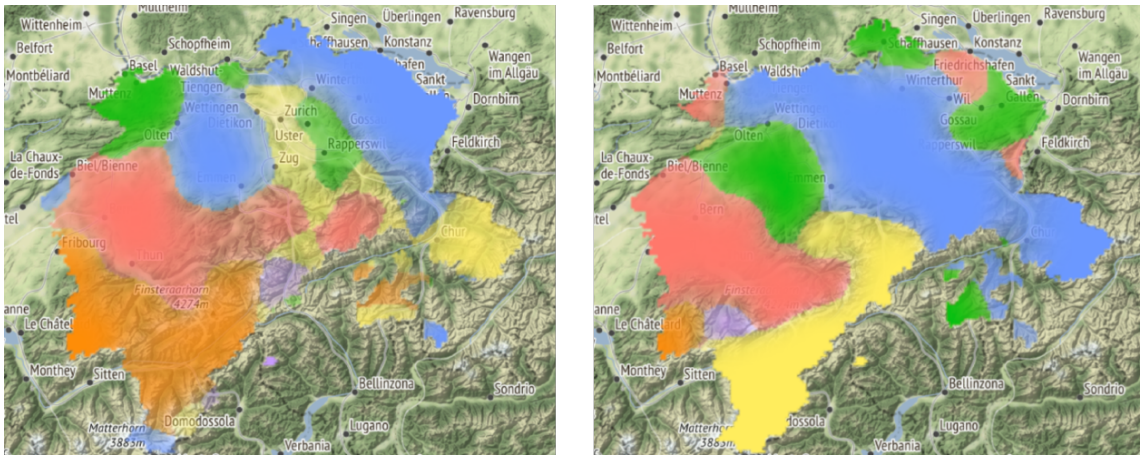

Figure 8: Spoken variation, pertaining to phonology (left) and lexicon (right), according to the digitised version of the SDS and the SADS, available at `http://dialektkarten.ch/` [Scherrer and Kellerhals, 2014].

As a consequence of this spoken variation, any processing of Swiss German language data that faithfully represents pronunciation is bound to suffer from issues relating to a high degree of noise and data sparsity. For acoustic modelling, this poses considerable challenges due to the fact that any given sample of training data will inevitably contain fewer examples, making it more difficult to learn good Gaussian mixture model representations for sounds in the language.

Additionally, a large degree of spoken variation raises a major question regarding pronunciation modelling. Assuming we can determine a single normalised lexical item, it is unclear whether it is best to model all possible pronunciations for a particular word or to approximate these using a single, somewhat 'common' form. Hain [2002] has shown that, in the case of English, using a simpler lexicon, containing only one pronunciation string per word outperforms systems that rely on a more detailed lexicon where all possible pronunciations are present. Hain notes that using multiple pronunciation strings "increases the confusability with other words as the distance in pronunciation between words usually becomes smaller" [2002, p. 130]. However, given that the phonological variation in a typical English ASR task is considerably less than that of any task involving Swiss German, it is unclear whether a more detailed pronunciation lexicon is preferable for Swiss German ASR.

A common method for avoiding some issues when handling dialectal varieties in ASR is to train a separate speech recogniser for each individual dialect [Elfeky

et al., 2018]. Once multiple models have been trained separately, they can be used in conjunction with dialect identification classifiers in order to decide which model should be used to decode a particular spoken utterance. However, naturally, such an approach requires a substantial amount of dialect-specific speech and language data for training multiple acoustic and language models. In the case of low-resourced Swiss German, the scarcity of relevant data across all variants renders such an approach unfeasible at this stage.

3.2 Written Variation

Due to the fact that there is no single dominant pronunciation for Swiss German, it has retained its dialectal status and remains a predominantly spoken language, while Standard German has long served as the official written form. As mentioned previously, Kolde [1981] describes the situation as one of 'medial diglossia', where multiple languages coexist side-by-side, each with its own specific function.

However, the boundaries of this medial diglossia are not at all set in stone nor completely static. Ricker-Abderhalden [1986] describes a noticeable increase in the use of Swiss German where Standard German had previously been the norm. She describes a *mundartwelle* ('dialect wave') that pertains not only to spoken communication but also to the written domain, evidenced by the influx of *mundartliteratur* ('dialect literature') and its use in advertising [Ricker-Abderhalden, 1986]. But perhaps by far the biggest 'wave' can be attributed to the rise of computer-mediated communication (CMC) [Siebenhaar, 2006]. The popularity of written communication through SMS and various forms of social media has lead to Swiss German being increasingly written, or perhaps more accurately *typed*, by younger generations.

Given that there is no orthographic norm or spelling rules in place, the written form of Swiss German adopted for communication can be described as 'spontaneous', often reflecting the author's personal preferences and influences. Major influencing factors can include their local dialect, personal interpretation of grapheme to phoneme rules and orthographic principles, and regional writing traditions, among others [Siebenhaar, 2006]. These factors differ not only from speaker to speaker but can also differ for a single speaker over time. As a consequence, spontaneously written Swiss German exhibits a great deal of surface-level variation. While it generally does not impede communication between native speakers [Siebenhaar, 2006], extensive spelling and lexical variation poses seemingly insurmountable challenges for automatic processing techniques such as ASR. Luckily, however, other textual representations exist.

3.2.1 Dieth Orthography

One writing 'system' that was intended to accommodate all dialects of Swiss German was proposed by the Swiss Linguist Eugen Dieth. The Dieth orthography [Dieth, 1986] outlines a method for writing *all* Swiss German variants in a manner which is close to the true sound of the spoken language, while still being easily readable. His aim was to establish a comfortable middle ground between the orthography of Standard German, which "reflects the pronunciation very poorly"[2] [Dieth, 1986, p. 14], and a true phonetic representation that would make it difficult for the lay person to read.

Dieth specified a number of basic principles for writing Swiss German varieties flexibly without introducing entirely new characters to the Standard German character set. He provided a number of examples demonstrating the 'correct' spelling of frequently used words and expressions in a variety of dialects, outlining some pragmatic solutions for difficult or questionable situations. At the same, the Dieth orthography allows an author to write according to what they hear. Thus, it can be seen more as a set of guidelines than a true orthographic system.

Dieth also distinguishes between a narrow spelling (*enge Dieth-Schreibung*) and broad spelling (*weite Dieth-Schreibung*). The former makes use of additional diacritics to represent certain phonetic differences, such as open/closed vowels, nasalisation, vocalisation and palatalisation, while the latter omits such fine-grained details for the sake of simplicity. Consequently, a broad Dieth spelling demands a higher degree of familiarity with the actual dialect in order to be able to reproduce the sounds accurately from a written text.

Despite the fact that the Dieth orthography provides a loosely phonemic spelling system that draws on familiar spelling norms from Standard German, it has never been widely adopted by native speakers. As with the adoption of any standardised orthography, learning to write according to the Dieth guidelines requires training and practice. As a result, it is primarily used by trained linguists looking to document spoken Swiss German. Inspecting language corpora that employ these guidelines reveals that spelling is often inconsistent due to the many arbitrary decisions that transcribers need to take while annotating. Nevertheless, since the Dieth orthography restricts the space of possible surface forms it provides a viable, albeit extremely scarce, textual representation for Swiss German NLP and ASR.

[2]Translated from the original, "Die heutige neuhochdeutsche Schreibung gibt die Lautung nur noch sehr unvollkommen wieder" [Dieth, 1986, p. 14].

3.2.2 Normalised Swiss German

Another possible method for representing Swiss German in text corpora is as a normalised form of the original surface text. Normalisation describes the task of "mapping variants of what can be identified as the same word to a single representation" [Samardžić et al., 2015, p. 1]. Automatic approaches to normalisation have been extensively investigated for various Swiss German corpora in order to cope with the large amount of surface form variation (see Ueberwasser and Seminar [2013]; Samardžić et al. [2016]; Lusetti [2018]).

Samardžić et al. [2015] propose a semi-automatic approach for normalising Dieth-transcribed spoken Swiss German using character-level machine translation. This normalised representation largely resembles Standard German spelling, yet differs in three major aspects.

Firstly, lexical mismatches occur where a Swiss German word has no etymologically related counterpart in Standard German. For example, the Swiss German form *öpper* corresponds to Standard German *jemand* ('someone') and Swiss German *velo* to Standard German *Fahrrad* ('bicycle'). Samardžić et al. [2015] handle such cases by normalising these surface forms with a reconstructed common Swiss German form. Therefore, *öpper* is normalised as *etwer* and *velo* as *velo*. Additionally, it should also be noted that lexical mismatches can potentially lead to ambiguous constructions. For instance, the normalised form *vorig* is commonly used in Swiss German in the sense of 'remaining', while in Standard German it exists as 'previous' [Scherrer et al., 2019].

Secondly, discrepencies between word boundaries occur due to extensive cliticisation of articles and pronouns in Swiss German. For example, the Swiss German form *hämmer* corresponds to Standard German *haben wir* ('have we') and Swiss German *echli* to Standard German *ein bisschen/wenig* ('a bit/little'). In such cases, the normalisation approach proposed by Samardžić et al. [2015] adopts the Standard German spelling conventions and splits a single-token surface form into its corresponding Standard German constituents.

Lastly, further discrepancies exist relating to morphosyntactic constructions in Swiss German that are not used in Standard German. For example, the Swiss German form *dure* ('through') would correspond to Standard German *durch* + direction and is thus normalised to *durchhin* instead of the true Standard German form *hindurch* [Clematide et al., 2016].

Due to these discrepancies, it is important to note that the resulting normalised

representation is by no means intended to provide a genuine translation into Standard German, but rather a word-level annotation layer designed to aid automatic processing of Swiss German text.

Table 1 demonstrates quite clearly the major differences between the three textual representations of Swiss German described above. Here, we provide just a sample of possible written forms for the short utterance "*Wo sind Sie gewesen?*" ("Where were you?"). The top portion of the table shows forms produced by native speakers. These were collected by asking a group of speakers to write the utterance spontaneously, as they would in a regular text message.[3] As can be seen, no two utterances overlap. Spellings differ considerably in terms of clitisation of the pronoun and grapheme combinations, some of which are not found in Standard German (e.g. 'xs'). If we compare these spontaneously written forms to three possible forms represented with the Dieth orthography, we see a considerable reduction in the number of different graphemes used to represent the main sounds, but variant spellings of the words are still present (e.g. *ggsii* vs. *gsii*). Comparing these further with the normalised representation at the bottom of the table shows how all variation is removed and, in this particular case, spelling resembles Standard German.

Spontaneously Written	wo sind si xii
	wo siter gsi
	wo sind sie xsi
	wo sitr gsi
	wo siter gsi
	wo sind sie gsii
	wo send sie gsi
	wo sendsi gsi
	wo syt dihr gsy
Dieth Transcribed	woo sind si gsii
	wo sind si gsii
	wo sind sii ggsii
Normalised	wo sind sie gewesen

Table 1: The utterance "*Wo sind Sie gewesen?*" ("Where were you?") as realised in Swiss German with three possible textual representations.

This small example demonstrates some of the challenges involved with automatic processing of Swiss German. Given the space of possible textual representations

[3]Some speakers were from different parts of Switzerland and thus speakers of different dialects, while others had grown up next door to each other and thus speak the same dialect.

for Swiss German, converting speech to text first involves defining a suitable target textual representation. Then, in order to gain accurate output transcriptions, we need to be able to model the language on the basis of this textual representation. While probabilistic LMs can typically learn this modelling through examples found in large text corpora, we are severely limited in the amount of language corpora available for Swiss German due to a long-standing medial diglossia. Furthermore, extensive variation in potential surface forms leads to a high degree of noise and data sparsity in any Swiss German corpus, making this task considerably more difficult.

3.3 Evaluating ASR for Non-Standardised Languages

Another challenge facing ASR for languages with non-standard orthography is how to evaluate system performance. As mentioned in Section 2.1.7, the standard metric for evaluating ASR system output relies on word-level string matches between a reference transcription and a system output transcription. However, in the context of non-standardised languages, there is rarely ever one single transcription that can be considered the 'correct' transcription. As shown in Table 1, Dieth-transcribed Swiss German is inherently lexically diverse with a number of potentially legitimate spellings for any given word. Relying on such a flexible orthography for ASR system evaluations will result in the system being unfairly penalised for producing alternative yet permissible spelling variants.

Conducting manual evaluations with human annotators can be seen as a valid solution for assessing system performance yet this is a prohibitively time consuming and expensive option. Therefore, in order to quickly compare the performance of multiple systems, an appropriate automatic evaluation metric is required. Ali et al. [2017] propose an adaption of WER called 'WERd' (word error rate for dialects) for evaluating ASR for non-standardised dialects of Arabic. Using a large lookup table of normalised word forms mapped to possible surface forms, they essentially allow a hypothesis word to be considered correct if it shares the same normalised form as the corresponding word in the reference transcription. The authors describe this simple adaption of the WER metric as a promising approach, avoiding the problems of using standard WER on dialectal and non-standardised language varieties. Furthermore, such an approach is practically free compared to the time and effort required to conduct manual evaluations for multiple ASR systems and thus facilitates experimentation and development in ASR for non-standardised languages. In Section 5.3.1, we propose a similar metric that exploits the word-level normalised annotations in our corpus to evaluate non-standardised Swiss German ASR output.

4 Corpora, Tools and Resources

The crux of most modern NLP technologies, especially ASR, is the amount of available language data. Of the approximately 7,000 languages spoken in the world today, only a small handful of them have the resources necessary for developing a broad range of language technology applications. Swiss German is one of the many low-resource languages, with only a few small data sets available for specific NLP applications. In this chapter, we describe the collection of Swiss German and Standard German corpora as well as additional linguistic resources used for the development of our ASR system and our experiments. In addition, we present the tools used in this project.

4.1 The ArchiMob Corpus of Spoken Swiss German

The ArchiMob corpus [Samardžić et al., 2016] is a large corpus of spoken Swiss German. Totalling roughly 70 hours of manually transcribed continuous speech, it constitutes the only publicly available resource suitable for the training and development of Swiss German ASR systems and thus forms the basis of our investigation into ASR for Swiss German.

ArchiMob was built on the basis of the oral history project *Archives de la Mobilisation* (Archimob)[1], which was launched by the Filmmaker Frédéric Gonseth, who set out to record first-person accounts of daily life, conceptions and experiences of Swiss citizens during the Second World War. Gonseth conducted a total of 555 interviews between 1999 and 2001 with informants from all over Switzerland. The interviews typically last one to two hours in length and were conducted with a semi-directive technique, allowing for a relatively free flow of ideas from the informants. Of these 555 interviews, 300 were conducted in Swiss German, providing an extensive resource for historical and linguistic studies.

[1] We follow the convention of Scherrer et al. [2019] and use *ArchiMob* to refer to the corpus and *Archimob* to refer to the original project, which can be found at `http://www.archimob.ch/d/archimob.html`.

The ArchiMob corpus consists of 44 interviews from the original Archimob project, which have been digitised and manually transcribed according to the Dieth orthographic guidelines (see Section 3.2.1). The work of transcribing took place over four distinct phases between 2006 and 2017 with phases differing from each other in terms of the tools used, the annotators and even the transcription guidelines. As a consequence of the intermittent workflow, inconsistencies exist in the annotations, leading to even more linguistic variation in the corpus.

4.1.1 Transcribing ArchiMob

In the first phase (2006–2012) 16 interviews were manually transcribed without a dedicated tool. The second phase (2011–2013) saw the adoption of the transcription tool FOLKER [Schmidt and Schütte, 2010] and resulted in an additional seven documents. In the third phase (2015), a further 11 documents were transcribed and in the fourth and final phase (2016–2017) the remaining 10 documents were completed. In these last two phases annotations were performed with the popular transcription tool EXMARaLDA [Schmidt, 2012]. In the earliest phases of the ArchiMob transcriptions, an attempt was made to write a partial narrow Dieth spelling, distinguishing vowel openness with additional grave accents (e.g. è, ì, ò, etc.), however, this was eventually abandoned and a broader Dieth spelling was adopted in subsequent phases.

The major benefit of using specialised transcription software is that these tools output time-stamped alignment information for every transcription unit and its corresponding audio segment. Since the first phase was done without specialised software, alignments had to be attained in an additional post-processing step. To do this, the authors used the automatic alignment tool WebMAUS [Kisler et al., 2012], which aligns a speech signal to a corresponding transcription using a process similar to forced Viterbi alignment. The resulting alignments were then manually checked and corrected using EXMARaLDA.

A total of five transcribers worked on the ArchiMob corpus throughout the various transcription phases. Naturally, these annotators were native Swiss German speakers but not necessarily native speakers of the particular dialect they were transcribing. Nonetheless, efforts were made to ensure that the transcriber was sufficiently familiar with whichever dialects they worked on. Annotators were trained to write Swiss German according to the Dieth orthographic guidelines in the hope of getting transcriptions that are as consistent as possible while still reflecting the sound of spoken Swiss German.

The ArchiMob transcriptions also include annotations of non-linguistic units and other conversational markers such as pauses and interruptions. Non-linguistic units can include vocalised noise, such as the speaker clearing their throat or coughing, as well as non-vocalised background noise, such as a passing motorbike or even the sound of body movements of the speaker. Being a corpus of natural spoken language, additional disfluencies such as truncated words and repetitions are frequent. Truncated words, which occur due to interruptions and false starts, are typically annotated, while word repetition events are not marked explicitly in the corpus.

4.1.2 Normalising ArchiMob Transcriptions

In Table 1, we showed how the Dieth transcriptions do indeed help to reduce the extreme variation associated with spontaneously written Swiss German. However, as is also clear from that example, the Dieth spelling does not provide standardisation. A number of factors can be seen to contribute towards a high degree of variation in the corpus transcriptions. These include:

- the extensive regional variation in spoken Swiss German. For example, a speaker from Bern may be recorded saying /hei/ while a speaker from Basel might say something closer to /hɜːn/ for the word *haben* ('have').
- changes in the transcription guidelines over time, particularly in the representation of vowel sounds.
- discrepancies between how different annotators perceive the sounds of the language and how they apply the transcription guidelines.
- inconsistencies in how a single annotator applies the transcription guidelines within a single interview. For example, a manual inspection of one interview reveals five unique surface forms used to represent Standard German *gewesen* ('been') as spoken by the same speaker.[2]

To combat this high degree of variation and to assist with further automatic processing, Samardžić et al. [2016] add a normalised annotation layer using the approach introduced in Section 3.2.2. While such an approach is unlikely to deliver perfect results, Samardžić et al. [2016] report an accuracy of around 84.13% on a held out test set of manually normalised utterances.

To assess the affect of the added normalisation layer, we calculate the type-token ratio (TTR), a simple measure of lexical diversity commonly used in corpus linguis-

[2]These five surface forms are *gsi*, *ggsii*, *gs*, *gsii* and *ggsìì*.

tic studies [Hess et al., 1984]. TTR considers the number of unique word forms (types) in relation to the total number of words (tokens). TTR is often reported as a percentage. For example, a TTR of 100% would imply that every word in the text appears only once. Comparing the TTR for both the Dieth surface-from transcriptions and the additional normalised annotation layer (see Table 2) shows a reduction of lexical variance as measured by TTR from 7.9% to 5.2%. While this reduction seems rather small at first glance, it corresponds to a relative reduction of 34.2% in the amount of lexical variation in written Swiss German afforded by the automatically normalised annotation layer.

	Tokens	**Types**	**TTR**
Dieth transcriptions	615k	48.5k	7.9%
Normalised annotations	615k	31.7k	5.2%

Table 2: Lexical unit variance in two textual representations of Swiss German in the ArchiMob corpus.

Given this reduction in lexical variation, the normalised annotation layer facilitates automatic processing of Swiss German in the ArchiMob corpus. For the purpose of language modelling, we expect this layer to provide a suitable textual representation of Swiss German that can be supplemented with additional corpus resources for Standard German to improve LM performance.

4.2 Additional Corpus Resources

While the ArchiMob corpus provides a valuable resource for continuous Swiss German speech, the number of corpus resources suitable for other Swiss German NLP tasks is slowly growing (see Samardžić et al. [2018]). Meanwhile, extensive resources exist for processing Standard German. Below we describe the corpora selected for the purpose of our experiments in language modelling given the two textual representations in the ArchiMob corpus. These additional resources allow us to experiment with augmented training data and larger, more general-purpose LMs for Swiss German ASR.

4.2.1 Swiss German Resources

Schawinski Transcripts

The Schawinski transcripts are a collection of manually transcribed interviews conducted by the acclaimed Swiss journalist Roger Schawinski. This corpus was developed by Dr. Anja Hasse as part of her doctoral research. Amounting to approximately 2,500 spoken utterances, these interviews have been transcribed according to the Dieth spelling guidelines and thus provide samples of spoken Swiss German somewhat similar to those found in the ArchiMob corpus in terms of surface representation. Nevertheless, topics covered and the age of the informants differ considerably from those in the ArchiMob corpus.

PAZTeK Corpus

The Phonogrammarchiv Zürich Text-Korpus (PAZTeK) comprises a collection of approximately 15,500 spoken Swiss German utterances, with transcriptions closely representing the Dieth spelling guidelines. This corpus is still under development as part of the long-running *Phonogramarchiv* project at the University of Zurich[3], which aims to digitise linguistic field recordings of language varieties collected between 1909 and 1996 from all over Switzerland.

NOAH's Corpus

NOAH's corpus [Hollenstein and Aepli, 2015] is a manually annotated corpus of spontaneously written Swiss German from different text genres. It comprises articles from Wikipedia, various news outlets, the Swatch Annual Report, literature and blog posts. Thus, unlike the corpora described above, NOAH's corpus does not contain spoken language data. Nevertheless, amounting to a total of 115,000 tokens, NOAH's corpus constitutes a significant contribution to NLP resources for Swiss German.

CH-Web Corpus

The CH-Web corpus[4] is a freely available corpus of web-crawled content compiled as part of the Leipzig Corpora Collection (LCC) [Goldhahn et al., 2012]. Consisting of approximately 100,000 sentences, the corpus contains both spontaneously written Swiss German and Standard German. Texts are taken from a range of web sources including blogs and web pages of local sports and cultural clubs, organisations and municipalities.

[3] `https://www.phonogrammarchiv.uzh.ch/de/projekte.html`

[4] `https://cls.corpora.uni-leipzig.de/en/gsw-ch_web_2017`

4.2.2 Standard German Resources

TüBa-D/S

The Tübingen Treebank of Spoken German (TüBa-D/S)[5] is a transcribed corpus of spontaneous speech in Standard German [Hinrichs et al., 2000]. The corpus contains approximately 38,000 sentences and is manually annotated with part-of-speech tags, syntactic phrase structure and function-argument structure, making it a rich resource for numerous NLP tasks related in spoken Standard German.

Tatoeba

Tatoeba[6] is an open-source database of translated sentences and short expressions in over 340 different languages. Originally intended as a large multilingual dictionary, Tatoeba provides users with the opportunity to see translation variants in a sentence context, which are submitted and verified by users. The entire database is available for download with sentences annotated with a language identifier. The Standard German section of the corpus contains approximately 485,000 sentences.

Open Subtitles

Open Subtitles[7] is a large repository of parallel corpora containing more than three million subtitles in 62 languages [Lison and Tiedemann, 2016]. Movie subtitles are essentially transcriptions of spoken language, albeit scripted and performed. The German section of the database totals more than 18 million sentences representing continuous spoken language.

4.2.3 Summary of Corpus Resources

Table 3 provides a brief overview of the corpora described above. As can be seen, the amount of data available for Standard German significantly outweighs that of both Dieth-transcribed and spontaneously written Swiss German combined. In this table, the 'text' column indicates the textual representation used, while 'genre' indicates the predominant style of language, either continuous speech (CS) or written language (WR). Lastly, in the 'domain' column, we categorise each corpus according to the original data source or a general language focus area if known.

[5] Samardžić et al. [2015] also make use of the TüBa-D/S corpus to build an extended LM for experiments on automatic normalisation in the ArchiMob corpus.

[6] `https://tatoeba.org/eng/`

[7] `http://opus.nlpl.eu/OpenSubtitles-v2018.php`

Corpus	Sentences	Tokens	Text	Genre	Domain
ArchiMob	82.4k	615k	Dieth+Norm.	CS	personal histories
Schawinski	2.5k	26.5k	Dieth	CS	celebrity interviews
NOAH	7k	97.4k	Spont.	WR	public media
PAZTeK	15.5k	237.5k	Dieth	CS	field recordings
CH Web	99.7k	1.5m	Spont./StG	WR	web crawl
TüBa-D/S	38.3k	304k	StG	CS	spontaneous dialogue
Tatoeba (de)	485.8k	3.8m	StG	CS/WR	translated sentences
Open Subtitles	18.7m	123m	StG	CS	scripted dialogue

Table 3: Overview of relevant text and speech corpora for Swiss German and Standard German.

Together, these additional resources provide a large collection of Swiss German and Standard German text data that is crucial for our experiments in investigating language model performance for Swiss German in the ArchiMob corpus.

4.3 Swiss German Pronunciation Dictionary for ASR

Contemporary ASR systems typically rely on large-scale pronunciation lexicons containing permissible words with an appropriate phone sequence representing the word's pronunciation. Given the extensive spoken variation of Swiss German, establishing such a resource for any normalised textual representation of Swiss German is hardly trivial. Recently however, a team of experts from the University of Zurich's University Research Priority Programme (URPP) Language and Space collaborated with Switzerland's largest telecommunications company, Swisscom, to tackle this task.

The result of this collaboration is a comprehensive pronunciation dictionary, described in Schmidt et al. [2020]. The dictionary contains 11,248 Standard German words manually annotated with their pronunciations from six major dialect areas, namely, Basel, Bern, Central Switzerland, St. Gallen, Wallis and Zurich. Pronunciations are represented using a set of 57 phone symbols from the Speech Assessment Methods Phonetic Alphabet (SAMPA) [Wells, 1997], which is an ASCII-compatible alternative to the International Phonetic Alphabet (IPA) and is most commonly used in ASR applications. We henceforth refer to this resource as the URPP-Swisscom SAMPA dictionary.

4.4 The Kaldi Speech Recognition Toolkit

Our experiments in Swiss German ASR make use of Kaldi [Povey et al., 2011], a popular, open-source toolkit for speech recognition. Originally conceived by a group of researchers taking part in a 2009 workshop at Johns Hopkins University, Kaldi was intended to serve as a tool for research and development of state-of-the-art acoustic modelling techniques. Since then, it has been in constant development under the leadership of Prof. Dr. Daniel Povey and it has established a strong and highly regarded position in the field of ASR, both in academic research and commercial applications.

The Kaldi toolkit is predominantly written in `C++` with command-line interface scripts written in Bash and some data processing scripts in Perl and Python. Since the main motivation behind the toolkit was to encourage and promote ASR research and development, particularly for acoustic modelling, Kaldi is shipped with numerous 'recipes', allowing users to hit the ground running. A Kaldi recipe essentially outlines the individual steps for developing a speech recogniser, including acoustic model training, decoding graph compilation and fine-tuning. Most of the recipes provided are based on licensed speech corpora available from the Linguistic Data Consortium (LDC)[8].

As mentioned in Section 2.1.6.1, Kaldi performs speech recognition according to the WFST framework described by Mohri et al. [2008]. To do this, Kaldi draws heavily on the open-source software library OpenFst [Allauzen et al., 2007], which implements advanced algorithms for common WFST operations.

While Kaldi offers many advanced techniques in acoustic modelling and allows for a great deal of experimentation, a major downside of the Kaldi toolkit is the lack of introductory-level documentation. As the developers point out on the official website, "much of Kaldi's documentation is written in such a way that it will only be accessible to an expert" [Kaldi, 2019].[9] Therefore, considerable time and effort is required to become familiar with ASR in Kaldi. In contrast, other ASR toolkits, such as the Hidden Markov Model Toolkit (HTK) [Young, 1993] provide extensive introductory-level documentation and are thus much more accessible but are often more restrictive in terms of licensing and experimental scope.

[8] `https://www.ldc.upenn.edu/`

[9] `https://kaldi-asr.org/doc/about.html`

4.5 Language Modelling Toolkits

Statistical language modelling has long been widely used in many areas of NLP. As a result, a number of toolkits exist to assist with building and evaluating statistical LMs. We leverage the following toolkits in order to investigate statistical language modelling approaches, various smoothing techniques and training data augmentation for the purpose of modelling spoken Swiss German.

SRILM Toolkit

The SRI Language Modelling (SRILM) toolkit [Stolcke, 2002; Stolcke et al., 2011] is a popular, Swiss-army-knife-style toolkit for statistical language modelling. It has support for numerous smoothing techniques and different types of statistical N-gram models, including Class-based models, Cache models and Skip N-gram models. SRILM also has extensive documentation and is a valuable tool for research and development of statistical N-gram LMs.

MITLM Toolkit

Another publicly available toolkit is the MIT Language Modelling (MITLM) toolkit [Hsu and Glass, 2008], which focuses on more efficient estimation of N-gram LMs with limited smoothing algorithms (mainly variations on Kneser-Ney smoothing). Unlike SRILM, MITLM optimises interpolation weights directly during training, which typically leads to slightly better test set perplexities [Hsu and Glass, 2008; Heafield, 2013]. The major disadvantage of this toolkit, however, is that documentation is extremely limited.

RNNLM Toolkit

The Recurrent Neural Network Language Modelling (RNNLM) toolkit [Mikolov et al., 2011b] was designed to promote the application of advanced language modelling techniques based on nerual networks. The toolkit implements the basic Elman RNNLM architecture, described in Section 2.2.2.3. As with most LM toolkits, it supports basic training and evaluation functions, as well as also providing support for N-best list rescoring and autoregressive text generation, which we exploit in Section 6.2.2 for the purpose of augmenting LM training data.

5 Experimental Setup

In this chapter we aim to address RQ1 and determine which target textual representation provides optimal performance for Swiss German ASR. To do this, we establish four basic system setups that rely on different textual representations or system components and conduct end-to-end evaluations to compare their overall performance. In Section 5.1, we provide details about the data sets used for model training, fine-tuning and testing. Then, in Section 5.2, we introduce the fundamental ASR system used in our experiments and describe the basic steps involved in developing a speech recogniser with Kaldi. Following this, in Section 5.3, we discuss the major decisions concerning ASR system design in regards to pronunciation modelling for each potential textual representation of Swiss German. We present different system setup designs and conduct exploratory experiments to identify optimum approaches and to establish rudimentary baselines for future experiments.

5.1 Training, Development & Evaluation Data

In order to train a speech recogniser, we require a large set of training data, comprising segmented audio recordings and their corresponding written transcriptions. For the purpose of fine tuning model parameters and evaluating system performance, we also need two smaller data sets in the same format. These are known as the development and the test sets.

As mentioned previously, the ArchiMob corpus boasts approximately 70 hours of continuous speech data, which corresponds to 82,440 transcribed spoken utterances. However, a number of factors result in a considerable reduction in the amount data available for developing an ASR system. Recordings that contain multiple people speaking simultaneously, extensive noise or confidential information are not suitable for use in system development. As a result of such factors, the total amount of valid ArchiMob speech data is reduced to roughly 60 hours with approximately 70,000 transcribed utterances.

In NLP, a general rule of thumb is to set aside 10% of the total data available

for testing and a further 10% for development. Popular ASR data sets, such as the Wall Street Journal challenge set [Paul and Baker, 1992] and Librispeech [Panayotov et al., 2015], which contain 80 and 1000 hours of speech data, respectively, use about 90% for training and split the remaining 10% equally for development and testing.

For the purpose of our experiments, we decide to use a larger portion of data for training (95%) and to reserve smaller portions for testing and development (approximately 2.5% for each). The motivation for this decision is twofold. Firstly, the 60 hours of speech data provided in the ArchiMob corpus is not only very little for the task at hand but also extremely noisy due to the high degree of linguistic variation in Swiss German, as discussed in Chapter 3. In order to get the most out of the limited data we have, using more utterances for training acoustic models is seen as beneficial. Secondly, our investigations on potential textual representations for ASR involve repeatedly decoding development and test utterances with numerous systems that share the same acoustic models. Thus, keeping these data sets small allows us keep the costs of our experiments to a minimum. The downside of this decision is that the evaluation of system performance is more susceptible to noise both in the test set and in the model itself, which can result in variance in the final word error rate metric of up to half a percentage point [Mikolov, 2012].

We make use of an existing test set, which was originally established by Samardžić et al. [2016] for evaluating part-of-speech tagging in the ArchiMob corpus. This test set contains a total 1,871 utterances, of which only 1,486 are valid for our task. For fine tuning system parameters, we hold out a development set (devset) of 2,267 randomly selected utterances, of which 1,710 are valid. Since the validity of utterances largely concerns the associated audio recording and not the written transcription itself, we disregard this validity requirement and use all utterances for the purpose of training, fine-tuning and testing LMs in isolation.

Training a full-scale chained acoustic model on the entire data set is an expensive process, typically taking anywhere between 35 to 50 hours (on our CPU infrastructure) for approximately 60 hours of audio material. In order to assess ASR performance for different textual representations, each with distinct pronunciation modelling approaches, an entire ASR system with a dedicated AM for each representation is required. Since we focus on the language modelling component and to keep costs down, we use a 22-interview subset of the ArchiMob corpus data, which we refer to as ArchiSmall. This provides a reduced-size training and devset that are sampled according to selected interview IDs[1]. Table 4 provides an overview of all

[1]ArchiSmall comprises the following interview IDs: 1008, 1044, 1048, 1055, 1063, 1138, 1143, 1147, 1188, 1189, 1195, 1205, 1209, 1224, 1228, 1235, 1240, 1248, 1255, 1259, 1295, 1300.

the data splits for both ArchiMob and ArchiSmall as used in our experiments.

	ArchiMob		ArchiSmall	
	Total	**Valid**	**Total**	**Valid**
Train	78,304	67,693	42,457	36,499
Dev	2,267	1,710	1,241	930
Test	1,871	1,486	1,871	1,486

Table 4: Utterance counts for training, development and test splits used in our experiments. Note, the test set is the same for both ArchiMob and ArchiSmall.

5.2 An ASR System for Swiss German

The ASR system used in this study was originally developed as part of a collaboration between the URPP Language and Space Lab and the Zurich-based start-up Spitch AG. The system is based on Kaldi's Wall Street Journal NNET2 recipe[2]. Developers at Spitch installed Kaldi on a 16-core CPU server instance[3] and adapted the original recipe to work with ArchiMob data, using an old EXMARaLDA XML (see Section 4.1.1) as the primary input format. As part of this project, we have made further adaptions to the system in order to use the current ArchiMob XML release (see Scherrer et al. [2019]) as the primary input format and to explore different language modelling techniques. All of the code is available on GitHub.[4]

Developing a speech recogniser with Kaldi typically consists of three main steps: AM Training, graph compilation and decoding, as discussed in Chapter 2. Below we describe these steps in the context of the NNET2 recipe provided with Kaldi.

AM Training

Kaldi's NNET2 recipe begins by extracting standard MFCC features from each audio file and performing feature normalisation to reduce noise contamination. It then trains a series of chained AMs to derive the final hybrid HMM AM. The process begins with a context-independent monophone model, trained from a flat start, and continues with a series of context-dependent triphone models trained with different objective functions. The training of each subsequent model benefits from having initial phone-signal alignments attained by performing forced Viterbi alignment given

[2] `https://github.com/kaldi-asr/kaldi`, commit 8cc5c8b32a49f8d963702c6be681dcf5a55eeb2e

[3] This server instance is provided by the University of Zurich's Science IT (S3IT) department.

[4] `https://github.com/tannonk/two-headed-master`

the previous model (see Section 2.1.3.3). A more detailed description of this training process is available in Nigmatulina [2020].

Graph Compilation

Typically, backoff LMs trained with the MITLM or SRILM toolkit are encoded in the popular ARPA file format.[5] In order to compile a functional decoding graph, we first need to generate an FST representation of the input LM. Then, given this FST representation, a trained acoustic model and a hardcoded pronunciation lexicon, Kaldi's `mkgraph.sh` script does most of the heavy lifting to combine these individual components and to compile the final decoding graph as described in Section 2.1.6.1.

Decoding

Once the graph has been compiled, a held out data set can be used for decoding. This initial decoding step corresponds to the development stage of a speech recogniser. First, the input utterances are processed and audio features are extracted as before. Then, we apply the newly compiled decoding graph to perform a hypothesis search. Decoding is run multiple times for varied weights applied to the LM component and word insertion penalties.[6] Once decoding has finished, we can determine which parameter values result in the best (i.e. lowest) WER score. This allows the user to then fix these parameters for the purpose of evaluation and applying the model to new speech signals.

5.2.1 Data Preparation & Preprocessing

As mentioned above, we use the current ArchiMob XML release format as initial input data. However, Kaldi requires utterance transcriptions to be encoded horizontally, in a one-utterance-per-line format. Each utterance must be prefixed by a unique ID, indicating the corresponding audio file. Therefore, starting from the current ArchiMob XMLs, we first generate a large comma-separated values (CSV) file containing the following nine columns:

utterance ID	unique enumerated utterance ID prefixed with speaker ID
Dieth	human annotated transcription of the utterance in Dieth orthography
normalised	automatically normalised transcription

[5] Here, the term 'backoff' refers to the standard LM format that contains hardcoded probabilities for all N-grams and should not be confused with the backoff smoothing technique described in Section 2.2.1.1.

[6] The default settings consider LMWTs ranging from 7 to 17 and WIP of 0.0, 0.5 and 1.0.

speaker ID	unique ID denoting the speaker
original audio ID	the original name of the audio file as listed in the XML corpus
anonymity	whether or not the utterance contains an anonymised name mention (e.g. m***)
speech in speech	whether or not the utterance overlaps with another utterance according to start and finish timestamps
missing audio	whether or not the utterance has a corresponding `wav` file
no relevant speech	whether or not the utterance is empty

The first three columns of the CSV file contain the main input data relevant for our Kaldi recipe for both textual representations. Columns 'speaker ID' and 'original audio ID' indicate the speaker and the original audio file names as listed in the XML format. These are used to automatically rename audio files according to the utterance ID in column 1. The last four columns of the CSV indicate the validity of a particular utterance. An utterance is deemed invalid if (a) it contains a person's name that has been manually anonymised, (b) multiple people are speaking simultaneously, (c) it has no corresponding audio file, or (d) it contains only silence and thus the transcription is empty.

While generating the CSV file, we do a small amount of preprocessing. As previously discussed, word boundaries in the Dieth surface forms and the normalised annotations do not always agree due to the extensive cliticisation in Swiss German (see Section 3.2.2). To ensure that a Dieth-transcribed utterance and its normalised counterpart have the same number of tokens, we use an underscore to glue normalised words consisting of multiple tokens together. As a result, the normalised form 'haben wir' ('have we') is instead represented in the system as 'haben_wir'. This step helps to better represent the pronunciation of these words later.

As mentioned in Section 4.1.1, the corpus transcriptions contain different meta-level annotations for non-speech events. In order to allow the system to distinguish from speech and non-speech sound units, we map these non-vocalised events to a single `<NON_SPOKEN_NOISE>` token. We reserve a special `<SPOKEN_NOISE>` token to cover language disfluencies such as truncated words or stutters, which are marked in the original corpus transcriptions.[7] Finally, where annotated, pauses or silences in the recordings are mapped to a single `<SIL_WORD>` token.

[7] In our Kaldi recipe, the `<SPOKEN_NOISE>` token covers all unknown or out-of-vocabulary words instead of the commonly used `<UNK>` token.

After generating the CSV, we split it into three smaller CSV files corresponding to training, development and testing sets. All input audio files are assumed to be 'chunked' (i.e. segmented) according to corpus utterance boundaries. Since Kaldi's I/O framework requires files to be sorted in a certain way, the audio files should be stored in a single directory and named according to the unique utterance IDs as they appear in column 1 of the CSV files. To run our train, validation and test scripts, we simply need to provide the relevant input CSV file and audio file directory as input.

5.3 Exploratory Experiments on Text Representation

The orthographic form used to represent spoken Swiss German (see Section 3.2) plays a significant role in the ASR system's design in two ways. Firstly, the LM needs to be tailored towards the target textual representation and thus differs between systems designed to produce Dieth-like transcriptions and those designed to produce normalised transcriptions. Second, the textual representation defines how words are represented in the system's vocabulary and thus affects the mapping of words to their pronunciation strings in the lexicon. Thereby, each potential textual representation has certain advantages and disadvantages.

5.3.1 Speech to Dieth Text

For Dieth transcriptions the major advantage concerns the creation of a pronunciation lexicon. Since Dieth spelling is, or at least intended to be, phonemic, it is possible to model the pronunciation of a given word by simply segmenting a word into its composite graphemes. Here, we take into consideration certain grapheme clusters which are often used to represent single sounds (e.g. 'sch' as /ʃ/ and 'ng' as /ŋ/) and avoid segmenting these. Instead these grapheme clusters[8] are mapped to a single appropriate phone symbol. For example, the Dieth surface form *afangen* (Standard German *anfangen*, English 'start') is represented by the phone sequence 'a f a ng e n'. While such an approach is unlikely to derive perfect pronunciation strings, it allows us to deduce a suitable pronunciation for all words in the training corpus relatively easily, albeit *ad hoc*.

On the other hand, using Dieth spelling for ASR has some significant drawbacks. Firstly, as discussed in Section 3.2.1, the Dieth orthography is not used by native

[8]The full list of grapheme clusters considered is provided in Table 12 in Appendix A.

Swiss German speakers and is thus extremely limited in its application. As a consequence, any further processing of the system's output, either by a human or a machine, is hardly straightforward.

Secondly, the significant amount of spelling variation in the Dieth transcriptions results in a higher degree of lexical diversity, introducing considerable noise and data sparsity. This sparsity is likely to make it more difficult for statistical acoustic and language models to capture patterns in the data and to estimate reliable probabilities.

Lastly, as mentioned in Section 3.3, evaluating system output with the standard WER metric is less meaningful for languages with a non-standardised orthography since there is rarely ever just one correct reference utterance. To account for this, we adapt the standard WER metric to allow alternative but permissible spelling variants, similarly to Ali et al. [2017]. We exploit the word-level normalised annotations to create a dictionary mapping between all surface form variants and their normalised forms. Using this mapping, a hypothesis word is considered correct if it shares the same normalised form as the corresponding word in the reference transcription. As we do not employ any further constraints, this metric is rather greedy and thus simpler than the implementation of WERd proposed by Ali et al. [2017]. Since our metric provides a more flexible error rate evaluation for Swiss German ASR, we henceforth refer to it as FlexWER.

5.3.2 Speech to Normalised Text

The normalised transcriptions provide a significantly more standardised textual representation of spoken Swiss German and thus do not suffer from the same drawbacks as a system designed to produce Dieth transcriptions. Firstly, since the normalised annotations closely resemble Standard German, albeit with some discrepancies, ASR system output could be further processed relatively easily with existing NLP tools designed for Standard German. Secondly, the reduced amount of lexical variation should aid statistical modelling by providing more surface-level patterns in the limited data available. And lastly, assuming that the normalisation is accurate in the reference utterance, the standard WER metric remains suitable for evaluation purposes.

Using the normalised transcriptions instead poses a more significant challenge for how to effectively model the pronunciation of words in the corpus. Since the normalised orthography differs greatly from the true pronunciation (e.g. /gsiː/ for *gewesen* ('been')) word-level pronunciations can not be derived from it directly. However,

one potential *ad hoc* solution is to make use of the corresponding Dieth transcriptions for each normalised word in the corpus. Using a similar approach to that described above, we can simply segment the relevant Dieth-transcribed word into its graphemes clusters to model the pronunciation of a given normalised word. A more elegant solution, which is more in line with the approach taken in conventional ASR systems for standardised languages, is made possible by exploiting the URPP-Swisscom SAMPA pronunciation dictionary (see Section 4.3). However, the low coverage of the pronunciation dictionary significantly reduces the size of the system's vocabulary. Of the approximately 31,000 word types in the normalised ArchiMob training set, only 7,373 (23%) of these currently appear in the dictionary.

5.3.3 System setups

With the above mentioned factors in mind, we identify four possible system setups to investigate.

Speech-to-Text-Dieth (STTD) is designed to produce Dieth-orthography transcriptions and relies on the rather naïve approach to derive a word's pronunciation directly from the surface form by segmenting grapheme clusters.

Speech-to-Text-Norm-Dieth (STTN-D) aims to produce normalised transcriptions and derives word-level pronunciations by splitting grapheme clusters from the corresponding Deith surface words.

Speech-to-Text-Norm-SAMPA (STTN-S) also aims to produce normalised transcriptions but takes advantage of the full scope of the URPP-Swisscom SAMPA dictionary, including all possible pronunciations from the six different dialect areas covered.

Speech-to-Text-Norm-ZRH (STTN-Z) is similar to STTN-S but aims to simplify the pronunciation lexicon by using a single possible pronunciation for each word. Here, we select only the phone sequence provided for Zurich Swiss German since this particular dialect is the single most represented dialect in the ArchiMob corpus.[9]

Table 5 shows a small sample of the pronunciation lexicon for each system defined above. As can be seen, STTD provides a one-to-one word-pronunciation mapping for each Dieth surface form type. STTN-D uses the same pronunciation representation as STTD but maps all possible pronunciations to a single normalised word form, resulting in an average of two pronunciations per word. STTN-S shows the greatest

[9]Swiss German from Zurich accounts for approximately 28% of the corpus, followed by Luzern (14%), Bern (12%) and Basel (12%).

amount of possible pronunciations, with one for each of the six dialects listed in the URPP-Swisscom SAMPA dictionary. Here, every word in the lexicon is represented by an average of 4.8 pronunciations. In contrast, STTN-Z is drastically simplified, offering only one possible pronunciation representation for each given normalised word type.

STTD		STTN-D		STTN-S		STTN-Z	
aagäu	a g äu	aargau	a g äu	aargau	A: r g A I	aargau	a: r g aU
aargau	a r g au	"	a r g au	"	a: R g E u		
aargou	a r g o u	"	a r g o u	"	a: R g aU		
aargäu	a r g äu	"	a r g äu	"	a: r g aU		
				"	a: r g o U		
				"	a: r k aU		

Table 5: The word *Aargau* as it appears in the pronunciation lexicon for each of the four system setups. Note, for pronunciation strings derived from Dieth spellings, common grapheme clusters representing a single sound (e.g. 'aa', 'äu' and 'au') are not separated according to the list provided in Table 12 in Appendix A.

5.3.4 Intermediary Results & Discussion

For the purpose of these exploratory experiments, we use modKN-smoothed trigram LMs (see Section 2.2.1.1). Each LM is trained on all utterances for a given textual representation in the ArchiMob training set. Before training LMs, special symbols representing non-spoken sounds and silences are removed from the utterance transcriptions.

Table 6 shows the performance of all four systems along with some simple statistics describing each system. In this table, 'types' refers to the number of unique surface forms found in the ArchiSmall training set transcriptions. The column 'vocab' indicates the size of the system vocabulary and thus the number of known words. This is dictated by the number of words modelled in the system's pronunciation lexicon. Since STTD and STTN-D derive pronunciations from the Dieth surface forms, both systems have full coverage of the relevant training set. In contrast, due to the limited coverage of the URPP-Swisscom SAMPA dictionary, only 32% of words in the normalised training set are modelled in the lexicons for STTN-S and STTN-Z. The column 'av. ppw' indicates the average number of pronunciations for each word in the lexicon. 'OOV' shows the portion of unknown, or out-of-vocabulary, words in the test set. 'WER' indicates the system word error rate achieved on test

set decoding, as calculated by Kaldi (i.e. the standard WER metric). The final column, 'FlexWER', shows the error rate attained by allowing for alternative but permissible Dieth spellings in the system output. This measure is only applicable for STTD.

setup	types	vocab	av. ppw	OOV	WER	FlexWER
STTD	27.5k	27.5k	1	32%	68.92%	**46.93**%
STTN-D	18.9k	18.9k	2	25%	69.51%	–
STTN-S	18.9k	6.2k	4.8	40%	65.44%	–
STTN-Z	18.9k	6.2k	1	40%	**62.32**%	–

Table 6: Results for exploratory experiments based on four different system setups.

If we consider the standard WER metric, STTN-Z clearly outperforms all other systems. This result supports the findings of Hain [2002]. Despite the extensive spoken variation in Swiss German, a simplified pronunciation lexicon leads to better results than one that is overly detailed (e.g. STTN-S). Surprisingly, even with the lowest OOV rate (25%), STTN-D is by far the worst performer. This indicates that relying on the Dieth transcriptions to derive possible pronunciations for normalised words is suboptimal, even though it provides greater lexical coverage in the system. The results for STTN-S and STTN-Z reveal that exploiting a dedicated resource for pronunciation modelling in which speech sounds are represented consistently rather than faithfully is beneficial, despite leading to a much higher OOV rate in the test set (40%).

As expected, STTD performs poorly according to the standard WER metric. However, if we consider the FlexWER score, we see not only a dramatic improvement but also the best performing system overall (46.93%). This demonstrates that allowing for a certain amount of lexical variation in representing spoken Swiss German with a non-standardised orthography in ASR is indeed necessary.

As can be seen from the results presented in Table 6, the WER for all systems is very high (62.3% – 69.51%). Naturally, we cannot expect groundbreaking results given the challenges posed by training and evaluating on the small and noisy data set used in these exploratory experiments. However, these scores are rather intended to function as an indicator for which approaches should be further investigated.

In summary, despite the high degree of spelling variation in the Dieth transcriptions, the system performs reasonably well if we are not deterred by a large amount of lexical variation in the system output and consider the appropriate FlexWER

metric. On the other hand, if our goal is to produce output transcriptions with less lexical variation, a normalised representation of Swiss German that more closely resembles Standard German does indeed provide a viable textual representation for the task. According to these experiments, relying on a simplified lexicon, in which each word has only one possible pronunciation, to model word-level pronunciation for normalised Swiss German delivers optimum results, despite the fact that the system suffers from low lexical coverage.

6 Experiments in Language Modelling

Having established basic system setups for two potential textual representations for Swiss German ASR, we now aim to address RQ2. Here, we describe a number of experiments designed to assess performance of various language models (LMs) for Dieth-transcribed and normalised Swiss German. In a first step, we compare different probabilistic N-gram LMs, investigating the effects of N-gram order and different smoothing techniques as implemented with two popular LM toolkits. Second, we investigate the possibility of augmenting LM training data with both out-of-domain (OOD) data and synthetically generated text data. In order to assess LM performance, we report test set perplexity (PPL) as an intrinsic evaluation measure (see Section 2.2.3). This investigation will allow us to identify the best performing models, which will later be incorporated into our ASR system for Swiss German and hopefully lead to improved overall system performance.

6.1 Comparing N-gram Language Models

Conventional ASR systems rely heavily on probabilistic N-gram LMs. These models are both quick to estimate from any example training corpus and efficient to apply in a wide range of tasks and have thus been used to establish a number of state-of-the-art benchmarks in ASR [Goodman, 2001; Mikolov, 2012]. Despite the fact that N-gram models suffer a few major short-comings, namely their limited ability to generalise to unseen contexts and failure to capture language dependencies that occur over arbitrarily long distances, they can be converted into a WFST representation, making them well suited for integration in the common ASR framework described in Section 2.1.6.1. Therefore, N-gram LMs remain not only an 'oldie-but-a-goodie' approach for practical language modelling applications but also an essential component for any ASR system that wants to exploit more advanced neural LMs through multi-pass decoding techniques (see Section 2.2.2).

6.1.1 Smoothing Techniques

N-gram LMs have been studied extensively and have been shown to deliver strong performance when combined with appropriate smoothing techniques. For example, Chen and Goodman investigate a number of different smoothing methods for English, inspecting the effect of N-gram order, training corpus size and the text domain. They report that the "relative performance of different smoothing methods can vary significantly as conditions vary" [Chen and Goodman, 1999, p. 375], yet they conclude that modified Kneser-Ney (modKN) smoothing consistently outperforms all other methods in their experiments.

This has lead to modKN smoothing becoming by far the most popular smoothing technique. However, given that little work has been done on language modelling for Swiss German, it is unclear whether other smoothing techniques would provide better performance given the high degree of lexical diversity and the resulting data sparsity in non-standardised writing. Motivated by Rusli [2017], who compares LMs with modKN and Witten-Bell (WB) smoothing for low-resource Bahasa Indonesian, we investigate these two smoothing techniques for modelling both Dieth-transcribed and normalised Swiss German. As mentioned in Section 2.2.1.1, modKN and WB are typically implemented as interpolated techniques, but can also be implemented using an alternative backoff approach. Therefore, we compare both backoff and interpolated versions of these to see which works best for our data.

6.1.2 N-Gram Order

It is commonly held that "the more information the N-gram gives us about the word sequence, the lower the perplexity" [Jurafsky and Martin, 2019, p. 38]. Chen and Goodman [1999] report that, given enough training data, using higher order N-gram LMs can significantly improve performance for some smoothing methods. However, as N-gram order increases, so too does the size of the LM. This becomes a considerable factor in ASR applications as larger LMs require significantly more memory in graph compilation and often lead to longer decoding times. For this reason, trigram models remain a popular choice for N-gram LMs in practice since they offer a good balance between size and performance. Given that utterances in the ArchiMob corpus are typically rather short, with an average length of only seven tokens, we expect that longer context histories will not greatly improve LM performance. Thus, we investigate N-gram models with orders ranging from two to six in order to determine optimum performance.

6.1.3 Test Set Perplexity Scores

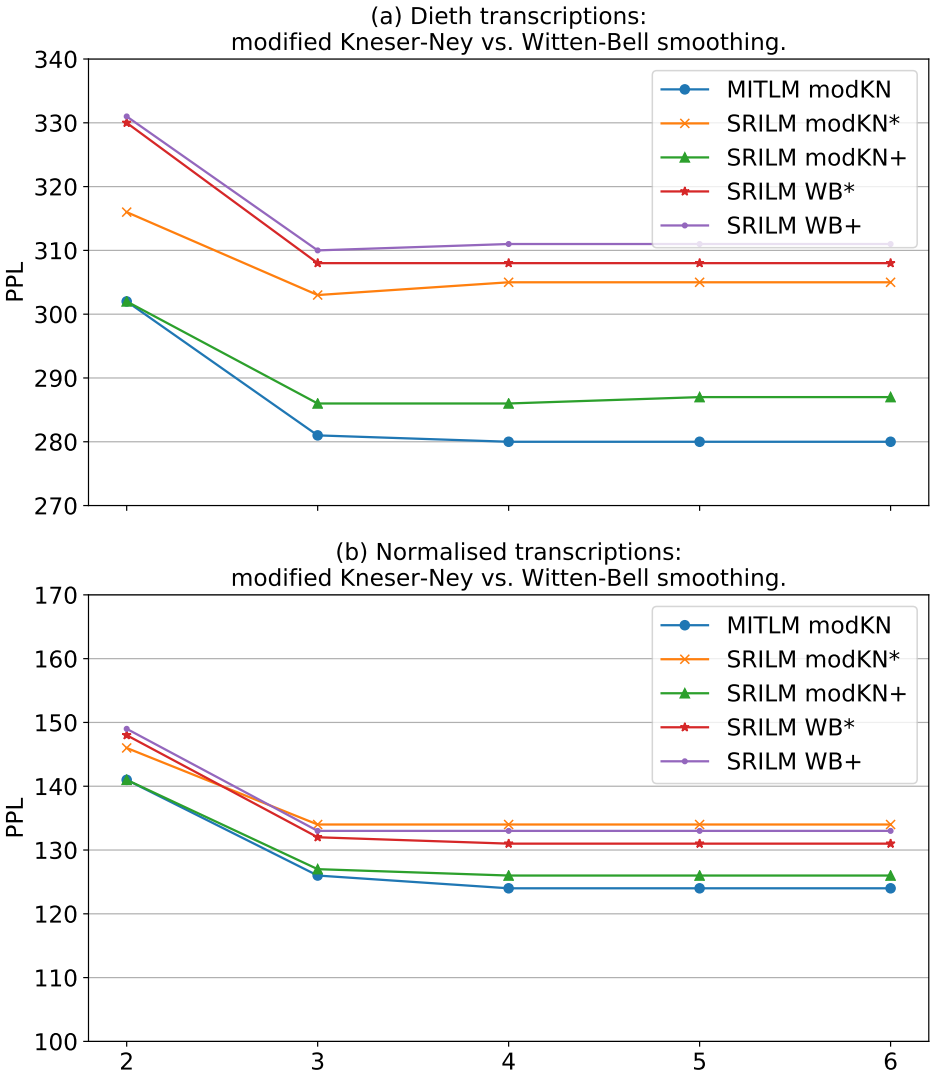

Figure 9: ArchiMob test set PPL scores attained with various LMs for both Dieth transcriptions (a) and normalised Swiss German (b). For models trained using the SRILM toolkit, $*$ represents a backoff version of the applied smoothing technique, while $+$ represents an interpolated version.

Figure 9(a) shows the test set PPL scores for all N-gram LMs of order two to six trained on Dieth transcriptions and evaluated on the corresponding ArchiMob test set. Looking at the graph, two findings are immediately clear. Firstly, PPL scores are particularly high, ranging from 280 to 330. This highlights the difficulty of the modelling task given this textual representation since PPL scores for regular language modelling tasks typically range from 100 to 200 [Mikolov, 2012]. Secondly, the graph reveals that trigram LMs offer a definite optimum for all smoothing techniques, with very little or no improvement gained as we move to higher order N-grams.

Comparing these different smoothing techniques reveals that modKN delivers performance gains over WB smoothing for all models, regardless of the N-gram order. Backoff WB smoothing appears to perform slightly better than the interpolated version (red and purple lines, respectively), however, the difference is negligible. For the three modKN-smoothed LMs the interpolated versions (blue and green lines) significantly outperform the backoff version (orange line). This supports the claim made by Goodman [2001] that interpolated Kneser-Ney models typically work better than their backoff counterparts. Finally, it can also be seen that MITLM's implementation of interpolated modKN smoothing proves to be the clear winner in terms of test set PPL thanks to its efficient weight optimisation performed during training (see Section 4.5).

Applying the same techniques to the normalised transcriptions, illustrated in Figure 9(b), shows a drastic improvement in language modelling ability across the board. Here, PPL scores are more than halved for all N-gram models in comparison to their Dieth-transcribed counterparts. Furthermore, the difference between smoothing techniques and implementations is greatly reduced. This shows that the reduction in lexical variation afforded by the automatic normalisation significantly aids statistical N-gram language modelling for Swiss German.

Looking closely at this graph, we can see that, when modelling normalised Swiss German, both versions of WB smoothing perform slightly better than backoff modKN for trigram models and above, however, the difference is rather negligible. Once again, interpolated modKN smoothing, as implemented by MITLM (blue line), outperforms all other smoothing techniques. Also apparent in these results is an ever so slight performance gain achieved by increasing the N-gram order from three to four. Thus, modelling normalised Swiss German benefits from slightly longer context histories.

These results demonstrate that trigram LMs with interpolated modKN smoothing provide a strong starting point for modelling both types of textual representations for Swiss German. For modelling Dieth-transcribed Swiss German, considerable improvements can be achieved by applying interpolated modKN smoothing, while for the normalised textual representation, less attention needs to be given to the type of smoothing implementation. The much lower PPL scores for normalised Swiss German prove that modelling this textual representation is a far easier task than modelling the alternative Dieth transcriptions. This makes sense when we consider PPL as the weighted average branching factor of a language, i.e. indicating "the number of possible next words that can follow any word" [Jurafsky and Martin, 2019, p. 37]. Here, we can attribute the high test set PPLs associated with the Dieth

transcriptions to the extensive lexical variation inherent in this non-standardised orthography.

In summary, according to this intrinsic evaluation, the reduced lexical variation afforded by the automatically normalised textual representation significantly reduces the complexity of the task and provides optimal performance for modelling Swiss German, which should, in turn, contribute to better ASR system performance. Additional experiments conducted to assess the effect of fine-tuning model interpolation weights on ArchiMob devset utterances and building open vocabulary LMs that include an `<unk>` token reveal that further reductions in test set PPL are possible for modelling Dieth-transcribed Swiss German, reaching as low as 250. However, these tweaks provided no significant improvements when modelling the normalised textual representation.

6.2 Modelling Swiss German with Additional Data

LMs for conventional ASR systems are typically trained on very large, heterogeneous corpora. These corpora often contain several million words and comprise both written text and transcribed speech from a variety of domains [Raju et al., 2019; Martins et al., 2004]. For standardised languages with large amounts of relevant text data, this can lead to relatively good results and thus raises the question as to whether we can exploit additional corpus data to improve LM performance for Swiss German and consequently improve ASR output.

As stated by Moore and Lewis, "it seems to be a universal truth that output quality [of a speech recogniser] can always be improved by using more language model training data, but only if the training data is reasonably well-matched to the desired output" [2010]. Usually this 'well-matchedness' pertains to the domain of the target text. For example, if the intended application of an ASR system is to transcribe parliamentary sessions, the language used would be expected to differ from that used in the domain of biomedicine, where an ASR system may be designed for dictation of patient records. While the domain of a general-purpose ASR system for Swiss German is not strictly defined to one specific area, in the context of the ArchiMob corpus, we can identify a general domain as personal oral histories relating to life in the early 20$^{\text{th}}$ century, with a particular focus on World War Two. However, in the broader context of Swiss German, clearly a more important criteria for finding well-matched text data is simply the orthography used to represent the language.

6.2.1 Out-of-Domain Data

In Section 4.2, we introduced a number of additional corpus resources that could potentially be used to extend LM training data for both Dieth-transcribed and normalised Swiss German. A summary of these corpora was provided in Table 3. As shown in that table, the vast majority of available data differs significantly from the ArchiMob corpus in terms of both textual representation and domain. With the exception of the Schawinski transcripts and parts of the PAZTeK corpus, which include Dieth-transcribed oral histories in the form of celebrity interviews and linguistic field recordings, most of the additional text data available represents Standard German or spontaneously written Swiss German from online media sources (NOAH's Corpus, CH-Web), conversational speech (TüBa-D/S) and movie subtitles (Open Subtitles). As such, these resources can largely be considered out-of-domain. Nevertheless, we investigate the effect of leveraging this additional data for the purpose of modelling the two types of textual representation in the ArchiMob corpus.

To model Dieth-transcribed Swiss German, we select those additional corpora that contain either Dieth-transcribed speech or spontaneously written Swiss German since these two spellings are at least intended to closely represent the sound of the language. These include the Schawinski Transcripts, NOAH's Corpus, PAZTeK and CH Web. The resulting data set contains around 125,000 utterances, totalling roughly 1.9 million words. For simplicity we henceforth refer to this data set as the out-of-domain Dieth data set (OOD-Dieth).

For modelling normalised Swiss German, we combine the remaining corpora that largely represent spoken Standard German. This includes TüBa-D/S, Tatoaba and a random sample of sentences from Open Subtitles. The resulting data set consists of a total of 924,000 utterances, amounting to roughly 6.8 million words. Since this data set is intended to resemble the normalised textual representation we refer to it as the out-of-domain normalised data set (OOD-Norm).

The majority of texts in both the OOD-Dieth and OOD-Norm come from written sources. Therefore, we perform some simple normalisation steps in an attempt to emulate spoken language as it is represented in the ArchiMob corpus. These include:

- mapping all URLs to a single placeholder token
- replacing Standard German 'ß' with Standard Swiss German 'ss'
- removing all punctuation symbols
- lowercasing all characters

- replacing all cardinal numbers with a randomly selected spelled-out form between one and ten extracted from the ArchiMob corpus

6.2.2 Synthetic Data

An alternative method for increasing LM training data, particularly for low-resource languages, is to use synthesised text data which can be automatically generated. In addition to their application in ASR systems, where the LM is used to assign probabilities to a hypothesis word sequence, LMs can be used to generate new text one word at a time based on the probabilities they have learned in training. The generation process begins with a start-of-sentence symbol `<s>` and outputs the next word by sampling from the most probable words given the current context history. This continues at each step (i.e. each output word) until the end-of-sentence symbol `</s>` is produced.

RNNLMs can also be used to perform this task, relying only on the previously generated word as input to output the current word in a process known as autoregressive generation [Jurafsky and Martin, 2019, ch. 9]. Since they are capable of generalising to more varied contexts and capture long-distance semantic and syntactic regularities in language, RNNLMs are also particularly good at this task compared to basic N-gram models [Huang et al., 2017].[1] In addition, using text generated by a trained RNNLM to estimate a novel N-gram LM can be seen as a legitimate method to approximate the enhanced predictive abilities of an RNNLM for use in first-pass decoding [Deoras et al., 2011].

Depending on the level of granularity that an RNNLM is trained on, the generated text can also differ significantly. While a word-level RNNLM can be used to generate novel sequences of words seen during training, a subword-level RNNLM, trained on sequences of characters, phonemes or even syllables, can be used to generate new 'words' and thereby expand training data vocabulary with potentially legitimate and illegitimate words [Huang et al., 2017]. One downside of the latter approach is that it also increases the amount of lexical variation and sparsity in the training data significantly and can thus reduce the effectiveness of the resulting LM. In order to explore both of these options, we use the RNNLM Toolkit [Mikolov et al., 2011b] to train two RNNLMs – one at word-level and one at the character-level – for the purpose of autoregressive generation of Swiss German text data.

[1] More recently, transformer-based architectures [Vaswani et al., 2017] such as the GPT-2 language model [Radford et al., 2019], which is trained on a curated 40-GB web corpus and contains approximately 1.5 billion model parameters, have been shown to significantly outperform recurrent architectures and are capable of generating far more realistic texts.

Word-level RNNLM

Starting with the full ArchiMob training set of transcriptions, we first trained a word-level RNNLM using a 100-dimensional hidden layer, class size of 100 for the output layer and limiting the number of timesteps for truncated BPTT to four (see Section 2.2.2.3).[2] The ArchiMob devset transcriptions are used to estimate PPL at each epoch. Early stopping ensures that the model does not overfit to the training data by ending the training once devset PPL stops improving. With these parameters, training ran for 16 epochs with an adaptive learning rate starting at 0.1 and reducing to 6.0×10^{-5}. Training took approximately 35 minutes and the test set PPL with this model was 307.

Once trained, we generated 1.2 million synthetic utterances, amounting to approximately 8.7 million words. We henceforth refer to this as the word-level synthesised data set (W-Synth).

Subword-level RNNLM

For the subword-level RNNLM, we start by segmenting all words in the ArchiMob training set according to the grapheme-phone clusters used to derive a word's pronunciation from Dieth spellings (see Section 5.3.1). Word boundaries are indicated with a special character, '`@`'. We then trained a subword-level RNNLM using a 200-dimensional hidden layer, output class size of one and 10 timesteps for truncated BPTT. The training ran for 14 epochs with an adaptive learning rate starting at 0.1 and finishing at 9.8×10^{-5}. Training time for this model was approximately 95 minutes.

Naturally, this model calculates test set PPL over sequences of grapheme-phones rather than words and is thus not directly comparable to the word-level PPL reported with other LMs. Graves [2014] proposes a simple method for approximating word-level PPL of subword-level models to allow for an effective comparison. If we know the average number subword units per word and we know the average number of bits required to encode the subword units (i.e. entropy), we can simply multiply these numbers and take the exponent in order to estimate average word-level PPL. In the ArchiMob test set, the average number of grapheme-phones per word is 3.7 and the grapheme-phone test set entropy is 2.34. Thus, applying Graves' formula, word-level PPL for this subword-level RNNLM can be approximated as $2^{2.34 \times 3.78} \approx 460$.

Once trained, we used this RNNLM to generate one million utterances. After con-

[2] These settings correspond to the default settings proposed by the authors in the toolkit's release notes `http://www.fit.vutbr.cz/~imikolov/rnnlm/`.

catenating consecutive graphemes back into words using the word boundary symbol as an indicator, we are left with approximately 6.5 million 'words'. This data set is henceforth referred to as the subword-level synthesised data set (SW-Synth).

6.2.3 Test Set Perplexity Scores

In order to inspect the effect of increasing LM training data, we concatenate each of four additional data sets with the relevant transcription type utterances from the ArchiMob training data. As a result, we get three unique extended LM training sets for Dieth-transcribed Swiss German and one for normalised Swiss German. Table 7 provides a brief overview of the size and scope of the data sets for extended LM training data.

Dataset	Utterances	Tokens	+ArchiMob Utterances	+ArchiMob Tokens
OOD-Dieth	125k	1.9m	201k	2.4m
W-Synth	1.2m	8.7m	1.3m	9.3m
SW-Synth	1m	6.5m	1.07m	7.1m
OOD-Norm	927k	6.8m	1m	7.3m

Table 7: Overview of additional data sets for LM training.

With each data set, we train multiple trigram LMs with interpolated modKN smoothing, each time increasing the amount of training data by 10,000 utterances. We use the MITLM toolkit to estimate the LMs and tune interpolation weights on the relevant ArchiMob devset before calculating test set PPL. As pointed out by Jurafsky and Martin, the PPL of two LMs "is only comparable if they use identical vocabularies" [2019, p. 38]. Therefore, to ensure a fair comparison, the vocabulary is always specified as the set of words in the ArchiMob training data and any out-of-vocabulary words (OOVs) are thus replaced with a special placeholder symbol, `<unk>`.[3]

Figure 10 depicts the relationship between test set PPL and increasing the number of training utterances from each data set. We limit the maximum number of training utterances in this experiment to 500,000 for readability. Recall that the first 76,000 utterances are from the ArchiMob training set and are thus in-domain. This threshold is shown on the graph by the vertical dotted line. As can be seen,

[3]We avoid limiting the vocabulary according to a word occurrence frequency threshold, as is a common approach in other tasks, in order to try to keep OOV words to a minimum.

for both text types, test set PPL improves quickly as we increase the amount of in-domain training instances and appears to reach optimum values at about 80,000 utterances. As soon as we begin to include additional OOD or synthesised utterances test PPL begins to rise rather quickly. Extending LM training data with all OOD and synthesised text data leads to considerably higher test set PPL scores for both Dieth-transcribed and normalised Swiss German.

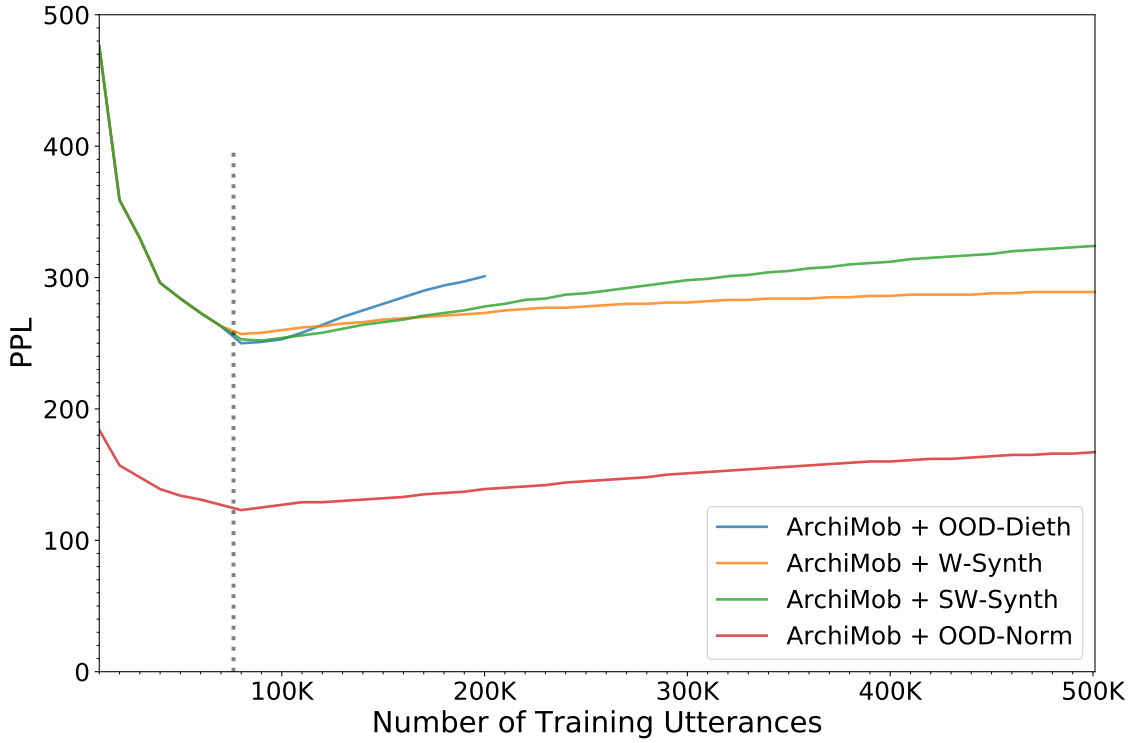

Figure 10: Test set PPL with increased LM training data for Dieth and normalised textual representations of Swiss German. Note, the vertical, dotted grey line indicates the upper limit of the ArchiMob training data (76k utterances).

For modelling Dieth-transcribed Swiss German, LM performance differs considerably depending on the nature of the extended training data. The effect of incorporating OOD data that more closely reflects ArchiMob's domain and the Dieth orthography is demonstrated relatively clearly (blue line). Here, the addition of a small amount of OOD data corresponding to the utterances taken from the Schawinski transcripts help to improve test set PPL. This corpus is by far the most similar to ArchiMob in terms of domain and textual representation. As we start to include OOD utterances from other resources, namely the PAZTeK corpus, NOAH's corpus and CH-Web, test set PPL rises rapidly.

Exploiting SW-Synth data (green line) also shows some slight improvement beyond the ArchiMob training data threshold but, interestingly, it shows a trajectory similar

to that of relying on OOD-Norm data for modelling normalised Swiss German (red line). In contrast, the W-Synth trained LM (orange line) performs slightly worse than all other models when trained on the first 80,000 utterances. Increasing the number of synthesised training instances, causes test set PPL to increase only slightly above the in-domain benchmark before it begins to plateau with around 350,000 training utterances. These results are further summarised in Table 8.

Training data	80k utterances	All utterances	Relative increase
ArchiMob + OOD-Dieth	250	301*	20.4%*
ArchiMob + W-Synth	257	289	12.5%
ArchiMob + SW-Synth	253	324	28%
ArchiMob + OOD-Norm	123	167	35.7%

Table 8: Test set PPL scores with increased training data. Relative increase shows the percentage increase between the test set PPL scored after training on the first 80k utterances in each data set and the test set PPL scored after training on all utterances in each set. *Note: for the LM trained on the concatenated ArchiMob and OOD-Dieth data set, all utterances amount to only 200k training utterances, unlike the others which amount to 500k each.

Figure 10 reveals that there is an evident sweet spot for the amount of training data for modelling Swiss German in our test set. A small amount of additional data, either OOD or subword-level synthetic, helps to achieve optimum test set PPL. We hypothesise that this is due to the fact that the open-vocabulary N-gram LM is able to leverage additional information to estimate decent probabilities for the special `<unk>` symbol, which covers all OOVs in the test set. Increasing the amount of training data beyond this point leads to poorer test set PPLs since the probability estimates of OOVs quickly grow further and further away from the true distribution of the test set. Since the vocabulary of word-level synthesised data matches that of the ArchiMob training data, no instances of OOVs are ever seen during LM training. As a consequence, this model assigns the probability estimates for OOVs arbitrarily and thus the model fails to account for them accurately in the test set.

6.3 Combining LMs with Linear Interpolation

In the previous section, we saw that expanding LM training data by concatenating multiple data sources together to train a single smoothed LM fails to improve LM

performance in terms of test set PPL. As pointed out by Hsu [2007], because each corpus differs in size and is not equally well-matched to the test set, summing N-gram counts over all available training data is a somewhat brute-force approach that rarely leads to optimum results. A more elegant and perhaps more effective approach is to train individual LMs on the different corpora separately and then combine them into a single LM using simple linear interpolation (see Section 2.2.1.1).[4] Combining two backoff LMs with linear interpolation results in a third backoff LM, which can easily be applied directly in first-pass decoding for ASR. While this function exists for most LM toolkits, the major advantage of using the MITLM toolkit is that the interpolation weights can be efficiently optimised on the relevant devset during training.

To investigate this approach, we train separate trigram LMs on the ArchiMob data and each additional data set. Each model is an open-vocabulary, modKN smoothed trigram model, trained with the MITLM toolkit. This time, we do not limit the vocabulary of the LMs. Instead we simply add allowance for the special `<unk>` symbol to be included arbitrarily. The motivation for not restricting the vocabulary is simply that we want to try to incorporate potentially useful information about new words from the LMs trained on the different corpora.

Dieth Models	
Training Corpus	**PPL**
ArchiMob (Dieth)	278
PAZTeK	525
NOAH	489
Schawinski	226
CH-Web	884
W-Synth	389
SW-Synth	360
All (brute force)	413
All (interpolated)	324

Normalised Models	
Training Corpus	**PPL**
ArchiMob (Norm)	124
Tatoeba	410
Tüba-D/S	219
Opensubs	747
All (brute force)	223
All (interpolated)	135

Table 9: Test set PPL scores for separately trained trigram LMs with modKN smoothing and their linearly interpolated combinations.

Table 9 shows the test set PPL scores for each of the individually trained LMs and their the resulting linearly interpolated LM for each textual representation. As men-

[4] Alternatively, models can also be combined with log linear interpolation, however, this is more computationally expensive [Mikolov et al., 2011a], and the resulting LM does not allow itself to be encoded as a standard backoff LM [Hsu, 2007].

tioned above, comparing the perplexity of LMs trained with different vocabularies is problematic, since LMs with small vocabularies can score high perplexities simply by assigning a high probability to OOVs [Buck et al., 2014]. Nevertheless, the PPLs reported here can be used to indicate roughly how well each individual data set matches the ArchiMob corpus data in terms of textual representation and domain.

For our non-normalised test set, CH-Web is clearly a poor match, while the Dieth transcriptions in ArchiMob and the Schawinski transcripts are much more well-suited. Likewise, for normalised Swiss German, the Open Subtitles corpus appears to be a very poor match for modelling text in the ArchiMob corpus. Nevertheless, the lower test set PPL scores for both interpolated models demonstrate that linear interpolation is a far more effective alternative for leveraging additional LM training data than simply combining all the data into a single training set. Unfortunately, the test set PPL scores for interpolated models are higher than those trained solely on the ArchiMob data, indicating that ASR performance is unlikely to improve for our current task. However, given a more general-purpose Swiss German language modelling task, these different corpora may prove considerably more useful.

7 Incorporating LM Improvements into Swiss German ASR

In the previous chapter we looked at ways of improving statistical N-gram language models for our Swiss German speech corpus. The advantage of these types of models is that they are encoded as backoff LMs and can easily be integrated into contemporary ASR systems. In this section, we see how the improvements gained in test set PPL correspond to improvements in an ASR system for Swiss German. Firstly, we consider the Dieth-transcribed representation of Swiss German and present an end-to-end evaluation of systems built with a selection of LMs from Chapter 6. Secondly, we compare the performance of multiple systems relying on the normalised textual representation of Swiss German. Following this, we discuss the results and contextualise the overall improvements.

7.1 System Evaluation

To inspect ASR performance given the different N-gram LMs described in the previous chapter, we compile multiple speech recognisers and run end-to-end evaluations with the ArchiMob test set. We train two AMs on the full ArchiMob training set using the best performing system setups from our exploratory experiments in Chapter 5: STTD and STTN-Z. Each AM is shared between all systems for the appropriate text representation. The evaluation metric we are aiming to optimise is, of course, word error rate (WER). However, since the regular WER is not well-suited to the non-standardised Dieth orthography, we again need to consider our flexible implementation (FlexWER).

7.1.1 ASR for Dieth-Transcribed Swiss German

Following our experiments from Section 6.1, all LMs are trained with the MITLM toolkit and smoothed with interpolated modKN smoothing. As a baseline, we use

a closed-vocabulary trigram LM trained on the full set of ArchiMob training utterances. As we showed in Figure 9(a) from Section 6.1, this LM provides optimum test set PPL over LMs with other smoothing techniques and larger N-gram orders. Thus, we consider this baseline to be a rather strong starting point.

Table 10 shows the performance of each system. Here, LMs are specified according to their N-gram order, while 'vocab' indicates whether or not they are open or closed vocabulary LMs. For example '3-gram C' indicates a trigram LM without the inclusion of an `<unk>` token as a placeholder for OOV words.

LM	Vocab	Training data	PPL	%WER	%FlexWER
3-gram (baseline)	C	ArchiMob (Dieth)	281	55.52	34.83
5-gram	C	"	282	55.55	34.86
4-gram	O	"	257	54.54	33.93
3-gram	O	OOD-Dieth (80k)	**250**	54.39	34.01
3-gram	O	OOD-Dieth (90k)	251	**54.36**	**33.73**
3-gram	O	OOD-Dieth (200k)	363	57.09	35.23
3-gram	O	All (interpolated)	327	55.28	34.68

Table 10: End-to-end evaluation of ASR for Dieth-transcribed Swiss German.

From these results, we see that the best performance is achieved by using a trigram LM trained on the first 90,000 utterances from the combined ArchiMob + OOD-Dieth dataset. This corresponds to the entire ArchiMob training set, the Schawinski transcripts and most of the PAZTeK corpus. With this LM, we achieve a WER of 54.36% and a FlexWER of 33.73%, with the standard WER showing a 2.1% relative improvement over the baseline. In contrast, extending LM training data with additional OOD examples has a considerably detrimental effect on system performance and results in the worst performing system, with a WER of 57.09% and a FlexWER of 35.23%. This supports the findings shown in the previous chapter, where we considered only an intrinsic evaluation of LMs. The success of exploiting OOD data for LM training relies heavily on the additional data being well-matched to the target domain and textual representation, which is a considerable challenge given the nature of Swiss German's various orthographic representations.

Comparing LM test set PPL with system WER for each system shows that, with the exception of the large interpolated model, there is generally a slight positive correlation between test set PPL and WER. While test set PPL scores fluctuate considerably relative to the baseline (+29.2/-11), changes in WER are remain rather

small (+2.8%/-2.1%).

Our flexible WER metric also shows a generally positive correlation with standard WER measure. According to the FlexWER measure, the best performing system achieves an improvement of 3.15% over the baseline. However, as mentioned in Section 5.3.1, it must also be noted here that our implementation of FlexWER is a 'greedy' measure and thus should be interpreted cautiously. In calculating this error rate, we rely solely on the mapping between normalised forms and Dieth surface forms and do not account for potential normalisation errors, which may lead to overly generous scores for the minimum edit distance between a hypothesis and reference transcription.

7.1.2 ASR for Normalised Swiss German

For the normalised textual representation, we consider only the standard WER metric for system evaluation. Again, all models are trained with the MITLM toolkit using modKN smoothing and we adopt the simple trigram model trained on the normalised ArchiMob training set with a closed vocabulary as a baseline. Table 11 provides the results for these systems.

LM	Vocab	Training data	PPL	%WER
3-gram (baseline)	C	ArchiMob (norm)	125	49.02
5-gram	C	"	124	**48.81**
5-gram	O	"	126	49.28
3-gram	O	OOD-Norm (80k)	123	49.26
5-gram	O	OOD-Norm (80k)	**121**	49.23
3-gram	O	OOD-Norm (1m)	210	51.47
3-gram	O	All (interpolated)	136	48.87

Table 11: End-to-end evaluation of ASR for normalised Swiss German.

Here, the correlation between test set PPL and WER is less apparent. Nevertheless, looking at the results of this end-to-end system evaluation, modelling normalised Swiss German does indeed benefit from higher order N-gram LMs, as was suggested by the PPL performance improvements shown in Figure 9(b) from Section 6.1.3. The best performing system makes use of a 5-gram LM, although the relative improvement over the baseline is extremely small (0.42%).

Unlike the systems that rely on Dieth transcriptions, the interpolated LM, which

combines separately trained LMs, does provide some slight improvement over the baseline (0.3%) for normalised Swiss German. We hypothesise that this is due to the fact the information captured in the different LMs is considerably less noisy and thus more representative of the normalised text type found in ArchiMob. Again, the worst performing system is that which uses the 'brute-force' N-gram LM, trained on the entire ArchiMob + OOD-Norm data set with a WER of 51.47%. This provides further support for the claims made by Hsu [2007] that such an approach is sub-optimal.

7.2 Discussion

Immediately evident from the scores reported in Table 10 is the large discrepancy between standard WER and FlexWER when evaluating non-standardised ASR output. Here, the FlexWER scores provide a much more optimistic view on system performance. Manual inspection of the system output transcriptions also suggests that these scores do indeed provide a more accurate idea of how well the system manages to recognise Swiss German words in the input speech utterances. Table 13 in Appendix A provides a small number of system output examples, demonstrating the effect of considering FlexWER as an evaluation metric.

While the system improvements reported here are indeed not large, they are in line with WER improvements reported in other studies. It appears that comparing different LMs typically leads to only small relative improvements, often of as little as 2% [cf. Arisoy et al., 2012; Mikolov, 2012]. This is due to the fact that, as an extrinsic evaluation metric, WER considers the entire system and assumes that reference transcriptions are perfect, which is unfortunately not always the case. Poor audio quality, performance of the acoustic model, the accuracy of the pronunciation lexicon and the language model all contributing factors for this evaluation metric and thus it is difficult improve by tweaking just a single component within a large and complex system.

End-to-end (E2E) approaches for ASR offer an alternative to the conventional systems that we have explored in this work. E2E ASR relies on large neural network models and a simplified training pipeline that aims to train and optimise the entire system as a whole [Kurata et al., 2019]. While these systems are gaining more and more attention, they typically require much more data than conventional systems as the large neural models tend to rapidly overfit to small amounts of training data [Kurata et al., 2019]. Thus, for low-resource languages and scenarios where limited data is available, traditional approaches remain an important research area.

If we compare the evaluations performed here to those done as part of the exploratory experiments on system setups in Section 5.3, we see a considerable leap in performance of approximately 14 percentage points for both textual representations. There are two major factors to account for this improvement. First, the amount of training data used to train the systems described here, is almost double that of the ArchiSmall subset that was used in the earlier experiments. Second, given that we sample utterances from the entire collection of the ArchiMob corpus to create our test set, it is likely that the evaluations performed in Section 5.3 involved utterances from unknown speakers, whereas the systems evaluated here have been trained and tested on utterances produced by the same speakers. Therefore, this evaluation fails to give an indication of speaker-independent ASR for Swiss German and is perhaps a little too optimistic. We believe that a test set involving only new speakers should be used in future to gain a more realistic understanding of how well these systems perform for Swiss German in general.

Another factor that should be considered is, of course, the amount of available data. ASR is a difficult task and relies on large volumes of annotated speech data, typically in the order of a 1,000 hours. However, our system is trained on less than 60 hours of speech data and tested on as little as 1.5 hours. As pointed out by Mikolov [2012], WER results reported on such small data sets are often noisy themselves and can show absolute variance of up to 0.5%. Therefore, more resources are necessary to guarantee the soundness and reliability of future experiments. Nevertheless, our findings seem relatively consistent and can thus serve as a baseline for further experimentation as well as providing an idea of the effect of improved N-gram language models on ASR for Swiss German.

8 Conclusion

In this work, we have addressed a major challenge in automatic speech recognition for the non-standardised low-resource language Swiss German. We aimed to investigate (a) how we could represent Swiss German orthographically for the purpose of automatically converting speech to text and (b) how we could best model Swiss German given the two potential textual representations. To this end, we conducted ASR system evaluations for both Dieth-transcribed and automatically normalised Swiss German and compared the performance of different system setups involving alternative methods for representing word-level pronunciations and various statistical N-gram language models.

Given that the conventional evaluation metric for ASR is not suitable evaluating performance on non-standardised languages, comparing system performance between standardised and non-standardised textual representations with the regular WER metric is hardly fair. Therefore, when evaluating ASR performance for non-standardised language, it is crucial to adopt an appropriate evaluation measure that takes into account potential lexical variation, as done by our FlexWER metric. Our investigations on various system setups showed that a partially phonemic, non-standardised textual representation of Swiss German can achieve transcription error rates as low as 33.73%, provided the evaluation metric accounts for permissible lexical variation in the output transcriptions. Alternatively, we showed that a normalised textual representation of Swiss German, which reduces lexical diversity in the surface text, can achieve error rates of 48.81% when only considering the standard WER evaluation metric.

Our best-performing system, according to the standard metric, leverages a simplified pronunciation lexicon in order to model word-level pronunciation for normalised Swiss German. Despite the high degree of spoken variation, attempting to cover pronunciations for multiple Swiss German varieties leads to greater confusability in the acoustic model and has a negative effect on overall performance. In addition, the lexicon used was relatively small, resulting in approximately 75% of word forms in the ArchiMob training set and 35% of the test set being out-of-vocabulary. Thus, a first step in future work should address expanding this lexicon to cover more word types

from the data set. This could be achieved through a data-driven approach, which exploits the word-level pronunciations in the existing URPP-Swisscom SAMPA dictionary.[1]

The decision of which textual representation to use for Swiss German ASR should consider the intended application of a speech recognition system. If output texts should represent spoken Swiss German in an approximated phonemic orthography, using a non-standardised textual representation such as the Dieth orthography provides a viable solution. On the other hand, if output texts are to be automatically processed in downstream tasks, such as parsing for natural language understanding, a normalised text representation should be adopted as the target text. In both cases, however, respective error rates of around 33% (FlexWER) and 48% indicate that output texts would likely require manual corrections before being further processed.

We investigated the performance, both intrinsically and extrinsically, of multiple statistical language models for Dieth-transcribed and automatically normalised textual representations of Swiss German. As part of this, we considered the effect of N-gram order, different smoothing techniques and exploiting additional language model training data. Our experiments showed that for modelling Dieth-transcribed Swiss German, the largest gains were achieved simply by modelling an open vocabulary with the inclusion of an `<unk>` symbol to represent unseen words. Estimating accurate probabilities for the OOV symbol during LM training is crucial for achieving good results. We found that exploiting a small amount of additional, out-of-domain data is most suitable for this purpose.

In the context of modelling normalised Swiss German, performance gains are possible by exploiting slightly longer context histories, using 5-gram language models smoothed with modified Kneser-Ney smoothing. While our results show only small improvements in our extrinsic evaluation of system performance over the baselines for each textual representation, the experiments have demonstrated some of the major challenges involved in representing and modelling non-standard and low-resource languages like Swiss German in ASR.

Our experiments have focused primarily on exploiting backoff N-gram language models that can easily be incorporated into conventional ASR system architectures. These language models provide a neat and tidy solution for relatively quick hypothesis search during decoding. Exploiting neural language models in Kaldi's ASR architecture requires multi-pass decoding techniques, such as N-best list reranking or lattice rescoring. Pilot experiments conducted as part of this work showed

[1] State-of-the-art sequence-to-sequence models can be applied to the task of grapheme to phoneme conversion, for example `https://github.com/cmusphinx/g2p-seq2seq`.

that building well-performing RNNLMs that can easily be integrated into the Kaldi framework is indeed challenging due to the small amount of training data and the fact that rescoring very large word lattices is memory intensive. Therefore, future research is needed to investigate possible performance improvements by exploiting neural network language models and potentially larger data sets.

References

Aggarwal, R. K. and Dave, M. (2011). Acoustic modeling problem for automatic speech recognition system: Conventional methods (Part I). *International Journal of Speech Technology*, 14(4):297–308.

Ali, A., Nakov, P., Bell, P., and Renals, S. (2017). WERd: Using Social Text Spelling Variants for Evaluating Dialectal Speech Recognition. In *arXiv:1709.07484 [Cs]*, pages 141–148, Okinawa, Japan.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In Holub, J. and Žďárek, J., editors, *Implementation and Application of Automata*, volume 4783, pages 11–23, Berlin, Heidelberg. Springer Berlin Heidelberg.

Arisoy, E., Chen, S. F., Ramabhadran, B., and Sethy, A. (2014). Converting Neural Network Language Models into Back-off Language Models for Efficient Decoding in Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):184–192.

Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. (2012). Deep Neural Network Language Models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-Gram Model? On the Future of Language Modeling for HLT*, pages 20–28, Montréal, Canada. Association for Computational Linguistics.

Beck, E., Zhou, W., Schlüter, R., and Ney, H. (2019). LSTM Language Models for LVCSR in First-Pass Decoding and Lattice-Rescoring. *arXiv:1907.01030 [cs, eess, stat]*.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Bucheli, C. and Glaser, E. (2002). The Syntactic Atlas of Swiss German Dialects: Empirical and Methodological Problems. *Syntactic microvariation*, 2:41–73.

Buck, C., Heafield, K., and Van Ooyen, B. (2014). N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland. European Language Resources Association.

CallMiner (2013). Phonetics vs. LVCSR: Under the Hood of Speech Analytics. https://callminer.com/blog/phonetics-vs-lvcsr-hood-speech-analytics/.

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

Chen, X., Liu, X., Gales, M. J. F., and Woodland, P. C. (2015). Investigation of back-off based interpolation between recurrent neural network and n-gram language models. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 181–186, Scottsdale, AZ, USA. IEEE.

Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., Jaitly, N., Li, B., Chorowski, J., and Bacchiani, M. (2018). State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778, Calgary, Canada. IEEE.

Clematide, S., Frick, K., Aepli, N., and Goldman, J.-P. (2016). Crowdsourcing Swiss Dialect Transcriptions for Assessing Factors in Writing Variations. *Bochumer Linguistische Arbeitsberichte*, pages 62–67.

Dave, N. (2013). Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition. *International journal for advance research in engineering and technology*, 1(6):1–4.

Deoras, A., Mikolov, T., Kombrink, S., Karafiat, M., and Khudanpur, S. (2011). Variational approximation of long-span language models for lvcsr. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5532–5535, Prague, Czech Republic. IEEE.

Dieth, E. (1986). *Schwyzertütschi Dialäktschrift*. Sauerländer, Aarau, second edition.

Donaj, G. and Kačič, Z. (2016). *Language Modeling for Automatic Speech Recognition of Inflective Languages: An Applications-Oriented Approach Using*

Lexical Data. SpringerBriefs in Speech Technology. Springer International Publishing.

Elfeky, M. G., Moreno, P., and Soto, V. (2018). Multi-Dialectical Languages Effect on Speech Recognition: Too Much Choice Can Hurt. *Procedia Computer Science*, 128:1–8.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2):179–211.

Fahy, J. (2016). German Speakers Re-learn their Language. https://www.swissinfo.ch/eng/society/tongue-twisting_german-speakers-re-learn-their-language/41976380.

Federal Statistical Office (2019). Statistical Data on Switzerland 2019.

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

Gold, B., Morgan, N., and Ellis, D. (2011). *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons, Inc., Hoboken, USA.

Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool, San Rafael, CA.

Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. In *Proceedings of the Eighth International Language Ressources and Evaluation (LREC)*, Istanbul, Turkey.

Goodman, J. T. (2001). A Bit of Progress in Language Modeling. *arXiv preprint cs/0108005*.

Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*.

Hain, T. (2002). Implicit Pronunciation Modelling in ASR. In *Proceedings of Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology (PMLA)*, pages 129–134, Colorado, USA.

Hannemann, M. (2013). Weighted Finite State Transducers in Automatic Speech Recognition.

Hasan, A. S. M., Islam, S., and Rahman, M. (2012). A comparative study of witten bell and kneser-ney smoothing methods for statistical machine translation. *Journal of Information Technology (JIT)*, 1:1–6.

Heafield, K. (2013). *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. PhD Thesis, Carnegie Mellon University, Pittsburgh, USA.

Hess, C. W., Ritchie, K. P., and Landry, R. G. (1984). The Type-Token Ratio and Vocabulary Performance. *Psychological Reports*, 55(1):51–57.

Hinrichs, E. W., Bartels, J., Kawata, Y., Kordoni, V., and Telljohann, H. (2000). The Tübingen Treebanks for Spoken German, English and Japanese. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 550–574. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hollenstein, N. and Aepli, N. (2015). A Resource for Natural Language Processing of Swiss German Dialects. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, Essen, Germany.

Hori, T., Cho, J., and Watanabe, S. (2018). End-to-end Speech Recognition with Word-based RNN Language Models. *arXiv:1808.02608 [cs]*.

Hsu, B.-J. P. (2007). Generalized linear interpolation of language models. In *Proceedings of the 2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 136–140, Kyoto, Japan. IEEE.

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.

Huang, G., da Silva, T. F., Lamel, L., Gauvain, J.-L., Gorin, A., Laurent, A., Lileikyte, R., and Messouadi, A. (2017). An investigation into language model data augmentation for low-resourced STT and KWS. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5790–5794, New Orleans, USA. IEEE.

Hui, J. (2019). Speech Recognition — Weighted Finite-State Transducers (WFST). https://medium.com/@jonathan_hui/speech-recognition-weighted-finite-state-transducers-wfst-a4ece08a89b7.

Jayaweera, M. P. and Dias, N. G. J. (2015). Application of witten-bell discounting techniques for smoothing in part of speech tagging algorithm for sinhala language. In *Proceedings of the International Postgraduate Research Conference*, Sri Lanka.

Jing, K. and Xu, J. (2019). A Survey on Neural Network Language Models. *arXiv:1906.03591 [cs]*.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Draft)*. Prentice Hall, New Jersey, USA, second edition.

Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Draft)*. Third edition.

Kaldi (2019). Kaldi: About the Kaldi project. https://kaldi-asr.org/doc/about.html.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

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

Klakow, D. and Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 181–184, Detroit, USA.

Kolde, G. (1981). *Sprachkontakte in Gemischtsprachigen Städten: Vergleichende Untersuchungen Über Voraussetzungen Und Formen Sprachlicher Interaktion Verschiedensprachiger Jugendlicher in Den Schweitzer Städten Biel/Bienne Und Fribourg/Freiburg*, volume 37. Steiner Franz Verlag.

Kurata, G., Audhkhasi, K., and Kingsbury, B. (2019). IBM Research advances in end-to-end speech recognition at INTERSPEECH 2019. https://www.ibm.com/blogs/research/2019/10/end-to-end-speech-recognition/.

Lecorvé, G. and Motlícek, P. (2012). Conversion of Recurrent Neural Network Language Models to Weighted Finite State Transducers for Automatic Speech Recognition. In *Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association*, Portland, Oregon, USA.

Lison, P. and Tiedemann, J. (2016). OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, Portorož, Slovenia.

Lusetti, M. (2018). *Normalization of Swiss German WhatsApp Messages with Statistical and Neural Sequence-to-Sequence Methods*. MA Thesis, University of Zurich, Zurich, Switzerland.

Maas, A. L., Qi, P., Xie, Z., Hannun, A. Y., Lengerich, C. T., Jurafsky, D., and Ng, A. Y. (2015). Building DNN Acoustic Models for Large Vocabulary Speech Recognition. *arXiv:1406.7806 [cs, stat]*.

MacCartney, B. (2005). NLP Lunch Tutorial: Smoothing.

Makhoul, J. and Schwartz, R. (1995). State of the art in continuous speech recognition. *Proceedings of the National Academy of Sciences*, 92(22):9956–9963.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT press.

Martins, C., Teixeira, A., and Neto, J. (2004). Language Models in Automatic Speech Recognition. *REVISTA DO DETUA*, 4.

Miao, Y., Gowayyed, M., and Metze, F. (2015). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174.

Mikolov, T. (2012). *Statistical Language Models Based on Neural Networks*. PhD Thesis, Brno University of Technology, Brno, Czech Republic.

Mikolov, T., Deoras, A., Kombrink, S., and Burget, L. (2011a). Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Proceedings of the Twelfth Annual Conference of the International Speech Communication Association*, pages 605–608, Florence, Italy.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. In *Proceedings of the*

Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH), pages 1045–1048, Japan.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Černocký, J. (2011b). RNNLM - Recurrent Neural Network Language Modeling Toolkit. In *Proceedings of the 2011 Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 196–201, Waikoloa, Hawaii, USA.

Mohri, M., Pereira, F., and Riley, M. (2008). Speech Recognition with Weighted Finite-State Transducers. In Benesty, J., Sondhi, M. M., and Huang, Y. A., editors, *Springer Handbook of Speech Processing*, pages 559–584. Springer Berlin Heidelberg, Berlin, Heidelberg.

Moore, R. C. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

Nigmatulina, I. (2020). Acoustic Modelling for Swiss German Automatic Speech Recognition (MA Thesis: Forthcoming).

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

Pappas, N. and Meyer, T. (2012). A Survey on Language Modeling using Neural Networks. Idiap-Rr Idiap-RR-32-2012, Idiap Research Institute, Martigny, Switzerland.

Paul, D. B. and Baker, J. M. (1992). The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the Workshop on Speech and Natural Language*, pages 357–362, New York.

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

Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafiat, M., Kombrink, S., Motlicek, P., Qian, Y., Riedhammer, K., Vesely, K., and Vu, N. T. (2012). Generating exact lattices in the WFST framework. In

Procceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4213–4216, Kyoto, Japan. IEEE.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8).

Raju, A., Filimonov, D., Tiwari, G., Lan, G., and Rastrow, A. (2019). Scalable Multi Corpora Neural Language Models for ASR. *arXiv:1907.01677 [cs]*.

Renals, S. (2018). Decoding, Alignment and WFSTs.

Ricker-Abderhalden, J. (1986). Mundart oder Schriftsprache? Von den Sprachschwierigkeiten in der deutschen Schweiz. *Die Unterrichtspraxis / Teaching German*, 19(2):162–177.

Rúnarsdóttir, A. V. (2018). *Re-Scoring Word Lattices from Automatic Speech Recognition System Based on Manual Error Corrections*. MA Thesis, Reykjavík University, Reykjavík, Iceland.

Rusli, I. (2017). Comparison of modified kneser-ney and witten-bell smoothing techniques in statistical language model of bahasa Indonesia. *arXiv preprint arXiv:1706.07786*.

Samardžić, T., Cieliebak, M., and Deriu, J. M. (2018). Future Actions for Swiss German — Workshop Results at SwissText 2018. In *Proceedings of the Third SwissText Analytics Conference*, pages 95–99, Winterthur, Switzerland.

Samardžić, T., Scherrer, Y., and Glaser, E. (2015). Normalising orthographic and dialectal variants for the automatic processing of Swiss German. In *Proceedings of the Seventh Language and Technology Conference (LTC)*, Poznań, Poland.

Samardžić, T., Scherrer, Y., and Glaser, E. (2016). ArchiMob — A Corpus of Spoken Swiss German.

Scherrer, Y. and Kellerhals, S. (2014). Digitizing the Linguistic Atlas of German-Speaking Switzerland. *Methods in Dialectology XV*.

Scherrer, Y., Samardžić, T., and Glaser, E. (2019). Digitising Swiss German: How to Process and Study a Polecentric Spoken Language. *Lang Resources & Evaluation*, 53:735–769.

Schmidt, L., Linder, L., Djambazovska, S., Lazaridis, A., Samardžić, T., and Musat, C. (2020). A Swiss German Dictionary: Variation in Speech and Writing. *arXiv:2004.00139 [cs]*.

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)*, pages 236–240, Istanbul, Turkey.

Schmidt, T. and Schütte, W. (2010). FOLKER: An Annotation Tool For Efficient Transcription Of Natural, Multi-Party Interaction.

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Schwenk, H. and Gauvain, J.-L. (2004). Neural Network Language Models for Conversational Speech Recognition. In *Proceedings of the Eighth International Conference on Spoken Language Processing (ICSLP)*, Jeju Island, Korea.

Siebenhaar, B. (2006). Code choice and code-switching in Swiss-German Internet Relay Chat rooms. *Journal of Sociolinguistics*, 10(4):481–506.

Singh, R. (2013). Part I: Designing HMM-based ASR systems Part II: Training continuous density HMMs.

Stolcke, A. (2002). SRILM — An Extensible Language Modeling Toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP)*, Denver, USA.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at Sixteen: Update and Outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Waikoloa, Hawaii, USA.

Taylor, P. (2009). *Text-to-Speech Synthesis*. Cambridge University Press.

Ueberwasser, S. and Seminar, D. (2013). Non-Standard Data in Swiss Text Messages with a Special Focus on Dialectal Forms. *Non-standard Data Sources in Corpus-based Research*, pages 7–24.

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 30*, pages 5998–6008. Curran Associates, Inc.

Watts, R. (2010). The Ideology of Dialect in Switzerland. In *Language Ideological Debates*, Language, Power and Social Process [LPSP]. De Gruyter.

Wells, J. C. (1997). SAMPA computer readable phonetic alphabet. *Handbook of Standards and Resources for Spoken Language Systems*, 4.

Widmer, C. (2018). Voice recognition for dialects. https://www.swisscom.ch/en/business/enterprise/themen/digital-business/language-recognition-swiss-german.html.

Witten, I. and Bell, T. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Xu, H., Chen, T., Gao, D., Wang, Y., Li, K., Goel, N., Carmiel, Y., Povey, D., and Khudanpur, S. (2018). A Pruned Rnnlm Lattice-Rescoring Algorithm for Automatic Speech Recognition. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5929–5933, Calgary, AB. IEEE.

Young, S. (1993). *The HTK Hidden Markov Model Toolkit: Design and Philosophy*. University of Cambridge, Department of Engineering Cambridge, England.

Young, S. (2008). HMMs and Related Speech Recognition Technologies. In Benesty, J., Sondhi, M. M., and Huang, Y. A., editors, *Springer Handbook of Speech Processing*, pages 539–558. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yu, D. and Deng, L. (2014). *Automatic Speech Recognition*. Springer, New York.

A Tables

Grapheme cluster	Phone symbol
ch	ch
sch	sch
tsch	tcsh, z ch
pf	pf
ng	ng
ph	ph
th	th
ts	z
tz	z
gg	gg
v	f
aa	a
ãã	ã
ee	e
èè	è
ẽẽ	ẽ
⋮	⋮

Grapheme cluster	Phone symbol
⋮	⋮
ii	i
ìì	ì
oo	o
òò	ò
õõ	õ
uu	u
ùù	ù
ää	ä
öö	ö
üü	ü
ai	ai
äi	äi
au	au
äu	äu
ei	ei
y	i, ü

Table 12: Grapheme-phoneme mappings used to derive pronunciation strings for Dieth transcribed Swiss German. This list was originally provided as part of the basic ASR system delivered by Spitch AG. For graphemes with two distinct phone symbols separated by a comma, words receive one pronunciation string for each phone symbol, e.g. the Swiss German word *deutsch* ('German') is represented in the lexicon with two pronunciation strings: 'd e u tcsh' and 'd e u z ch'.

Inspecting the Dieth-transcriptions in the left-hand column of Table 13, examples (a) and (b) demonstrate instances where the FlexWER metric clearly contributes to a drastic decrease in overall system error counts, since almost all of the words in the hypotheses are considered incorrect substitutions according to the standard WER, but not according to FlexWER. Examples (c), (d) and (e) show instances where the FlexWER correctly identifies truly permissible lexical variants but also manages to accurately penalise incorrect substitutions. This shows that the FlexWER metric provides a more accurate indication of how well the system manages to recognise words in the spoken input utterance.

On the right-hand side of Table 13, we provide the normalised textual representation outputs for the same test set utterances. Examples (a), (b) and (c) demonstrate instances where the problems associated with lexical variation in the Dieth-transcriptions are entirely avoided thanks to relying on a normalised textual representation, with the only true recognition error being an insertion error in example (a). On the other hand, examples (d) and (e) show the obvious limitations of relying on a pronunciation lexicon with extremely low lexical coverage. Here, majority of the errors are likely due to the fact that the words *Feuerwehr* ('fire brigade'), *abspritzen* ('hose down') and *Zimmerherr* ('tenant') do not appear in the pronunciation lexicon. Furthermore, these errors also appear to affect the surrounding words, leading to additional substitution errors (e.g. *er* ('he') in example (d) and '*harte*' ('hard') in example (e)).

		Dieth	Normalised
(a)	**R**	dä han ich chönä verschtaa	dann habe ich können verstehen
	H	de han i chönne verschtoo	dann habe ich können verstehen mann
		WER: 80% / FlexWER: 0%	WER: 20%
		then I could understand	
(b)	**R**	wohäär händ si d imfòrmazioone alli us	woher haben sie die informationen alle aus
	H	wohär händ sii di informazioone alli uus	woher haben sie die informationen alle aus
		WER: 71.43% / FlexWER: 0%	WER: 0%
		from where do you have the information all from	
(c)	**R**	wiso händ si mich ned psuecht	wieso haben sie mich nicht besucht
	H	wiso händ si mich nöd gsuecht	wieso haben sie mich nicht besucht
		WER: 33.33% / FlexWER: 16.67%	WER: 0%
		why did you not visit me	
(d)	**R**	d füürweer hät dänn die müesen abschprüze	die feuerwehr hat dann die müssen abspritzen
	H	vil wèèr hät dän die müsen ab schprüze	** er hat dann die müssen **
		WER: 85.71% / FlexWER: 57.14%	WER: 42.86%
		the fire brigade then had to hose them down	
(e)	**R**	dèè zimmerhèr isch schwigersoon woorde ja	der zimmerherr ist schwiegersohn geworden ja
	H	de zimmerhèr di schwigersoon worde ja	der zimmer harte schweigen sohn worden ja
		WER: 50% / FlexWER: 16.67%	WER: 83.33%
		the tenant became son-in-law yes	

Table 13: Examples of system output for selected test set utterances. Each example is shown with its reference transcription (i.e. ground-truth) in both Dieth-transcribed and normalised text types. Rows prefixed with **R** indicate the reference transcription, while rows prefixed with **H** contain the hypothesis (i.e. system output). Underneath each example, we report the utterance-level WER and FlexWER scores and provide an approximate English translation. Words marked in blue indicate permissible substitutions according to our FlexWER metric, while words in red indicate incorrect substitutions. Deletions are marked with ** and words pertaining to insertions are underlined.

B Figures

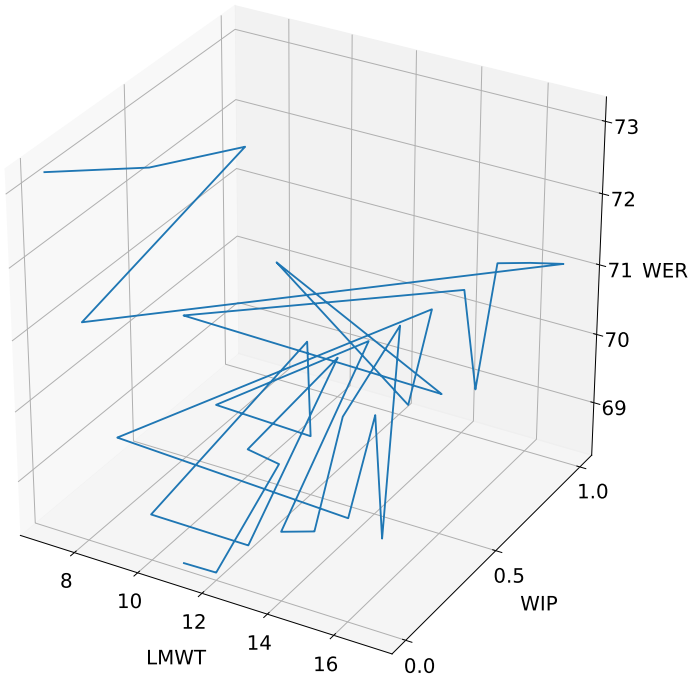

Figure 11: Relationship between the language model weight (LMWT) and word insertion penalty (WIP) on the word error rate (WER) of a system. Here, the LMWT is plotted along the x-axis and the WIP is plotted on the y-axis. The z-axis indicates the resulting WER. As can be seen, in this example, the lowest WER is achieved with with a LMWT of 11 (or 12) combined with a WIP or 0.0. Our experiments revealed that in most cases, LMWTs of around 10 - 12 resulted in the best WER, combined with WIP for 0.0 for the Dieth-transcribed textual representations of Swiss German, and 1.0 for normalised Swiss German.

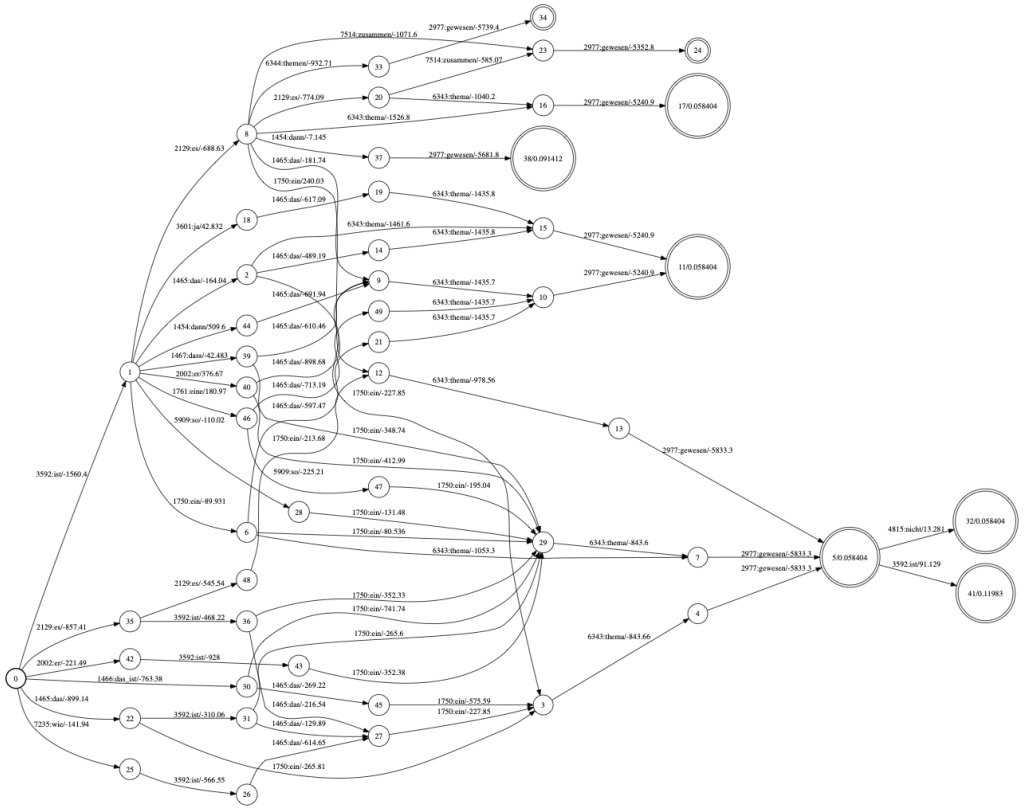

Figure 12: Output word lattice for the normalised Swiss German utterance '*Ist es ein Thema gewesen?*' ('Was it an issue?'). Following the lattice from left to right, a large number of potential hypotheses can be derived. Nodes represent timesteps between words and edges encode particular words. Each word is associated with its AM score (preceding the colon ':') and the LM score (following the forward slash '/'). LM scores are represented as negative log probabilities.