

SPEAKNOW: A SPEECH-TO-TEXT SYSTEM FOR THE HILIGAYNON LANGUAGE USING KALDI TOOLKIT

A Special Problem

Presented to

the Faculty of the Division of Physical Sciences and Mathematics

College of Arts and Sciences

University of the Philippines Visayas

Miag-ao, Iloilo

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science by

GONZALES, Benjie Jr.

PANIZALES, John Patrick

PIORQUE, Lester

Francis D. DIMZON

Adviser

June 8, 2023

Approval Sheet

The Division of Physical Sciences and Mathematics, College of Arts and
Sciences, University of the Philippines Visayas

certifies that this is the approved version of the following special problem:

**SPEAKNOW: A SPEECH-TO-TEXT SYSTEM FOR THE
HILIGAYNON LANGUAGE USING KALDI TOOLKIT**

Approved by:

Name

Signature

Date

(Adviser)

(Co-Adviser)

(Reader)

(Division Chair)

Division of Physical Sciences and Mathematics

College of Arts and Sciences

University of the Philippines Visayas

Declaration

I/We, GONZALES, PANIZALES and PIORQUE, hereby certify that this Special Problem, including the pdf file, has been written by me/us and is the record of work carried out by me/us. Any significant borrowings have been properly acknowledged and referred.

Name

Signature

Date

(Student)

(Student)

(Student)

Dedication

We, the SpeakNow team, would like to dedicate this special problem to all the individuals who have supported us throughout this journey. To our families, whose unwavering love and encouragement have been our pillars of strength, we extend our heartfelt gratitude. To our project adviser, Sir Francis D. Dimzon, for his guidance, expertise, and invaluable insights, we are deeply grateful. To our friends and colleagues, for their camaraderie and support, thank you for being by our side. Finally, we dedicate this work to the countless individuals whose lives may be positively impacted by the knowledge gained from our research. May this paper contribute to the advancement of knowledge, inspire future explorations, and make a meaningful difference in the world.

Acknowledgment

We would like to express our heartfelt gratitude to our esteemed project adviser, Sir Francis D. Dimzon, for his invaluable guidance, expertise, and unwavering support throughout the duration of this special problem. His deep knowledge, insightful feedback, and mentorship have been pivotal in shaping the direction and quality of our research. We are truly grateful for his dedication and commitment to our academic growth.

Moreover, we sincerely appreciate the participants of this project, the eighteen Hiligaynon speakers, whose involvement and cooperation were integral to its success. Their dedication of time and effort made significant contributions to the results of our special problem. We are deeply grateful for their valuable contributions and trust in our research.

Furthermore, we would like to acknowledge the invaluable assistance and support provided by our colleagues and friends who have stood by us throughout this undertaking. Their encouragement, discussions, and exchange of ideas have played a crucial role in shaping and refining our analysis.

Lastly, we would like to thank our families for their unwavering support and understanding. Their encouragement, patience, and belief in our abilities have been a constant source of motivation during challenging times.

Together, the collective contributions and support of our project adviser, participants, colleagues, and families have made this research endeavor possible. We are deeply grateful for their involvement and contributions, as they have played a significant role in the successful completion of this special problem.

Abstract

This research paper presents a speech-to-text (STT) system for the Hiligaynon language using an Automatic Speech Recognition (ASR) model trained with Kaldi. The ASR model was trained on a corpus of approximately 3,500 Hiligaynon words. From this corpus, a subset of 1,000 words was randomly selected and recorded by 18 speakers (9 males and 9 females). The models trained included Monophone, Delta-based Triphone, LDA+MLLT, LDA+MLLT+SAT, and DNN, employing a six-fold cross-validation scheme. The DNN model yielded the lowest average Word Error Rate (WER) score and was chosen for the STT system development. In addition, the Whisper toolkit was utilized to transcribe the same audio dataset. Comparing different pre-trained model sizes, it was found that performance was proportional to the model size. The small model achieved a WER of 25.50%. This research contributes to the development of an effective STT system for Hiligaynon, emphasizing the significance of acoustic model selection and the influence of model size on transcription performance.

Keywords: Automatic Speech Recognition (ASR), Hiligaynon language, acoustic modeling, neural networks, Whisper, OpenAI, Word Error Rate (WER), Speech-to-Text (STT).

Contents

1	Introduction	1
1.1	Overview of the Current State of Technology	1
1.2	Problem Statement	3
1.3	Research Objectives	3
1.3.1	General Objective	3
1.3.2	Specific Objectives	4
1.4	Scope and Limitations of the Research	4
1.5	Significance of the Research	5
2	Review of Related Literature	6
2.1	Automatic Speech Recognition	6
2.1.1	Lexicon Model	7

2.1.2	Acoustic Model	8
2.1.3	Language Model	9
2.1.4	Decoding	10
2.2	The Language Dilemma	10
2.3	Hiligaynon Speech Recognition	11
2.4	The Hiligaynon Language	12
2.5	Kaldi ASR Toolkit	14
2.6	Acoustic Models in Kaldi	14
2.6.1	Monophone Model	15
2.6.2	Triphone Model	15
2.6.3	LDA + MLLT Model	15
2.6.4	LDA + MLLT + SAT Model	16
2.6.5	DNN Model	16
2.6.6	Whisper by OpenAI	17
3	Research Methodology	19
3.1	Research Activities	19
3.1.1	Data Gathering and Preprocessing	19

<i>CONTENTS</i>	ix
3.1.2 Preprocessing	21
3.1.3 Training	21
3.1.4 Evaluation	32
3.1.5 System Development	33
4 Results and Discussions/Analyses	35
4.1 Results of Monophone model	36
4.2 Results of Triphone model	38
4.3 Result of LDA + MLLT model	40
4.4 Result of LDA + MLLT + SAT model	42
4.5 Result of DNN model	44
4.6 Summary of results	46
4.7 Results on the performance of Whisper	47
5 Conclusion	48
5.1 Recommendations	49
6 References	50
A Appendix	55

A.1 Code snippets	55
-----------------------------	----

List of Figures

- 3.1 Screenshot of the system when called without an argument. . . . 33
- 3.2 Screenshot of the system with an audio file's file name as an argument. 34

List of Tables

2.1	Table of the 25 significant sounds in Hiligaynon (Wolfenden, 2019)	13
4.1	Results of Monophone model	37
4.2	Results of Triphone model	39
4.3	Results of LDA + MLLT model	41
4.4	Results of LDA + MLLT + SAT model	43
4.5	Results of DNN model	45
4.6	Summary of WER means of acoustic models in each fold	46
4.7	WER and SER results when transcribing in Hiligaynon using different sizes of pre-trained models trained with Whisper	47

Chapter 1

Introduction

1.1 Overview of the Current State of Technology

The technology known as Automatic Speech Recognition (ASR) enables computers to recognize spoken language and convert it into text. It is a fast developing field that might completely alter how we engage with technology. Since Hiligaynon is one of the most extensively used languages in the Philippines and is spoken in the Western Visayas area, it is a suitable topic for an ASR special problem. With over 7 million speakers, it is the third most spoken language in the Philippines (PSA, 2023).

The Visayan language family includes Hiligaynon, which is spoken in the Philippines' central and southern regions. It is well-known for its extensive vocabulary and intricate verbal structure, which incorporates numerous Spanish and English words. The language is used in education, the media, and government, among

other places. However, due to limited availability of ASR systems for Hiligaynon, the alternatives for speech-to-text transcription and other ASR-related applications are currently constrained. Furthermore, Aquino et al. (2019) noted the lack of access to phonetically-annotated audio data for Philippine languages, which is crucial for training ASR systems. Currently, much of the focus is centered towards the Tagalog language.

Some studies have given attention to Hiligaynon within the context of speech recognition. Billones & Dadios (2014) developed a breast self-examination (BSE) multimedia training system incorporating Hiligaynon speech recognition limited to a five-word vocabulary for motion control commands. The study used Mel frequency cepstrum coefficients for feature extraction and genetic algorithm for pattern recognition coupled with an adaptive database yielding an accuracy of 97.50%. Gavieta et al. (2022) also conducted a study developing a speech recognition system for Hiligaynon using words commonly used in households and for directions. Their system trained on a total of 240 words using different acoustic models namely: monophone, delta-based triphone, delta + delta-delta triphone, LDA + MLLT, SAT, and DNN. Results of their study show that the DNN model performed best after yielding a mean word error rate (WER) of 0.35% following a 5-fold cross validation scheme. Additionally, Aquino et al. (2019) implemented three methods for the low-resource automatic phonetic transcription of Tagalog, Cebuano and Hiligaynon. The three methods were grapheme-to-phoneme conversion (G2P), automatic speech recognition, and adaptive alignment. Their results show that the G2P method is a good alternative to manual phonetic transcription, whereas ASR models prove to be beneficial in specific moments where only recorded audio of speech utterances is provided.

There are numerous advantages to developing an ASR system for Hiligaynon, including enhancing accessibility for Hiligaynon speakers and encouraging the language's preservation and development. Additionally, it would aid in the development of computational linguistics and ASR technology. The creation of an ASR system for Hiligaynon is, in general, a significant and pressing special issue with the potential to have a significant impact.

1.2 Problem Statement

Despite the advancements in Automatic Speech Recognition (ASR) technology for widely spoken languages such as English and Filipino, there is a significant gap in the availability of ASR models for regional languages in the Philippines. With the country being home to a diverse range of 187 languages, including Hiligaynon, there is a need to address the lack of ASR models specifically tailored to these local languages. This special problem aims to fill this gap by developing an ASR model specifically for Hiligaynon, which would serve as a foundational milestone towards advancing speech-to-text innovation for regional languages in the Philippines.

1.3 Research Objectives

1.3.1 General Objective

The aim of this project is to develop an Automatic Speech Recognition System for the Hiligaynon language using Kaldi.

1.3.2 Specific Objectives

Specifically, the project targets to:

1. Train an ASR model with Hiligaynon words using Kaldi.
2. Train different acoustic models and compare the results.
3. Evaluate the performance of the models in terms of the Word Error Rate (WER) via cross-validation.
4. Evaluate the current performance of OpenAI's Whisper when transcribing in the Hiligaynon language.
5. Construct a speech-to-text (STT) system that employs the best performing model trained using Kaldi.

1.4 Scope and Limitations of the Research

The system is specific to the Hiligaynon language. The words used in the audio data are limited to two-to-three-syllable Hiligaynon words taken from a personally compiled collection of Hiligaynon words contributed by Francis D. Dimzon. The system is also limited to the features offered by Kaldi - an open source speech recognition toolkit used for training the models.

1.5 Significance of the Research

ASR systems offer a wide-range of benefits such as improved accessibility through the use of spoken language data. However for under resourced languages such as Hiligaynon, data used for training is limited. Large-scale ASR initiatives which aim to support languages world-wide are currently limited to Tagalog/Filipino, in the case of Philippine languages such as Mozilla’s DeepSpeech and Open AI’s whisper. A common constraint for Hiligaynon ASR studies is the fact that these systems have been trained on small text and/or speech corpora such as in the case of Billones & Dadios (2014) and Aquino et al. (2019). As noted by Aquino et al. (2019), access to phonetically-annotated audio data for Philippine languages apart from Tagalog is limited. Considering these circumstances, the development for an effective ASR system for Hiligaynon is yet to come (Gavieta et al., 2022). Focusing attention to developing ASR systems that cater to under resourced languages can help in extending these benefits to speakers of such languages. This project aims to take another step towards developing an automatic speech recognition (ASR) system for the Hiligaynon Language, adding more words to the vocabulary of the system and incorporating more speakers for more variability.

Chapter 2

Review of Related Literature

2.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) systems have a rich history dating back to the 1950s when Bell Labs developed "Audrey," an early digit recognizer (Hannun, n.d.). At that time, Audrey was limited to transcribing spoken numbers. However, researchers made significant advancements in the following decade, enabling Audrey to transcribe simple spoken words like "hello." Traditionally, ASR relied on classical Machine Learning techniques such as Hidden Markov Models (Huang et al., 2014). While these models were once the industry standard, their accuracy had plateaued in recent years.

The advent of advanced Deep Learning technology, which has also driven progress in fields like self-driving cars, brought about a shift in ASR approaches (Summa Linguae, 2021). These newer methods have substantially improved accuracy com-

pared to classical models. Additionally, access to ASR technology has become more accessible. Previously, customers had to navigate lengthy and costly enterprise software contracts to license ASR technology. However, in the present day, State-of-the-Art ASR technology is readily available to developers, startups, and even Fortune 500 companies through simple APIs like AssemblyAI's Speech-to-Text API (Summa Linguae, 2021).

The process of automatic speech recognition involves a sequence of combinations in predicting transcriptions, including the lexicon model, acoustic model, language model, and decoding (Huang et al., 2014). Overall, the evolution of ASR systems from early digit recognition to the incorporation of Deep Learning techniques has significantly improved accuracy and accessibility in recent years.

2.1.1 Lexicon Model

A lexicon model, also known as a pronunciation dictionary, is a fundamental component of Automatic Speech Recognition (ASR) systems. It plays a crucial role in mapping words or units of speech to their corresponding pronunciations (Hatfield, 2023). For example, in English, the word "cat" might be represented with the phonemic transcription /kæt/. The lexicon model provides valuable information about pronunciation variants of words, including phonetic transcriptions or pronunciation rules. It serves as a reference for the acoustic and language models, enabling accurate recognition and decoding of spoken words.

The lexicon model's primary function is to convert spoken words into text, making it an essential component of ASR systems (Fasold & Connor-Linton, 2014). By

utilizing the lexicon model, ASR systems can match the audio input to the appropriate word and generate the corresponding textual output. This process enables the transcription of spoken words into written form, facilitating various applications such as transcription services, voice assistants, and language processing (Fasold & Connor-Linton, 2014).

2.1.2 Acoustic Model

The acoustic model is a key component of Automatic Speech Recognition (ASR) systems, responsible for converting speech signals into phonetic representations. It learns to associate audio features with corresponding phonetic units, enabling the recognition of spoken words (Klatt, 1979). Techniques such as Hidden Markov Models, Gaussian Mixture Models, and Deep Neural Networks are used to train the acoustic model and improve transcription accuracy. The accurate representation of acoustic properties is crucial for achieving high transcription accuracy and overall system performance (Pisoni & Luce, 1987).

Recent advancements in acoustic modeling, particularly the application of Deep Neural Networks, have contributed to significant progress in ASR technology (Veselý et al., 2013). These advancements have resulted in improved accuracy and robustness in converting spoken language into written text. The acoustic model's ability to capture and map acoustic characteristics to phonetic units plays a vital role in achieving these improvements (Veselý et al., 2013). Ongoing research and development aim to further enhance acoustic models, thus advancing the capabilities of ASR technology in various applications.

The acoustic model is a critical component in the pipeline for converting spoken words into text. Its accurate representation of acoustic properties is essential for achieving high transcription accuracy and system performance. Advancements in acoustic modeling techniques, particularly the use of Deep Neural Networks, have greatly contributed to the progress in ASR technology, resulting in improved accuracy and robustness (Veselý et al., 2013). Ongoing research and development continue to enhance acoustic models, further advancing ASR capabilities.

2.1.3 Language Model

A language model incorporates linguistic knowledge to estimate the likelihood of word sequences in a given language (Klatt, 1979). It helps distinguish between potential word sequences generated by the acoustic (Gales et al., 2008). For example, given the sequence of phonetic units /k/ /æ/ /t/, the language model calculates the probability of different word sequences that could correspond to those phonetic units (Gales et al., 2008). The language model is typically an N-gram model in which the probability of each word is conditioned only on its N-1 predecessors (Gales et al., 2008). The N-gram parameters can be estimated from large text corpora using techniques such as Maximum Likelihood Estimation or Bayesian Inference (Gales et al., 2008). Language models can also be based on neural networks, such as Recurrent Neural Networks or Transformers. These models can capture long-range dependencies between words and can be trained on large text corpora using techniques such as Maximum Likelihood Estimation or Contrastive Learning. The language model is a crucial component of ASR systems and is used to improve the accuracy of the transcription by selecting the

most likely word sequence given the input audio features.

2.1.4 Decoding

Decoding is the process of determining the most likely word sequence or transcription given the outputs of the acoustic model and language model. It is a crucial step in speech recognition systems. During decoding, the system combines the information from the acoustic model, which produces a sequence of phonetic units, and the language model, which estimates the likelihood of word sequences (Zenkel et al., 2017).

2.2 The Language Dilemma

While there is a wealth of English-oriented ASR systems, other languages, especially lesser known languages tend to struggle in these situations. For instance, most leading tech companies tend to focus on developing speech recognition technologies for the English language such as DeepSpeech, Mozilla’s speech to text engine and OpenAI’s Whisper. In the context of Philippine languages, research in speech processing technologies for the Filipino language is not unheard of. As an example, Dimzon & Pascual (2020) was able to develop an “Automatic Phoneme Recognizer for Children’s Filipino Read Speech” using a 5-active state HMM model. This led to an accuracy of 57.47%. Increasing Gaussian mixtures from 1 to 6 have also been found to increase accuracy by 10% and vocal track length normalization has also been seen to improve accuracy by 2%. However, developing efficient ASR systems for Filipino have yet to be seen Dimzon & Pascual

(2020).

2.3 Hiligaynon Speech Recognition

Billones & Dadios (2014) conducted a study, where they created a 5-word vocabulary speech recognition system for Hiligaynon terms used as motion commands implemented for a breast self-examination (BSE) multimedia training system. The study aimed at raising awareness about breast cancer among the local female population of Western Visayas. Their focus was on developing a system that utilized Hiligaynon speech recognition with a limited vocabulary of five words. The researchers selected five commonly used Hiligaynon words ("idalom," "ibabaw," "wala," "tuo," and "patiyog") as representative chromosomes for the system. They collected a total of 200 audio samples by recording 40 samples for each word. The system employed Mel frequency cepstrum coefficients (MFCC) for feature extraction and genetic algorithms for pattern recognition. Additionally, an adaptive database was integrated into the system to enhance training and classification accuracy for the Hiligaynon words. Through the combination of these models and the adaptive database, the system achieved an impressive accuracy rate of 97.50% in recognizing the distinct Hiligaynon words.

A special problem by Gavieta et al. (2022) developed a speech recognition system for Hiligaynon. Their system was built using Kaldi and was trained using 240 Hiligaynon words commonly used in Households, as well as words used to indicate direction. Audio samples were produced from 10 speakers, five being male and the other five being female. Their data was fed to several training models namely,

monophone, triphone(delta, delta + delta-delta), Linear Discriminate Analysis Maximum Likelihood Linear Transform (LDA+MLLT), Speaker Adapting Training (SAT), and a Deep Neural Network (DNN) model. Their results show that the DNN model performs the best, yielding a mean WER of 0.35%.

Additionally, Aquino et al. (2019) conducted a study where they implemented three techniques in the context of automated phonetic transcriptions. Their system consists of three subsystems: grapheme-to-phoneme conversion (G2P), an automatic speech recognition (ASR) tool, and an adaptive substitution face. The study used data from the ISIP Project 6 Database of the UP Digital Signal Processing Laboratory. Their results show that the G2P approach is an effective alternative to manual phonetic transcription, even going beyond human accuracy. On the other hand, performance of ASR tools have been found to be satisfactory especially when the only available data is audio data.

2.4 The Hiligaynon Language

The Hiligaynon language has 25 significant sounds - 5 vowels and 20 consonants, and an accent (Wolfenden, 2019). According to Wolfenden's Hiligaynon Reference Grammar, vowels take up the nucleus slot of syllables while consonants take up the margin slots. Wolfenden (2019) defines the term "significant sounds" as the irreducible set of sounds which are key to understanding a language and hence must be symbolized in the orthography. Table 2.1 shows the 25 significant sounds plus accent of the Hiligaynon language as described by Wolfenden (2019).

Hiligaynon, also known as Ilonggo, is an Austronesian language spoken in the

Table 2.1: Table of the 25 significant sounds in Hiligaynon (Wolfenden, 2019)

Vowels	Consonants	
i	p	s
e	b	h
a	t	m
o	d	n
u	k	ng
	g	l
	c	r
	j	w
	f	y
	v	
Stress (not symbolized)		

Western Visayas region of the Philippines, particularly in the provinces of Iloilo, Guimaras, Negros Occidental, and Capiz. It is one of the major languages of the Philippines, spoken by millions of people as a first or second language.

Hiligaynon has a rich and varied vocabulary, with many loanwords from Spanish, English, and other languages. According to Hiligaynon Reference Grammar by Wolfenden 2019, Hiligaynon has a complex verb conjugation and tense system, with a range of tense markers including markers for past, present, and future tense, as well as markers for perfective and imperfective aspect. The book also notes that Hiligaynon has a number of mood markers, including markers for indicative, imperative, and subjunctive mood.

Additionally, Hiligaynon Reference Grammar by Wolfenden 2019 describes the phonemic alphabet of Hiligaynon as consisting of 28 letters: A, B, C, D, E, F, G, H, I, J, K, L, M, N, Ñ, O, P, Q, R, S, T, U, V, W, X, Y, and Z. The book notes that the letters C, F, J, Q, V, X, and Z are not used as frequently in Hiligaynon as in other Philippine languages, and that the letter Ñ is used to represent the

Spanish sound "ny."

2.5 Kaldi ASR Toolkit

Povey et al. (2011) described Kaldi as a modern toolkit for speech recognition. It is designed to be extensible and has one of the least restrictive licenses making it more accessible. Several studies have incorporated Kaldi into their implementations.

For instance, Upadhyaya et al. 2017 , developed a continuous Hindi speech recognition model using Kaldi, citing the toolkit for its ability to create high quality lattices and sufficient speed for real time recognition. It also said that the mentioned toolkit is actively maintained and accessible.

Additionally, not only is Kaldi able to support conventional models such as Gaussian mixture models (GMMs) but is also able to implement deep neural network based structures. For example, Kipyatkova & Karpov 2016 developed a "DNN-Based Acoustic Modeling for Russian Speech Recognition Using Kaldi." The paper mentioned using DNN implementations in Kaldi, ultimately choosing Dan's implementation because of its support for parallel training on multiple CPUs.

2.6 Acoustic Models in Kaldi

In the realm of speech recognition research, the Kaldi toolkit has gained prominence as an open-source framework offering a diverse array of acoustic models.

These models play a pivotal role in effectively capturing and representing speech data, thereby contributing to advancements in automatic speech recognition research. Notably, the following acoustic models are widely employed in Kaldi:

2.6.1 Monophone Model

The monophone model represents each phoneme individually, serving as an initial building block for speech recognition systems. For example, in the word 'cat,' the monophone model would have separate models for the /k/, /æ/, and /t/ phonemes.

2.6.2 Triphone Model

The triphone model enhances the representation of phonemes by incorporating context-dependent information. It captures variations in phoneme sounds based on neighboring phonemes. For instance, in the word 'cat,' the triphone model would consider the context of the previous and next phonemes, such as /k/ in the context of /æ/ and following /t/.

2.6.3 LDA + MLLT Model

The LDA + MLLT model combines Linear Discriminant Analysis (LDA) with Maximum Likelihood Linear Transformations (MLLT). LDA reduces the dimensionality of acoustic features and provides a discriminative representation. MLLT then refines this representation to account for speaker and channel variations.

This can be visualized as a transformation of the acoustic feature space, aligning similar speech patterns while differentiating between different speakers.

2.6.4 LDA + MLLT + SAT Model

The SAT (Speaker Adaptive Training) model extends the capabilities of the tri-phone model by incorporating speaker adaptation techniques. It accommodates individual speaker characteristics by adapting the model parameters to match the characteristics of the target speaker. This adaptation can be imagined as adjusting the model to fit the specific speaker’s speech patterns, reducing inter-speaker variability.

2.6.5 DNN Model

The DNN (Deep Neural Network) model employs deep neural networks, such as feed-forward or recurrent neural networks, to capture intricate representations of acoustic features. With multiple hidden layers, these models can learn complex patterns and relationships within the input data. The DNN model can be depicted as a series of interconnected nodes, simulating the human brain’s neural connections.

In Kaldi, various DNN setups have been implemented, the latest of which is the nnet3 setup. It is intended to support more general kinds of networks without any actual coding via a config-file-driven approach and is compatible with parallel training across GPUs on multiple machines (*Deep Neural Networks in Kaldi*, n.d.). According to Kaldi’s documentation, a nnet3 architecture consists of a list of the

name of the components along with a graph that specifies how these components are organized and work together (*Data types in the "nnet3" setup*, n.d.). Part of the graph's function is to allow recurrent neural nets, which is useful for speech recognition where different timelines can have varying effects on each other (*Data types in the "nnet3" setup*, n.d.).

2.6.6 Whisper by OpenAI

Several open-source speech recognition systems have been launched which offer models trained using hundreds-to-thousands-hours-long audio data, yielding good accuracy.

OpenAI's Whisper in particular, is trained using 680,000 hours of "multilingual and multitask" weakly supervised audio data from the web Radford et al. (2022). Including a diverse dataset makes Whisper a robust system that is sensitive to accents and background noise and is able to support transcription to multiple languages, as well as translation from other languages to English.

What makes Whisper different from other speech recognition systems is its deviation from "self-supervision and self-training techniques" typical of large-scale speech recognition systems. This makes Whisper efficient at dealing with large volumes of data.

Currently, the only Philippine language mentioned in Whisper's documentation is Tagalog, where the system performs with a 13.8% word error rate (WER) with the Fleurs dataset using the large-v2 model Radford et al.. It was also noted that Whisper's performance is proportional to the amount of data trained on a

particular language, leaving under-resourced languages at a disadvantage.

Chapter 3

Research Methodology

This chapter presents the methodology used to develop an automatic speech recognition system for the Hiligaynon language. This chapter is divided into the following major parts: Data Gathering and Preprocessing, Training, Evaluation and System Development.

3.1 Research Activities

3.1.1 Data Gathering and Preprocessing

Word Selection and Dictionary Creation

Initially, a set of one thousand words from the corpus were considered for recording. The selection was limited to two-to-three-syllable words only. From this set, a script was created for each speaker to read. Each script contained 500 words

which were randomly chosen from the previous 1000-word dictionary.

The same words were also included in the lexicon along with their phonetic transcriptions which were manually transcribed. Transcription was based off of the phonemes described in Wolfenden's Hiligaynon Reference Grammar.

Later, another set of words were added to the lexicon along with their phonetic transcriptions which were also manually transcribed. However, these words were not considered for recording. All in all, a total of about 3500 words were included in the project's lexicon.

Selection of Speakers and Equipment Used

A total of 18 speakers were gathered for audio recording. Nine of the speakers were male and the other nine were female. The speakers were either native speakers of Hiligaynon or are fluent in the said language. For recording, an external microphone was utilized alongside Audacity, a free and open-source digital audio editor.

Recording

Recording sessions were done in areas with minimal background noise. Each of the speakers uttered 25 words per recording. A total of 20 audio files were produced from each speaker. A naming system was developed for each audio file that identifies the gender of the speaker, the speaker number, and the audio file number in the following format: <gender>_<speaker-number>_<file-number >. The male gender was represented by a 1 and the number 2 was used to represent

the female gender.

3.1.2 Preprocessing

Audacity was used for preprocessing the audio files. Normalization was first applied, followed by compression. After which, noise reduction was applied. The audio files were then exported using the WAV format.

3.1.3 Training

Kaldi, an open-source speech recognition toolkit was used for training the ASR models.

A six-fold cross validation scheme was followed for training. Each fold contains data from three speakers. For each iteration, one fold was set for use as testing data while the remaining folds were set for use as training data. A total of 6 iterations were executed, with each fold acting as testing data exactly once.

For each iteration, a set of metadata files were created. These files contain information specific to the audio data within the context of each iteration. These files can be divided into two types:

Acoustic Data

These files contains information related to each audio file/data.

- **wav.scp:** This file contains information that maps a file id to the location

of the file. Each line in this file is written in the following format: `<file_id
><path_to_file >`.

- **text:** This file contains information about the file and the corresponding words uttered in that particular audio file. It is written in the following format `<utterance_id> word1 word2 word3...`
- **utt2spk:** This file contains information about the mapping of a specific file to it's corresponding speaker. It is written in the following format.
`<utterance_id><speaker_id>`.
- **spk2gender:** This file contains information indicating a specific speaker's gender. It is written in the following format `<speaker_id><gender>`.
- **corpus.txt:** This file contains all the words uttered in all of the recordings. Each line represents the words uttered in a specific file.

Language Data

These files contain information which were used for language modeling.

- **lexicon.txt:** This file contains information about all the words considered in the project's dictionary together with their phonemic transcriptions. Silence phones are also included in the lexicon.
- **nonsilence_phones:** This file contains all of the non-silent phones included in the project.
- **silence_phones.txt and optional_silence.txt:** This files contains the silence phones included in the project.

Training scripts were sourced from Kaldi's builtin scripts for different acoustic models namely: monophone, triphone, LDA+MLLT, LDA+MLLT+SAT and DNN. A single script is used calling the different training and decoding scripts for each type of model.

The script first starts with data preparation, where the script checks for the format of user-prepared files. Then, it proceeds to feature extraction. Mel-Frequency Cepstral Coefficients (MFCCs) are the type of features extracted for use for all the training models. Following which is Cepstral Mean and Variance Normalization or cmvn. This is followed by the building of the language model with a language model order equal to three. Following the building of the language model, the script proceeds to the training and decoding for the different types of acoustic model.

Following feature extraction and language modelling, each of the different models generally proceeds to building a graph derived from the data from the acoustic features and the language model. Features from the utterances are then used with the graph to generate alignments which are subsequently used for decoding.

Monophone Training

The following lines show the configuration options specified for the monophone training.

```
1 # Begin configuration section.
2 nj=4
3 cmd=run.pl
4 scale_opts="--transition-scale=1.0 --acoustic-scale=0.1 --
```

```

    --self-loop-scale=0.1"
5 num_iters=40      # Number of iterations of training
6 max_iter_inc=30 # Last iter to increase #Gauss on.
7 initial_beam=6 # beam used in the first iteration (set
    smaller to speed up initialization)
8 regular_beam=10 # beam used after the first iteration
9 retry_beam=40
10 totgauss=1000 # Target #Gaussians.
11 careful=false
12 boost_silence=1.0 # Factor by which to boost silence
    likelihoods in alignment
13 realign_iters="1_2_3_4_5_6_7_8_9_10_12_14_16_18_20_23_26
    _29_32_35_38";
14 config= # name of config file.
15 stage=-4
16 power=0.25 # exponent to determine number of gaussians
    from occurrence counts
17 norm_vars=false # deprecated, prefer --cmvn-opts "--norm
    -vars=false"
18 cmvn_opts= # can be used to add extra options to cmvn.
19 delta_opts= # can be used to add extra options to add-
    deltas
20 # End configuration section.

```

Note that the `-nj` option is overwritten to 2 from the calling script.

The following lines show the configuration options specified for the Triphone training.

```

1 # Begin configuration.
2 stage=-4 # This allows restarting after partway, when
   something when wrong.
3 config=
4 cmd=run.pl
5 scale_opts="--transition-scale=1.0 --acoustic-scale=0.1
   --self-loop-scale=0.1"
6 realign_iters="10 20 30";
7 num_iters=35 # Number of iterations of training
8 max_iter_inc=25 # Last iter to increase #Gauss on.
9 beam=10
10 careful=false
11 retry_beam=40
12 boost_silence=1.0 # Factor by which to boost silence
   likelihoods in alignment
13 power=0.25 # Exponent for number of gaussians according
   to occurrence counts
14 cluster_thresh=-1 # for build-tree control final bottom
   -up clustering of leaves
15 norm_vars=false # deprecated. Prefer --cmvn-opts "--
   norm-vars=true"
16 # use the option --cmvn-opts "--norm-

```

```

                                means=false"
17 cmvn_opts=
18 delta_opts=
19 context_opts=    # use"--context-width=5 --central-
                    position=2" for quinphone
20 # End configuration.

```

Note that the main script calls the Triphone training script specifying 3200 for `|num-leaves|` and 30000 for `|tot-gauss|`. Also, prior to graph compilation, the Triphone training script utilized a decision tree to help augment the accuracy of the acoustic model.

LDA + MMLT Training

The following configurations were specified for LDA + MMLT Training:

```

1 # Begin configuration.
2 cmd=run.pl
3 config=
4 stage=-5
5 scale_opts="--transition-scale=1.0_--acoustic-scale=0.1_
              --self-loop-scale=0.1"
6 realign_iters="10_20_30";
7 mllt_iters="2_4_6_12";
8 num_iters=35    # Number of iterations of training
9 max_iter_inc=25 # Last iter to increase #Gauss on.

```

```
10 beam=10
11 retry_beam=40
12 boost_silence=1.0 # Factor by which to boost silence
    likelihoods in alignment
13 power=0.2 # Exponent for number of gaussians according
    to occurrence counts
14 randprune=4.0 # This is approximately the ratio by which
    we will speed up the
15                # LDA and MLLT calculations via randomized
                pruning.
16 cluster_thresh=-1 # for build-tree control final bottom
    -up clustering of leaves
17
18 # apply CMVN to the second feature stream?
19 normft2=true
20
21 # Do additional LDA after pasting the features
22 dim2=40
23 extra_lda=false
24
25 # End configuration.
```

Note that the following parameters were specified from the main script: `-left-context=3 -right-context=3` with 2000 for numleaves and 20000 for totgauss.

For the LDA + MLLT model, an additional process called the Linear Discriminant

Analysis for feature space reduction to reduce burden on computing resources. Furthermore, Maximum Likelihood Linear Transform is also applied to improve the accuracy of the acoustic model.

SAT Training

The following are the configurations specified for SAT training.

```
1 # Begin configuration section.
2 stage=-5
3 exit_stage=-100 # you can use this to require it to exit
   at the
4                 # beginning of a specific stage.  Not
   all values are
5                 # supported.
6 fml1r_update_type=full
7 cmd=run.pl
8 scale_opts="--transition-scale=1.0_--acoustic-scale=0.1_
   --self-loop-scale=0.1"
9 beam=10
10 retry_beam=40
11 careful=false
12 boost_silence=1.0 # Factor by which to boost silence
   likelihoods in alignment
13 context_opts= # e.g. set this to "--context-width 5 --
   central-position 2" for quinphone.
14 realign_iters="10_20_30";
```

```
15 fmlr_iters="2_4_6_12";
16 silence_weight=0.0 # Weight on silence in fMLLR
    estimation.
17 num_iters=35 # Number of iterations of training
18 max_iter_inc=25 # Last iter to increase #Gauss on.
19 power=0.2 # Exponent for number of gaussians according
    to occurrence counts
20 cluster_thresh=-1 # for build-tree control final bottom
    -up clustering of leaves
21 phone_map=
22 train_tree=true
23 tree_stats_opts=
24 cluster_phones_opts=
25 compile_questions_opts=
26 # End configuration section.
```

A process that sets SAT apart is its use of Feature space Maximum Likelihood Linear Regression (fMLLR) for speaker adaptive-purposes to improve accuracy of the acoustic model based on the characteristics of a speaker. The SAT model builds on the alignments generated by the LDA+MLLT model, after which it applies fMLLR transforms before proceeding to graph compilation and subsequently, decoding.

TDNN Training

The following configurations are specified for TDNN training.

```
1 decode_nj=1
2 nj=4
3 train_set=train
4 test_sets=test
5 gmm=tri3b
6 nnet3_affix=
7
8 # The rest are configs specific to this script. Most of
   the parameters
9 # are just hardcoded at this level, in the commands
   below.
10 affix=1a    # affix for the TDNN directory name
11 tree_affix=
12 train_stage=-10
13 get_egs_stage=-10
14 decode_iter=
15
16 # training options
17 # training chunk-options
18 chunk_width=140,100,160
19 xent_regularize=0.1
20 bottom_subsampling_factor=1  # I'll set this to 3 later,
   1 is for compatibility with a broken ru.
```

```
21 frame_subsampling_factor=3
22 langs="default" # list of language names
23
24 # The amount of extra left/right context we put in the
    egs. Note: this could
25 # easily be zero, since we're not using a recurrent
    topology, but we put in a
26 # little extra context so that we have more room to play
    with the configuration
27 # without re-dumping egs.
28 egs_extra_left_context=5
29 egs_extra_right_context=5
30
31 # The number of chunks (of length: see $chunk_width
    above) that we group
32 # together for each "speaker" (actually: pseudo-speaker,
    since we may have
33 # to group multiple speaker together in some cases).
34 chunks_per_group=4
35
36 # training options
37 srand=0
38 remove_egs=true
39 reporting_email=
40
```

```
41 #decode options
42 test_online_decoding=true # if true, it will run the
    last decoding stage.
43
44
45 # End configuration section.
```

The script builds on the alignments from the SAT training. The script also extracts ivectors for speaker-specific data which will be fed to the TDNN network. From the ivectors and the alignments, a topology is created, which is subsequently used to build a tree. A graph is then compiled which is used for decoding.

3.1.4 Evaluation

Evaluation for DNN model trained using Kaldi

Performance of the models were measured in terms of WER which was done by getting the average WER for each iteration of the training.

Evaluation of transcriptions using Whisper

The same dataset used in this project was used for transcription using Whisper. Transcription was done using pre-trained models of the following sizes: tiny, base and small.

The generated transcriptions were then formatted in the same format as Kaldi's

transcriptions, specifically in the following format: <file_id>word1 word2 word3... for compatibility purposes.

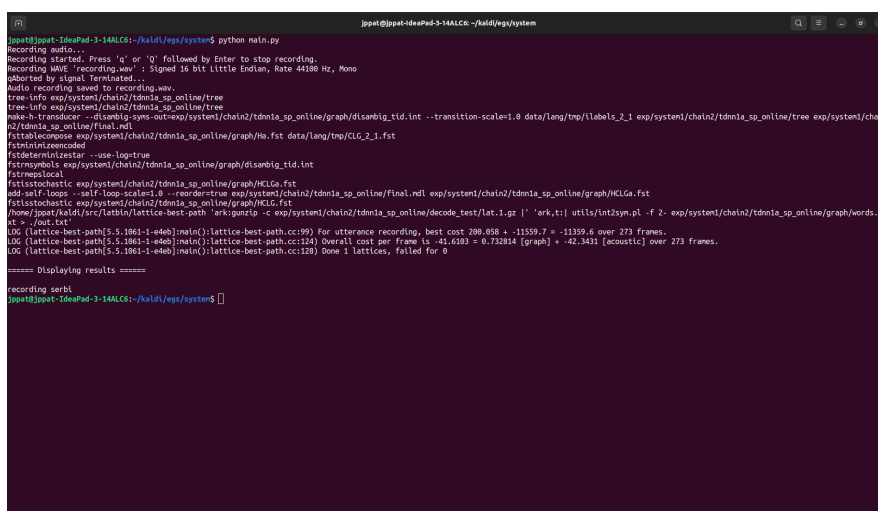
The formatted reference transcription was then compared with the reference transcription for the calculation of WER using Kaldi's compute-wer tool.

3.1.5 System Development

A speech-to-text system with a command line interface was developed using the best-performing DNN model during training.

The system can be called with or without an argument. When called without an argument, the system starts recording. To stop the recording, the user is prompted to press q.

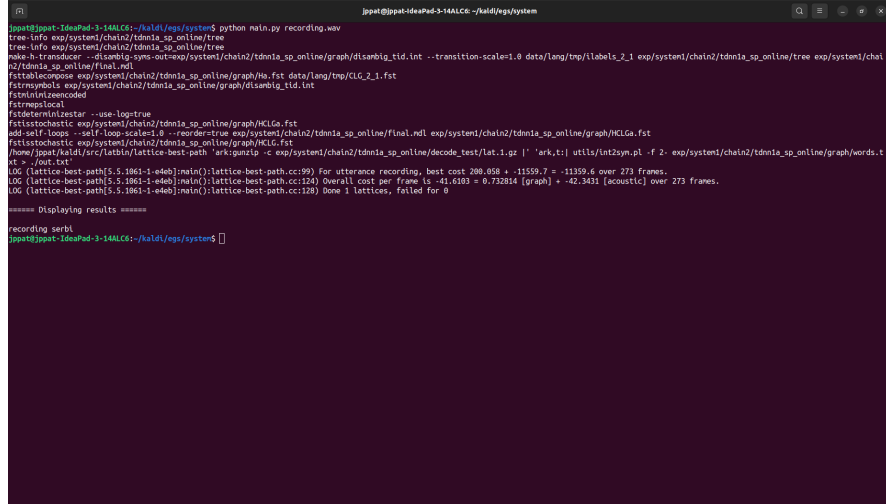
After recording, the system proceeds to decoding. The resulting transcriptions are then printed on the terminal.



```
jppat@jppat-ideaPad-3-14ALC6:~/kaldi/egs/system$ python main.py
Recording audio...
Recording started. Press 'q' or 'Q' followed by Enter to stop recording.
Recording WAVE 'recording.wav' : Signed 16 Bit Little Endian, Rate 44100 Hz, Mono
Aborted by signal terminated...
Audio recording saved to 'recording.wav'.
tree-info exp/system/chain2/tdmia_sp_online/tree
make-h-transducer --disambig-syms-out=exp/system/chain2/tdmia_sp_online/graph/disambig_tld.int --transition-scale=1.0 data/lang/tmp/labels_2_1_exp/system/chain2/tdmia_sp_online/tree exp/system/chain2/tdmia_sp_online/final.mdl
fsttablecompose exp/system/chain2/tdmia_sp_online/graph/Hs.fst data/lang/tmp/CLG_2_1.fst
fstminimizecoded
fsttobinlattice --use-logtrue
fststrings exp/system/chain2/tdmia_sp_online/graph/disambig_tld.int
fsttopological
fstisstochastic exp/system/chain2/tdmia_sp_online/graph/HCLGa.fst
add-self-loop --self-loop-scale=1.0 --read-er-tree exp/system/chain2/tdmia_sp_online/final.mdl exp/system/chain2/tdmia_sp_online/graph/HCLGa.fst
fstisstochastic exp/system/chain2/tdmia_sp_online/graph/HCLG.fst
/home/jppat/kaldi/src/lattice/lattice-best-path 'ark:gunzip -c exp/system/chain2/tdmia_sp_online/decode_test/lat.1.gz |' 'ark,tt:| utils/int2syn.pl -f 2- exp/system/chain2/tdmia_sp_online/graph/words.t
v> -> /out-1.txt'
LOG (lattice-best-path[5.5.1061-1-e4eb]:main):lattice-best-path.ccc:99) For utterance recording, best cost 200.058 + -11559.7 = -11359.6 over 273 frames.
LOG (lattice-best-path[5.5.1061-1-e4eb]:main):lattice-best-path.ccc:124) Overall cost per frame is -45.6103 = 0.732834 [graph] + -42.3431 [acoustic] over 273 frames.
LOG (lattice-best-path[5.5.1061-1-e4eb]:main):lattice-best-path.ccc:128) Done 1 lattices, failed for 0
===== Displaying results =====
recording serbi
jppat@jppat-ideaPad-3-14ALC6:~/kaldi/egs/system$
```

Figure 3.1: Screenshot of the system when called without an argument.

The system also accepts one argument, which is the file name of the audio to be transcribed. Given this argument, the system proceeds to decoding and then it also prints the resulting transcriptions on the terminal.



```

jppat@jppat-IdeaPad-3-14ALC6:~/hald/egs/system$ python main.py recording.wav
tree-info exp/system1/chain2/tdmia_sp_online/tree
tree-info exp/system1/chain2/tdmia_sp_online/tree
make-h-transducer --disambig-syms-out=exp/system1/chain2/tdmia_sp_online/graph/disambig_tid.int --transition-scale=1.0 data/lang/tmp/tlabels_2_1 exp/system1/chain2/tdmia_sp_online/tree exp/system1/chain2/tdmia_sp_online/final.mdl
fstrmsymbols exp/system1/chain2/tdmia_sp_online/graph/Ha.fst data/lang/tmp/CLG_2_1.fst
fstrmsymbols exp/system1/chain2/tdmia_sp_online/graph/disambig_tid.int
fstcompile --cs exp/system1/chain2/tdmia_sp_online/graph/disambig_tid.int
fstcompile --cs exp/system1/chain2/tdmia_sp_online/graph/Ha.fst
fstcompile --cs exp/system1/chain2/tdmia_sp_online/graph/HCLGa.fst
add-self-loops --self-loop-scale=1.0 --reorder-trues exp/system1/chain2/tdmia_sp_online/final.mdl exp/system1/chain2/tdmia_sp_online/graph/HCLGa.fst
fstcompile --cs exp/system1/chain2/tdmia_sp_online/graph/HCLGa.fst
/home/jppat/hald/src/lattice/lattice-best-path 'ark:gunzip -c exp/system1/chain2/tdmia_sp_online/decode_test/lat.1.gz |' 'ark,t| utils/int2sym.pl -f 2- exp/system1/chain2/tdmia_sp_online/graph/words.txt > ./out.txt'
LOG (lattice-best-path[5.5.1061-1-e4eb]:main():lattice-best-path.ccc:99) For utterance recording, best cost 200.858 + .11559.7 = .11359.6 over 273 frames.
LOG (lattice-best-path[5.5.1061-1-e4eb]:main():lattice-best-path.ccc:124) Overall cost per frame is -41.6183 = 0.732814 [graph] + -42.3431 [acoustic] over 273 frames.
LOG (lattice-best-path[5.5.1061-1-e4eb]:main():lattice-best-path.ccc:128) Done 1 lattices, failed for 0
===== Displaying results =====
recording serbi
jppat@jppat-IdeaPad-3-14ALC6:~/hald/egs/system$

```

Figure 3.2: Screenshot of the system with an audio file's file name as an argument.

Chapter 4

Results and Discussions/Analyses

This chapter provides a summary and analysis of the results obtained by decoding different training models, such as monophones, triphones, LDA + MLLT, SAT, and DNN. The information presented includes data extracted from each decoding model's WER files output, including the number of errors made (insertions, deletions, and substitutions), as well as the word and sentence error rates (WER and SER), and the average percentage of errors for each model.

The same information is also presented but for the evaluation of the transcriptions generated by Whisper using the same audio dataset used in the project.

To ensure dependable outcomes, this project underwent rigorous testing using a 6-fold cross-validation technique for each training model. The project involved a total of eighteen speakers, with three speakers assigned for testing in each fold, while the remaining speakers were used for training. The subsequent tables will display the outcomes obtained from this procedure.

4.1 Results of Monophone model

The performance of Monophone model across six folds after acoustic model training is presented in the following tables.

In Table 4.1a, the results for the first fold indicate a Word Error Rate (WER) ranging from wer_7 to wer_17, with an average of 6.35%. The highest WER is 9.07% and the lowest is 5.60%. The Sentence Error Rate (SER) exhibits an average of 69.70%, with the highest and lowest scores at 81.67% and 63.33%, respectively.

Table 4.1b displays the second fold results, with a WER average of 1.87%. The highest WER is 3.07% and lowest is 1.53%. The SER shows an average of 35.45%, with the highest and lowest scores also at 3.07% and 1.53%, respectively.

Table 4.1c presents the outcomes for the third fold, showing an average WER of 0.68%. The highest WER is 0.80%, and the lowest at 0.60%. The average SER is 16.82%, with the highest SER at 20.00% and the lowest at 15.00%.

Table 4.1d presents the results for the fourth fold, indicating a WER average of 0.62%. The highest WER is 1.00%, while the lowest is 0.53%. The average SER is 12.88%, with the highest at 18.33% and the lowest at 11.67%.

In Table 4.1e, the data gathered for the fifth fold reveals a WER average of 4.90%. The highest WER is 8.53% and the lowest is 3.20%. Additionally, the average SER is 30.61%, with the highest SER recorded at 51.67% and the lowest at 23.33%.

Lastly, Table 4.1f presents the results for the sixth fold, with a WER average of 0.44%. The highest WER is 0.67%, and the lowest at 0.33%. The average SER is calculated as 8.94%, with the highest SER at 13.33% and the lowest at 6.67%.

Table 4.1: Results of Monophone model

	WER	SER	ins	del	sub
wer_7	9.07	81.67	5	20	111
wer_8	7.87	78.33	5	20	93
wer_9	6.87	75.00	5	22	76
wer_10	6.53	73.33	4	25	69
wer_11	6.27	71.67	4	25	65
wer_12	5.73	66.67	4	30	52
wer_13	5.67	65.00	4	32	49
wer_14	5.60	65.00	3	34	47
wer_15	5.73	63.33	3	38	45
wer_16	5.93	63.33	3	43	43
wer_17	5.60	63.33	3	43	38
MEAN	6.35	69.70			

(a) First fold decoding results

	WER	SER	ins	del	sub
wer_7	3.07	48.33	1	0	45
wer_8	2.47	41.67	1	0	36
wer_9	2.13	36.67	0	0	32
wer_10	1.73	35.00	0	0	26
wer_11	1.67	33.33	0	0	25
wer_12	1.67	33.33	0	0	25
wer_13	1.67	33.33	0	0	24
wer_14	1.60	33.33	0	0	24
wer_15	1.53	31.67	0	0	23
wer_16	1.53	31.67	0	0	23
wer_17	1.53	31.67	0	0	23
MEAN	1.87	35.45			

(b) Second fold decoding results

	WER	SER	ins	del	sub
wer_7	0.80	20.00	1	0	11
wer_8	0.67	16.67	1	0	9
wer_9	0.67	16.67	1	0	9
wer_10	0.67	16.67	1	0	9
wer_11	0.67	16.67	1	0	9
wer_12	0.67	16.67	1	0	9
wer_13	0.67	16.67	1	0	9
wer_14	0.67	16.67	1	0	9
wer_15	0.67	16.67	1	0	9
wer_16	0.67	16.67	1	0	9
wer_17	0.60	15.00	1	0	8
MEAN	0.68	16.82			

(c) Third fold decoding results

	WER	SER	ins	del	sub
wer_7	1.00	18.33	3	2	10
wer_8	0.80	16.67	3	2	7
wer_9	0.67	13.33	3	2	5
wer_10	0.60	11.67	3	2	4
wer_11	0.60	11.67	3	2	4
wer_12	0.53	11.67	3	2	3
wer_13	0.53	11.67	3	2	3
wer_14	0.53	11.67	3	2	3
wer_15	0.53	11.67	3	2	3
wer_16	0.53	11.67	3	2	3
wer_17	0.53	11.67	3	2	3
MEAN	0.62	12.88			

(d) Fourth fold decoding results

	WER	SER	ins	del	sub
wer_7	8.53	51.67	5	7	116
wer_8	7.40	40.00	5	7	99
wer_9	6.53	36.67	4	8	86
wer_10	5.33	33.33	2	9	69
wer_11	4.60	28.33	2	9	58
wer_12	4.27	26.67	1	9	54
wer_13	3.73	26.67	1	9	46
wer_14	3.47	23.33	1	9	42
wer_15	3.47	23.33	1	9	42
wer_16	3.40	23.33	1	9	41
wer_17	3.20	23.33	1	10	37
MEAN	4.90	30.61			

(e) Fifth fold decoding results

	WER	SER	ins	del	sub
wer_7	0.67	13.33	3	0	7
wer_8	0.67	13.33	3	0	7
wer_9	0.47	10.00	1	0	6
wer_10	0.47	10.00	1	0	6
wer_11	0.47	10.00	1	0	6
wer_12	0.40	8.33	1	0	5
wer_13	0.33	6.67	1	0	4
wer_14	0.33	6.67	1	0	4
wer_15	0.33	6.67	1	0	4
wer_16	0.33	6.67	1	0	4
wer_17	0.33	6.67	1	0	4
MEAN	0.44	8.94			

(f) Sixth fold decoding results

4.2 Results of Triphone model

The tables provided contain data for six fold Triphone models after training the acoustic model.

Table 4.2a represents the data gathered for the first fold Triphone model after training the acoustic model. The word error rates (WER) range from 1.07% to 2.40%, with a mean of 3.09%. The corresponding sentence error rates (SER) range from 20.00% to 35.00%, with a mean of 25.46%.

Table 4.2b represents the data gathered for the second fold Triphone model. The WERs range from 2.73% to 5.07%, with a mean of 3.56%, and the SERs range from 46.67% to 63.33%, with a mean of 54.46%.

Table 4.2c represents the data gathered for the third fold Triphone model. The WERs range from 1.07% to 2.40%, with a mean of 1.45%, and the SERs range from 20.00% to 35.00%, with a mean of 25.46%.

Table 4.2d represents the data gathered for the fourth fold Triphone model. The WERs range from 0.67% to 1.13%, with a mean of 0.79%, and the SERs range from 15.00% to 21.67%, with a mean of 16.97%.

Table 4.2e represents the data gathered for the fifth fold Triphone model. The WERs range from 11.20% to 14.20%, with a mean of 12.09%, and the SERs range from 43.33% to 63.33%, with a mean of 50.60%.

Table 4.2f represents the data gathered for the sixth fold Triphone model. The WERs range from 0.33% to 0.67%, with a mean of 0.45%, and the SERs range from 6.67% to 13.33%, with a mean of 0.24%.

Table 4.2: Results of Triphone model

	WER	SER	ins	del	sub
wer_7	2.40	35.00	2	0	34
wer_8	1.87	31.67	2	0	26
wer_9	1.73	31.67	1	0	25
wer_10	1.67	30.00	1	0	24
wer_11	1.47	26.67	1	0	21
wer_12	1.33	23.33	1	0	19
wer_13	1.20	21.67	1	0	17
wer_14	1.07	20.00	1	0	15
wer_15	1.07	20.00	1	0	15
wer_16	1.07	20.00	1	0	15
wer_17	1.07	20.00	1	0	15
MEAN	3.09	25.46			

(a) First fold decoding results

	WER	SER	ins	del	sub
wer_7	5.07	63.33	4	0	72
wer_8	4.40	60.00	2	0	64
wer_9	4.13	56.67	2	0	60
wer_10	3.87	56.67	1	0	57
wer_11	3.80	55.00	1	0	56
wer_12	3.27	53.33	0	0	49
wer_13	3.07	51.67	0	0	46
wer_14	3.07	51.67	0	0	46
wer_15	2.87	50.00	0	0	43
wer_16	2.87	50.00	0	0	43
wer_17	2.73	46.67	0	0	41
MEAN	3.56	54.46			

(b) Second fold decoding results

	WER	SER	ins	del	sub
wer_7	2.40	35.00	2	0	34
wer_8	1.87	31.67	2	0	26
wer_9	1.73	31.67	1	0	25
wer_10	1.67	30.00	1	0	24
wer_11	1.47	26.67	1	0	21
wer_12	1.33	23.33	1	0	19
wer_13	1.20	21.67	1	0	17
wer_14	1.07	20.00	1	0	15
wer_15	1.07	20.00	1	0	15
wer_16	1.07	20.00	1	0	15
wer_17	1.07	20.00	1	0	15
MEAN	1.45	25.46			

(c) Third fold decoding results

	WER	SER	ins	del	sub
wer_7	1.13	21.67	5	2	10
wer_8	1.00	20.00	5	2	8
wer_9	0.87	18.33	5	2	6
wer_10	0.80	16.67	4	2	6
wer_11	0.73	16.67	4	2	5
wer_12	0.73	16.67	4	2	5
wer_13	0.73	16.67	4	2	5
wer_14	0.67	15.00	4	2	4
wer_15	0.67	15.00	4	2	4
wer_16	0.67	15.00	4	2	4
wer_17	0.67	15.00	4	2	4
MEAN	0.79	16.97			

(d) Fourth fold decoding results

	WER	SER	ins	del	sub
wer_7	14.20	63.33	11	31	171
wer_8	13.40	56.67	8	31	162
wer_9	12.67	55.00	5	34	151
wer_10	12.00	51.67	1	36	143
wer_11	11.93	51.67	1	37	141
wer_12	11.93	50.00	1	41	137
wer_13	11.73	48.33	1	48	127
wer_14	11.40	46.67	1	49	121
wer_15	11.27	45.00	1	50	118
wer_16	11.27	45.00	1	52	116
wer_17	11.20	43.33	1	56	111
MEAN	12.09	50.60			

(e) Fifth fold decoding results

	WER	SER	ins	del	sub
wer_7	0.67	13.33	3	0	7
wer_8	0.67	13.33	3	0	7
wer_9	0.67	13.33	3	0	7
wer_10	0.47	10.00	1	0	6
wer_11	0.47	10.00	1	0	6
wer_12	0.40	8.33	1	0	5
wer_13	0.33	6.67	1	0	4
wer_14	0.33	6.67	1	0	4
wer_15	0.33	6.67	1	0	4
wer_16	0.33	6.67	1	0	4
wer_17	0.33	6.67	1	0	4
MEAN	0.45	0.24			

(f) Sixth fold decoding results

4.3 Result of LDA + MLLT model

Table 4.3a presents data from the first fold LDA + MLLT model after acoustic model training. Word errors (wer_{.7} to wer_{.17}) range from 52.53% to 62.53%, with a mean word error rate of 57.27%. Sentence errors have a constant rate of 100%.

Similarly, Table 4.3b displays data from the second fold, with a mean word error rate of 4.79% and a range of 3.60% to 6.47% for word errors. Sentence error rates have a mean of 48.33%, with the highest at 65.00% and the lowest 48.33%.

Continuing, Table 4.3c contains data from the third fold, showing a mean error rate of 2.21% for words (ranging from 1.93% to 2.73%) and a mean sentence error rate of 29.69% with highest of 35.00% and lowest of 28.33%).

Likewise, Table 4.3d and Table 4.3e present data from the fourth and fifth folds, respectively, with word error rates of 0.71% and 16.30%, and varying sentence error rates.

Finally, Table 4.3f captures data from the sixth fold, showing a mean word error rate of 0.45%, with a range of 0.33% to 0.67%. The mean sentence error rate is 9.24%, ranging from 6.67% to 13.33%.

Table 4.3: Results of LDA + MLLT model

	WER	SER	ins	del	sub		WER	SER	ins	del	sub
wer_7	52.53	100.00	1	661	126	wer_7	6.47	65.00	2	4	91
wer_8	52.87	100.00	1	680	112	wer_8	6.13	65.00	2	4	86
wer_9	53.40	100.00	0	702	99	wer_9	5.67	65.00	2	4	79
wer_10	54.27	100.00	0	724	90	wer_10	5.53	61.67	2	4	77
wer_11	56.13	100.00	0	760	82	wer_11	4.80	56.67	2	3	67
wer_12	57.20	100.00	0	786	72	wer_12	4.47	56.67	1	3	63
wer_13	58.53	100.00	0	813	65	wer_13	4.33	55.00	1	3	61
wer_14	59.80	100.00	0	839	58	wer_14	4.07	55.00	0	4	57
wer_15	60.60	100.00	0	862	47	wer_15	3.87	51.67	0	5	53
wer_16	62.07	100.00	0	887	44	wer_16	3.80	50.00	0	5	52
wer_17	62.53	100.00	0	896	42	wer_17	3.60	48.33	0	5	49
MEAN	57.27	100.00				MEAN	4.79	48.33			

(a) First fold decoding results

(b) Second fold decoding results

	WER	SER	ins	del	sub		WER	SER	ins	del	sub
wer_7	2.73	35.00	1	12	28	wer_7	0.93	21.67	4	2	8
wer_8	2.67	33.33	1	13	26	wer_8	0.87	20.00	4	2	7
wer_9	2.33	30.00	1	13	21	wer_9	0.87	20.00	4	2	7
wer_10	2.20	30.00	1	13	19	wer_10	0.80	18.33	4	2	6
wer_11	2.13	28.33	1	12	19	wer_11	0.73	18.33	4	2	5
wer_12	2.13	28.33	1	12	19	wer_12	0.67	16.67	4	2	4
wer_13	2.13	28.33	1	12	19	wer_13	0.67	16.67	4	2	4
wer_14	2.13	28.33	1	12	19	wer_14	0.60	15.00	4	2	3
wer_15	2.07	28.33	1	12	18	wer_15	0.60	15.00	4	2	3
wer_16	1.93	28.33	1	12	16	wer_16	0.53	13.33	4	2	2
wer_17	2.00	28.33	1	14	15	wer_17	0.53	13.33	4	2	2
MEAN	2.21	29.69				MEAN	0.71	17.12			

(c) Third fold decoding results

(d) Fourth fold decoding results

	WER	SER	ins	del	sub		WER	SER	ins	del	sub
wer_7	16.73	60.00	0	160	91	wer_7	0.67	13.33	3	0	7
wer_8	16.73	58.33	0	165	86	wer_8	0.67	13.33	3	0	7
wer_9	16.60	56.67	0	169	80	wer_9	0.67	13.33	3	0	7
wer_10	16.47	56.67	0	169	78	wer_10	0.47	10.00	1	0	6
wer_11	16.27	55.00	0	173	71	wer_11	0.47	10.00	1	0	6
wer_12	16.27	53.33	0	174	70	wer_12	0.40	8.33	1	0	5
wer_13	16.20	51.67	0	175	68	wer_13	0.33	6.67	1	0	4
wer_14	16.13	50.00	0	178	64	wer_14	0.33	6.67	1	0	4
wer_15	16.07	48.33	0	178	63	wer_15	0.33	6.67	1	0	4
wer_16	15.93	46.67	0	179	60	wer_16	0.33	6.67	1	0	4
wer_17	15.93	46.67	0	181	58	wer_17	0.33	6.67	1	0	4
MEAN	16.30	53.03				MEAN	0.45	9.24			

(e) Fifth fold decoding results

(f) Sixth fold decoding results

4.4 Result of LDA + MLLT + SAT model

Table 4.4a shows data from the first fold LDA + MLLT + SAT model after acoustic model training. Word errors range from 2.13% to 2.87%, with a mean word error rate of 2.40%. The sentence error rate has a mean of 45.61%, with the highest at 48.33% and the lowest at 43.33%.

Similarly, Table 4.4b presents data from the second fold, with a mean word error rate of 1.72% and a range of 1.47% to 2.00% for word errors. The mean sentence error rate is 35.00%, with the highest at 40.00% and the lowest at 31.67%.

Continuing, Table 4.4c contains data from the third fold, showing a mean error rate of 1.71% for words (ranging from 1.67% to 1.80%) and a mean sentence error rate of 19.39% (highest: 21.67%, lowest: 18.33%).

Likewise, Table 4.4d and Table 4.4e present data from the fourth and fifth folds, respectively, with word error rates of 0.66% and 12.75%, and varying sentence error rates.

Finally, Table 4.4f captures data from the sixth fold, showing a mean word error rate of 0.99%, with a range of 0.73% to 1.33%. The mean sentence error rate is 17.58%, ranging from 13.33% to 21.67%.

Table 4.4: Results of LDA + MLLT + SAT model

	WER	SER	ins	del	sub
wer_7	2.87	48.33	6	0	37
wer_8	2.67	46.67	6	0	34
wer_9	2.67	46.67	6	0	33
wer_10	2.60	46.67	6	0	33
wer_11	2.27	45.00	5	0	29
wer_12	2.27	45.00	5	0	29
wer_13	2.27	45.00	5	0	29
wer_14	2.27	45.00	5	0	29
wer_15	2.20	45.00	5	0	28
wer_16	2.20	45.00	5	0	28
wer_17	2.13	43.33	5	0	27
MEAN	2.40	45.61			

(a) First fold decoding results

	WER	SER	ins	del	sub
wer_7	2.00	40.00	0	0	30
wer_8	1.93	38.33	0	0	29
wer_9	1.87	38.33	0	0	28
wer_10	1.80	36.67	0	0	27
wer_11	1.67	33.33	0	0	25
wer_12	1.67	33.33	0	0	25
wer_13	1.67	33.33	0	0	25
wer_14	1.67	33.33	0	0	25
wer_15	1.60	33.33	0	0	24
wer_16	1.53	33.33	0	0	23
wer_17	1.47	31.67	0	0	22
MEAN	1.72	35.00			

(b) Second fold decoding results

	WER	SER	ins	del	sub
wer_7	1.80	21.67	1	12	14
wer_8	1.73	20.00	1	12	13
wer_9	1.73	20.00	1	12	13
wer_10	1.73	20.00	1	12	13
wer_11	1.73	20.00	1	12	13
wer_12	1.73	20.00	1	12	13
wer_13	1.67	18.33	1	12	12
wer_14	1.67	18.33	1	12	12
wer_15	1.67	18.33	1	12	12
wer_16	1.67	18.33	1	12	12
wer_17	1.67	18.33	1	12	12
MEAN	1.71	19.39			

(c) Third fold decoding results

	WER	SER	ins	del	sub
wer_7	0.80	16.67	4	3	5
wer_8	0.80	16.67	4	3	5
wer_9	0.80	16.67	4	3	5
wer_10	0.67	16.67	4	3	3
wer_11	0.60	15.00	4	3	2
wer_12	0.60	15.00	4	3	2
wer_13	0.60	15.00	4	3	2
wer_14	0.60	15.00	4	3	2
wer_15	0.60	15.00	4	3	2
wer_16	0.60	15.00	4	3	2
wer_17	0.60	15.00	4	3	2
MEAN	0.66	15.61			

(d) Fourth fold decoding results

	WER	SER	ins	del	sub
wer_7	12.73	40.00	2	133	56
wer_8	12.67	40.00	2	134	54
wer_9	12.73	38.33	2	137	52
wer_10	12.67	38.33	2	138	50
wer_11	12.60	38.33	2	139	48
wer_12	12.60	36.67	1	145	43
wer_13	12.60	36.67	1	148	40
wer_14	12.80	35.00	1	154	37
wer_15	12.93	35.00	1	157	36
wer_16	12.93	35.00	1	158	35
wer_17	13.00	35.00	1	159	35
MEAN	12.75	37.12			

(e) Fifth fold decoding results

	WER	SER	ins	del	sub
wer_7	1.33	21.67	4	0	16
wer_8	1.20	20.00	4	0	14
wer_9	1.20	20.00	4	0	14
wer_10	1.07	20.00	2	0	14
wer_11	0.93	16.67	2	0	12
wer_12	0.93	16.67	2	0	12
wer_13	0.93	16.67	2	0	12
wer_14	0.87	16.67	2	0	11
wer_15	0.87	16.67	2	0	11
wer_16	0.80	15.00	2	0	10
wer_17	0.73	13.33	2	0	9
MEAN	0.99	17.58			

(f) Sixth fold decoding results

4.5 Result of DNN model

Table 4.5a presents data for the first fold DNN model after training the acoustic model, with a mean word error rate of 2.30% and a mean sentence error rate of 42.27%. The highest word error rate observed is 3.40%, and the lowest is 1.73

Table 4.5b shows data for the second fold DNN model after training the acoustic model. The mean word error rate is 1.90%, and the mean sentence error rate is 36.97%. The highest word error rate is 2.40%, and the lowest is 1.60%.

Table 4.5c provides data for the third fold DNN model after training the acoustic model. The word error rate is consistently 0.73%, and the sentence error rate is 18.33%.

Table 4.5d displays data for the fourth fold DNN model after training the acoustic model, with a mean word error rate of 0.40% and a mean sentence error rate of 10.00%. The highest word error rate observed is 0.47%, and the lowest is 0.33%.

Table 4.5e exhibits data for the fifth fold DNN model after training the acoustic model, showing a mean word error rate of 8.29% and a mean sentence error rate of 28.18%. The highest word error rate is 11.20%, and the lowest is 6.47%.

Table 4.5f includes data for the sixth fold DNN model after training the acoustic model. The mean word error rate is 0.45%, and the mean sentence error rate is 8.48%. The highest word error rate observed is 0.93%, and the lowest is 0.20%.

Table 4.5: Results of DNN model

	WER	SER	ins	del	sub
wer_7	3.40	56.67	6	0	45
wer_8	3.07	51.67	5	0	41
wer_9	2.80	51.67	5	0	37
wer_10	2.67	50.00	4	0	36
wer_11	2.13	38.33	4	0	28
wer_12	2.13	38.33	4	0	28
wer_13	1.93	36.67	4	0	25
wer_14	1.93	36.67	4	0	25
wer_15	1.73	35.00	4	0	22
wer_16	1.73	35.00	4	0	22
wer_17	1.73	35.00	4	0	22
MEAN	2.30	42.27			

(a) First fold decoding results

	WER	SER	ins	del	sub
wer_7	2.40	41.67	1	0	35
wer_8	2.20	38.33	1	0	32
wer_9	2.13	38.33	1	0	31
wer_10	2.00	38.33	0	0	30
wer_11	1.93	38.33	0	0	29
wer_12	1.87	38.33	0	0	28
wer_13	1.73	35.00	0	0	26
wer_14	1.67	35.00	0	0	25
wer_15	1.67	35.00	0	0	25
wer_16	1.67	35.00	0	0	25
wer_17	1.60	33.33	0	0	24
MEAN	1.90	36.97			

(b) Second fold decoding results

	WER	SER	ins	del	sub
wer_7	0.73	18.33	1	0	10
wer_8	0.73	18.33	1	0	10
wer_9	0.73	18.33	1	0	10
wer_10	0.73	18.33	1	0	10
wer_11	0.73	18.33	1	0	10
wer_12	0.73	18.33	1	0	10
wer_13	0.73	18.33	1	0	10
wer_14	0.73	18.33	1	0	10
wer_15	0.73	18.33	1	0	10
wer_16	0.73	18.33	1	0	10
wer_17	0.73	18.33	1	0	10
MEAN	0.73	18.33			

(c) Third fold decoding results

	WER	SER	ins	del	sub
wer_7	0.47	11.67	3	2	2
wer_8	0.47	11.67	3	2	2
wer_9	0.47	11.67	3	2	2
wer_10	0.47	11.67	3	2	2
wer_11	0.47	11.67	3	2	2
wer_12	0.40	10.00	3	2	1
wer_13	0.33	8.33	3	2	0
wer_14	0.33	8.33	3	2	0
wer_15	0.33	8.33	3	2	0
wer_16	0.33	8.33	3	2	0
wer_17	0.33	8.33	3	2	0
MEAN	0.40	10.00			

(d) Fourth fold decoding results

	WER	SER	ins	del	sub
wer_7	11.20	36.67	24	6	138
wer_8	10.47	36.67	19	6	132
wer_9	9.60	33.33	13	6	125
wer_10	9.27	33.33	11	6	122
wer_11	8.33	31.67	9	6	110
wer_12	7.60	30.00	7	6	101
wer_13	7.47	30.00	7	6	99
wer_14	7.33	28.33	7	7	96
wer_15	7.00	26.67	6	9	90
wer_16	6.47	26.67	5	9	83
wer_17	6.47	26.67	5	9	83
MEAN	8.29	30.00			

(e) Fifth fold decoding results

	WER	SER	ins	del	sub
wer_7	0.93	20.00	4	0	10
wer_8	0.73	15.00	3	0	8
wer_9	0.67	13.33	3	0	7
wer_10	0.60	11.67	2	0	7
wer_11	0.53	10.00	2	0	6
wer_12	0.33	5.00	1	0	4
wer_13	0.33	5.00	1	0	4
wer_14	0.27	3.33	1	0	3
wer_15	0.20	3.33	0	0	3
wer_16	0.20	3.33	0	0	3
wer_17	0.20	3.33	0	0	3
MEAN	0.45	8.48			

(f) Sixth fold decoding results

4.6 Summary of results

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Mean
Monophone	6.35	1.87	0.68	0.62	4.90	0.44	2.48
Triphone	3.09	3.56	1.45	0.79	12.09	0.45	3.57
LDA + MLLT	57.27	4.79	2.21	0.71	16.30	0.45	13.62
SAT	2.40	1.72	1.71	0.66	12.75	0.99	3.37
DNN	2.30	1.90	0.73	0.40	8.29	0.45	2.35

Table 4.6: Summary of WER means of acoustic models in each fold

Table 4.6 displays the compiled mean word error rates for each acoustic model across the different folds. The monophone model achieved the lowest word error rate of 0.44% on the sixth fold. Similarly, the triphone model obtained a word error rate of 0.45% on the sixth fold. The LDA + MLLT model exhibited optimal performance on the sixth fold, attaining a word error rate of 0.45%. Conversely, the SAT model yielded the lowest word error rate of 0.66% on the fourth fold. Notably, the DNN model showcased exceptional efficacy with the dataset, achieving a word error rate of 0.40% on the fourth fold and yielding the lowest mean of 2.35% among all the means obtained from different acoustic models.

Each model demonstrated satisfactory word error rates across different folds. Among these models, the Deep Neural Network or DNN model stood out with the lowest word error rate of 0.40% on the fourth fold. Consequently, this model was selected for the development of an automatic speech recognition system designed for the Hiligaynon language.

4.7 Results on the performance of Whisper

Model size	WER	SER	Insertion	Deletion	Substitution
tiny	62.11	100.00	631	63	4896
base	49.02	100.00	429	19	3964
small	25.50	99.72	86	13	2196

Table 4.7: WER and SER results when transcribing in Hiligaynon using different sizes of pre-trained models trained with Whisper

It is seen that the performance of the transcription is proportional to the size of the pre-trained model used. The small model performed best with a WER of 25.50% and SER of 99.72%.

It is important to note that the language specified for decoding was Tagalog since it is the closest language available in the Whisper system. However, not specifying the language does lead Whisper to detect the language as Tagalog, based on the text. Moreover, Wolfenden (2019) mentions similarities between the phonology of Tagalog and Hiligaynon, hence the researchers found it reasonable to proceed with decoding.

Furthermore, the calculation of the WER scores is based on the assumption that we follow the spelling of the words as indicated in the source corpus for Hiligaynon used in this project. Hence, transcriptions made by Whisper may have influences based on the Tagalog language model leading to some words transcribed in a variation of its spelling, to which a human may think of as similar words but different to a model. This is evident particularly on the 'i' and 'e' or 'o' and 'u' sounds where the difference may translate to an error.

Chapter 5

Conclusion

In this paper, the researchers trained different ASR models in Kaldi to build a speech-to-text (STT) system with a command line interface. The following acoustic models were considered for training: Monophone, Triphone, LDA + MLLT, SAT and DNN. Eighteen speakers (9 male and 9 female) were gathered for recording. Each speaker spoke 500 randomly chosen words from a dictionary of 1000 words which were considered for recording. However, there is a total of 3,500 words which were included in the system's lexicon. Each word in the lexicon comes with their manually transcribed phonemic transcriptions which is used for training. A six-fold cross validation scheme was followed during training. Results show that the DNN model yielded the lowest word error rate of 2.35% followed by the Monophone model at 2.48%. The following are the results of the remaining models in ascending order: SAT at 3.37%, Triphone at 3.57%, and LDA +MLLT at 13.62%. Considering this, the DNN model was used to build the speech-to-text (STT) system. Specifically, the model produced during the fourth iteration of the

training of the DNN model was used after yielding the lowest WER of 0.40%.

Additionally, Whisper was also used to generate transcriptions of the same audio dataset using different sizes of pre-trained models available in Kaldi; specifically the tiny, base and small models which contain multilingual data. It's important to note that the language parameter passed to the Whisper system was Tagalog. Considering that it is the only Philippine language officially supported by Whisper, as well as considering the phonetic similarities of Tagalog and Hiligaynon, the researchers found it reasonable to proceed with the transcription. Results show that the size of the model is proportional to the performance of the system. The small model yielded the best WER of 25.50%, followed by the base model at 49.02%, and finally by the tiny model at 62.11%.

5.1 Recommendations

Future researchers may conduct further studies by adding to the complexity of the words used in training the system. The researchers may add more words beyond 3 syllables, or train using grammatically correct sentences instead of training just by word-for-word. They may also try to investigate deeper into the linguistic properties of Hiligaynon and incorporate those insights, for instance, in a building a more nuanced phonetic dictionary for the system. Furthermore, other researchers may want to explore techniques on how to make the system more sensitive to the speakers' emotion.

Chapter 6

References

References

- Aquino, A., Tsang, J. L., Lucas, C. R., & de Leon, F. (2019, 8). G2P and ASR techniques for low-resource phonetic transcription of Tagalog, Cebuano, and Hiligaynon. *2019 International Symposium on Multimedia and Communication Technology (ISMAC)*. Retrieved from <http://dx.doi.org/10.1109/ismac.2019.8836168> doi: 10.1109/ismac.2019.8836168
- Billones, R. K. C., & Dadios, E. P. (2014, 11). Hiligaynon language 5-word vocabulary speech recognition using Mel frequency cepstrum coefficients and genetic algorithm. *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. Retrieved from <http://dx.doi.org/10.1109/hnicem.2014.7016247> doi: 10.1109/hnicem.2014.7016247
- Data types in the "nnnet3" setup.* (n.d.). https://kaldi-asr.org/doc/dnn3_code_data_types.html. (Accessed on June 8, 2023)
- Deep neural networks in kaldi.* (n.d.). <https://kaldi-asr.org/doc/dnn.html>. (Accessed on June 8, 2023)
- Dimzon, F. D., & Pascual, R. M. (2020, 12). An Automatic Phoneme Recognizer for Children's Filipino Read Speech. *2020 IEEE International Confer-*

- ence on Teaching, Assessment, and Learning for Engineering (TALE)*. Retrieved from <http://dx.doi.org/10.1109/tale48869.2020.9368399> doi: 10.1109/tale48869.2020.9368399
- Fasold, R. W., & Connor-Linton, J. (2014). *An introduction to language and linguistics*. Cambridge university press.
- Gales, M., Young, S., et al. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3), 195–304.
- Gavieta, D. M., Honeyman, J., & Samson, A. M. (2022). *Hilispeech: A hiligaynon speech recognition system*. Division of Physical Sciences and Mathematics, College of Arts and Sciences, University of the Philippines Visayas. (Unpublished special problem paper)
- Hannun, A. (n.d.). *Future speech*. <https://awni.github.io/future-speech/>. (Accessed on April 28, 2023)
- Hatfield, H. (2023). *Dynamic approaches to phonological processing*. Cambridge University Press. doi: 10.1017/9781009258661
- Huang, X., Baker, J., & Reddy, R. (2014, 01). A historical perspective of speech recognition. *Communications of the ACM*, 57, 94-103. doi: 10.1145/2500887
- Kipyatkova, I., & Karpov, A. (2016). DNN-Based Acoustic Modeling for Russian Speech Recognition Using Kaldi. *Speech and Computer*, 246–253. Retrieved from http://dx.doi.org/10.1007/978-3-319-43958-7_29 doi: 10.1007/978-3-319-43958-7_29

- Klatt, D. H. (1979). Speech perception: a model of acoustic–phonetic analysis and lexical access. *Journal of Phonetics*, 7(3), 279–312. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0095447019310599> doi: [https://doi.org/10.1016/S0095-4470\(19\)31059-9](https://doi.org/10.1016/S0095-4470(19)31059-9)
- Pisoni, D. B., & Luce, P. A. (1987). Acoustic-phonetic representations in word recognition. *Cognition*, 25(1-2), 21–52.
- Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... Vesely, K. (2011, December). The kaldi speech recognition toolkit. In *Ieee 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society. (IEEE Catalog No.: CFP11SRW-USB)
- PSA. (2023, 3). *Tagalog is the most widely spoken language at home (2020 census of population and housing)*. Press Release. (<https://psa.gov.ph/content/tagalog-most-widely-spoken-language-home-2020-census-population-and-housing>)
- Radford, A., Kim, J., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022, 12). Robust speech recognition via large-scale weak supervision.
- Summa Linguae. (2021). *Speech recognition software: History and future*. Retrieved from <https://summalinguae.com/language-technology/speech-recognition-software-history-future/> (Accessed on April 28, 2023)
- Upadhyaya, P., Farooq, O., Abidi, M. R., & Varshney, Y. V. (2017, 3). Continuous hindi speech recognition model based on Kaldi ASR toolkit. *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. Retrieved from <http://dx.doi.org/10.1109/wispnet.2017.8299868> doi: 10.1109/wispnet.2017.8299868

- Veselý, K., Ghoshal, A., Burget, L., & Povey, D. (2013, 01). Sequence-discriminative training of deep neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTER-SPEECH*, 2345-2349.
- Wolfenden, E. (2019). *Hiligaynon Reference Grammar* (Open Access ed.). University of Hawaii Press. Retrieved from <https://core.ac.uk/display/211329359>
- Zenkel, T., Sanabria, R., Metze, F., Niehues, J., Sperber, M., Stüker, S., & Waibel, A. (2017). Comparison of decoding strategies for ctc acoustic models. *arXiv preprint arXiv:1708.04469*.

Appendix A

Appendix

A.1 Code snippets

Training the different ASR acoustic models

```
1  ### Mono Training
2
3  if [ $stage -le 0 ]; then
4
5      echo
6      echo "Starting stage 0"
7      echo
8      echo
9      echo "==== MONO TRAINING ====="
10     echo
```

```
11
12 steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/
    train data/lang exp/system1/mono || exit 1
13 data/train data/lang exp/system1/mono || exit 1
14
15 # Graph compilation
16 utils/mkgraph.sh data/lang exp/system1/mono exp/system1/
    mono/graph || exit 1
17 fi
18
19 if [ $stage -le 3 ]; then
20 # Decoding
21 echo
22 echo "====_MONO_DECODING_===="
23 echo
24 steps/decode.sh --nj $decode_nj --cmd "$decode_cmd" exp/
    system1/mono/graph data/test exp/system1/mono/decode
25 echo -e "Mono_training_done.\n"
26
27 fi
```

```
1 if [ $stage -le 4 ]; then
2     echo
3     echo "Starting_stage_4"
4     echo
5
```

```
6 echo
7 echo "====_TRIPHONE_TRAINING_===="
8 echo
9
10 steps/align_si.sh --boost-silence 1.25 --nj $nj --cmd "
    $train_cmd" data/train data/lang exp/system1/mono exp
    /system1/mono_ali
11
12 data/train data/lang exp/system1/mono_ali exp/system1/
    tri1
13
14 steps/train_deltas.sh --boost-silence 1.25 --cmd "
    $train_cmd" 3200 30000 data/train data/lang exp/
    system1/mono_ali exp/system1/tri1
15
16 ## Graph compilation
17 utils/mkgraph.sh data/lang exp/system1/tri1 exp/system1
    /tri1/graph
18
19 # Decoding
20 echo
21 echo "====_TRIPHONE_DECODING_===="
22 echo
23
24 steps/decode.sh --nj $decode_nj --cmd "$train_cmd" exp/
```

```

system1/tri1/graph  data/test exp/system1/tri1/decode
/tri1/decode
25
26 fi

```

```

1 # train an LDA+MLLT system.
2 if [ $stage -le 5 ]; then
3     echo
4     echo "Starting stage 5"
5     echo
6     echo "====_TRIPHONE_LDA_MLLT_TRAINING_===="
7     steps/align_si.sh --nj $nj --cmd "$train_cmd" data/
        train data/lang exp/system1/tri1 exp/system1/
        tri1.ali
8     steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-
        opts "--left-context=3--right-context=3" 2000
        20000 data/train data/lang exp/system1/tri1.ali
        exp/system1/tri2b
9
10    # Graph compilation
11    utils/mkgraph.sh data/lang exp/system1/tri2b exp/
        system1/tri2b/graph
12
13    # Decoding
14    echo "====_TRIPHONE_LDA_MLLT_DECODING_===="
15    steps/decode.sh --nj $decode_nj --cmd "$train_cmd" exp

```

```

    /system1/tri2b/graph  data/test exp/system1/tri2b/
    decode
16 fi

1 # Train tri3b, which is LDA+MLLT+SAT
2 if [ $stage -le 6 ]; then
3     echo
4     echo "Starting stage 6"
5     echo
6
7     # Align utts using the tri2b model
8     steps/align_si.sh --nj $nj --cmd "$train_cmd" --use-
        graphs true data/train data/lang exp/system1/tri2b
        exp/system1/tri2b.ali
9     steps/train_sat.sh --cmd "$train_cmd" 2000 20000 data/
        train data/lang exp/system1/tri2b.ali exp/system1/
        tri3b
10    utils/mkgraph.sh data/lang exp/system1/tri3b exp/
        system1/tri3b/graph
11    steps/decode_fmllr.sh --nj $decode_nj --cmd "
        $train_cmd" exp/system1/tri3b/graph data/test exp/
        system1/tri3b/decode
12 fi
13
14 if [ $stage -le 8 ]; then
15     echo
```



```
16  echo "Starting stage 8"
17  echo
18  steps/align_fmllr.sh --nj $nj --cmd "$train_cmd" data/
    train data/lang exp/system1/tri3b exp/system1/
    tri3b_ali
19
20  # decode using the tri3b model
21  utils/mkgraph.sh data/lang exp/system1/tri3b exp/
    system1/tri3b/graph
22  steps/decode_fmllr.sh --nj $decode_nj --cmd "
    $decode_cmd" exp/system1/tri3b/graph data/test exp/
    system1/tri3b/decode
23  fi
```

```
1  # Train a chain model
2  if [ $stage -le 9 ]; then
3      echo
4      echo "Starting stage 9: Chain Model"
5      echo
6      local chain2/run_tdnn.sh
7  fi
8
9  if [ $stage -le 10 ]; then
10     echo
11     echo "Starting stage 10: Display results"
12     echo
```

```
13  for x in exp/system1/*/decode*;
14      do
15          [ -d $x ] && grep WER $x/wer_* | utils/
              best_wer.sh;
16      done
17
18  for x in exp/system1/chain2/*/decode*;
19      do [ -d $x ] && grep WER $x/wer_* | utils/
              best_wer.sh; done
20  fi
```

Code snippets for the speech-to-text system

```
1  # Prepare online decoding
2  os.makedirs("./exp/system1/chain2/tdnn1a_sp_online",
              exist_ok=True)
3  bash_out = s.run("steps/online/nnet3/
              prepare_online_decoding.sh --mfcc-config conf/
              mfcc_hires.conf "data/lang exp/system1/nnet3/
              extractor exp/system1/chain2/tdnn1a_sp exp/system1/
              chain2/tdnn1a_sp_online", stdout=f, text=True, shell=
              True)
4
5  # Create decoding graph
```

```

6 os.makedirs("./exp/system1/chain2/tdnn1a_sp_online/graph
    ", exist_ok=True)
7 bash_out = s.run("utils/mkgraph.sh --self-loop-scale 1.0
    " "data/lang_exp/system1/chain2/tdnn1a_sp_online_exp
    /system1/chain2/tdnn1a_sp_online/graph", stdout=f,
    text=True, shell=True)
8
9 # Decode using the created graph
10 os.makedirs("./exp/system1/chain2/tdnn1a_sp_online/
    decode_test", exist_ok=True)
11 bash_out = s.run("steps/online/nnet3/decode.sh --acwt
    1.0 --post-decode-acwt 10.0 --nj 1" "exp/system1/
    chain2/tdnn1a_sp_online/graph." "exp/system1/chain2
    /tdnn1a_sp_online/decode_test", stdout=f, text=True,
    shell=True)
12
13 # GET TRANSCRIPTION
14 gz_location = "exp/system1/chain2/tdnn1a_sp_online/
    decode_test/lat.1.gz"
15 words_txt_loc = "exp/system1/chain2/tdnn1a_sp_online/
    graph/words.txt"
16 command = "~/kaldi/src/latbin/lattice-best-path" \ "ark
    :`gunzip -c {0}|' " \
17         "'ark,t:|utils/int2sym.pl -f 2-" \

```

```
18         "{1}_>_/out.txt' ".format(gz_location,
19         words_txt_loc)
bash_out = s.run(command, stdout=f, text=True, shell=
    True)
```

Transcribing using Whisper

```
1 #!/bin/bash
2
3 # Directory path
4 directory="./audio"
5
6 # Loop through each file in the directory
7 for file in "$directory"/*.wav; do
8     if [ -f "$file" ]; then
9         echo "transcribing_file:_" "$file"."
10         whisper "$file" --language Tagalog --model base
11     fi
12 done
13
14 echo "transcribing_done"
```