Fine-tuning ASR Models for Very Low-Resource Languages: A Study on Mvskoke

Julia Mainzinger

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2024

Committee:

Gina-Anne Levow

Rolando Coto-Solano

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Fine-tuning ASR Models for Very Low-Resource Languages: A Study on Mvskoke

Julia Mainzinger

Chair of the Supervisory Committee:
Gina-Anne Levow
Linguistics

Recent advancements in multilingual models for automatic speech recognition (ASR) have significantly improved accuracy for languages with extremely limited resources. This study focuses on ASR modeling for the Mvskoke language, an indigenous language of America, by fine-tuning three multilingual wav2vec2.0 models: XLSR-53, MMS-300M, and MMS-1B-l1107. Training data is prepared using language documentation resources, and two evaluation sets are designed, one clean and one noisy, to evaluate performance in different settings. Parameter efficiency of adapter training is compared with training entire models, as well as examining the impact of the number of languages used during pre-training. The study also investigates how performance varies with different amounts of training data, by testing models trained with 10, 60, 120, and 243 minutes of data. A trigram language model is trained using cultural documents and transcripts of interviews, and the ASR models are evaluated with and without language model decoding. The findings show that both MMS models outperform XLSR-53 with higher amounts of training data. Notably, training an adapter for the MMS-1B-l1107 proves to be both parameter-efficient and capable of achieving high accuracy with a relatively small amount of data. ASR accuracy begins to converge around 2-4 hours of training data. While using a language model generally improves metrics such as word error rate, it can sometimes degrade the output. The study introduces the first ASR models successfully developed for the Mvskoke language.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

Table Number Page

LIST OF ABBREVIATIONS

ASR: Automatic speech recognition. The technology of modeling human speech, usually translating spoken language into text.

CER: Character error rate. A metric to evaluate a model's performance on speech tasks. Its score represents the minimum edit distance between two character sequences.

CNN: Convolutional neural network. A type of feed-forward neural network, often used for processing continuous signals which have been discretized in a grid-like topology (such as audio and images).

CTC: Connectionist temporal classification. A function that calculates the loss between two unaligned sequences.

DNN: Deep neural network.

E2E: End-to-end (model). A single model that is trained to learn all the steps between an initial input and final output result.

HMM: Hidden Markov model. A statistical model in which emissions (outputs) are determined by a sequence of hidden states.

LM: Language model.

MMS: Massively multilingual speech. A series of speech representation models.

OOV: Out-of-vocabulary. A word not present in a language model's vocabulary set.

SSL: Self-supervised learning. A technique of machine learning which utilizes large amounts of unlabeled training data.

WER: Word error rate. A metric to evaluate a model's performance on speech tasks. Its score represents the minimum edit distance between two word sequences.

ACKNOWLEDGMENTS

My deepest thanks to my advisor, Gina-Anne Levow, for guiding me through this process, providing valuable instruction through both course instruction and personal communication. Thank you to my reader, Rolando Coto-Solano, for providing feedback and mentorship. Thank you to Dr. Jack Martin for generously allowing me to build from his decades of work on Mvskoke language documentation. I am also grateful to the students of the Linguistics Undergraduate Research Apprenticeship Program (LURAP) for their assistance editing audio recordings - Liyana Alam, Bill Yu, Anika Pontis, Myranda Fraker, Cassie Hu, Luke Eriksen, Isabella Lufschanowski, and Kim Dang. Thank you to tribal elders, leaders, and friends who encouraged me through this process and provided wisdom and insight. Thank you to Dr. Monte Randall and the faculty at the College of the Muscogee Nation for their feedback on my work. And finally thank you to my husband, David Mainzinger, and the many friends and family who supported me, cared for and loved my children, and kept us going through the chaos. *Mvto.*

Chapter 1

INTRODUCTION

Endangered languages are often overlooked in research on speech technology and other NLP applications. Research obstacles include data scarcity and the effort it takes to collect new data, as well as funding and a perceived limited impact on small speech communities. However, these technologies can be hugely beneficial to assisting community-led language revitalization efforts and are worthy of the effort it takes, if it is done with consideration and care for the speech community.

Automatic Speech Recognition (ASR) is the technology of modeling human speech and representing it as text. This technology can help speed up transcription and documentation work, as well as be a stepping stone to other applications such as spoken term detection, which can help in identifying certain topics or key information contained in recordings. Other useful applications for the speech community are speech-to-text input and automatic subtitling. These applications can be helpful in encouraging use of the language and promoting language education.

ASR is a relatively mature technology when applied to high-resource languages (Baevski et al., 2020). But it is only more recent advancements such as model size and multilinguality that have enabled comparable accuracy for resource-constrained situations (Pratap et al., 2023). This work focuses on the evaluation and analysis of two highly multilingual speech models when trained for Mvskoke, a language indigenous to the southeastern United States (Martin and Mauldin, 2000).

1.0.1 The Mvskoke Language

The Mvskoke language is spoken by members of the Muscogee (Creek) Nation and Seminole Nation in Oklahoma, and members of the Seminole tribe of Florida. It is estimated that less than 300 first-language speakers remain, and nearly all are over the age of 60[1]. Recent years have seen an interest among tribal members to revitalize the language, which has led to several new initiatives such as a Master-Apprentice Program at the College of the Muskogee Nation, and new educational

[1]This estimate is from personal communication with a member of Ekvn-Yefolecv, a community of Mvskoke people.

and preservation resources being created and collected by the Language Program at the Muscogee Creek Nation tribal government. ASR can assist in some of these efforts.

The language is synthetic and agglutinative, with a traditional orthography of 20 latin letters (Martin, 2011; Frye, 2020). The orthography is relatively transparent and allows for spelling variations. The advantage of a transparent orthography is that transcriptions can remain relatively close to the speech signal. The flexibility of the orthography, however, can cause ASR error rates to appear higher since spelling may be inconsistent across reference transcriptions.

1.1 Significance

The significance of the proposed experimentation is first to test the performance of a pre-trained speech model when fine-tuned for a low data endangered language. Pre-trained end-to-end neural networks have been explored in other low-resource settings, but further experimentation on a new language could yield interesting results. There are many factors that can affect performance, including phonological features of the language, morphology, and the amount and type of data available. Furthermore, one of the pre-trained models, the Massively Multilingual Speech (MMS) project, is newly released and there is very little literature yet published on the performance of MMS when adapted for a new low-resource language.

From another perspective, the process of building an ASR system can result in valuable tools and datasets for the Mvskoke language, including forced alignment systems, a speech corpus, and audio query tools such as spoken term detection. These tools could of course be of great significance to the Mvskoke-speaking community itself.

1.2 Research Questions

This thesis examines state-of-the-art approaches to ASR for the Mvskoke language with the aim of optimizing the utilization of available resources. It evaluates the accessibility and labor intensity associated with different types of data, emphasizing the balance between enhanced model accuracy and human effort. This study seeks to identify the types and quantities of data that yield the most significant improvements in accuracy.

Specifically, the questions to be explored are as follows:

- Pre-training: How does pre-training affect accuracy?
- Data: How much data is needed before results begin to converge?
- Language model: Does including a language model improve accuracy?

1.3 Study Contributions

This study explores these research questions by fine-tuning three wav2vec2.0 models on Mvskoke language training data: XLSR-53, MMS-300M, and MMS-1B-l1107. Both XLSR-53 and MMS-300M have 300 million parameters, while MMS-1B-l1107 contains 1 billion parameters. XLSR-53 is pre-trained on 53 languages, while both MMS models are pre-trained on 1,406 languages. The models are evaluated using varying amounts of training data to assess their performance in extremely low-data scenarios. They are then evaluated on both clean and noisy evaluation sets. Additionally, a trigram language model is trained and incorporated into the models at the inference stage. Models are evaluated both with and without a language model.

Results show that both MMS models outperform XLSR-53 with higher amounts of training data. The best model overall is MMS-1B-l1107 with a WER of 37% and CER of 5% on the clean evaluation set. Using a language model improves accuracy for all of the models, although gains are less as model accuracy improves.

The results of the study contribute to research on speech recognition for languages with limited available data. This study demonstrates that several key advancements in ASR technology such as multilingual pre-training and adapters are effective for improving accuracy for low-resource languages. Three widely-used pre-trained models are evaluated on a new language not present in the original pre-training. The results can guide future researchers in selecting pre-trained ASR models for similarly low-resourced languages.

1.4 Outline of thesis

Chapter 2 contains a survey of the current methods in transfer learning and ASR for endangered languages. Chapter 3 lays out the methodology for experimentation. Chapter 4 contains a brief

overview of the implementation. Chapter 5 shows the results of the experiments, and Chapter 6 discusses those results. Chapter 7 contains the conclusions and directions for future work.

Chapter 2

LITERATURE SURVEY

Automatic speech recognition (ASR) is the technology of modeling human audio speech as text. This kind of technology can help endangered language communities overcome the "transcription bottleneck" by generating transcriptions from recordings. However, low-resource languages face a shortage of training data compared to high-resource languages, making it difficult to train accurate models. Additionally, the task of preparing parallel speech and text data is hugely time-consuming. Transfer learning from models pre-trained on other languages can help alleviate some of these challenges.

2.1 Advances in ASR

The process of speech recognition involves interpreting a time-varying acoustic signal using some sort of stochastic process (Dighe et al., 2020). The earliest stages began in the 1930s, and we are now approaching nearly 100 years of advancement in this field (Juang and Rabiner, 2005). In the 1980s, a statistical method known as Hidden Markov Model (HMM) became the preferred method, and this method forms the backbone of many ASR systems used today (Juang and Rabiner, 2005). HMMs are now typically integrated with deep neural networks (DNN) to form "hybrid" DNN-HMM architectures (Dighe et al., 2020). Today, with advances in computing capacity and data availability, HMM pipelines are being replaced by fully neural end-to-end (E2E) architectures. E2E pipelines require less human effort, but they also require more data, which can be disadvantageous in a low-resource setting. In order to take advantage of E2E technology, low-resource languages can utilize transfer learning of pre-trained models. This section details these various approaches.

2.1.1 HMM-based Methods

HMM-based methods hypothesize word or phone sequences from a sequence of acoustic features. This outputs a set of states that can be used to calculate the probability of words or phones given

an observed sequence of tokens. The output probabilities can then be decoded by the Viterbi algorithm and then converted to words by means of a manually designed pronunciation dictionary (Rabiner and Juang, 1986). In general, HMM-based systems may need less data than fully neural approaches, due to their reliance on domain expertise. However, they can be more labor intensive - although modern tools have made the process more accessible.

The HMM approach has been improved by neural networks in terms of accuracy and ability to utilize larger datasets. Neural networks can be integrated into the pipeline to form "hybrid" HMM-based systems such as a Deep Neural Network - Hidden Markov Model (DNN-HMM). In this setup, the DNN component acts as an acoustic model, generating a probability distribution for a sequence of acoustic frames. These distributions are then used as emission probabilities for the HMM, which outputs acoustic probabilities that can be combined with a language model during decoding (Dighe et al., 2020; Dahl et al., 2012). However, this approach has led to a disconnect between the phonetic objective functions used to train the neural networks and the lexical inputs used to train the speech recognition systems. Furthermore, HMMs also lead to cascade models, where the errors compound between stages, which can lead to worse transcriptions at the end of the process. These issues can be avoided by end-to-end (E2E) ASR systems, which map acoustic sequences directly to orthographic representations (Graves and Jaitly, 2014).

2.1.2 End-to-End Methods

The goal of the neural end-to-end (E2E) approach is to train a single neural network without requiring explicit feature extraction and domain-specific knowledge. E2E systems have largely replaced HMMs because these methods can be trained directly on text transcripts paired with audio, avoiding the need for a pronunciation dictionary. Furthermore, an E2E model can be trained to directly optimize desired metrics such as word error rate, building an acoustic representation that is better integrated with the transcription process. The E2E approach outperforms the HMM-neural network hybrid approach on larger datasets (Graves and Jaitly, 2014). However, in low-data settings, the HMM hybrid sometimes still gives better accuracy over the E2E approach (Jimerson et al., 2023).

The earliest E2E systems for ASR began with Connectionist Temporal Classification (CTC),

which was introduced by Graves et al. (2006). The CTC function adapts the beam search algorithm to calculate grapheme probabilities directly from a sequence of inputs corresponding to audio frames (Graves et al., 2006). It accomplishes this by collapsing sequences of labels that indicate tokens which cover multiple acoustic frames, then determining where word breaks occur. This contrasts with the HMM-based approach which seeks to map each acoustic frame to an output label (Graves and Jaitly, 2014; Li, 2022). The CTC method removes the difficulty of requiring prior alignment between input and target sequences, and puts the training at grapheme-level instead of frame-level. This allows the model to dynamically learn alignment during training.

The CTC approach was further improved by the introduction of using Transformer in the encoder, which uses an attention mechanism to capture long-term dependencies (Li, 2022). This has allowed continued success of the CTC-based E2E methods in recent years. However, the E2E approach still requires large amounts of data, which led to research in self-supervised learning techniques, discussed in the next section.

Both HMM-based and E2E systems can give comparable results for low-resource languages. Brixey and Traum (2022) use Kaldi to build an HMM-hybrid system for Choctaw, a Muskogean language, and achieve a WER of 49.35% with about 45 minutes of data and a small lexicon. An ASR system for Yoloxóchitl Mixtec compares HMM and E2E and finds E2E outperforms with a WER of 16.0% and has been incorporated into documentation workflow (Shi et al., 2021; Amith et al., 2021). A fully-convolutional neural network (CNN) for Seneca outperforms HMM models (Thai et al., 2020). Jimerson et al. (2023) show that an HMM-hybrid trained from scratch can outperform pre-trained neural networks for some languages, but is worse for others. This shows that there is no clear choice for system architecture, and that choice of architecture may in fact be dependent on the features of the language. The current work fine-tunes multilingual E2E transformer models due to the ease of implementation, promising results seen in similar situations, and desire to evaluate the most recent models and methods.

2.2 ASR for Low-Resource Languages

ASR approaches in low-resource settings utilize techniques that maximize the amount and types of data available. Self-supervised learning (SSL), multilingual pre-training, and using text data in language models can provide large benefits with small amounts of data. This section details these

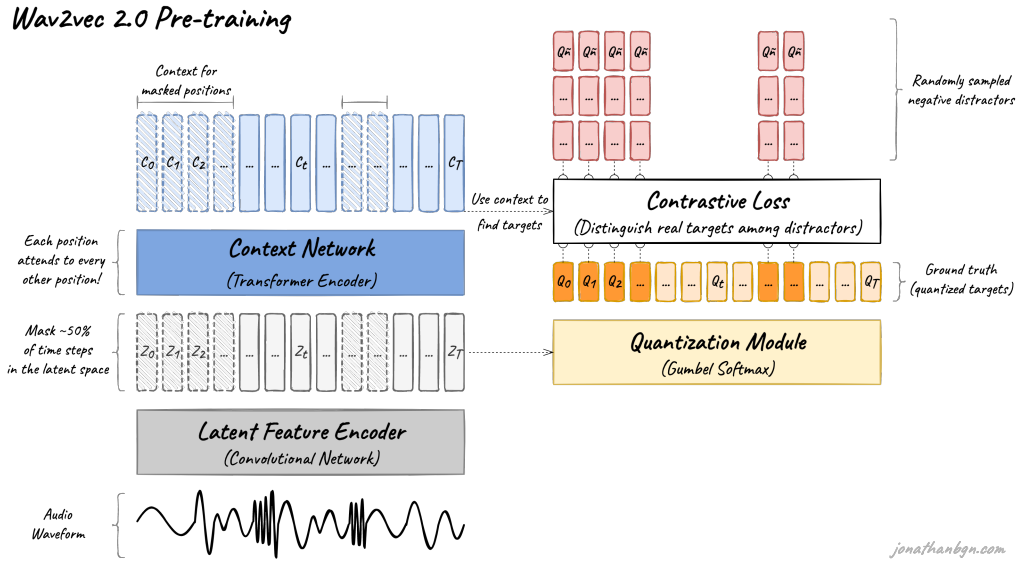

Figure 2.1: **Wav2vec2.0 architecture**. As described in (Baevski et al., 2020) and illustrated by (Boigne, 2021).

techniques and applications for low-resource languages.

2.2.1 Self-supervised learning

Neural networks need large amounts of data to be successful, on the order of thousands of hours of transcribed speech. Since this amount of labeled data is not available for the majority of languages, researchers explored self-supervised learning (SSL) methods, a machine learning technique by which unlabeled data is used to learn general representations. Wav2vec and wav2vec2.0 are frameworks that use SSL to encode speech representations in a masked fashion similar to masked language modeling (Baevski et al., 2020).

The wav2vec architecture is designed to predict speech units for masked parts of speech audio (Schneider et al., 2019). Wav2vec2.0 was introduced in Baevski et al. (2020) with an improved design that includes the Transformer architecture. In wav2vec2.0, raw audio is encoded using a convolutional neural network (CNN), and then the speech is masked and fed to a Transformer encoder to build contextualized representations. The model is then trained by solving a contrastive

task. The transformer layer improves accuracy and reduces data needs (Baevski et al., 2020). In order to be used for ASR, wav2vec2.0 can be fine-tuned with labeled data using CTC.

2.2.2 Pre-trained Models

Wav2vec2.0 refers only to the framework for pretraining - the data used in the pre-training depends on the task. In the case of low-resource ASR, large multilingual datasets are commonly used. Multilingual pre-training seeks to model all possible phonemes in order to predict speech units for the languages represented in the data. This is advantageous for low-resource languages, requiring less labeled data when fine-tuning for a particular language.

Several iterations of wav2vec2.0 have been pre-trained using large multilingual datasets. A popular choice is XLSR-53, a wav2vec2.0 model trained on 53 languages (Conneau et al., 2020). An ASR model for Cherokee using a fine-tuned XLSR-53 has a WER of 64% (Zhang et al., 2022). In their paper on endangered languages of Nepal, Meelen et al. (2024) demonstrate an effective ASR pipeline using XLSR-53 and show the relationship between dataset size and model performance.

In 2023, Meta released the Massively Multilingual Speech (MMS) model, a model pre-trained on 491K hours of unlabeled audio data in 1,406 languages (Pratap et al., 2023). MMS, like XLSR-53, relies on the wav2vec2.0 architecture. It has base models with both 300M and 1B parameters. MMS shows improved performance on most low-resource languages when compared to previous pre-trained models, although there is a degradation among some languages including higher-resource languages. The authors of MMS also fine-tuned the 1B base model for the task of ASR using labeled data of New Testament texts in 1,107 languages (Pratap et al., 2023).

For this work, both XLSR-53 and MMS are evaluated. In this way, results can be compared to other low-resource experiments. Two separate MMS models are considered - one is the self-supervised base 300M parameter model, and the other is the 1B base model after undergoing fine-tuning for ASR. The XLSR-53 model is purely self-supervised, and also has 300M parameters. Further details about these models are discussed in Section 3.2.

2.2.3 Adapters

Houlsby et al. (2019) introduced adapter modules, which allow fine-tuning pretrained models by

adding only a few trainable parameters per task rather than training all of the existing parameters. The recent Massive Multilingual Speech (MMS) models include adapters that are trainable for certain tasks such as ASR, and have been shown to be more memory efficient and yield better performance for low-resource languages (Pratap et al., 2023). Figure 2.2 shows this architecture.

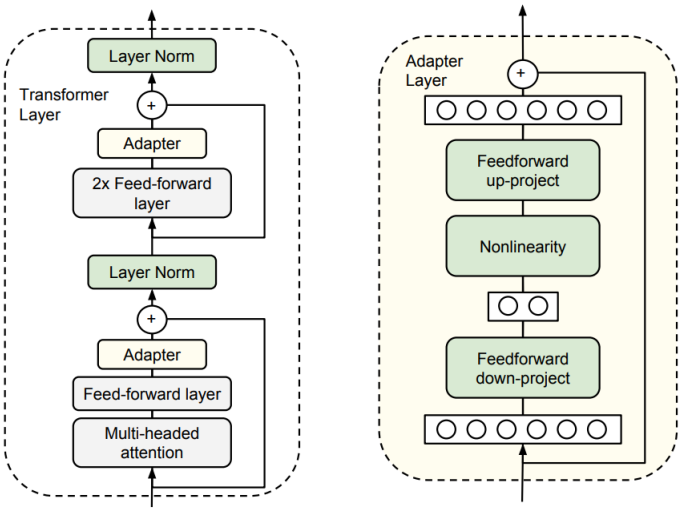

Figure 2.2: **Adapter architecture.** Adapters are added at every block of the transformer. These additional layers serve as a language-specific adapter to allow parameter-efficient fine-tuning (Houlsby et al., 2019).

2.2.4 Language Model Decoding

A language model (LM) generally refers to a text-based model that predicts the next word or character in a sequence. Incorporating a language model into an ASR system can be advantageous because text data is often more readily available than transcriptions of speech data. Moreover, language models provide longer-range, top-down constraints on the transcription output compared to the grapheme-sequence models produced by most CTC-based ASR systems. Nonetheless, with very large speech corpora it may be possible to learn the language model directly from the transcripts (Graves and Jaitly, 2014).

Typically, the language model is trained separately from the ASR model and then incorporated

into the model's inference stage. In the wav2vec2.0 architecture, the acoustic model is trained to produce CTC-encoded texts. The simplest inference is to apply an argmax function to the CTC output to give the most likely grapheme sequence. However, this approach doesn't take into account that a single output can have many alignments, and thus can sometimes miss more probable outputs (Jurafsky and Martin, 2020).

One solution to this problem is to use a modified beam search algorithm. The beam search algorithm maintains a set of possible character sequences. New hypotheses are generated at each time step by extending each existing hypothesis with all possible output characters, retaining only the top candidates. In the context of CTC decoding, repeated characters are handled by collapsing repeats and removing blank characters from the output prefixes. A language model can add additional information to the acoustic model outputs, helping predict the better path. This can be achieved by using the language model's predictions to increase the score for the paths that are more probable, thereby establishing more likely character or word sequences.

ASR systems for low-resource languages can experience an increase in accuracy by using language model decoding, although Jimerson et al. (2023) demonstrate that using a language model always increases accuracy, but the gains are minimal in comparison with other factors such as model architecture. On the other hand, Orken et al. (2020) show that ASR for two agglutinative languages, Turkish and Tatar, see marked improvement from use of a language model. Because there is a sizable collection of text documentation for Mvskoke, this work investigates the performance of the multilingual models with and without language model decoding.

2.3 Chapter Summary

This chapter reviews the recent advances in ASR technology that have enabled improved accuracy for low-resource languages. The first section gives an overview of two current ASR approaches, HMMs and neural E2E methods, and compares the advantages of each. HMM-based methods require more manual effort, but E2E methods require more data. With regards to HMM and E2E in low-resource ASR systems, recent works show that both approaches can give comparably good performance. This work chooses to focus on training pre-trained E2E multilingual transformer models. The second section outlines the specific technologies that are commonly used in low-resource ASR. Wav2vec uses SSL to represent speech in pre-trained models. Adapters help reduce memory re-

quirements while effectively fine-tuning large pre-trained models. Finally, language modeling has the potential to increase accuracy using text data, which can be easier to gather.

Chapter 3

METHODOLOGY

The goal of this work is to evaluate the effectiveness of fine-tuning multilingual models for a very low resource language. This is one state-of-the-art path for ASR that requires less manual work than other methods such as an HMM, and generally requires less data due to the existing pre-trained acoustic information of the multilingual models. The experimentation includes both training an adapter for a pre-trained ASR model as well as training an entire pre-trained model. Additionally, other aspects that are evaluated are how much data is required and whether or not a language model can improve results. This section describes the experimental setup including data preparation, pre-processing pipeline, training configurations, and evaluation methods.

3.1 Data

3.1.1 Data Sources

The texts and recordings used in these experiments primarily come from language documentation work conducted over the last few decades. Two documentation books, Haas et al. (2015) and Gouge et al. (2004) are collections of stories, historical letters, and other cultural documents. A portion of these texts were recorded in a studio setting by two female speakers. From 2015-2017, the Seminole Nation's Pumvhakv School conducted video interviews of fluent Seminole and Mvskoke speakers. The entire New Testament has also been recorded. These resources can be accessed on the Muskogee (Seminole/Creek) Documentation Project website [1]. In order to incorporate male speakers and spontaneous speech, a small segment of the New Testament was selected, as well as a few short sections of recorded interviews. Sources are listed in Table 3.2.

[1] https://muskogee.pages.wm.edu

3.1.2 Splits

Train and development sets are split 90/10 at run-time. Two evaluation sets are kept separate from the training set. "Eval (clean)" is read speech from the same documentation sources as the training set, and "eval (other)" is noisier speech, consisting of one overlapping male speaker and one held-out female speaker. In the transcripts for all the audio data, there are a total of 6,840 utterances and 19,154 words, for an average of 2.8 words per utterance. The train and "eval (other)" sets include both read and spontaneous speech, while the "eval (clean)" set is only read speech. Other features of the datasets are shown in Table 3.3.

3.1.3 Language Model

The text data for the language model (LM) includes the two books above as well as the transcriptions from a series of interviews conducted by the Pumvhakv School in 2015. For these experiments, the interview recordings are not used for training due to noise including nature sounds, speech errors, and singing, but the transcriptions provide valuable vocabulary. The texts and transcriptions of the evaluation set were excluded from the text training data. The text corpus used for LM training has 118,021 words and 27,795 unique words.

3.1.4 Phrase-level Forced Alignment

A portion of the transcriptions by the linguist are segmented at the phrase level in ELAN. However, sometimes the phrases are quite long, up to 30 seconds or longer. This proved a huge burden on the memory at training time, so all of the original transcriptions needed new phrase-level segmentation. Since splicing at regular intervals can lead to splitting at the middle of the word, this required forced alignment to find word boundaries. Therefore a forced alignment pipeline was designed to prepare the data. This provided the additional benefit of generating new phrase-audio pairs from previously unaligned transcriptions.

One difficulty with forced alignment for long audio files is that if there are any discrepancies between the speaker and the human transcription - for example, speech errors or background sounds, then the entire file will often be misaligned. Therefore the first phase of the forced alignment pipeline is to broadly segment long speech recordings into sections with the relevant transcriptions, in order

to contain widespread misalignment.

The first step in the process was to generate a rough transcription using ASR. An initial ASR model was fine-tuned utilizing all of the usable data - that is, the phrases that were short enough before causing memory errors. At the start, this amounted to about 1 hour of transcribed audio. For this, the MMS-1B-l1107 pre-trained model was chosen. This model is used to generate rough transcriptions with time stamps marking the beginning and end of each word.

After the automatic transcription is generated, the output is replaced by the correct transcriptions. This is achieved by stepping through each word of the ASR output and building phrases from the actual transcriptions, using the character error rate as a rough score to find matching phrases. The goal of this step is only to have broad segmentation, so that long files are split up for another pass of finer-grained word alignment.

The final step is a second pass of word alignment, using the broad segmentation as boundaries for the fine-grained alignment. This step employs a pytorch wav2vec2 forced alignment pipeline (Ansel et al., 2024), which utilizes a base pre-trained MMS model [2]. This final step gives word-level alignments for the true transcription.

The resulting alignments are then manually reviewed and adjusted for accuracy. Approximately three-quarters of the transcriptions processed through this pipeline aligned successfully. Those that failed contained too many speech errors, leading to significant portions of the recordings lacking proper alignment. In the more successful cases, misalignments were confined to the broadly segmented regions, requiring only minimal manual corrections, primarily near the boundaries of the larger segments. Using this method, the data was able to be increased to 4 hours.

The data is then saved as .TextGrid files with word-level segments. The audio files are stored alongside the .TextGrid files. This allows for dynamic data preparation by using a Python script that concatenates words to create segments of desired lengths. Finally, a metadata file is created which maps audio segments to transcript phrases, which is then uploaded to Hugging Face as an audio dataset.

---

[2] https://pytorch.org/audio/stable/generated/torchaudio.pipelines.MMS_FA.html

3.1.5 Prepared Datasets

The final data prepared for training and testing is the set of audio phrase pairs that are clean and high-quality, extracted from the original sources. Two evaluation sets were kept separate from the training set. "Eval (clean)" is read speech from the same documentation sources as the training set, and "eval (other)" is noisier speech, consisting of one overlapping male speaker and one held-out female speaker. In the transcripts for all the audio data, there are a total of 6,840 utterances and 19,154 words, for an average of 2.8 words per utterance. The train and "eval (other)" sets include both read and spontaneous speech, while the "eval (clean)" set is only read speech. Other features of the datasets are shown in Table 3.3.

3.2 Models

The study evaluates several multilingual wav2vec2.0 models. As mentioned earlier in Section 2.2.1, wav2vec2.0 is trained by solving a contrastive task over masked speech (Baevski et al., 2020). When trained with multilingual data, these models can jointly learn speech representations that are shared across languages (Conneau et al., 2020). This type of pre-training can be very beneficial for low-resource languages because models are able to leverage acoustic information learned from a large amount of data available in other languages.

XLSR-53 was one of the first models to use entirely unlabeled speech and quickly became popular for low-resource ASR, significantly outperforming previous baselines (Conneau et al., 2020). It is pre-trained on 53 languages using several public datasets, encompassing both low-resource languages with only a few hours of data and high-resource languages with hundreds or thousands of hours. The publicly released model is trained on a total of 56,000 hours of unlabeled speech data and contains 300 million parameters (Conneau et al., 2020).

The authors of XLSR also fine-tuned the model for the purpose of analysis. Results indicate that gains are more substantial for low-resource languages than for high-resource languages due to a "transfer-interference trade-off" (Arivazhagan et al., 2019). Data from multiple languages improves speech representations that benefit low-resource languages, but the model's capacity must be shared across languages, which diminishes performance on high-resource languages (Conneau et al., 2020). The experimentation in this present work evaluates the performance of XLSR-53 when fine-tuned

on Mvskoke, thereby allowing comparison of performance with other publications on low-resource languages.

This study also evaluates models recently introduced by Meta's Massively Multilingual Speech (MMS) project (Pratap et al., 2023). Like XLSR-53, MMS models also feature wav2vec2.0 architecture, but are trained on many more languages. The authors of MMS pre-train models on unlabeled data from 1,406 languages (Pratap et al., 2023). The base models are available in 300 million and 1 billion parameter versions. Of particular interest in this study is the MMS-1B-l1107, a model that was fine-tuned for ASR from the MMS-1B base model (Pratap et al., 2023). This model features an adapter with 2 million parameters on top of the base 1 billion parameters, based off of a method introduced by Houlsby et al. (2019), discussed above in Section 2.2.3. The adapter layers allow the large multilingual acoustic knowledge to be fine-tuned for a new language in a computationally efficient fashion.

In order to compare MMS with XLSR-53, the similarly-sized MMS-300M (Pratap et al., 2023) is also trained. Both MMS-300M and XLSR-53 undergo full parameter training, while only the adapter is trained for MMS-1B-l1107. MMS-1B is not included for this experiment due to memory constraints of the hardware used. Table 3.1 shows more details about these models.

	train type	parameters	hours of audio	# unlabeled langs	# labeled langs
XLSR-53	full	317 million	50K	53	0
MMS-300M	full	317 million	491K	1,406	0
MMS-1B-l1107	adapter	965 million	491K	1,406	1,107

Table 3.1: **Models fine-tuned in this experiment**. XLSR-53 (Conneau et al., 2020) and MMS-300M (Pratap et al., 2023) are trained on unlabeled data, and MMS-1B-l1107 (Pratap et al., 2023) is fine-tuned for ASR on labeled data from 1,107 languages.

Source	Type	Description	Phrase Aligned	# of Speakers	Length (hours)
Gouge	Read	Studio	Yes	1 Female	5.28
Haas/Hill	Read	Studio	Partial	2 Female	1.43
New Testament	Read	Speech errors	No	1 Male	40.52
Interviews	Spontaneous	Noisy	Yes	30+ Male & Female	16.38

Table 3.2: **Data sources and description**. Portions of these sets were prepared as phrase-audio pairs.

3.3 Implementation

Implementation follows the steps detailed by Patrick von Platen to fine-tune the MMS adapter using Huggingface Transformers[3] (Wolf et al., 2019). For the MMS-1B, the base model is frozen and only the adapter layer is trained. For the other two models, the entire model weights are trained. The data is split into sets of 10, 60, 120, and 243 minutes. Early stopping criteria ends training before overfitting. The best model is saved with the lowest character error rate (CER), and then evaluated on the clean and noisy evaluation sets.

The language model is a trigram model trained with KenLM (Heafield, 2011). This LM is then used in a CTC decoder after the models are trained.

3.4 Experimental Design

3.4.1 Paradigms

In these experiments, the goal is to evaluate the ASR system from three aspects: pre-training, data, and language modeling. To facilitate this, the experiment has variables in each category as follows.

Research Question 1. Pre-training: How does pre-training affect accuracy?

All models used in this experiment are Wav2Vec2 models. The models differ by the number of languages involved, number of parameters, and adapter vs non-adapter versions. Models that are tested are shown in Table 3.1.

[3] https://huggingface.co/blog/mms_adapters

	train+dev	eval clean	eval other
Total Length	4.1h	21m	27.6m
Avg. Length	2.6s	2.5s	2.5s
Fem. Speakers	2	2	1
Male Speakers	2	0	1

Table 3.3: **Prepared audio datasets**. Train and development sets are split 90/10 at run-time, and the evaluation sets are held out for testing. The eval (clean) set contains clean read speech from known female speakers. The eval (other) set consists of noisy read data from a known male speaker and conversational speech from a held-out female speaker.

Research Question 2. Data: How much data before results begin to converge?
In order to see the effect of data size, the data is segmented into smaller portions and training run at intervals, with accuracy measured at the following steps:

1. 10 minutes
2. 60 minutes
3. 120 minutes
4. full set (243 minutes)

Research Question 3. Language model: Does including a language model improve accuracy?
To see the effect of language modeling, the models are trained with and without a language model, and then evaluated. Thus the variables are:

1. With language model
2. Without language model

3.5 ASR Training

ASR training implementation follows the steps detailed by Patrick von Platen to fine-tune the MMS adapter using Huggingface Transformers[4] (Wolf et al., 2019). The steps include preprocessing the data, training, and evaluation.

3.5.1 Preprocessing

The first step is to preprocess the data. Specifically, punctuation and nasalization markers are removed. Nasalization markers are inconsistently applied across the transcriptions — they are absent in the New Testament and are seldom used by community members outside of linguistic documentation. Removing these markers improves the consistency of the training data. The tokenization vocabulary is then built to represent the characters present in the transcriptions. Following this, a Wav2Vec2 tokenizer and feature extractor are initialized and uploaded to HuggingFace as a processor to be used for training.

The training dataset is split into train and dev, 90% and 10% respectively, at pre-processing time. The eval (clean) and eval (other) sets are reserved for evaluation after training is completed.

3.5.2 Training

All the models used in this experimentation are Wav2Vec2 with CTC. The models are initialized from pre-trained models from HuggingFace. During training, the data is dynamically split into sets of 10, 60, 120, and 243 minutes. Training ends when the early stopping criteria is met before over-fitting or maximum epoch limit is reached. The hyperparameters are as follows:
Training batch size: 2
Training Epochs: 4
Gradient Accumulation Steps: 4
Learning Rate:1e-3
Maximum epochs: 30
Best model metric: CER
Early stopping threshold: 0.003

[4] https://huggingface.co/blog/mms_adapters

Early stopping patience: 3

For MMS-1B-l1107, the base model is frozen and only the adapter layer is trained. For the other two models, the entire model weights are trained. The best model is saved with the lowest character error rate (CER), and then evaluated on the clean and noisy evaluation sets. Table A.1 in Appendix A details the epoch counts for each training run.

3.5.3 Model Evaluation

Performance is evaluated using word error rate (WER) and character error rate (CER). Word error rate is calculated by comparing predicted words with reference words and counting substitutions (S), deletions (D), and insertions (I), and then dividing by the number of words in the reference (N) (Jurafsky and Martin, 2020). Then the error rate is computed by the following formula:

$$WER = \frac{S + I + D}{N} \tag{3.1}$$

Similarly, the character error rate is calculated in the same way, but with the errors counted at the character-level instead of the word-level. Because Mvskoke is a primarily oral language, and spelling is not entirely standardized, the character error rate is useful in evaluating how readable the transcription is. Many Mvskoke speakers will still be able to read transcriptions with some spelling errors and still have a good idea of what is being said in the recording. These kinds of transcriptions may not necessarily be candidates for archival linguistic documentation, but they are still nonetheless useful for referencing the contents of an audio recording.

The models are evaluated on both the held-out test set, which is clean recordings similar to that in the training set, and noisy recordings. This is to evaluate the usefulness of ASR in a practical setting, which will likely be spontaneous speech in a noisy environment. The models are also evaluated both with and without language model decoding, discussed further in the next section.

3.6 Language Model Training

The data that is used to train the language model comes from the transcripts of the audio training dataset as well as transcripts of interviews and the New Testament. Many of the interviews are too

noisy to be used for training in speech recognition, but the transcripts are useful as text data for language modeling. Table 3.4 describes the features of the text data.

Source	Word Count	# Unique Words
Gouge stories	14,667	4,727
Haas-Hill documents	35,771	12,513
Interview transcriptions	69,412	16,288
Final, cleaned	117,825	27,794

Table 3.4: **Text used for training the language model**. Total and unique word counts from each source. The "Final" dataset represents the text corpus when transcripts of recordings from the audio eval set have been removed.

To build the language model, an n-gram model is trained using KenLM. The text corpus is cleaned with the exact same functions as the pre-processing during the ASR model training, to maintain consistency of the vocabulary. These functions remove punctuation and special characters. After cleaning, the language model is trained by the KenLM "lmplz" module with only the "order" flag specified to 3 to build a trigram model. KenLM estimates unpruned language models with "modified Kneser-Ney smoothing" (Heafield, 2011; Chen and Goodman, 1999).

The resulting language model contains the n-gram counts shown in Table 3.5.

1-grams	27,794
2-grams	89,074
3-grams	112,597

Table 3.5: **Language Model n-gram counts** when built using KenLM

Table 3.6 shows out-of-vocabulary (OOV) rates and perplexity of the three datasets used for ASR training and evaluation, when analyzed with the given language model. The "train+dev" set contains a small amount of OOV tokens, despite most of the training data being present in the language model text corpus. This is due to a portion of the New Testament that is present in the

ASR training data which was not used for language model training.

	train+dev	eval (clean)	eval (other)
OOV count	302	209	623
OOV rate	1.9%	17.8%	27.3%
Mean perplexity	138.1	573.5	1309.6

Table 3.6: **OOV rates and perplexity of ASR datasets** when analyzed using the 3-gram KenLM language model.

After training, the language model is inserted at the decoding stage. Decoding without the language model is a simple argmax function applied to the CTC output of the ASR model. When decoding with the language model, the output is decoded using a modified beam search as discussed in Section 2.2.4. This is implemented using the pyctcdecode library (Lopez et al., 2024), which allows insertion of a KenLM language model. The following default parameters and insertion weights are used:
Alpha: 0.5 (weight for language model during shallow fusion)
Beta: 1.5 (weight for length score adjustment of during scoring)
Unknown score offset: -10.0 (amount of log score offset for unknown tokens)
Beam width: 100 (maximum number of beams at each step in decoding)

Outcomes of model training and language model decoding are presented in the next chapter.

3.7 Chapter Summary

This chapter discusses the data, experimental procedures, and evaluation methods used in this study. It describes the data sources and how the data is prepared to be used for training and evaluation. The experimental design shows the experimental paradigms as well as training setup, including hyperparameters, models used, and data included. Models are evaluated using WER and CER, and clean and noisy data sets. The trigram language model is trained using transcripts and documentation sources, and the resulting model is incorporated into the ASR models at inference.

Chapter 4

RESULTS

This chapter details results from all experiments. The focus of these experiments is to evaluate the pre-trained models, measure the affect of data amount, and assess accuracy with and without a language model.

4.1 Training Runs

The XLSR-53 tended to train fastest, whereas MMS-300M and MMS-1B-l1107 generally took longer. One outlier is that the MMS-300M, when given the full data set, took significantly more training epochs than the other two models. Early stopping criteria ended training around 10-13 epochs for most models except for two models with the 10 minute dataset and the MMS-300M at 243 minutes, the latter of which trained for 23 epochs. Table A.1 and Table 4.2 in Appendix A contain the information for all training runs and evaluation results.

The most significant factor influencing the total runtime is the evaluation time per epoch. The evaluation time for MMS-1B-l1107 is more than twice as long than the smaller models, since the entire model is used during inference. For example, with 60 minutes of training data, both XLSR-53 and MMS-300M took about 30 minutes for evaluation at each epoch, while the MMS-1B-l1107 took 71 minutes. Additionally, larger models require more disk space; the 300 million parameter models occupy 1.2 GB, whereas the 1 billion parameter models requires 3.6 GB.

However, adapter training offers several key advantages in terms of memory efficiency. It enables leveraging larger models while only moderately increasing training time per sample. Table 4.1 shows several training metrics along with training samples per second. MMS-300M can handle 12.3 samples per second, while MMS-1B-l1107 calculates an average of 11.4 samples per second, but considering that the latter has more than triple the number of parameters, it is clear that adapter training is beneficial in terms of computational efficiency. This is achieved because only 2 million parameters weights are updated rather than the full 1 billion, lessening the burden on memory

usage. Furthermore, attempting to train the full MMS-1B model often led to out-of-memory errors, whereas the MMS-1B-l1107 adapter was able to be trained without such issues.

Another advantage is the size of the model weights file after fine-tuning. The adapter file for MMS-1B-l1107 is a mere 8.4 MB, whereas a fine-tuned complete model is the same size as the base model. Storing adapter files is therefore significantly more memory efficient than storing fully fine-tuned models, particularly when fine-tuning a base model for many languages. In such cases, instead of deploying an entire model for each language, only the adapters need to be deployed, significantly decreasing storage requirements.

Model	xlsr-53	mms-300m	mms-1b-l1107
Model parameters	317M	317M	1B
Parameters trained	317M	317M	1M
Model size	1.2 GB	1.2 GB	3.6 GB
Fine-tuned weights size	1.2 GB	1.2 GB	8.4 MB
Average training time (min)	85	388	196
Average training samples per second	14.7	12.3	11.4

Table 4.1: **Training times**, and storage and memory metrics.

4.2 Error Rates

Table 4.2 compares the error rates for both evaluation sets for the models trained with 243 minutes of data. The MMS-1B-l1107 performed best overall when trained on the entire dataset, with 37% word error rate (WER) and 5% character error rate (CER) on the "clean" evaluation set. Language model decoding reduces the WER to 31% and the CER is unchanged. WER and CER are reduced by 6 and 1 percentage points when using the LM on the "other" evaluation set. MMS-300M performs better than XLSR-53 on the "clean" dataset but comparable or slightly worse on the "other" set. This could possibly be due to overfitting, since MMS-300M trained longer before reaching stopping criteria. Transcription samples are shown and discussed in more detail in Section 5.3.

Model	WER clean	CER clean	WER other	CER other
XLSR-53	51	9	97	47
XLSR-53 + LM	36	7	90	47
MMS-300M	48	8	96	48
MMS-300M + LM	33	6	92	48
MMS-1B-L1107	37	**5**	82	33
MMS-1B-L1107 + LM	**31**	**5**	**76**	**32**

Table 4.2: **Eval Results - Clean and Other.** Error percentages for the "clean" and "other" evaluation sets for each model trained on the complete dataset (243 minutes).

4.3 Results by Data Size

Results for each dataset size are shown in Figure 4.1. Interestingly, the XLSR-53 performed better than the MMS-300M on smaller amounts of data. However, more data (4 hours) improves the MMS-300M to a point that surpasses XLSR-53, although this could be due partially to the fact that the former trained longer.

Table 4.3 shows resulting error rates on the "clean" eval set for each model trained with 10, 120, and 243 minutes of data. There is a dramatic difference between MMS-1B-L1107 and the other two models for the smaller amounts of data (10 minutes). This model does impressively well with just 10 minutes of data, with a WER of 53% and CER of 8%. This margin shrinks as more data is added, though the MMS-1B-L1107 consistently performs better. This makes it a good choice for low-resource languages with very small amounts of labeled data. See Table 4.2 in Appendix A for all results.

4.4 Language Model Decoding

Language model (LM) decoding improves all of the models by several percentage points. The performance improvement is less for the better models, but even the best model (MMS-1B-l1107) improves slightly in WER. Figure 4.2 shows the decrease in error rate for each model with the LM.

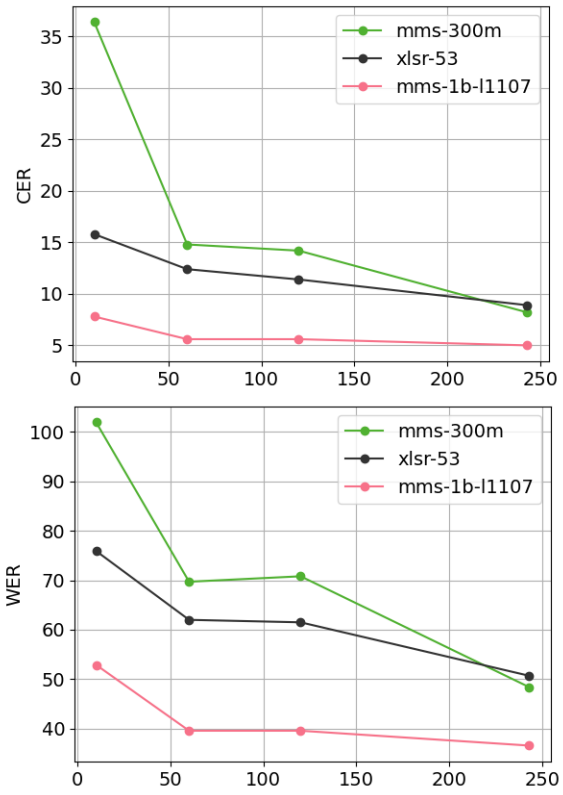

Figure 4.1: **Error rates by data size.** Word error rate and character error rate for each model given the length of training data in minutes without LM decoding.

However, in the best model, the CER is not improved.

4.5 Chapter Summary

The MMS-1B-l1107 model trained on the full dataset (243 minutes) performed best overall. MMS-300M and XLSR-53 perform less well on smaller amounts of data, but improve with more data. The MMS-1B-l1107 training is faster and more computationally efficient due to only training the adapter instead of the entire model. Decoding with a language model improves error rates but also degrades some transcriptions.

Model	WER				CER			
	10	60	120	243	10	60	120	243
XLSR-53	76	62	62	51	16	12	11	9
XLSR-53 + LM	50	45	40	36	12	11	10	7
MMS-300M	102	70	71	48	36	15	14	8
MMS-300M + LM	74	45	43	33	26	11	10	6
MMS-1B-L1107	53	40	40	37	8	6	6	**5**
MMS-1B-L1107 + LM	**36**	**34**	**34**	**31**	**7**	**5**	**5**	**5**

Table 4.3: **Results by data size.** Error rate percentages for different training data amounts in minutes, with and without language model (LM) decoding, on the eval "clean" set.

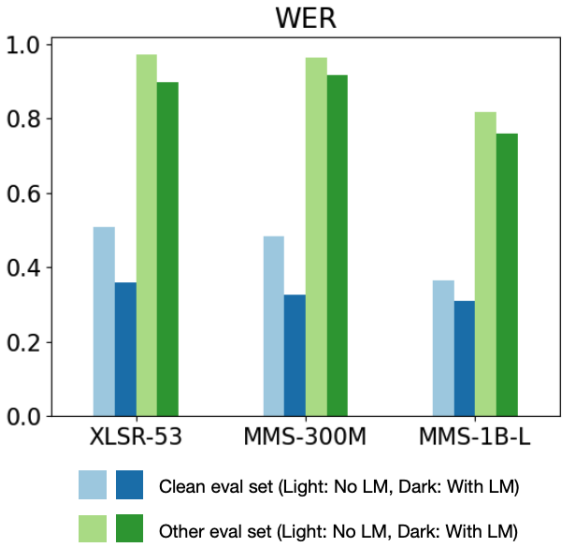

Figure 4.2: **Reduction in WER with LM.** Evaluation sets word error rates when decoding with and without a trigram language model, for each model trained on 243 minutes of audio data. MMS-1B-L is the MMS-1B-L1107 model.

Chapter 5

DISCUSSION

This chapter discusses the results in the context of the three research questions defined in section 3.4.1.

5.1 RQ1: How does pre-training affect accuracy?

The goal of evaluating different models is to see if the languages included in pre-training, as well as amount of parameters and type of adapter training, have a significant effect on performance.

5.1.1 MMS vs XLSR

These experiments show that MMS, which is pre-trained on 1,406 languages, performs better than XLSR-53 when given the full dataset. However, XLSR-53 outperforms the similarly-sized MMS-300M on smaller amounts of data. Other papers have shown that XLSR-53 outperforms MMS in some situations, such as Uralic languages and Arabic, both of which have tens of thousands of hours of training data available (Mihajlik et al., 2023; Younis and Mohammad, 2023). Mvskoke on the other hand only has a few hours of data, possibly making MMS the better candidate. This is consistent with the findings of the original authors of MMS, that higher-resource languages show some degradation in MMS compared with previous models that cover fewer languages, but that most extremely low-resource languages benefit from the large amount of languages represented in MMS (Pratap et al., 2023).

5.1.2 Adapters and Parameters

The MMS-1B-l1107 outperforms other models considerably. However, this model is different from others in three ways: it has more parameters (1 billion), it features an adapter, and it is fine-tuned for the task of ASR specifically. Because each these aspects did not have a control in the experiments, it is hard to make any conclusions about the effectiveness of each feature. It can

be stated, however, that this model is both computationally efficient and gives state-of-the-art performance. In the case of Mvskoke, it is a good choice for field applications.

5.2 RQ2: How much data before results begin to converge?

It is clear that models improve most dramatically between 10 minutes and 60 minutes of training data. Performance is only improved slightly between 1 hour and 2 hours, and in fact the 2-hour version of MMS-300M is 1 percentage point *worse* than 1-hour version. At some point between 2 hours and 4 hours, there is another slight performance increase - more so in the lower two performing models. For the best model, there is only an improvement of 3% WER and 1% CER.

Due to the arduous effort of manually transcribing recordings, the goal of experimenting with varying amounts of data is to see if there is a point at which gathering more data is not worth chasing after a few percentage points. The results in this work suggest there may be a "sweet spot" for low-resource languages where near-maximum performance can be achieved with the least amount of data. Furthermore, this sweet spot may be earlier for better models such as the MMS-1B-l1107. For the case of Mvskoke, increasing data beyond 4 hours may not be significantly helpful.

5.3 RQ3: Does including a language model improve accuracy?

LM does improve accuracy for all models, at least slightly. The LM is more helpful in the lower data situations (10 minutes), but it becomes less significant as more data is added and model accuracy is improved. For the CER of the best model, the effect is negligible, although WER is still improved. However, one important factor to consider is the application in which transcriptions will be used. This section discusses specific example outputs and how the LM errors could influence decisions about whether or not to use LM decoding in field applications.

5.3.1 Good transcriptions

Table 5.1 lists some example model transcriptions. Unsurprisingly, the models reach the highest accuracy on read speech from speakers present in the training data. Many predictions on the known speaker are perfect transcriptions, such as shown in example 1. For noisier data, the language model helps normalize some predictions, however in some cases it is detrimental to the transcription.

5.3.2 Spelling changes

The LM makes occasional minor spelling changes that are acceptable alternative spellings, such as in example 2, where the LM substitutes a common alternative spelling for the same word. This causes an increase in error rate, but it is still a good transcription, which calls for consideration where there may be better measures than WER and CER for a flexible orthography.

5.3.3 When the LM Degrades Transcriptions

The model also misses some word boundaries, and the LM is unable to correct many of these mistakes, such as in example 3. Furthermore, the language model occasionally breaks apart long out-of-vocabulary words into more common words, degrading the transcription, such as in example 4. Here the output without LM decoding makes a closer transcription. “vcvkvhoyvte hvmkat” (”one of the ones who had followed”) is transcribed as “vcakkvhoyvte hvmkat” without an LM, which is phonetically similar, but is changed to “vcakv oketv hvmkat” by the LM, which is nonsensical. So although the WER goes down overall for the whole evaluation, some information may be lost. This may be dis-preferred for some applications such as spoken term detection, where adherence to the source signal is crucial for indexing audio recordings (Le Ferrand et al., 2021). The final example, example 5, shows that LMs can improve the transcription on familiar words.

5.4 Chapter Summary

This chapter addresses each research question stated in Chapter 3. It concludes that the best overall model for Mvskoke is MMS-1B-l1107. It is shown that the amount of data used for training becomes less significant as model accuracy increases. Finally, it discusses individual transcription outputs and how the LM can both help and harm ASR output.

1.	**Perfect transcription**		
Eval (clean)	Known female speaker, read		
Translation	*"A log was lying in the water"*		
Reference	etot uewvn akwakkēt	CER	WER
No LM	etot uewvn akwakkēt	0	0
With LM	etot uewvn akwakkēt	0	0
2.	**Minor spelling changes**		
Eval (clean)	Known female speaker, read		
Translation	*"We don't want you. Go back," he was told*		
Reference	ceyacēkot os yefulkvs kihocen	CER	WER
No LM	ceyacēkot os yefulkvs kihocen	0	0
With LM	ceyac**e**kot os yefulkvs kihocen	3	25
3.	**Missed word boundary**		
Eval (other)	Held-out female speaker		
Translation	*"'Wring its neck,' he told me."*		
Reference	nokfiyvs kihcen cvkihcen	CER	WER
No LM	**nokfiyvskihcen** cvkihcen	12	67
With LM	**nokfiyvskihcen** cvkihcen	12	67
4.	**LM degrades transcription**		
Eval (other)	New Testament, male, read		
Translation	*"one of the ones who had followed"*		
Reference	vcvkvhoyvte hvmkat	CER	WER
No LM	vc**ak**kvhoyvte hvmkat	8	5
With LM	**vcakv oketv** hvmkat	32	100
5.	**LM improves transcription**		
Eval (other)	Held-out male, spontaneous interview		
Translation	*"November"*		
Reference	ohrolopē eholē	CER	WER
No LM	orrolop**e v** eho**f**lē	38	150
With LM	ohrolopē eholē	0	0

Table 5.1: **Example output** of ASR predictions from MMS-1B-l1107 trained on the full dataset.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Summary

This study shows that fine-tuning multilingual transformer models is an effective method for training ASR systems in low-resource language contexts. Fine-tuning the adapter for a 1 billion parameter model, MMS-1B-l1107, yields better results when compared to training entire models such as XSLR-53 and MMS-300M. However, the performance of such systems depends highly on the recording quality and type of speech. Although language modeling improves overall accuracy measures such as WER and CER, it can also degrade the output in some cases.

6.2 Future Work

There are several opportunities for future improvements and extensions of this study. One area could be to enhance the quality and diversity of the training data. For practical applications, an ASR model would likely need to process spontaneous speech. However, the models in this experiment struggled with conversational and noisy recordings. To address this, incorporating more interview recordings into the training data could be beneficial. Additionally, increasing the diversity of speakers by gathering data from a broader range of individuals might improve accuracy, although this may necessitate the collection of new recordings.

Another area for development is the creation of a sub-word or character-level language model. Such a model would be valuable for preserving the fidelity of the speech signal, which is helpful for certain downstream applications such as query-by-example. A character-level model may be able to better handle the agglutinative nature of the Mvskoke language. Moreover, exploring more advanced language models, such as those based on transformers, could yield better results than the n-gram models used in this study.

Finally, another future direction could be to incorporate the ASR model into a keyword-spotting or sparse transcription system. The high error rates for noisy recordings in this study mean that

manual transcription may still be faster than correcting ASR output. Sparse transcription can be helpful in situations where high ASR error rates lead to low-quality transcriptions (Bird, 2021). Transcribing only high-confidence words can be useful for indexing recordings and providing an overview of recorded content that can then be used for knowledge gathering.

6.3 Limitations

Due to the computational effort, each model was only trained once for each data amount (10, 60, 120, and 243 minutes). The datasets were shuffled randomly at runtime when selecting the splits, for example one 10 minute set is slightly different than another 10 minute set. This creates some variability in the results, and is not as robust as training the models multiple times and taken an average of performance.

This study also does not include the MMS-1B, the adapter-less version of the MMS-1B-l1107, because of the computational requirements of training such a large model. Because of this, conclusions cannot be made about the performance of an adapter model compared to a model with an equal amount of parameters. This study does not seek to fully evaluate adapter architecture, rather only to say that it is an effective method for this setting.

Finally, the transformer architecture was not evaluated alongside other architectures. In low-resource settings, model architecture can affect performance significantly, and no single architecture is best for every language (Jimerson et al., 2023). This study only evaluates the models stated here and their performance on the Mvskoke language.

BIBLIOGRAPHY

Jonathan D Amith, Jiatong Shi, and Rey Castillo Garcia. 2021. End-to-end automatic speech recognition: Its impact on the workflow for documenting Yoloxóchitl Mixtec. In *First Workshop on NLP for Indigenous Languages of the Americas. 11 June 2021. https://www. aclweb. org/anthology/2021. americasnlp-1.8. pdf.*

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 929–947, New York, NY, USA. Association for Computing Machinery.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. arXiv:1907.05019.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: a framework for self-supervised learning of speech representations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Steven Bird. 2021. Sparse Transcription. *Computational Linguistics*, 46(4):713–744.

Jonathan Boigne. 2021. An illustrated tour of wav2vec2.0. *jonathanbgn.com*.

Jacqueline Brixey and David Traum. 2022. Towards an automatic speech recognizer for the Choctaw language. *1st Workshop on Speech for Social Good (S4SG)*, 24-25.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition. arXiv:2006.13979.

George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Pranay Dighe, Afsaneh Asaei, and Hervé Bourlard. 2020. On quantifying the quality of acoustic models in hybrid DNN-HMM ASR. *Speech Communication*, 119:24–35.

Melanie Frye. 2020. Improving Mvskoke (creek) language learning outcomes: A frequency-base approach. *Thesis, University of Oklahoma*.

Earnest Gouge, Edited, Translated by Jack B. Martin, and Juanita McGirt. 2004. *Totkv Mocvse / New Fire: Creek Folktales*. Norman: University of Oklahoma Press.

A. Graves and N. Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. *31st International Conference on Machine Learning, ICML 2014*, 5:1764–1772.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. In *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, volume 2006, pages 369–376.

Mary R. Haas, James H. Hill, Jack B. Martin, Margaret McKane Mauldin, and Juanita McGirt. 2015. *Creek (Muskogee) Texts*. University of California Publications.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Robert Jimerson, Zoey Liu, and Emily Prud'hommeaux. 2023. An (unhelpful) guide to selecting the best ASR architecture for your under-resourced language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1008–1016, Toronto, Canada. Association for Computational Linguistics.

Biing-Hwang Juang and Lawrence R Rabiner. 2005. Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1(67):1.

Daniel Jurafsky and James H. Martin. 2020. *Speech and Language Processing. 3rd Edition Draft.*

Eric Le Ferrand, Steven Bird, and Laurent Besacier. 2021. Phone based keyword spotting for transcribing very low resource languages. In *Proceedings of the The 19th Annual Workshop of the Australasian Language Technology Association*, volume 19 of *Proceedings of the Australasian Language Technology Workshop*, pages 79–86. Australasian Language Technology Association. Publisher Copyright: © ALTA 2021. All rights reserved.; 19th Workshop of the Australasian Language Technology Association, ALTA 2021 ; Conference date: 08-12-2021 Through 10-12-2021.

Jinyu Li. 2022. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1):–. doi:10.1561/116.0000005.

Jeremy Lopez, Georg Kucsko, Pat O'Neill, and Patrick von Platen. 2024. Pyctcdecode. `https://github.com/kensho-technologies/pyctcdecode`.

Jack B. Martin. 2011. *A Grammar of Creek (Muskogee)*. University of Nebraska Press.

Jack B. Martin and Margaret McKane Mauldin. 2000. *A Dictionary of Creek/Muskogee*. University of Nebraska Press.

M Meelen, A O'Neill, and R Coto-Solano. 2024. End-to-end speech recognition for endangered languages of Nepal. *Proceedings of the Seventh Workshop on the Use of Computational Methods in the Study of Endangered Languages*.

Péter Mihajlik, Máté Kádár, Gergely Dobsinszki, Yan Meng, Meng Kedalai, Julian Linke, Tibor Fegyó, and Katalin Mady. 2023. What kind of multi- or cross-lingual pre-training is the most effective for a spontaneous, less-resourced ASR task? In *2nd Annual Meeting of the ELRA/ISCA Special Interest Group on Under-resourced Languages (SIGUL 2023)*.

Mamyrbayev Orken, Keylan Alimhan, Bagashar Zhumazhanov, Tolganay Turdalykyzy, and Farida Gusmanova. 2020. End-to-end speech recognition in agglutinative languages. *Intelligent Information and Database Systems*, pages 391–401.

Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, Alexei Baevski, Yossi Adi, Xiaohui Zhang, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. 2023. Scaling speech technology to 1,000+ languages. arXiv:2305.13516.

L. Rabiner and B. Juang. 1986. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1):4–16.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. *CoRR*, abs/1904.05862.

Jiatong Shi, Jonathan D. Amith, Rey Castillo García, Esteban Guadalupe Sierra, Kevin Duh, and Shinji Watanabe. 2021. Leveraging end-to-end ASR for endangered language documentation: An empirical study on Yolóxochitl Mixtec. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1134–1145, Online. Association for Computational Linguistics.

Bao Thai, Robert Jimerson, Raymond Ptucha, and Emily Prud'hommeaux. 2020. Fully convolutional ASR for less-resourced endangered languages. In *Proceedings of the 1st Joint Workshop*

on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pages 126–130, Marseille, France. European Language Resources association.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Hiba Adreese Younis and Yusra Faisal Mohammad. 2023. Arabic speech recognition based on self supervised learning. In *2023 16th International Conference on Developments in eSystems Engineering (DeSE)*, pages 528–533.

Shiyue Zhang, Ben Frey, and Mohit Bansal. 2022. How can NLP help revitalize endangered languages? a case study and roadmap for the Cherokee language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1529–1541, Dublin, Ireland. Association for Computational Linguistics.

Appendix A

DATA TABLES

Model	length (min)	WER	CER	data instances	total steps	total epochs	max epochs	time (min)	best epoch
mms-1b-l1107	5	0.63	0.20	105	154	11	30	35	11
mms-1b-l1107	10	0.48	0.13	210	324	12	30	66	12
mms-300m	10	0.94	0.34	210	702	26	30	23	26
xlsr-53	10	0.67	0.19	210	810	30	30	85	30
mms-1b-l1107	60	0.31	0.10	1264	1896	12	30	340	12
mms-1b-all	60	0.39	0.11	1264	1896	12	30	335	12
mms-300m	60	0.57	0.16	1264	2054	13	30	296	10
xlsr-53	60	0.55	0.15	1264	2054	13	30	173	13
mms-1b-l1107	120	0.29	0.09	2528	3160	10	30	35	10
mms-300m	120	0.58	0.16	2528	4108	13	30	31	13
xlsr-53	120	0.52	0.14	2528	3476	11	30	27	11
mms-1b-l1107	243	0.28	0.08	5116	7040	11	30	343	11
mms-300m	243	0.36	0.10	5116	14720	23	30	1,203	23
xlsr-53	243	0.41	0.12	5116	8320	13	30	57	13

Table A.1: **Training information for each train run.** "Length" is the number of minutes in the training dataset. Each run stopped after 3 epochs where the CER did not improve by more than 0.003 (indicated by "total epochs"), and the best model was saved ("best epoch").

Model	length (min)	WER no LM clean	CER no LM clean	WER LM clean	CER LM clean	WER no LM other	CER no LM other	WER LM other	CER LM other
mms-1b-l1107	5	0.64	0.11	0.61	0.13	0.95	0.40	0.84	0.37
mms-1b-l1107	10	0.53	0.08	0.36	0.07	0.92	0.37	0.76	0.33
mms-300m	10	1.02	0.36	0.74	0.26	1.02	0.67	1.02	0.64
xlsr-53	10	0.76	0.16	0.50	0.12	1.01	0.52	0.96	0.52
mms-1b-l1107	60	0.40	0.05	0.31	0.05	0.82	0.34	0.75	0.32
mms-300m	60	0.70	0.15	0.45	0.11	0.99	0.51	0.94	0.52
xlsr-53	60	0.62	0.12	0.45	0.11	0.98	0.48	0.91	0.48
mms-1b-l1107	120	0.40	0.06	0.34	0.05	0.83	0.34	0.77	0.32
mms-300m	120	0.71	0.14	0.43	0.10	1.00	0.53	0.95	0.52
xlsr-53	120	0.62	0.11	0.40	0.10	0.98	0.49	0.91	0.50
mms-1b-l1107	243	0.37	0.05	0.31	0.05	0.82	0.33	0.76	0.32
mms-300m	243	0.48	0.08	0.33	0.06	0.96	0.48	0.92	0.48
xlsr-53	243	0.51	0.09	0.36	0.07	0.97	0.47	0.90	0.47

Table A.2: **Eval Results.** This shows the WER and CER results with and without language model (LM) decoding, on “clean” and “other” eval sets.